HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Dương Đỗ Nhuận

PHÁT HIỆN XÂM NHẬP MẠNG SỬ DỤNG HỌC MÁY

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Dương Đỗ Nhuận

PHÁT HIỆN XÂM NHẬP MẠNG SỬ DỤNG HỌC MÁY

CHUYÊN NGÀNH : **HỆ THỐNG THÔNG TIN**

MÃ SỐ : **8.48.01.04**

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. HOÀNG XUÂN DẬU

HÀ NỘI – 2020

LỜI CAM ĐOAN

Luận văn này là thành quả của quá trình học tập nghiên cứu của em cùng sự giúp đỡ, khuyến khích của các quý thầy cô sau 02 năm em theo học chương trình đào tạo Thạc sĩ, chuyên ngành Hệ thống thông tin của trường Học viện Công nghệ Bưu chính Viễn thông.

Em cam đoan đây là công trình nghiên cứu của riêng em. Nội dung của luận văn có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn sách, tạp chí được liệt kê trong danh mục các tài liệu tham khảo và được trích dẫn hợp pháp.

Tác giả

Dương Đỗ Nhuận

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
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
DAE	Phương pháp học sâu Denoise Autoencoder
DoS	Tấn công từ chối dịch vụ - Denial of Service
DT	Thuật toán học máy cây quyết định - Decision Tree
FN	Sai âm - False Negative
FP	Sai dương - False Positive
GD	Gradient Descent
HIDS	Hệ thống phát hiện xâm nhập ở mức máy – Host Intrusion Detection System
IDS	Hệ thống phát hiện xâm nhập - Intrusion Detection System
KNN	Thuật toán k láng giềng ngắn nhất - K- nearest Neighbors
NB	Thuật toán Naive Baves
NIDS	Hệ thống phát hiện ở mức mạng – Network Intrusion Detection System
R2L	Tấn công từ xa đến cục bộ - Remote to Local
ReLU	Rectified Linear Unit
RF	Thuật toán rừng ngẫu nhiên - Random Forest
ROC	Receiver Operating Characteristics
SAE	Phương pháp học sâu Stacked Autoencoder
SDAE	Phương pháp học sâu Stacked Denoise Autoencoder
SGD	Stochastic Gradient Descent
SVM	Thuật toán máy véc tơ hỗ trợ - Support Vector Machine
TP	Đúng dương - True Positive
TN	Đúng âm - True Negative
U2R	User to Root

Danh mục các bảng

Danh mục các hình

MỤC LỤC

PHẦN MỞ ĐẦU

1. Tính cấp thiết của đề tài

Hiện nay, với sự phát triển nhanh chóng của CNTT cũng như tự động hóa, thế giới bước vào kỷ nguyên 4.0, nhiều ngành nghề, lĩnh vực trong xã hội áp dụng, triển khai các ứng dụng CNTT để tăng năng suất lao động cũng như chất lượng sản phẩm. Các hệ thống CNTT có kết nối Internet ngày càng được mở rộng, nhiều tính năng, được sử dụng rất rộng rãi và có đóng góp rất quan trọng vào phát triển kinh tế. Tuy vậy, khi các hệ thống CNTT có kết nối Internet được ứng ngày càng rộng rãi, nguy cơ mất ATTT cũng ngày càng lớn xuất phát từ các truy cập, xâm nhập bất hợp pháp để đánh cắp thông tin nhạy cảm, hoặc phá hoại hệ thống. Thời gian gần đây, các vụ xâm nhập vào các hệ thống CNTT trên toàn cầu diễn ra với tần suất ngày càng nhiều và mức độ phá hoại ngày càng nghiêm trọng.

Tháng 4 năm 2015, văn phòng quản lý nhân sự của Mỹ phát hiện ra rằng hệ thống mạng của họ bị hacker xâm nhập hệ thống lấy đi thông tin có giá trị của ít nhất 25,1 triệu người bao gồm Số an sinh xã hội (SSN), 5,6 triệu dấu vân tay. Vụ việc đã được các quan chức liên bang Mỹ mô tả là một trong những vi phạm dữ liệu chính phủ nghiêm trọng nhất trong lịch sử Hoa Kỳ. Vụ tấn công này được cho là thực hiện bởi các hacker tới từ Trung Quốc. Tuy nhiên Chính phủ Trung Quốc phủ nhận mọi liên quan đến vụ tấn công mạng này.

Trong năm 2017, Equifax là một trong ba tổ chức báo cáo tín dụng tiêu dùng lớn nhất, cùng với Experian và TransUnion đã bị hacker xâm nhập vào hệ thống máy chủ và lấy đi khoảng 209,000 thông tin thẻ tín dụng và số an sinh xã hội. Nguyên nhân của vụ việc là Equifax đã không thực hiện cập nhật các bản vá lỗ hổng bảo mật của nền tảng Apache Struts sử dụng trong hệ thống máy chủ của họ.

Xâm nhập trái phép có thể được thực hiện trong một thời gian dài trước khi chúng bị phát hiện và đó là những gì đã xảy ra trong trường hợp của Marriott Hotels khi cơ sở dữ liệu Starwood của hãng bị xâm nhập. Hacker đã xâm nhập vào cơ sở dữ liệu và đánh cắp dữ liệu của khoảng 500 triệu khách hàng của Marriott. Sự cố được cho là bắt đầu từ năm 2014. Chính phủ Mỹ cho rằng vụ việc do các hacker liên quan đến chính phủ của Trung Quốc, tuy nhiên phía Trung Quốc phủ nhận cáo buộc này. Marriott phải đối mặt với khoản tiền phạt 123 triệu USD do không thể bảo vệ dữ liệu khách hàng.

Trong tháng 11/2019, một sự cố nghiêm trọng trong lĩnh vực tài chính - ngân hàng đã xảy ra gây hoang mang dư luận, đó là sự cố lộ 02 triệu mục dữ liệu của một ngân hàng lớn tại Việt Nam. Theo đó, trên diễn đàn của giới hacker Raidforums, nơi

chuyên đăng và rao bán những cơ sở dữ liệu bị hack đã tải lên một tập tin dữ liệu được cho là có chứa thông tin người dùng của một ngân hàng tại Việt Nam bao gồm mã khách hàng, họ tên, ngày tháng năm sinh, số điện thoại, email, địa chỉ nhà riêng và nơi công tác của khoảng 02 triệu người Việt Nam.

Trong một cuộc tấn công thư điện tử doanh nghiệp (Business email compromise – BEC) năm 2020, tin tặc đã sử dụng một số kỹ thuật tấn công, lừa đảo hết sự tinh vi, phức tạp và được tổ chức tốt để chiếm quyền điều khiển một số tài khoản thư điện tử cao cấp của ba công ty tài chính có trụ sở ở Anh và Israel. Sau đó tin tặc lừa ba công ty này chuyển khoản tổng cộng 1,3 triệu đô la cho các tài khoản ngân hàng của chúng – trong khi các nạn nhân nghĩ rằng họ đã chuyển tiền theo hợp đồng đầu tư với một số công ty khởi nghiệp.

Như vậy, ảnh hưởng từ những nguy cơ mất ATTT nói chung từ những hành vi xâm nhập trái phép nói riêng ngày càng lớn và phức tạp. Việc nghiên cứu các công nghệ, giải pháp để xây dựng các hệ thống phát hiện xâm nhập (Intrustion Detection System - IDS) hoạt động hiệu quả, chính xác là rất cấp thiết để có thể đối phó với các hành vi xâm nhập mạng trái phép ngày càng phức tạp, tinh vi hiện nay.

Trong thời gian gần đây, học máy (Machine Learing) được áp dụng, đầu tư phát triển và đã giải quyết được nhiều vấn đề khó khăn, phức tạp trong nhiều lĩnh vực của xã hội. Do đó, tôi đã quyết định chọn đề tài "***Phát hiện xâm nhập mạng sử dụng học máy***" để có cơ sở nghiên cứu xây dựng, triển khai hệ thống phát hiện xâm nhập mạng hiệu quả, chính xác trong mạng CNTT tại cơ quan tôi đang làm việc.

2. Tổng quan vấn đề nghiên cứu

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. Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào thư mục tương ứng. Trong học máy, có nhiều thuật toán được áp dụng cho việc "học" của hệ thống như: Linear Regression (Hồi quy tuyến tính), Logistic Regression (Hồi quy logistic), Linear Discriminant Analysis (Phân tích phân loại tuyến tính), DT (Decision Tree- Cây quyết định), NB (Naive Bayes), KNN (K-Nearest Neighbors), Học Vector Quantization, SVM (Support Vector Machine), RF (Random Forest) và mạng nơ ron nhân tạo. Mỗi thuật toán có một vài điểm mạnh, điểm yếu riêng và phù hợp với một số lớp bài toán.

Mạng nơ ron nhân tạo (Artificial Neural Network – ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thần kinh, bao gồm số lượng lớn các neural được gắn kết để xử lý thông tin. ANN giống như bộ não con người, được

học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu trữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).

Deep Learning là một nhánh của lĩnh vực 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. Một quan sát (như hình ảnh) có thể được biểu diễn bằng nhiều cách như một vector của các giá trị cường độ cho mỗi điểm ảnh hoặc trừu tượng hơn như là tập hợp các cạnh, các khu vực hình dạng cụ thể,… Một vài đại diện khiến cho việc học các nhiệm vụ dễ dàng hơn. Phương pháp này đã cải thiện đáng kể công nghệ tiên tiến trong nhận dạng giọng nói, nhận dạng đối tượng trực quan, phát hiện đối tượng và nhiều lĩnh vực khác như khám phá thuốc và bộ gen. Deep learning phát hiện ra cấu trúc phức tạp trong các tập dữ liệu lớn bằng cách sử dụng thuật toán backpropagation (lan truyền ngược) để chỉ ra cách một máy nên thay đổi cấc tham số bên trong được sử dụng để tính toán biểu diễn trong mỗi lớp từ biểu diễn trong lớp trước. Deep Learning đã mang lại những đột phá trong việc xử lý hình ảnh, video, nhận dạng giọng nói và âm thanh. 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.

Deep Learning có rất nhiều thuật toán như Convolutional Neural Network (CNN), Deep Belief Network (DBN), Deep Neural Netwwork (DNN), Recurrent Neural Network (RNN), Boltzman Machine (BM) và Autoencoder (AE).

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.

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 vì vậy ta có cấu trúc Stacked Denoise Autoencoder (SDAE). Việc 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 dữ 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.

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.

Phát hiện xâm nhập là quá trình theo dõi các sự kiện xảy ra trong một hệ thống máy tính hoặc mạng máy tính và phân tích chúng để tìm ra các dấu hiệu sự cố 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 dựa trên tiêu chuẩn bảo mật.

Hệ thống IDS là một hệ thống (có thể là thiết bị phần cứng hay phần mềm) nhằm giám sát lưu lượng mạng theo dõi, thu thập thông tin để phát hiện xâm nhập mạng và đưa ra cảnh báo.

Hiện nay, nhiều nghiên cứu đã áp dụng thành công các thuật toán học máy để hệ thống IDS có khả năng tự học và cập nhật các cuộc tấn công mới. Nhưng để hạn chế báo động nhầm và tăng khả năng dự đoán các cuộc tấn công thì ngoài khả năng tự quyết định, IDS cần phải có tư duy phân tích. Vì vậy ta cần phải ứng dụng học máy vào IDS. 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.

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 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. Quá trình này 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 (Stacked Denoise Autoencoder) 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.

3. Mục tiêu nghiên cứu của đề tài

Nghiên cứu về mạng học sâu AE (Autoencoder) và áp dụng vào khâu tiền xử lý dữ liệu trong Hệ thống phát hiện xâm nhập mạng (IDS) để xác định tấn công xâm nhập, góp phần tăng mức độ hiệu quả, chính xác trong hoạt động của hệ thống IDS.

4. Đối tượng và phạm vi nghiên cứu của đề tài

Đối tượng nghiên cứu của luận văn là mạng học sâu AE (Autoencoder) và Hệ thống IDS.

Phạm vi nghiên cứu của luận văn là áp dụng AE để xác định một dữ liệu đầu

vào có phải là tấn công xâm nhập hay không.

5. Phương pháp nghiên cứu của đề tài

- Về mặt lý thuyết:

+ Thu thập, khảo sát, phân tích, nghiên cứu các tài liệu và thông tin có liên quan đến một số hình thức tấn công xâm nhập mạng và một số phương pháp phát hiện xâm nhập mạng.

+ Tìm hiểu về mạng học sâu AE (Autoencoder) và áp dụng AE vào vào khâu tiền xử lý dữ liệu trong hệ thống phát hiện xâm nhập mạng. Sau khâu tiền xử lý dựa trên AE, sử dụng một số thuật toán SVM, RF, KNN, NB, DT để xác định một số hành vi là tấn công xâm nhập mạng hay không.

- Về mặt thực nghiệm:

+ Sử dụng các thuật toán SVM, RF, KNN, NB, DT trong mạng học sâu AE (Autoencoder) trên bộ dữ liệu NSL-KDD để xác định xâm nhập mạng.

+ Đánh giá, so sánh mức độ hiệu quả khi ứng dụng AE vào hệ thống phát hiện xâm nhập mạng.

CHƯƠNG 1
TỔNG QUAN VỀ PHÁT HIỆN XÂM NHẬP MẠNG

1.1 Khái quát về tấn công xâm nhập mạng

Tấn công, xâm nhập là một, hoặc một chuỗi các hành động vi phạm các chính sách an ninh, an toàn của tổ chức, cơ quan, gây tổn hại đến các thuộc tính bí mật, toàn vẹn và sẵn sàng của thông tin, hệ thống và mạng [1]. Một cuộc tấn công vào hệ thống máy tính hoặc các tài nguyên mạng thường được thực hiện bằng cách khai thác các lỗ hổng bảo mật tồn tại trong hệ thống. Trong thế giới kết nối mạng sâu và rộng hiện nay, hầu hết các dạng tấn công, xâm nhập đều được thực hiện thông qua hệ thống mạng, nhất là mạng Internet, nên có thể xem tấn công, xâm nhập mạng đồng nhất với tấn công, xâm nhập nói chung.

Có thể chia các dạng tấn công, xâm nhập theo mục đích thực hiện thành 4 loại chính như sau [1]:

- Giả mạo (Fabrication) là dạng tấn công thực hiện việc giả mạo thông tin (email, địa chỉ IP…) và thường được sử dụng để đánh lừa người dùng thông thường;
- Chặn bắt (Interception) là dạng tấn công thường liên quan đến việc nghe lén thông tin trên đường truyền và chuyển hướng thông tin để sử dụng trái phép;
- Gây ngắt quãng (Interruption) dạng tấn công làm ngắt, hoặc chậm kênh truyền thông, hoặc làm quá tải hệ thống, ngăn cản việc truy cập dịch vụ của người dùng hợp pháp;
- Sửa đổi (Modification) dạng tấn công thực hiện việc sửa đổi thông tin trên đường truyền hoặc sửa đổi dữ liệu file.

Theo hình thức thực hiện, có thể chia các dạng tấn công, xâm nhập thành 2 kiểu chính như sau:

- Tấn công chủ động (Active attack) là một đột nhập, xâm nhập về mặt vật lý vào hệ thống, hoặc mạng. Các tấn công chủ động thực hiện sửa đổi dữ liệu trên đường truyền, sửa đổi dữ liệu trong file, hoặc giành quyền truy cập trái phép vào hệ thống máy tính hoặc hệ thống mạng.
- Tấn công thụ động (Passive attack) là kiểu tấn công thường không gây ra thay đổi trên hệ thống. Các tấn công thụ động điển hình là nghe lén và giám sát lưu lượng trên đường truyền.

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. Các mạng không được thiết lập để định tuyến không đối xứng sẽ không bị ảnh hưởng bởi phương pháp tấn công này.

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. Trong hầu hết các trường hợp, mục đích là bắt đầu tình huống từ chối dịch vụ (DoS) hoặc thiết lập một kênh mà qua đó kẻ tấn công có thể truy cập từ xa vào mạng. Việc thực hiện các cuộc tấn công như vậy khó hơn khi các nhà thiết kế mạng giữ kích thước bộ đệm tương đối nhỏ và / hoặc cài đặt logic kiểm tra ranh giới xác định mã thực thi hoặc độ dài chuỗi URL trước khi cho phép ghi dữ liệu vào bộ đệm.

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. Khi hệ thống không xác minh đầu vào hoặc kiểm tra các ký tự, tập lệnh CGI bí mật có thể dễ dàng thêm nhãn thư mục ".." hoặc dấu "|" cho bất kỳ tên đường dẫn tệp nào và do đó truy cập các tệp không được phép qua ứng dụng Web.

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". Các cuộc tấn công theo giao thức cụ thể có thể dễ dàng xâm nhập hoặc thậm chí làm hỏng các thiết bị được nhắm mục tiêu trên mạng.

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. Trojan có thể được đưa vào mạng từ các kho lưu trữ tệp trực tuyến không bị nghi ngờ, đặc biệt nhất là các hệ thống trao đổi tệp ngang hàng.

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. Một trong các dạng sâu phổ biến nhất là sâu mạng (network worm) sử dụng kết nối mạng để lây lan từ máy này sang máy khác. Worms thường lây lan qua tệp đính kèm email hoặc giao thức Trò chuyện chuyển tiếp Internet (IRC). Worms khi không bị phát tiêu thụ nhiều tài nguyên mạng, chẳng hạn như CPU hoặc băng thông mạng, các hoạt động được ủy quyền bị thực thi quá mức. Một số Worm chủ động tìm kiếm thông tin bí mật — chẳng hạn như các tệp có chứa từ khóa có giá trị như "tài chính" hoặc "SSN" và truyền đạt dữ liệu đó cho những kẻ tấn công đang chờ đợi bên ngoài mạng.

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

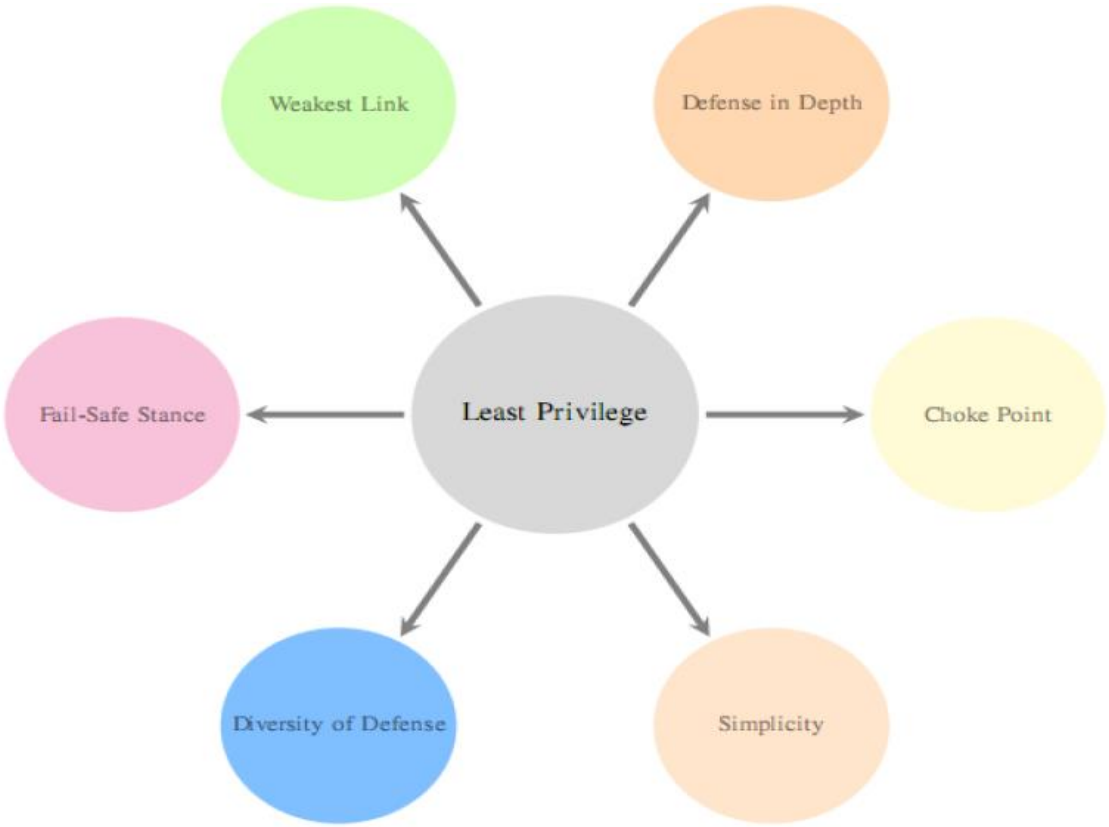
1.3.1. Chiến lược an toàn hệ thống

Xây dựng chiến lược an toàn hệ thống là một bài toán lớn, cần những chuyên gia về an toàn thông tin (ATTT), an ninh mạng (ANM) có khả năng tổ chức, quản lý ATTT, quản lý rủi ro có thể sảy ra đối với hệ thống tại một số thời điểm.

1.3.2 Tính logic, khoa học, an toàn ở mức cao

Đây là những yếu tố hết sức cần thiết đối với hệ thống thông tin hiện nay. Một hệ thống phải đảm bảo tính logic, khoa học và có khả năng bảo vệ tốt. Tính logic, khoa học ngoài việc bảo đảm dịch vụ hệ thống chạy tốt còn giúp rút ngắn thời gian phân tích, khắc phục sự cố an toàn thông tin.

Để đảm bảo cho khả năng bảo vệ tốt, ngoài việc đầu tư các trang thiết bị, phần mềm bảo đảm an toàn thông tin, chúng ta rất cần một lực lượng giám sát ATTT 24/7 để có thể phân tích, xử lý kịp thời các dấu hiệu, nguy cơ mất ATTT mà các trang thiết bị bảo đảm ATTT không thể tự động xử lý được.



Hình 1.1. Một số chiến lược ATTT

1.3.3 Quyền tối thiểu (Least Privilege)

Trong bảo đảm ATTT, đây là nguyên tắc cơ bản nhất. Trong nguyên tắc này, bất kỳ đối tượng (người dùng, admin, thiết bị, ứng dụng, …) trong hệ thống chỉ được cấp những quyền để đối tượng này thực hiện các hoạt động trong phạm vi cho phép.

1.3.4 Phòng thủ theo chiều sâu (Defense in Depth)

Trong chiến lược này, các cơ chế và kiểm soát bảo mật được phân lớp cẩn thận trong toàn bộ mạng máy tính để bảo vệ tính bí mật, tính toàn vẹn và tính khả dụng của mạng và dữ liệu bên trong. Kẻ tấn công có thể vượt qua một vài cơ chế hoặc lớp bảo mật nhưng chưa chắc có thể tấn công thành công vào hệ thống.

1.3.5 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, điều khiển, tăng cường các hình thức giám sát, bảo đảm ATTT nâng cao. Phương pháp này học hỏi từ một số hệ thống thực tế, ví dụ trạm thu phí, máy an ninh tại siêu thị, máy đo thân nhiệt tại Sân bay, quầy bán vé tại rạp chiếu phim.

1.3.6 Liên kết yếu nhất (Weakest Link)

Mỗi một hệ thống thông tin luôn có những điểm yếu. Không chuyên gia nào dám khẳng định một hệ thống thông tin nào dó hoàn hảo về bảo mật, không thể bị tấn

công. Thông thường, kẻ tấn công thường tìm điểm yếu nhất của hệ thống để tấn công, do vậy ta cần phải liên tục gia cố, tăng cường bảo mật cho các yếu điểm hệ thống. Trong nhiều hệ thống thông tin, con người là điểm yếu nhất, dễ bị khai thác qua đó hacker có thể tấn công chiếm quyền điều khiển hoặc phá hoại hệ thống.

1.3.7 Lập trường thất bại an toàn (Fail-Safe Stance)

Trong một số trường hợp đặc biệt, hệ thống thông tin nên có cơ chế 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. Đây là một hình thức đánh đổi chấp nhận được.

1.3.8 Phòng thủ đa dạng (Diversity of Defense)

Ý tưởng đằng sau sự đa dạng của hệ thống phòng thủ là việc sử dụng các hệ thống bảo mật từ các nhà cung cấp khác nhau có thể làm giảm nguy cơ xảy ra lỗi, lỗ hổng phổ biến hoặc lỗi cấu hình ảnh hưởng đến cả hệ thống. Tuy nhiên, cần có một sự cân bằng về độ phức tạp và chi phí. Việc mua sắm và lắp đặt nhiều hệ thống khác nhau sẽ khó khăn hơn, mất nhiều thời gian hơn và tốn kém hơn so với việc mua và lắp đặt đồng bộ một hệ thống duy nhất từ một hãng sản xuất.

1.3.9 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. Khi một hệ thống thông tin rắc rối, không rõ ràng về logic, chúng ta không hiểu được nó vận hành thế nào, luồng dữ liệu đi như thế nào, chúng ta không thể đánh giá được nó có an toàn hay không. Thứ hai, các chương trình, hệ thống càng phức tạp thì nguy cơ lỗi, nguy cơ tồn tại lỗ hổng bảo mật càng cao.

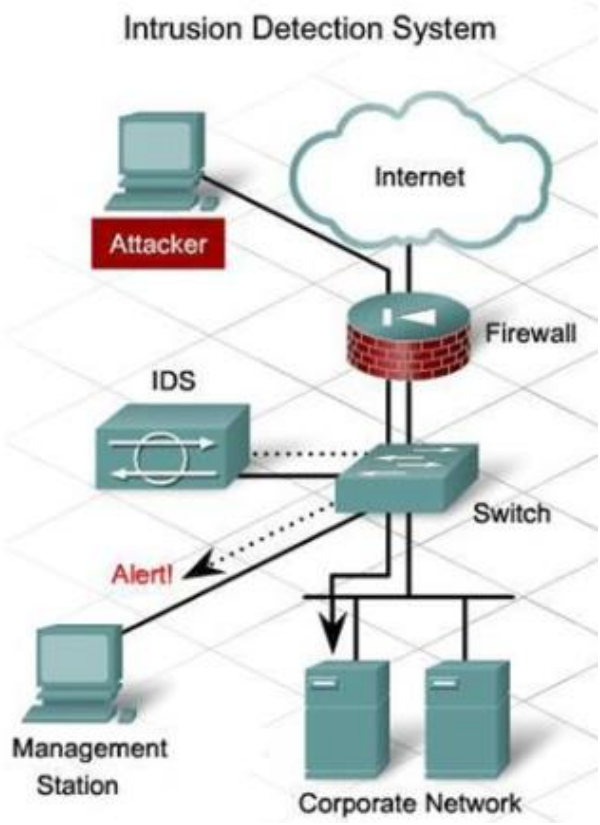
1.4 Khái quát về phát hiện xâm nhập mạng

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 [2].

Hệ thống phát hiện tấn công, xâm nhập (Intrusion Detection System - IDS) là một lớp bảo vệ quan trọng trong các lớp đảm bảo ATTT cho hệ thống thông tin và mạng theo mô hình phòng thủ có chiều sâu (defence in depth). IDS là hệ thống phát hiện tấn công, xâm nhập. Các hệ thống IDS có thể được đặt trước hoặc sau tường lửa trong mô hình

mạng, tùy theo mục đích sử dụng. Hình 1.7 cung cấp vị trí các hệ thống IDS và IPS trong sơ đồ mạng, trong đó IDS thường được kết nối vào bộ switch phía sau tường lửa.



Hình 1.2 Sơ đồ vị trí IDS trong mạng [1, 129]

Nhiệm vụ chính của các hệ thống IDS bao gồm [2]:

- Giám sát lưu lượng mạng hoặc các hành vi trên một hệ thống để nhận dạng các dấu hiệu của tấn công, xâm nhập;

- Khi phát hiện các hành vi tấn công, xâm nhập, thì ghi logs các hành vi này cho phân tích bổ sung sau này;

- Gửi thông báo cho người quản trị về các các hành vi tấn công, xâm nhập đã phát hiện được.

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.

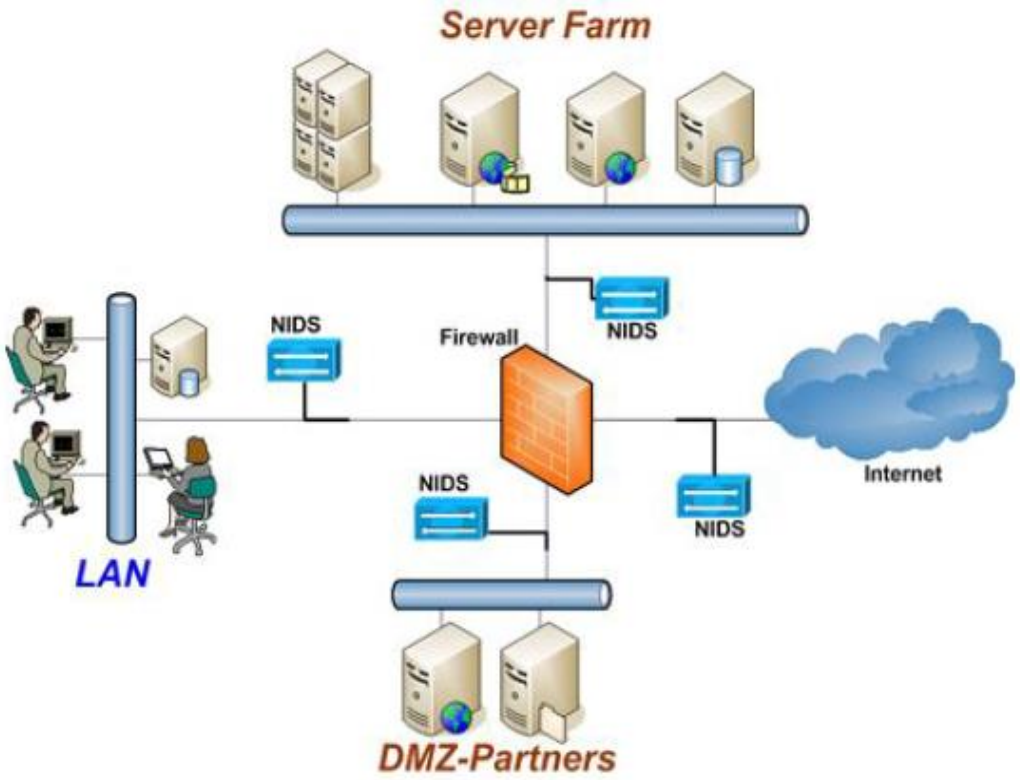
Đối với phân loại theo nguồn dữ liệu, có 02 loại hệ thống IDS.

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.

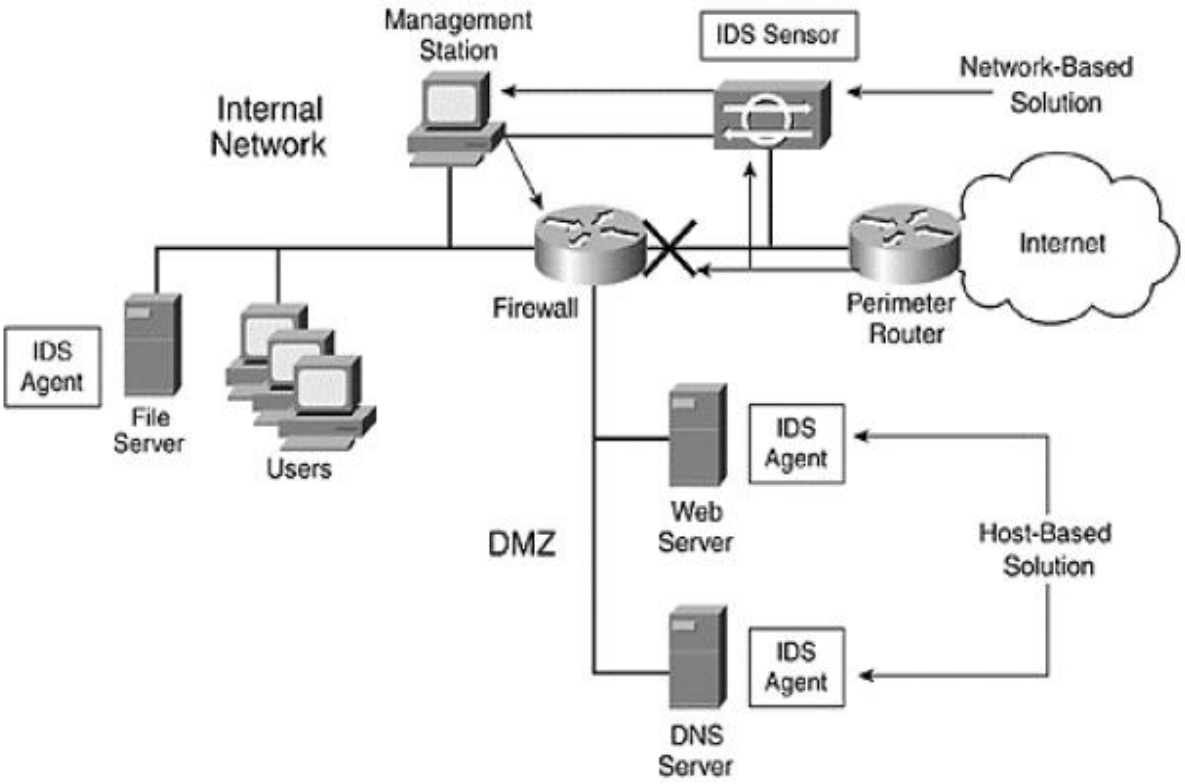


Hình 1.3. Các NIDS được bố trí để giám sát phát hiện xâm nhập tại cổng vào mạng và cho từng phân đoạn mạng [1, 130]

1.4.2.2 Hệ 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, vì vậy nó cung cấp một cơ chế phân tích toàn diện các hoạt động và phát hiện một cách chính xác các thành phần tấn công.

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.



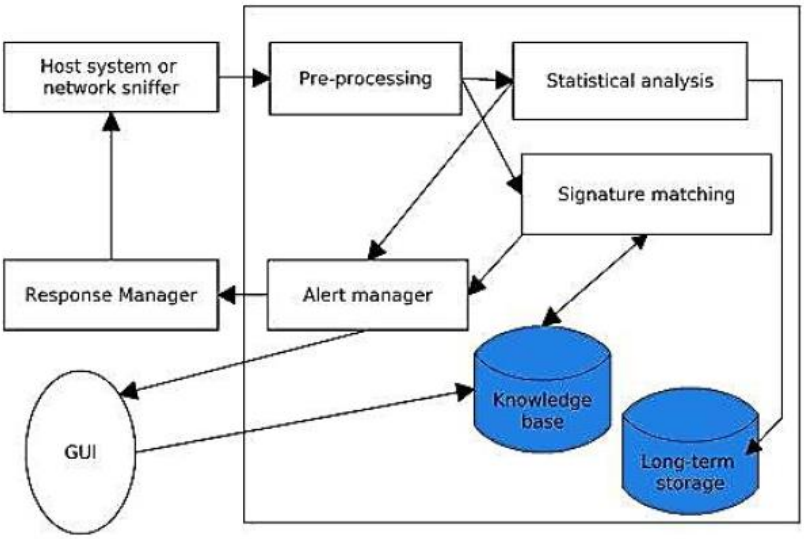
Hình 1.4. Sử dụng kết hợp NIDS và HIDS để giám sát lưu lượng mạng và các host [1, 131]

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.

Trong kỹ thuật này cần xây dựng cơ sở dữ liệu các chữ ký, hoặc các dấu hiệu của các loại tấn công, xâm nhập đã biết. Bước tiếp theo là sử dụng cơ sở dữ liệu các chữ ký để giám sát các hành vi của hệ thống, hoặc mạng, và cảnh báo nếu phát hiện chữ ký của tấn công, xâm nhập.



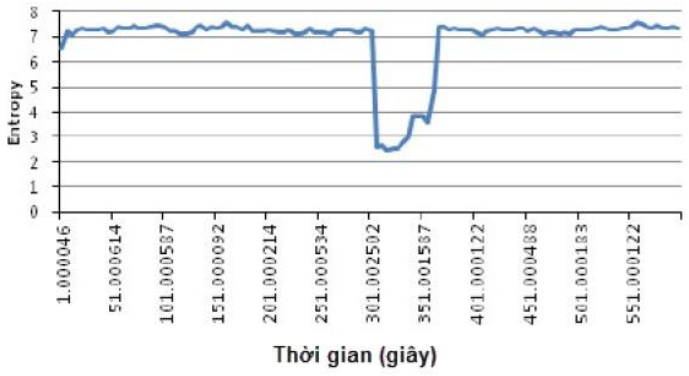
Hình 1.5. Giám sát phát hiện xâm nhập dựa trên chữ ký [1, 131]

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. Cần thường xuyên cập nhật các đặc trưng (dấu hiệu) về các cuộc tấn công mới và thời gian phát hiện tăng khi cơ sở dữ liệu lớn.

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 giai 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.



Hình 1.6. Giá trị entropy của IP nguồn của các gói tin từ lưu lượng hợp pháp (phần giá trị cao, đều) và entropy của IP nguồn của các gói tin từ lưu lượng tấn công DDoS (phần giá trị thấp)

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.

1.5 Kết luận chương

Chương 1 đã giới thiệu tổng quan về xâm nhập mạng và một số dạng tấn công xâm nhập mạng điển hình. Ngoài ra, chương này cũng giới thiệu khái quát về phát hiện xâm nhập mạng, phân loại phương pháp phát hiện xâm nhập mạng và một số giải pháp phát hiện, phòng chống tấn công xâm nhập mạng cơ bản.

Trong Chương 2, với nội dung là **PHÁT HIỆN XÂM NHẬP DỰA TRÊN HỌC SÂU**, luận văn sẽ tiếp tục đi tìm hiểu khái quát về học máy máy và học sâu, đồng thời nghiên cứu xây dựng mô hình ứng dụng Autoencoder trong phát hiện xâm nhập mạng.

CHƯƠNG 2
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 [6]. Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác (spam) hay không và tự động xếp thư vào thư mục tương ứng. Trong học máy, có nhiều thuật toán được áp dụng cho việc "học" của hệ thống như: Linear Regression (Hồi quy tuyến tính), Logistic Regression (Hồi quy logistic), Linear Discriminant Analysis (Phân tích phân loại tuyến tính), DT (Decision Tree- Cây quyết định), NB (Naive Bayes), KNN (K-Nearest Neighbors), Học Vector Quantization, SVM (Support Vector Machine), RF (Random Forest) và mạng nơ ron nhân tạo. Mỗi thuật toán có một vài điểm mạnh, điểm yếu riêng và phù hợp với một số lớp bài toán.

Mạng nơ ron nhân tạo (Artificial Neural Network – ANN) là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thần kinh, bao gồm số lượng lớn các neural được gắn kết để xử lý thông tin [5]. ANN giống như bộ não con người, được học bởi kinh nghiệm (thông qua huấn luyện), có khả năng lưu trữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).

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. Một quan sát (như hình ảnh) có thể được biểu diễn bằng nhiều cách như một vector của các giá trị cường độ cho mỗi điểm ảnh hoặc trừu tượng hơn như là tập hợp các cạnh, các khu vực hình dạng cụ thể,… Một vài đại diện khiến cho việc học các nhiệm vụ dễ dàng hơn. Phương pháp này đã cải thiện đáng kể công nghệ tiên tiến trong nhận dạng giọng nói, nhận dạng đối tượng trực quan, phát hiện đối tượng và nhiều lĩnh vực khác như khám phá thuốc và bộ gen. Deep learning phát hiện ra cấu trúc phức tạp trong các tập dữ liệu lớn bằng cách sử dụng thuật toán backpropagation (lan truyền ngược) để chỉ ra cách một máy nên thay đổi các tham số bên trong được sử dụng để tính toán biểu diễn trong mỗi lớp từ biểu diễn trong lớp trước. Deep Learning đã mang lại những đột phá trong việc xử lý hình ảnh, video, nhận dạng giọng nói và âm thanh. 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 [5].

Trong Deep Learning, có 03 dạng học chính là học có giám sát, học nửa giám sát và học không giám sát.

Deep Learning có rất nhiều thuật toán như Convolutional Neural Network (CNN), Deep Belief Network (DBN), Deep Neural Netwwork (DNN), Recurrent Neural Network (RNN), Boltzman Machine (BM) và Autoencoder (AE).

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

2.2.1 Học sâu sử dụng Autoencoder

2.2.1.1 Mô tả

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 (tầng ẩn) 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.

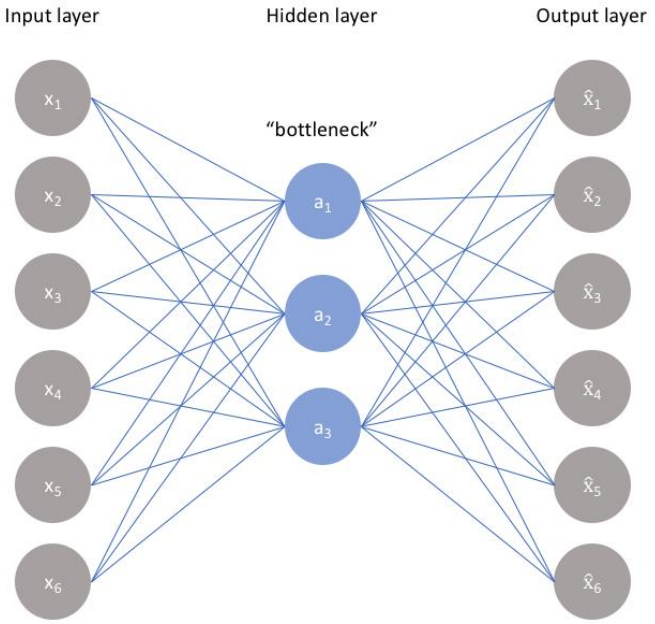
Cấu trúc Autoencoder được chia thành bộ mã hóa và bộ giải mã, bao gồm cả tầng đầu vào (input layer), tầng ẩn (hidden layer), tầng đầu ra (output layer). Autoencoder có một hoặc nhiều tầng ẩn có chức năng mã hóa để tạo ra dữ liệu có chứa các thuộc tính cơ bản nhất có thể mô tả đầy đủ dữ liệu đầu vào. 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 tầng ẩn 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, nén dữ liệu. 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:

$$h(W, b(x)) \approx x \qquad (1)[11, 179]$$

Trong đó W là trọng số, b là bias. Quá trình học được mô tả như là một hàm giảm thiểu lỗi tái thiết lại dữ liệu.

Cụ thể, 02 phần bộ mã hóa và giải mã, có thể được định nghĩa như sau:

$$\begin{aligned} &\phi : \mathcal{X} \to \mathcal{F} \\ &\psi : \mathcal{F} \to \mathcal{X} \\ &\phi, \psi = \underset{\phi,\psi}{\arg\min} \|X - (\psi \circ \phi)X\|^2 \end{aligned} \qquad (2)\ [11, 179]$$



Hình 2.1 Sơ đồ cấu trúc mạng Autoencoder [10,4]

Trường hợp đơn giản nhất, khi có một tầng ẩn, tầng encoder của autoencoder lấy input $x \in R^d = X$ và ánh xạ tới

$$z \in R^p = F: z = \delta(Wx + b) \quad (3)$$

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 vecto sai lệch. Sau đó, giai đoạn giải mã của autoencoder là lấy z để tái thiết x' giống x:

$$x' = \delta'(W'z + b') \quad (4)$$

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.

Autoencoder được đào tạo để giảm thiểu lỗi tái cấu trúc (*như lỗi bình phương*), hay thường được gọi là loss function (*hàm mất mát - một phương pháp đánh giá thuật toán mô hình hóa dữ liệu. Nếu các dự đoán sai lệch quá nhiều so với kết quả thực tế, loss function sẽ có giá trị rất lớn, hàm mất mát giúp giảm sai số trong dự đoán*):

$$||x - x'||^2 = ||x - \delta'\big(W'\big(\delta(Wx + b)\big) + b'\big)||^2 \quad (5)[10, 4]$$

Mô hình mạng nơ-ron nhân tạo nói chung và mạng Autoencoder nói riêng cho phép liên kết có trọng số các phần tử phi tuyến *(các nơ-ron đơn lẻ)* tạo nên dạng hàm tổng hợp từ các hàm thành phần. Do vậy, sau một quá trình điều chỉnh sự liên kết *(trọng số)* cho phù hợp *(tức quá trình học)*, các phần tử phi tuyến đó sẽ tạo nên một

hàm phi tuyến phức tạp có khả năng xấp xỉ hàm biểu diễn quá trình cần nghiên cứu (*hàm mục tiêu*). Kết quả là đầu ra của nó sẽ tương tự với đầu vào của tập dữ liệu dùng để luyện mạng. Khi đó, ta nói mạng Autoencoder đã học được mối quan hệ tương quan đầu vào - đầu ra của quá trình học hay mạng Autoencoder đã học được các đặc trưng nhất của dữ liệu và lưu lại mối quan hệ tương quan này thông qua bộ trọng số liên kết giữa các nơ-ron. Do đó, 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.

2.2.1.2 Xác định thuộc tính và siêu tham số của mạng

Trước khi xây dựng một mạng để huấn luyện và lấy các đặc trưng của dữ liệu ta cần phải xác định được các thuộc tính cần thiết cũng như các siêu tham số của mạng. Đây là các yếu tố cần xác định trước để có thể xây dựng được mạng và huấn luyện mạng một cách tốt nhất.

a) Thuộc tính

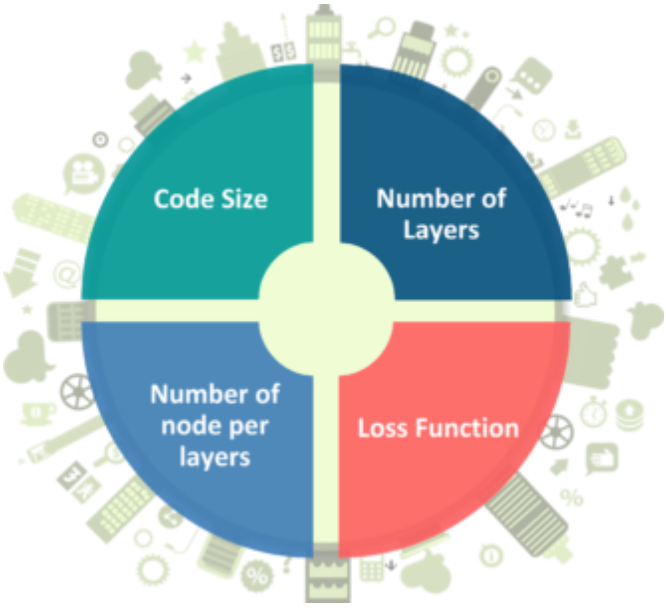
- Dữ liệu cụ thể (Data-Specific): Autoencoder chỉ có thể nén dữ liệu theo những gì mà nó được đào tạo. Vì trong quá trình huấn luyện chúng đã học các đặc trưng của dữ liệu đã được dùng trong quá trình đó, chúng khác hoàn toàn so với thuật toán nén dữ liệu tiêu chuẩn như gzip. Vì vậy chúng ta không thể mong đợi một mạng autoencoder được đào tạo bằng dữ liệu về chữ số viết tay có thể nén dữ liệu về hình ảnh và decode ra được hình ảnh tương tự.

- Loss: autoencoder có một tầng ẩn **h** dùng để mô tả một mã (code) được dùng để đại diện cho dữ liệu đầu vào. Mạng được xem như bao gồm hai phần một là encoder function: h = f(x) và một là decoder tạo ra một bản tái cấu trúc đầu vào r = g(h). Nếu một autoencoder thành công chỉ đơn giản là học cách thiết lập: g(f(x)) = x. Nhưng thay vào đó, autoencoder được thiết kế để không học cách copy hoàn hảo. Thông thường việc học bị hạn chế, nó chỉ cho phép học copy gần đúng và chỉ copy đầu vào gần giống dữ liệu đầu vào. Bởi vì mô hình buộc phải ưu tiên những khía cạnh của đầu vào nên copy, nó thường học các thuộc tính hữu ích của dữ liệu.

- Không giám sát (*Unsupervised*): để đào tạo một mạng autoencoder chúng ta không cần làm bất cứ điều gì đặc biệt chỉ cần đưa dữ liệu đầu vào. Autoencoder được coi là một kỹ thuật học không giám sát vì chúng không cần nhãn (*label*) đánh dấu để đào tạo. Tuy nhiên để chính xác hơn, chúng ta tự giám sát (*supervised*) bằng cách tự tạo nhãn riêng từ dữ liệu đào tạo.

b) Siêu tham số (*Hyperparameter*)

Hyperparameter là những thông số cần phải thiết lập trước khi xây dựng Autoencoder và DAE như số tầng ẩn, số nút, chọn công thức loss function sử dụng, kích thước mã (code size), learning rate (*tốc độ học*), … Sau khi đã xây dựng mạng và huấn luyện, ta cần dựa vào kết quả huấn luyện để thay đổi các thông số làm sao cho ra kết quả tốt nhất. Ngoài ra, ta có thể sử dụng một số kỹ thuật để ước lượng được một khoảng giá trị tốt nhất.



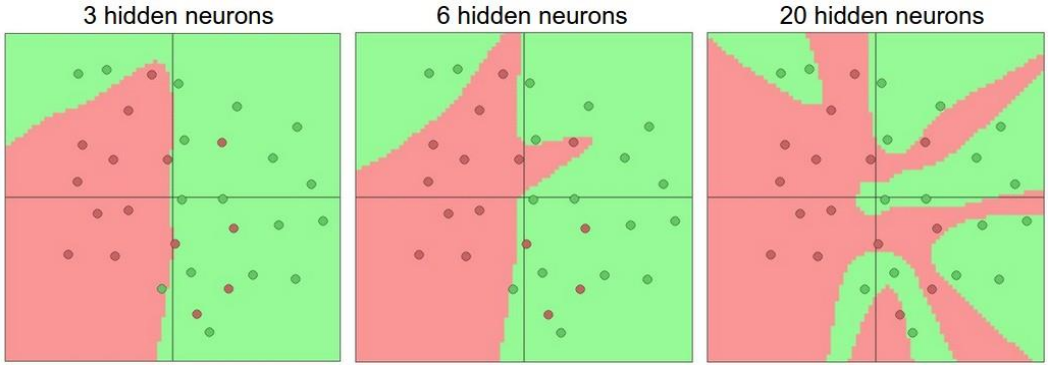
Hình 2.2 Thành phần của Hyperparameter

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

- Code size: Số lượng nút trong tầng ẩn giữa. Kích thước tầng ẩn càng nhỏ thì dữ liệu càng nén nhiều.

- Số lớp: Ta có thể thấy nerural network có nhiều tầng ẩn hơn có thể biểu diễn các function phức tạp hơn. Tuy nhiên, đây là điều tốt vì chúng ta có thể học cách phân loại dữ liệu phức tạp hơn và nhược điểm là nó quá phức tạp để huấn luyện. Ví dụ: Mô hình với 20 tầng ẩn phù hợp với tất cả các loại dữ liệu huấn luyện nhưng tốn khá nhiều chi phí tài nguyên để phân chia không gian thành nhiều vùng khác nhau. Mô hình với 03 tầng ẩn chỉ có khả năng để phân loại dữ liệu theo các vùng rộng. Mô hình dữ liệu mô tả dưới dạng 02 đốm màu và các điểm màu đỏ bên trong vùng xanh là các ngoại lệ (hình dưới). Trong thực tế, mô hình 03 layer có thể khái quát hóa tốt hơn trên bộ dữ liệu thử nghiệm. Vì vậy, dựa trên những ý kiến trên ta có thể thấy dường như các mạng neural nhỏ có thể được ưa thích hơn nếu dữ liệu không quá phức tạp để giảm thiểu chi phí tài nguyên. Trong thực tế, chúng ta nên sử dụng các phương pháp để kiểm soát chi phí thay vì số lượng lớp neural. Lý do là các mạng neural nhỏ hơn khó huấn luyện hơn

với các phương pháp Gradient Descent *(thuật toán tối ưu hóa được sử dụng để giảm thiểu một số hàm bằng cách di chuyển lặp đi lặp lại theo hướng dốc xuống nhanh nhất được xác định bởi giá trị âm của dốc. Trong học máy, chúng ta sử dụng gradient descent để cập nhật các tham số của mô hình)*. Các loss function tương đối ít cực tiểu, nhưng lại dễ hội tụ hơn và lại cho kết quả loss function cao. Ngược lại, các mạng neural lớn hơn chứa nhiều điểm cực tiểu lại cho kết quả loss function tốt hơn nhiều. Trong thực tế, nếu huấn luyện một mạng nhỏ, kết quả loss function cuối cùng có thể cho một phương sai tốt - trong trường hợp gặp may mắn và hội tụ ở một điểm tốt nhưng trong một số trường hợp ta bị mắc kẹt ở một trong những điểm cực tiểu xấu sẽ cho kết quả loss function lớn. Mặt khác, nếu huấn luyện một mạng neural lớn, ta sẽ có được các giải pháp khác nhau và phương sai mất mát cuối cùng có thể nhỏ hơn nhiều.



Hình 2.3 So sánh mô hình phân bố giữa các cấu trúc sử dụng số tầng ẩn khác nhau

- 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. Ngoài ra, encoder và decoder đối xứng nhau về cấu trúc layer.

2.2.1.3 Phương pháp huấn luyện mạng AE và DAE

a) Phương pháp 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. Đây là 02 loss function được sử dụng phổ biến

- Loss function root-mean-square error: Độ sai lệch được coi như khoảng cách giữa giá trị dự đoán và giá trị thật. Khoảng cách càng xa, độ sai lệch càng cao. Hàm này chính là tổng bình phương của sự khác biệt giữa các biến mục tiêu dự đoán và thực tế chia cho số điểm dữ liệu

$$\text{MSE}= \frac{1}{n}\left(\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2\right) \qquad (6)\ [6, 108]$$

Trong đó: n là số điểm dữ liệu, y_i là giá trị dự đoán thứ i, $\tilde{y}_i$ là giá trị thật thứ i.

Nếu các giá trị đầu vào nằm trong phạm vi [0,1] thì chúng ta thường sử dụng lỗi bình phương trung bình.

- Loss funciton entropy: Dùng để tính độ sai lệch giữa dự đoán dạng xác xuất và giá trị đúng. Ví dụ, khi thuật toán dự đoán 80% output là giá trị 1 và 20% output là giá trị 0 trong khi output đúng là giá trị 1, entropy sẽ được sử dụng để tính độ sai lệch.

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & y = 1 \\ -\log(1 - h_\theta(x)), & y = 0 \end{cases} \quad (7)\ [6, 132]$$

Trong cấu trúc mạng AE và DAE chúng ta sẽ dụng loss function root-mean-square error. Điều chúng ta muốn làm là làm sao huấn luyện mạng để được loss function nhỏ nhất. Điều đó đồng nghĩa với việc tìm vector hệ số w sao cho giá trị của hàm mất mát này càng nhỏ càng tốt [7]. Giá trị của w làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là *điểm tối ưu* (*optimal point*). Để làm được điều này chúng ta cần các thuật toán tối ưu hóa như Gradient Descent, Adam, Adadelta, RMSprop, Adagrad,…

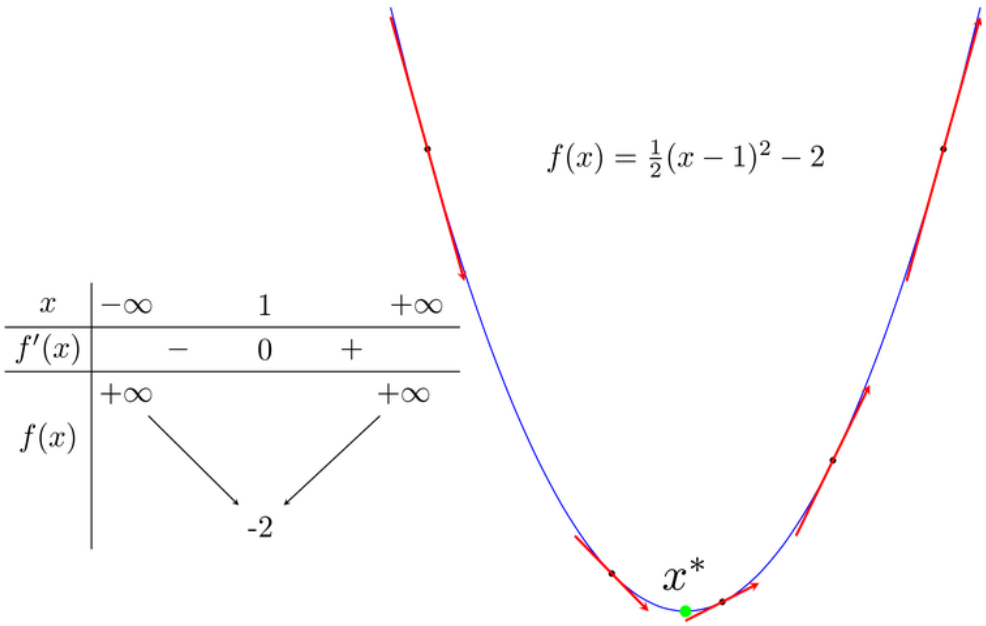
b) Phương pháp Gradient Descent

Việc tìm cực tiểu toàn cục (*global minimum*) của các hàm mất mát là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm cực tiểu cục bộ (*local minimum*), và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Các điểm local minimum là nghiệm của phương trình đạo hàm bằng 0. Nếu bằng một cách nào đó có thể tìm được toàn bộ (*hữu hạn*) các điểm cực tiểu, ta chỉ cần thay từng điểm local minimum đó vào hàm số rồi tìm điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp trong đạo hàm của phương trình, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0 [8]. Gradient Descent (*viết gọn là GD*) và các biến thể của nó là một trong những phương pháp được dùng nhiều nhất.

- Gradient Descent hàm một biến:



Hình 2.4 Đồ thị hàm một biến

Giả sử x_t là điểm ta tìm được sau vòng lặp thứ t. Ta cần tìm một thuật toán để đưa x_t về càng gần x^* càng tốt.

Nếu đạo hàm của hàm số tại x_t: $f'(x_t) > 0$ thì x_t nằm về bên phải so với x^* (*và ngược lại*). Để điểm tiếp theo $x_t + 1$ gần với x^* hơn, chúng ta cần di chuyển x_t về phía bên trái, tức về phía âm. Nói các khác, chúng ta cần di chuyển ngược dấu với đạo hàm:

$$x_t + 1 = x_t + \Delta \quad (8)$$

Trong đó: Δ là một đại lượng ngược dấu với đạo hàm $f'(x_t)x_t$ càng xa x^* về phía bên phải thì f'(xt) càng lớn hơn 0 (và ngược lại).

$$x_{t+1} = x_t - \eta f'(x_t) \quad (9)$$

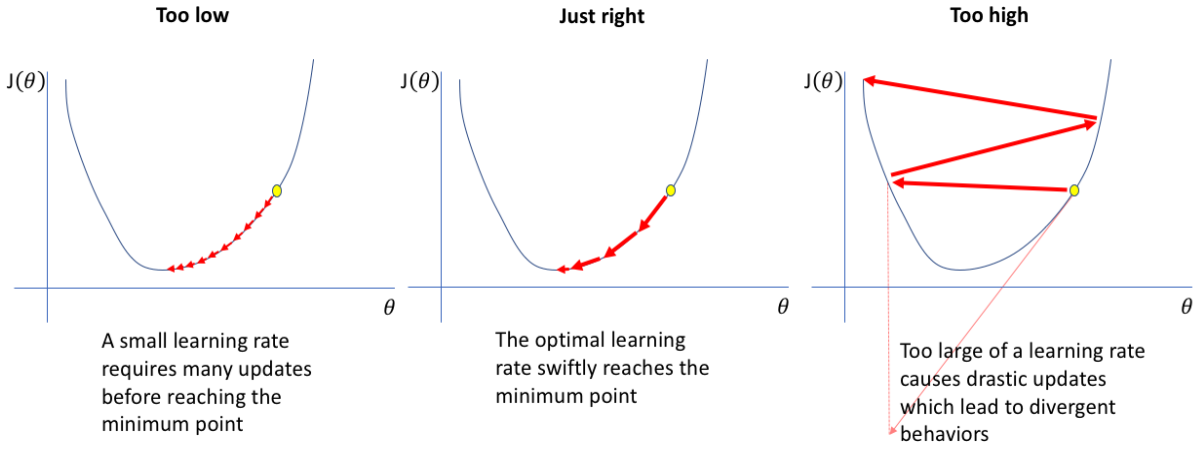
Trong đó: η là một số dương được gọi là *learning rate* (tốc độ học). Dấu trừ thể hiện việc chúng ta phải *đi ngược* với đạo hàm.

- Gradient Descent hàm đa biến: Giả sử ta cần tìm global minimum cho hàm $f(\theta)$ trong đó θ là một vector, thường được dùng để ký hiệu tập hợp các tham số của một mô hình cần tối ưu (*trong mạng neural thì các tham số chính là hệ số w*). Đạo hàm của hàm số đó tại một điểm θ bất kỳ được ký hiệu là $\nabla_\theta f(\theta)$. Tương tự như hàm 01 biến, thuật toán GD cho hàm nhiều biến cũng bắt đầu bằng một điểm dự đoán θ_0, sau đó, ở vòng lặp thứ t, quy tắc cập nhật là:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta_t) \quad (10)$$

- 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. Trong thực tế, khi việc tính toán trở nên phức tạp, *learning rate* quá thấp sẽ ảnh hưởng tới tốc độ của thuật toán rất nhiều, thậm chí không bao giờ tới được đích.

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.



Hình 2.5 Sơ đồ biểu diễn khả năng hội tụ của learning rate khác nhau

Việc lựa chọn learning rate rất quan trọng trong việc huấn luyện mạng AE và DAE. Việc lựa chọn giá trị này phụ thuộc nhiều vào bài toán chúng ta thực hiện và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất. Ngoài ra, tùy vào bài toán của chúng ta, GD có thể làm việc hiệu quả hơn bằng cách chọn ra learning rate phù hợp hoặc chọn learning rate khác nhau ở mỗi vòng lặp.

c) Stochastic Gradient Descent

Trong thuật toán tối ưu GD, có rất nhiều biến thể nhưng ta sẽ chọn thuật toán SGD sử dụng trong việc huấn luyện AE và DAE. Trong thuật toán SGD, tại một thời điểm, ta chỉ tính đạo hàm của loss function dựa trên một điểm dữ liệu x_i rồi cập nhật trọng số mạng dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả.

Mỗi lần duyệt một lượt qua *tất cả* các điểm trên toàn bộ dữ liệu được gọi là một epoch (*giai đoạn mới*). Với GD thông thường, mỗi epoch ứng với 01 lần cập nhật trọng số mạng nơron. Với SGD, mỗi epoch ứng với N lần cập nhật trọng số mạng với N là số điểm dữ liệu. Nhìn từ một phía, việc cập nhật từng điểm có thể làm giảm đi tốc độ thực hiện một epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (*thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt*). Vì vậy, SGD phù hợp với bài toán có lượng cơ sở dữ liệu mạng lớn cần xử lý và yêu cầu mô hình thay đổi liên tục như của chúng ta.

Một điểm cần lưu ý đó là: Sau mỗi epoch, chúng ta cần shuffle (*xáo trộn*) thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD. Về mặt toán học, quy tắc cập nhật của SGD là:

$$w = w - \eta \nabla_w J(w; x_i; y_i)$$ (11) [6, 294-296]

Trong đó J(w,x_i,y_i) là hàm mất mát với 01 cặp điểm dữ liệu (input, label) là (x_i,y_i).

d) Chỉ số AUC (Area Under the ROC Curve): ROC curve (**receiver operating characteristic curve**) – đường cong ROC là một đồ thị thể hiện hiệu suất của quá trình phân loại tại tất cả các ngưỡng phân loại. Đường cong này vẽ 02 tham số:

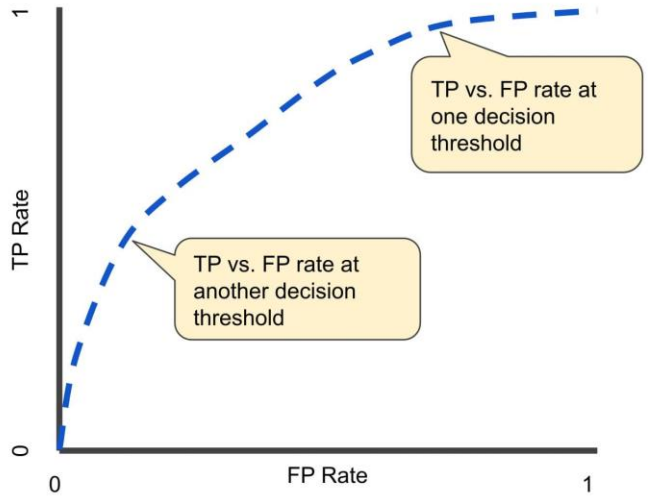
- True Positive Rate (TPR)
- False Positive Rate (FPR)

True Positive Rate được tính theo công thức sau: $TPR = \frac{TP}{TP + FN}$

False Positive Rate (FPR) được tính theo công thức sau: $FPR = \frac{FP}{FP + TN}$

TP là giá trị True Positive (đúng dương), FN là giá trị False Negative (Sai âm), FP là giá trị False Postive (Sai dương), TN là giá trị True Negative (Đúng âm).

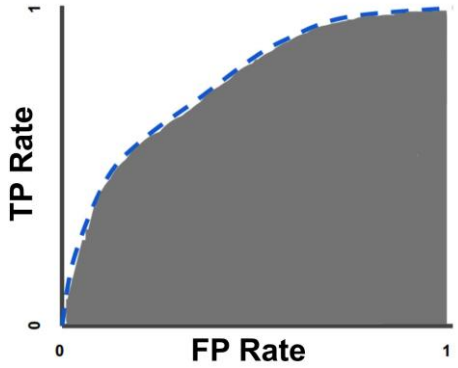
Đường cong ROC vẽ TPR và FPR tại các ngưỡng phân loại khác nhau. Càng giảm ngưỡng phân loại càng nhiều phần tử positive, vì thế tăng cả False Positives và True Positives. Biểu đồ dưới đây thể hiện đường cong ROC cơ bản.



Hình 2.6 Biểu đồ đường cong ROC ở những ngưỡng phân loại khác nhau.

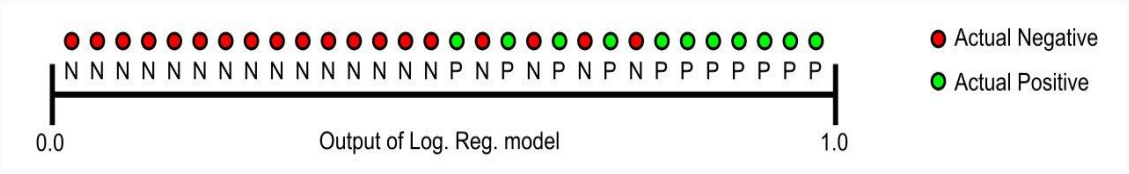
Để tính toán các điểm trên đường cong ROC, chúng ta có thể đánh giá, ước lượng mô hình hồi quy logistic nhiều lần với nhiều ngưỡng phân loại khác nhau nhưng nó sẽ không hiệu quả. Tuy nhiên, có một phương pháp hiệu quả như sau: thuật toán dựa trên sắp xếp gọi là AUC *(Area Under the ROC Curve)* có thể cung cấp thông tin cho chúng ta.

AUC đo diện tích phần dưới của cả đường cong ROC (*tích phân từ (0,0) đến (1,1)*).



Hình 2.7 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. Một cách giải thích AUC là xác suất mà mô hình xếp hạng một ví dụ positive (dương tính) ngẫu nhiên cao hơn một ví dụ negative (tiêu cực) ngẫu nhiên. Ví dụ: đưa ra các ví dụ sau, được sắp xếp từ trái sang phải theo thứ tự tăng dần của các dự đoán hồi quy logistic:



Hình 2.8. Các dự đoán được xếp hạng theo thứ tự tăng dần điểm hồi quy logistic

AUC biểu diễn xác suất 01 mẫu positive ngẫu nhiên (màu xanh) được đặt ở bên phải của 01 mẫu âm tính (*negative*) ngẫu nhiên (màu đỏ).

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

AUC được mong muốn vì 02 lý do sau đây:

- AUC có tỷ lệ không thay đổi. Giá trị này tính toán đo mức độ xếp hạng của các dự đoán thay vì giá trị tuyệt đối của chúng

- AUC có tính bất biến trong phân loại theo ngưỡng. Nó đo lường chất lượng của các dự đoán của mô hình bất kể ngưỡng phân loại nào được chọn.

Tuy nhiên, cả 02 lý do trên đều đi kèm với những lưu ý có thể hạn chế tính hữu ích của AUC trong một số trường hợp sử dụng nhất định

- Bất biến tỷ lệ không phải lúc nào cũng mong muốn. Ví dụ: Đôi khi, chúng tôi thực sự cần đầu ra xác suất được hiệu chỉnh tốt và giá trị AUC sẽ không cho ta biết về điều đó.

- Bất biến ngưỡng phân loại không phải lúc nào cũng mong muốn. Trong trường hợp có sự chênh lệch lớn về chi phí (*cost*) của false negative (âm tính giả) và false positive (dương tính giả), điều quan trọng là phải giảm thiểu một loại lỗi phân loại. Ví dụ: Khi thực hiện phát hiện thư rác email, ta có thể muốn ưu tiên giảm thiểu các false positive (ngay cả khi điều đó dẫn đến sự gia tăng đáng kể các âm tính giả). AUC không phải là một số liệu hữu ích cho loại tối ưu hóa này.

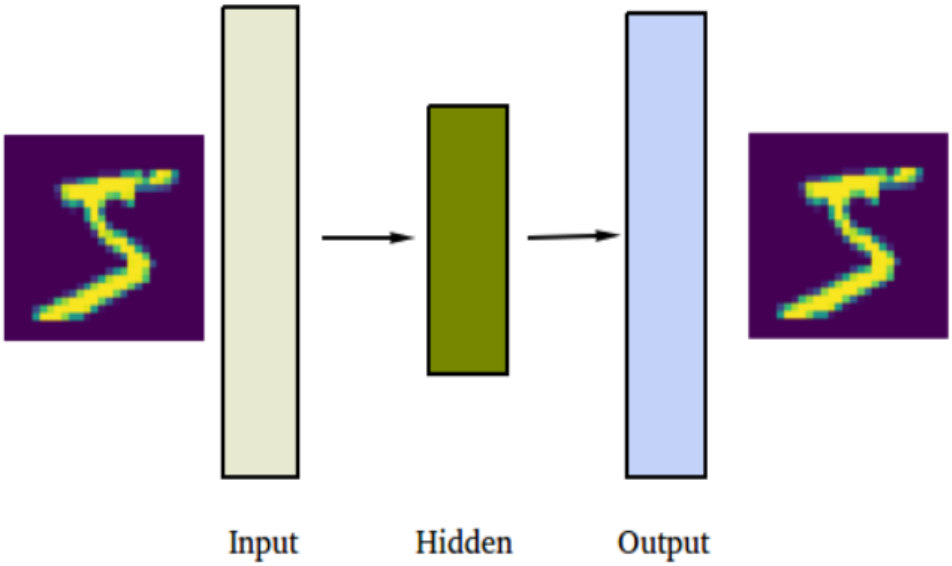
2.2.2 Phân loại Autoencoder

2.2.2.1 Autoencoder chưa hoàn thành (Undercomplete Autoencoder)

Nhiệm vụ của Autoencoder là biến đổi đầu vào thành đầu ra của mạng sao cho giống nhất có thể, nhưng chúng ta thường không quan tâm đến đầu ra của quá trình giải mã (decoder). Thay vào đó, trong quá trình huấn luyện autoencoder để thực hiện nhiệm vụ sao chép đầu vào sẽ dẫn đến các tầng ẩn ta có thể học được các đặc trưng hữu ích của dữ liệu. Một cách để có được các đặc trưng hữu ích từ autoencoder là giảm dần số lượng nút của các tầng ẩn để có kích thước mã (code) nhỏ hơn. 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.

Do Undercomplete Autoencoder có khả năng học các đặc trưng của dữ liệu. Chính vì vậy dữ liệu mã được sinh ra sẽ ít các đặc trưng hơn, giúp quá trình học các thuật toán sau này tốn ít thời gian và độ chính xác cũng cao hơn khi dữ liệu mã biểu diễn tốt dữ liệu đầu vào. Chính vì vậy, tôi đã chọn mạng này để tăng khả năng phân lớp của các thuật toán sau này.

Mục tiêu của quá trình học được mô tả đơn giản là giảm thiểu loss function.



Hình 2.9 Sơ đồ mạng Undercomplete Autoencoder

2.2.2.2 Regularized Autoencoder (Autoencoder đúng quy tắc)

Undercomplted autoencoder với kích thước mã nhỏ hơn kích thước đầu vào, có thể học các đặc trưng nổi bật nhất của dữ liệu đầu vào. Chúng ta có thể thấy rằng autoencoder không được học được những đặc trưng hữu ích nếu encoder và decoder lớn (*kích thước và số lượng lớp trong mạng lớn*).

Một vấn đề tương tự xảy ra nếu tầng ẩn có số lượng nút bằng với số đầu vào, và trong trường hợp overcomplete (*tầng ẩn có kích thước lớn hơn đầu vào*). Trong những trường hợp này, quá trình mã hóa (encoder) tuyến tính và quá trình giải mã (decoder) tuyến tính có thể học cách sao chép đầu vào thành đầu ra mà không cần học bất cứ điều gì về phân phối dữ liệu.

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. Thay vì giới hạn sức chứa mô hình bằng cách giữ encoder và decoder ít layer và kích thước mã nhỏ, regularized autoencoder sử dụng loss function, khuyến khích mô hình có các thuộc tính khác bên cạnh khả năng sao chép đầu vào thành đầu ra. Các tính chất này bao gồm độ thưa của biểu diễn (*sparsity of the representation*), độ nhỏ đạo hàm của biểu diễn (*smallness of the derivative of the representation*), nhiễu hoặc thiếu các đầu vào. 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. Trong thực tế chúng ta có thể tìm thấy 02 loại Regularized Autoencoder: Sparse Autoencoder và Denoising Autoencoder (*DAE*).

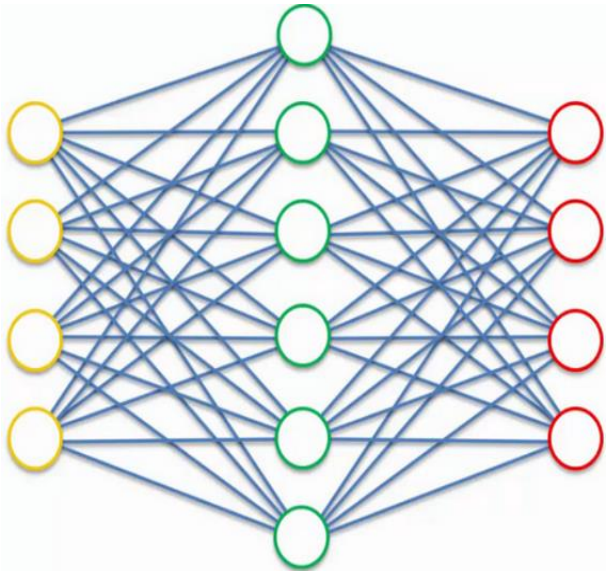
Sparse Autoencoder có số nút ở tầng ẩn lớn hơn nút đầu vào nhưng nó vẫn có thể học được các đặc trưng từ dữ liệu. Nó có nhược điểm là hạn chế sao chép đầu vào thành đầu ra. Sparse Autoencoder có hình phạt thưa thớt (*sparsity penalty*) Ω(h) có giá trị gần bằng 0 nhưng không bằng không. Hình phạt thưa thớt được sử dụng bên cạnh lỗi tái cấu trúc (*reconstruction error*) hay loss function. Việc này giúp tránh được overfitting *(hiện tượng mô hình tìm được* quá *khớp với dữ liệu huấn luyện. Việc* quá khớp *này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa).*

$$L=|x-g(f(x))|+\Omega(h) \qquad (12)\ [6,\ 505\text{-}507]$$

Sparse Autoencoder lấy các giá trị cao nhất trong tầng ẩn và loại bỏ phần còn lại. Điều này ngăn mạng sử dụng tất cả các nút ẩn tại cùng một thời điểm.

Đối với DAE (*làm nhiễu dữ liệu đầu vào trước khi cho vào mạng để huấn luyện*), do mục đích của nghiên cứu này là xây dựng mạng có thể học được các đặc trưng của dữ liệu để tăng khả năng phân loại của các thuật toán, nên DAE sẽ được xây dựng mạng dựa theo cấu trúc mạng của Undercomplete Autoencoder. Sau đó, ta có

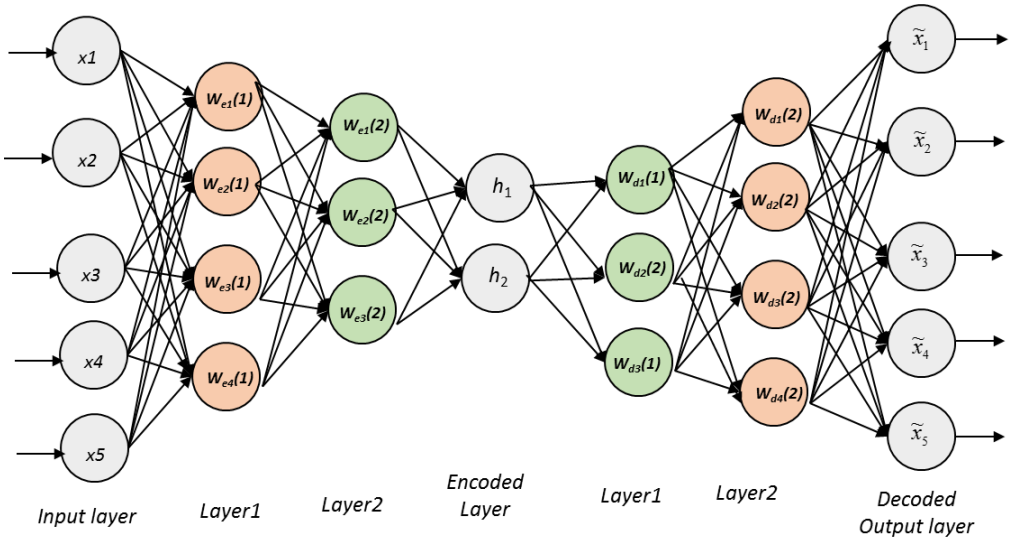
thể so sánh xem việc làm nhiễu dữ liệu đầu vào của DAE có thể học các đặc trưng tốt hơn Undercomplete Autoencoder hay không.



Hình 2.10 Sơ đồ cấu trúc mạng Regularized Autoencoder có số nút tầng ẩn lớn hơn đầu vào

2.2.2.3 Stacked Autoencoder (SAE)

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



Hình 2.11 Sơ đồ cấu trúc mạng Stacked Autoencoder

Stacked Autoencoder (*SAE*) là sự xếp chồng lên nhau của AE vì vậy nó có nhiều tầng ẩn 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 tầng ẩn 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.

Việc xác định số tầng ẩn và nút mỗi lớp được sử dụng để xây dựng mạng SAE còn phụ thuộc vào số đặc trưng của dữ liệu đầu vào và quá trình chạy thử nghiệm cấu trúc mạng. Khi chạy thử nghiệm, nếu loss function chấp nhận được thì ta sẽ chọn cấu trúc mạng như vậy, nếu không ta tiếp tục điều chỉnh số tầng ẩn và số nút mỗi lớp để được kết quả tốt nhất.

2.2.2.4 Denoise Autoencoder (DAE)

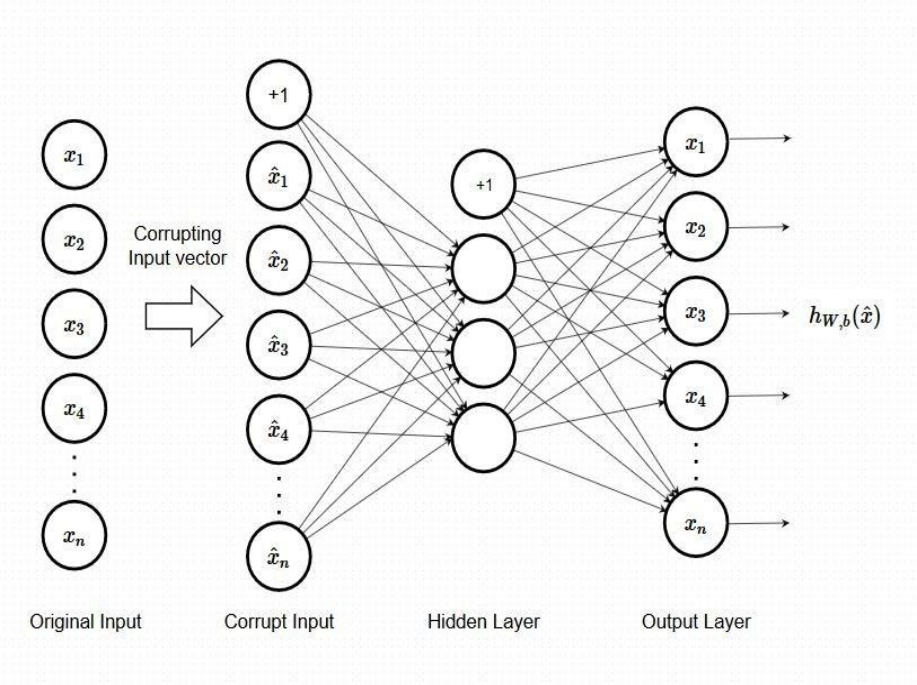
DAE được phát triển từ Autoencoder nhưng mạnh mẽ hơn. Đầu vào của DAE là dữ liệu bị làm nhiễu và chúng ta sẽ học các đặc trưng của dữ liệu từ dữ liệu nhiễu. Nhưng sau quá trình giải mã, đầu ra sẽ là dữ liệu ban đầu trước khi bị làm nhiễu. Từ đó, ta có thể thấy khả năng khái quát hóa cả DAE tốt hơn so với Autoencoder. Hơn nữa, DAE có thể xếp chồng lên nhau để có được feature tốt hơn. Vì vậy, ta có cấu trúc Stacked Denoise Autoencoder (SDAE). Đào tạo mạng SDAE theo layer-wise vì mỗi DAE với một tầng ẩn 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: tầng đầu vào (input layer), tầng ẩn (hidden layer) và tầng đầu ra (output layer), trong đó tầng đầu vào và tầng ẩn là lớp mã hóa (encoder) còn tầng đầu ra và tầng ẩn là lớp giải mã (decoder). Cấu trúc này tương tự như Autoencoder.

Với dữ liệu ban đầu ta có thể tạo ra một dữ liệu nhiễu bằng cách đặt một vài thành phần về 0 hoặc là thêm nhiễu Gaussian (*nhiễu thống kê có hàm mật độ xác suất - probability density function - PDF bằng hàm mật độ xác suất của phân phối chuẩn*) vào để làm dữ liệu đầu vào cho DAE. Khi làm nhiễu Gaussian ta sẽ lấy dữ liệu đầu vào cộng với một nhân tố nhiễu (*noise factor*) nhân với một giá trị ngẫu nhiên (*random*) từ 0 đến 1. Giá trị của nhân tố nhiễu thường rất nhỏ. Sau khi có kết quả, ta sẽ đưa các giá trị về khoảng [0,1].

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:

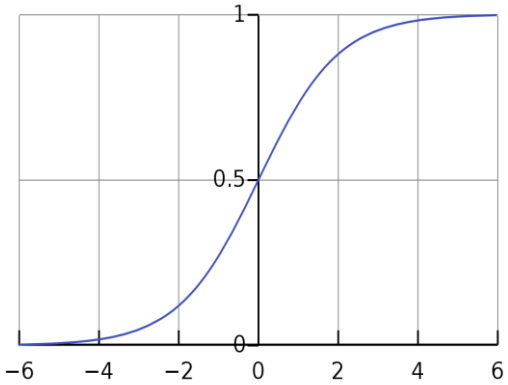
$$z = \delta(W\bar{x} + b) \tag{13}$$



Hình 2.12 Sơ đồ cấu trúc mạng Denoise Autoencoder

Trong đó, z là đầu ra của quá trình encoder và có thể gọi nó là đặc trưng đại diện cho dữ liệu đầu vào hoặc gọi là code. z € R^n, n là số nút của tầng ẩn, W € R^{nxd} là trọng số từ input layer tới tầng ẩn, d là số chiều của dữ liệu, b là bias, (W $\tilde{x}$ + b) là đầu vào của tầng ẩn và δ là hàm kích hoạt của tầng ẩn. Chúng ta có thể chọn Sigmod ($\frac{1}{1+e^{-x}}$), tanh (*là hàm thay đổi tỷ lệ của hàm sigmoid, sao cho đầu ra của nó nằm trong khoảng từ -1 đến 1*), ReLU *(một hàm tuyến tính mảnh sẽ xuất đầu vào (input) trực tiếp nếu nó là dương, nếu không, nó sẽ xuất ra bằng không. Nó đã trở thành chức năng kích hoạt mặc định cho nhiều loại mạng nơ-ron vì nếu một mô hình sử dụng nó sẽ dễ dàng đào tạo hơn và thường đạt được hiệu suất tốt hơn)* là các hàm hay được chọn để sử dụng. Nếu ta chọn ReLU làm hàm kích hoạt trong quá trình học thì ta có:

$$\delta(W\tilde{x} + b) = \max(0, W\tilde{x} + b) \quad (14)$$



Hình 2.13 Đồ thị hàm Sigmoid

Nếu như giá trị của $W\tilde{x} + b$ nhỏ hơn 0, đầu ra của tầng ẩn sẽ là 0. ReLU có thể học các đặc trưng tốt. ReLU có thể đào tạo mạng với dữ liệu lớn nhanh hơn và hiệu quả hơn các hàm kích hoạt khác.

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*):

$$x' = \delta(W'z + b') \quad (15)$$

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. Nếu x thay đổi trong (0,1) thì ta chọn softplus function (*$ln(1+e^x)$ – là hàm có đạo hàm đúng bằng hàm sigmoid*) làm hàm decoding. 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.

2.2.2.5 Stacked Denoise Autoencoder

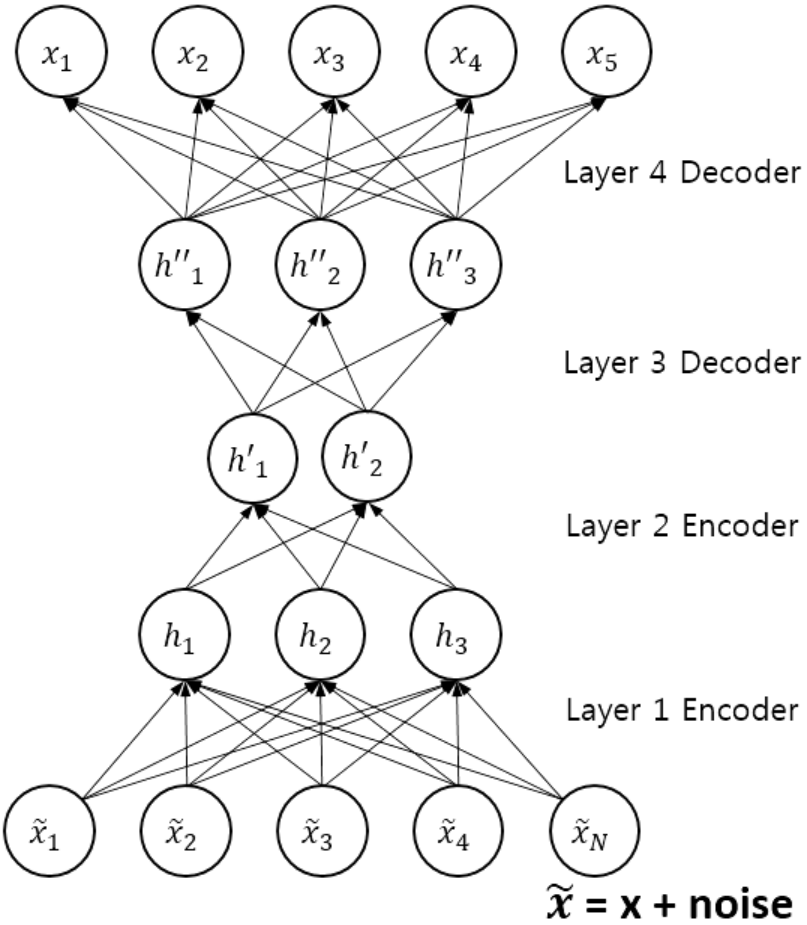
DAE có thể xếp chồng lên nhau để xây dựng deep network (mạng học sâu) có nhiều hơn 01 tầng ẩn. SDAE bao gồm hai phần: encoder và decoder. Trong phần encoder đầu ra của lớp đầu tiên đóng vai trò là dữ liệu đầu vào của lớp mã hóa thứ hai. Giả sử có L lớp ẩn trong encoder, chúng ta có hàm kích hoạt của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

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

Với k = 0 thì $z^{(0)}$ chính là dữ liệu đầu vào $\tilde{\mathbf{x}}$ của input layer. Output của lớp mã hóa cuối cung z^L là các đặc trưng tốt được tạo ra trong quá trình huấn luyện mạng SDAE. 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 2.14 Sơ đồ cấu trúc mạng Stacked Denoise Autoencoder

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.

2.2.3 Ứng dụng Autoencoder trong tiền xử lý dữ liệu

Hiện nay, nhiều nghiên cứu đã áp dụng thành công các thuật toán học máy để hệ thống IDS có khả năng tự học và cập nhật các cuộc tấn công mới. Nhưng để hạn chế báo động nhầm và tăng khả năng dự đoán các cuộc tấn công thì ngoài khả năng tự quyết định, IDS cần phải có tư duy phân tích. Vì vậy ta cần phải ứng dụng học máy vào IDS. 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.

Việc ứng dụng mạng học sâu có hai 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.

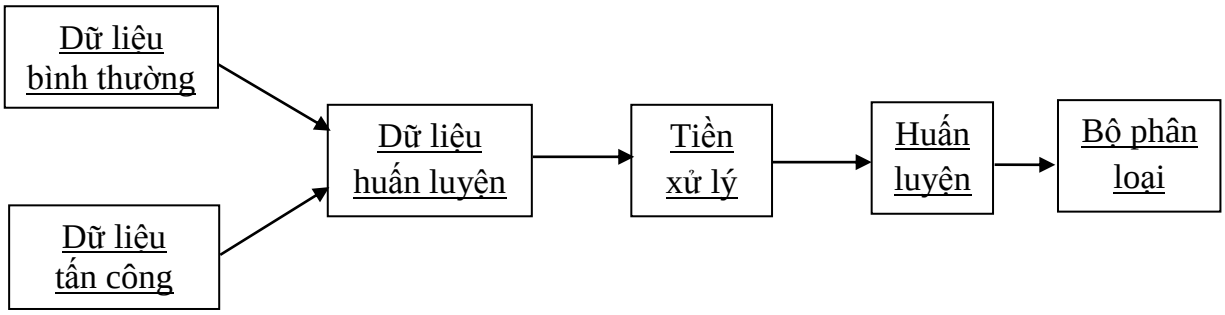
Có 02 giai đoạn trong quá trình phát hiện xâm nhập là: ***Learning Feature*** và ***Classifier***. Trong giai đoạn Learning Feature, các dữ liệu của mạng sẽ được đưa vào các mạng AE và DAE ta sẽ được mã chứa các đặc trưng đại diện nhất của dữ liệu. Các đặc trưng này có thể mô tả được dữ liệu đầu vào. Quá trình này 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 (*Stacked Denoise Autoencoder*) 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 học sâu

Mô hình phát hiện tấn công xâm nhập đề xuất được triển khai trong 02 giai đoạn: (1) giai đoạn huấn luyện và (2) giai đoạn phát hiện. Ta sẽ dùng các bộ dữ liệu được dán nhãn như NSS-KDD. Sau đó, ta sẽ đưa tất cả dữ liệu số và chia dữ liệu theo tỷ lệ 70% để huấn luyện (training) và 30% để kiểm thử (test) để kiểm tra mức độ phát hiện chính xác của mô hình sau khi được huấn luyện.

2.3.1 Giai đoạn huấn luyện

Giai đoạn này sẽ được tiến hành theo các bước sau như biểu diễn trên Hình 2.15 dưới đây.



Hình 2.15: Mô hình phát hiện tấn công xâm nhập: Giai đoạn huấn luyện

- Thu thập dữ liệu huấn luyện là bộ dữ liệu NSS-KDD gồm dữ liệu bình thường và dữ liệu là tấn công xâm nhập;

- Dữ liệu huấn luyện được tiền xử lý để chọn và trích xuất các đặc trưng sử dụng SAE, SDAE. Sau tiền xử lý, dữ liệu được chuyển đổi thành dạng véc tơ.

- Dữ liệu huấn luyện dưới dạng là các véc tơ được đưa vào bước huấn luyện để xây dựng bộ phân loại *(sử dụng các thuật toán phân loại)*, hoặc mô hình sử dụng cho giai đoạn phát hiện.

Cụ thể, trong bước chọn và trích xuất các đặc trưng của dữ liệu đầu vào (*bước tiền xử lý*), 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. Như vậy, bước này có một số bước con như sau:

- Bước 1. Chuẩn bị dữ liệu: Dữ liệu huấn luyện sẽ được chia thành các phần như X_train, Y_train. Trong đó X_train là dữ liệu được dùng để huấn luyện, Y_train là các nhãn của X_train có 02 loại bao gồm normal: 0 và attack: 1.

- 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 Nơron phát hiện xâm nhập mạng dựa trên SAE và SDAE được xây dựng với cấu trúc: Input Layer, 02 tầng ẩn và tầng đầu ra (*như hình dưới*). Số nút mỗi layer phụ thuộc vào số lượng đặc trưng của từng loại dữ liệu. Mục đích của việc huấn luyện là xác định bộ trọng số và giảm loss function. Vì vậy, tôi đã chọn thuật toán tối ưu Stochastic Gradient Descent. 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.

- Bước 3. Tiến hành huấn luyện (*trong giai đoạn tiền xử lý dữ liệu*): 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.

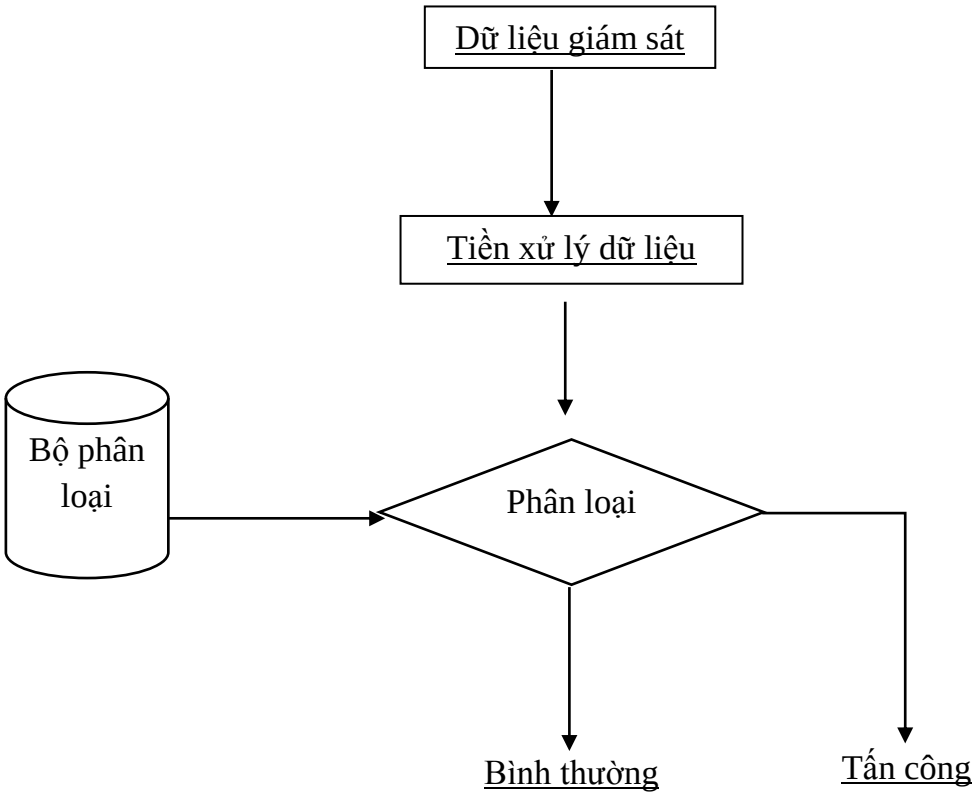
Sau khi huấn luyện xong một lần, ta xem xét loss function đạt tiêu chuẩn không, sau đó sẽ sử dụng mạng SAE và DAE để chuyển đổi X_train thành Z_train, chứa các đặc trưng của dữ liệu. Tiếp theo, Z_train, Y_train sẽ được classifier để học cách phân loại bằng các thuật toán phân loại.

- 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.

Các hệ số (*trọng số*) trong mô hình phát hiện liên tục được cập nhật đến khi giá trị sai số (*được tính theo chỉ số AUC*) trong phát hiện đạt đến một giá trị đủ bé chấp nhận được. Sau đó, ta có thể áp dụng mô hình phát hiện để kiểm thử trên tập dữ liệu X test (*30% của bộ dữ liệu ban đầu*)

2.3.2 Giai đoạn phát hiện

Sau khi hoàn thành giai đoạn huấn luyện, ta sẽ sử dụng mô hình phát hiện thu được để thực hiện đối với X test. Giai đoạn này được tiến hành theo các bước sau như biểu diễn trên Hình 2.16 dưới đây

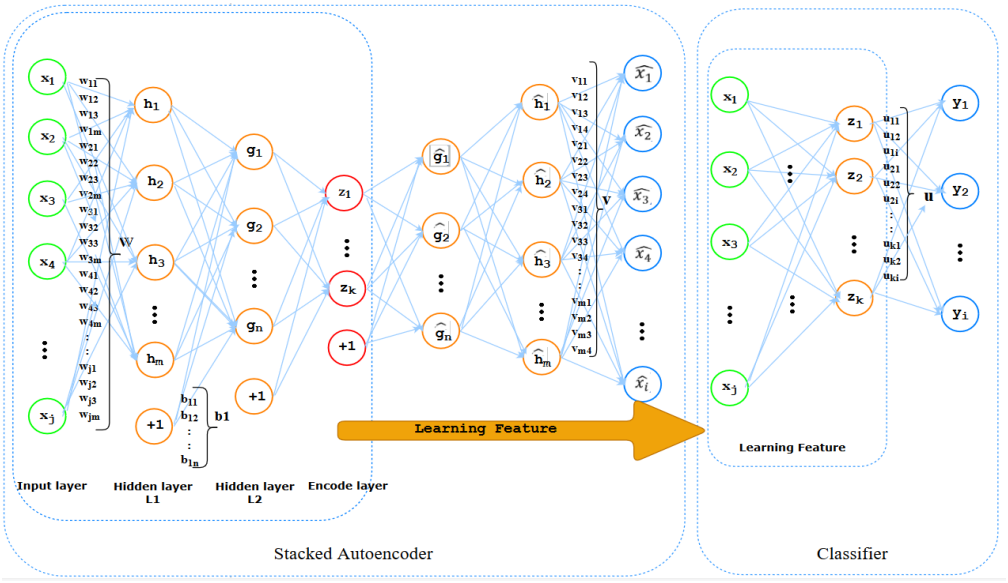


Hình 2.16: Mô hình phát hiện tấn công xâm nhập: Giai đoạn phát hiện

+ X test (*30% của bộ dữ liệu đầu vào ban đầu*) được trích chọn đặc trưng (*tiền xử lý*) theo mô hình nhận được từ giai đoạn huấn luyện. Kết quả của tiền xử lý là véc tơ được sử dụng cho bước tiếp theo;

+ Véc tơ được phân loại sử dụng Bộ phân loại (*bao gồm các thuật toán phân loại SVM - Support Vector Machine, DT - Decision Tree, RF - Random Forest, NB - Naive Baves, K Neighbors*) đã xây dựng trong giai đoạn huấn luyện. Kết quả của bước này là trạng thái dữ liệu đầu vào: Bình thường hoặc Tấn công.

Cụ thể, các thuật toán trong Bộ phân loại 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 để đánh giá mức độ chính xác của mô hình phát hiện.



Hình 2.17 Tổng thể mô hình ứng dụng SAE và SDAE vào phát hiện xâm nhập mạng

2.4 Kết luận chương

Chương 2 đã tìm hiểu khái quát về học máy máy và học sâu, đồng thời nghiên cứu xây dựng mô hình ứng dụng Autoencoder trong phát hiện xâm nhập mạng với 02 giai đoạn: Giai đoạn huấn luyện và giai đoạn phát hiện. Giai đoạn phát hiện sử dụng một số thuật toán phân loại có giám sát.

Trong Chương 3 **CÀI ĐẶT VÀ THỬ NGHIỆM**, nội dung chủ yếu là giới thiệu tập dữ liệu được sử dụng để huấn luyện cho học máy, phương pháp trích chọn đặc trưng sử dụng AE, các bước làm trong pha huấn luyện và phát hiện xâm nhập, các kết quả được dùng để đánh giá mức độ hiệu quả khi sử dụng AE.

CHƯƠNG 3
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 máy HP core i7, Ram 16G, ổ cứng HDD 1TB. Ngoài ra 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 số học 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 [9]. Tensor là cấu trúc dữ liệu trong tensorflow đại diện cho tất cả các loại dữ liệu. Nói cách khác, tất cả các kiểu dữ liệu khi đưa vào trong Tensorflow thì đều được gọi là Tensor. Do vậy, ta có thể hiểu được Tensorflow là một thư viện mô tả, điều chỉnh dòng chảy của các Tensor. Tensor có 03 thuộc tính cơ bản: là rank, shape và type. Rank là số bậc của tensor. Việc phân rank này khá quan trọng vì nó đồng thời cũng giúp phân loại dữ liệu của Tensor. Shape là chiều của Tensor. Kiểu dữ liệu của các phần tử (*elements*) trong Tensor. Vì 01 Tensor chỉ có duy nhất 01 thuộc tính Type nên từ đó suy ra chỉ có duy nhất một kiểu Type duy nhất cho toàn bộ phần tử có trong Tensor hiện tại.

Ngoài thư viện Tensorflow, ta còn sử dụng thư viện Sklearn là thư viện mã nguồn mở hỗ trợ hầu hết các thuật toán của học máy một cách đơn giản mà chúng ta không cần phải cài đặt, lập trình lại.

PyQt là thư viện lập trình giao diện của python được tôi lựa chọn để xây dựng ứng dụng thể hiện quá trình huấn luyện mạng.

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 với dữ liệu train;
- 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 (*tấn công giả mạo/ lừa đảo*). 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%.

Bảng 3.3 Các thuộc tính của tập dữ liệu Phishing Website Data

1	having_IP_Address	16	SFH
2	URL_Length	17	Submitting_to_email
3	Shortening_Service	18	Abnormal_URL
4	having_At_Symbol	19	Redirect
5	double_slash_redirecting	20	on_mouseover
6	Prefix_Suffix	21	RightClick
7	having_Sub_Domain	22	popUpWidnow
8	SSLfinal_State	23	Iframe
9	Domain_registeration_length	24	age_of_domain
10	Favicon	25	DNSRecord
11	port	26	web_traffic
12	HTTPS_token	27	Page_Rank
13	Request_URL	28	Google_Index
14	URL_of_Anchor	29	Links_pointing_to_page
15	Links_in_tags	30	Statistical_report

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 tra được phân thành 04 nhóm:

- Denial of Service (DoS) gồm các kiểu tấn công như: neptune, smurf, pod, teardrop, ... Trong trường hợp này, kẻ tấn công làm cho các tài nguyên tính toán hoặc bộ nhớ quá tải để xử lý các yêu cầu hợp lệ hoặc từ chối người dùng hợp lệ truy cập dịch vụ;
- Remote to Local (R2L) gồm các kiểu tấn công như: đoán mật khẩu (*guess-passwd*), ftp-write, imap, phf, ... Trong trường hợp này, kẻ tấn công tuy không có tài

khoản nhưng có khả năng gửi các gói tin đến một máy qua mạng, sẽ khai thác một số lỗ hổng để đạt được quyền truy cập cục bộ như là người sử dụng của máy đó;

- User to Root (U2R), gồm các kiểu tấn công như: buffer-overflow, load-module, perl, rootkit, ... Trong trường hợp này, kẻ tấn công bắt đầu với một quyền truy cập bình thường và sau đó khai thác một số lỗ hổng để đạt được quyền truy cập có đầy đủ quyền (*root*) trên hệ thống.

- Probe, gồm các kiểu tấn công như: port-sweep, ip-sweep, nmap, ... Trong trường hợp này, kẻ tấn công nỗ lực thu thập thông tin về mạng máy tính nhằm phá vỡ khả năng kiểm soát an ninh, an toàn thông tin của nó.

Bảng 3.2 Các kiểu tấn công trong tập dữ liệu NSL-KDD

Phân lớp	**Tên tấn công**	**Số bản ghi**	**Tỷ lệ %**
Normal		67.343	53, 45
Probe	ipsweep, mscan, nmap, portsweep, saint, satan	11.656	9, 26
DoS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm	45.927	36, 46
U2R	Buffer_overflow, httptunnel, loadmodule, perl, ps, rootkit, sqlattack, xterm	52	0, 04
R2L	ftp_write, guess_passwd, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xlock, xsnoop	995	0, 79
Tổng cộng		**125.973**	**100**

Bảng 3.3 Các thuộc tính của tập dữ liệu NSL-KDD

1	duration	22	Is_guest_login
2	protocol type	23	Count
3	service	24	Srv_count
4	flag	25	Serror_rate
5	Src_bytes	26	Srv_serror_rate
6	Dst_bytes	27	Rerror_rate
7	land	28	Srv_rerror_rate
8	Wrong_fragment	29	Same_srv_rate
9	urgent	30	Diff_srv_rate

10	hot	31	Srv_diff_host_rate
11	Num_failed_logins	32	Dst_host_count
12	Logged_in	33	Dst_host_srv_count
13	Num_compromised	34	Dst_host_same_srv_rate
14	Root_shell	35	Dst_host_diff_srv_rate
15	Su_attempted	36	Dst_host_same_src_port_rate
16	Num_root	37	Dst_host_srv_diff_host_rate
17	Num_file_creations	38	Dst_host_serror_rate
18	Num_shells	39	Dst_host_srv_serror_rate
19	Num_access_files	40	Dst_host_rerror_rate
20	Num_outbound_cmds	41	Dst_host_srv_rerror_rate
21	Is_hot_login		

3.3. Trích chọn đặc trưng sử dụng AE

Ta xây dựng mô hình mạng Nơron phát hiện xâm nhập mạng dựa trên SAE và SDEA cấu trúc: Tầng đầu vào, 02 tầng ẩn và tầng đầu ra. Số nút mỗi layer phụ thuộc vào số lượng đặc trưng của từng loại dữ liệu. Ta 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.

3.3.1 Phương pháp xây dựng mạng Nơron SAE

```
def build(self, input_dim):
    self.x = tf.placeholder(name='x', dtype=tf.float32, shape=[None, input_dim])
    # Encode
    # x -> z_mean, z_sigma -> z
    f1 = fc(self.x, self.hidden_layers[0], scope='ae_enc_fc1', activation_fn=tf.nn.relu)
    f3 = fc(f1, self.hidden_layers[1], scope='ae_enc_fc3', activation_fn=tf.nn.relu)
    self.z = fc(f3, self.hidden_layers[2], scope='ae_enc_fc5_mu', activation_fn=None)
    # Decode
    # z,y -> x_hat
    g2 = fc(self.z, self.hidden_layers[1], scope='ae_dec_fc2', activation_fn=tf.nn.relu)
    g3 = fc(g2, self.hidden_layers[0], scope='ae_dec_fc3', activation_fn=tf.nn.relu)
    self.x_hat = fc(g3, input_dim, scope='ae_dec_fc5', activation_fn=tf.sigmoid)
    # Loss
    recon_loss = tf.reduce_mean(tf.square(self.x - self.x_hat), 1)  # (((self.x - y)**2).mean(1)).mean()
```

```
self.recon_loss = tf.reduce_mean(recon_loss)
self.train_op = tf.train.AdamOptimizer(
  learning_rate=self.learning_rate).minimize(self.recon_loss)
return
```

3.3.2 Phương pháp xây dựng mạng Nơron SDAE

```
def build(self, input_dim):
    self.x = tf.placeholder(name='x', dtype=tf.float32, shape=[None, input_dim])
    Xnoise = self.x + self.noise_factor * tf.random_normal(tf.shape(self.x))
    Xnoise = tf.clip_by_value(Xnoise, 0., 1.)
    # Encode
    f1 = fc(Xnoise, self.hidden_layers[0], scope='dae_enc_fc1', activation_fn=tf.nn.relu)
    f2 = fc(f1, self.hidden_layers[1], scope='dae_enc_fc2', activation_fn=tf.nn.relu)
    self.z = fc(f2, self.hidden_layers[2], scope='dae_enc_fc3_mu', activation_fn=None)
    # Decode
    g1 = fc(self.z, self.hidden_layers[1], scope='dae_dec_fc2', activation_fn=tf.nn.relu)
    g2 = fc(g1, self.hidden_layers[0], scope='dae_dec_fc1', activation_fn=tf.nn.relu)
    self.x_hat = fc(g2, input_dim, scope='dae_dec_xhat', activation_fn=tf.nn.sigmoid)
    recon_loss = tf.reduce_mean(tf.square(self.x - self.x_hat), 1)
    self.recon_loss = tf.reduce_mean(recon_loss)
    self.train_op = tf.train.AdamOptimizer(
     learning_rate=self.learning_rate).minimize(self.recon_loss)
    return
```

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:

- Khai báo hàm chuyển đổi:

```
def transformer(self, x):
    z = self.sess.run(self.z, feed_dict={self.x: x}) #Function of Tensorflow
    return z
```

- Chuyển đổi dữ liệu

z_train = model1.transformer(X_train)# Chuyen doi du lieu X_train thanh z_train

z_test = model1.transformer(X_test))# Chuyen doi du lieu X_test thanh z_test

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 (Support Vector Machine)

auc_svm, t1 = classifier.svm(z_train, Y_train, z_test, Y_test)

- Decision Tree *(DT)*

auc_dt, t2 = classifier.decisiontree(z_train, Y_train, z_test, Y_test)

- RF *(Random Forest)*

auc_rf, t3 = classifier.rf(z_train, Y_train, z_test, Y_test)

- Naive Baves *(NB)*

auc_nb, t4 = classifier.naive_baves(z_train, Y_train, z_test, Y_test)

- K Neighbors

auc_kn, t5 = classifier.KNeighbors(z_train, Y_train, z_test, Y_test)

3.4.1 Phương pháp sử dụng mạng Nơron SAE

```
def AE_trainer(learning_rate=1e-3, batch_size=100, num_epoch=10, hidden_layers=[7, 4, 2], input_dim=0, step=20, X_train=[], X_test=[], Y_train=[], Y_test=[], dt=[]):
    model1 = AE(learning_rate=learning_rate, batch_size=batch_size, hidden_layers=hidden_layers, input_dim=input_dim)
    for epoch in range(num_epoch):
        num_sample = len(X_train)
        for iter in range(num_sample // batch_size):
            X_mb, _ = dt.train.next_batch(batch_size)
        # Execute the forward and the backward pass and report computed losses
            recon_loss = model1.run_single_step(X_mb)
        if epoch % step == 0:
            print('[Epoch {}] Recon loss: {}'.format(
                epoch, recon_loss))
            chartcolumn.take_display_traning('Epoch ' + str(epoch) + ' Recon loss: ' + str(recon_loss))
            # Training Process
            z_train = model1.transformer(X_train)
            s = time.time()
            z_test = model1.transformer(X_test)
            e = time.time()
```

```
                t_tr = (e - s) / float(len(X_test))
                # Clasifier process using
                auc_svm, t1 = classifier.svm(z_train, Y_train, z_test, Y_test)
                auc_dt, t2 = classifier.decisiontree(z_train, Y_train, z_test, Y_test)
                auc_rf, t3 = classifier.rf(z_train, Y_train, z_test, Y_test)
                auc_nb, t4 = classifier.naive_baves(z_train, Y_train, z_test, Y_test)
                auc_kn, t5 = classifier.KNeighbors(z_train, Y_train, z_test, Y_test)
                AE_recon_loss_.append(recon_loss)
                AE_auc_svm_.append(auc_svm)
                AE_auc_dt_.append(auc_dt)
                AE_auc_rf_.append(auc_rf)
                AE_auc_nb_.append(auc_nb)
                AE_auc_kn_.append(auc_kn)
                AE_t1_.append((t1 + t_tr))
                AE_t2_.append((t2 + t_tr))
                AE_t3_.append((t3 + t_tr))
                AE_t4_.append((t4 + t_tr))
                AE_t5_.append((t5 + t_tr))
        print('Done AE!')
            return model1
```

3.4.2 Phương pháp sử dụng mạng Nơron SDAE

```
    def DAE_trainer(learning_rate=1e-3, batch_size=100, num_epoch=10,
hidden_layers=[7, 4, 2], input_dim=0, step=20, X_train=[], X_test=[], Y_train=[],
Y_test=[], dt=[], noise_factor=0.25):
    model1 = DAE(learning_rate=learning_rate, batch_size=batch_size,
hidden_layers=hidden_layers, input_dim=input_dim,
    noise_factor=noise_factor)
    for epoch in range(num_epoch):
        num_sample = len(X_train)
        for iter in range(num_sample // batch_size):
            X_mb, _ = dt.train.next_batch(batch_size)
         # Execute the forward and the backward pass and report computed losses
            recon_loss = model1.run_single_step(X_mb)
        if epoch % step == 0:
            chartcolumn.take_display_traning('Epoch ' + str(epoch) + ' Recon loss:
' + str(recon_loss))
            # Training Process
            z_train = model1.transformer(X_train)
```

```
            s = time.time()
            z_test = model1.transformer(X_test)
            e = time.time()
            t_tr = (e - s) / float(len(X_test))
           # Clasifier Process
            auc_svm, t1 = classifier.svm(z_train, Y_train, z_test, Y_test)
            auc_dt, t2 = classifier.decisiontree(z_train, Y_train, z_test, Y_test)
            auc_rf, t3 = classifier.rf(z_train, Y_train, z_test, Y_test)
            auc_nb, t4 = classifier.naive_baves(z_train, Y_train, z_test, Y_test)
            auc_kn, t5 = classifier.KNeighbors(z_train, Y_train, z_test, Y_test)
            DAE_recon_loss_.append(recon_loss)
            DAE_auc_svm_.append(auc_svm)
            DAE_auc_dt_.append(auc_dt)
            DAE_auc_rf_.append(auc_rf)
            DAE_auc_nb_.append(auc_nb)
            DAE_auc_kn_.append(auc_kn)
            DAE_t1_.append((t1 + t_tr))
            DAE_t2_.append((t2 + t_tr))
            DAE_t3_.append((t3 + t_tr))
            DAE_t4_.append((t4 + t_tr))
            DAE_t5_.append((t5 + t_tr))
    print('Done DAE!')
    return model1
```

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 tầng ẩn và số lượng nút mỗi layer là [25, 15]. Tôi đã thử huấn luyện với số lượng tầng ẩn là 03 và 04 hidden và 01 thì kết quả AUC không tốt bằng khi sử dụng 02 tầng ẩn.

Để so sánh việc sử dụng các mạng SAE và SDAE và không sử dụng mạng trong quá trình phân loại (*classifier*), tôi tiến hành 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.

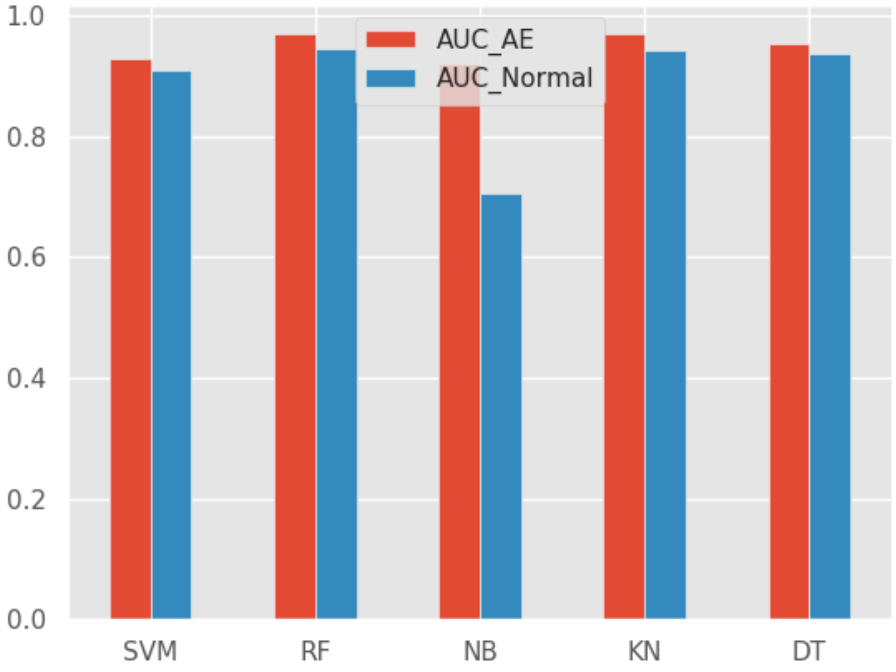
Tuy nhiên, việc sử dụng mạng học sâu có kết quả khác nhau đối với các thuật toán phân loại khác nhau.

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 AUC 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 AUC tăng khoảng 20 % đối với mạng SAE còn các thuật toán khác chỉ tăng từ 02 đến 03%.

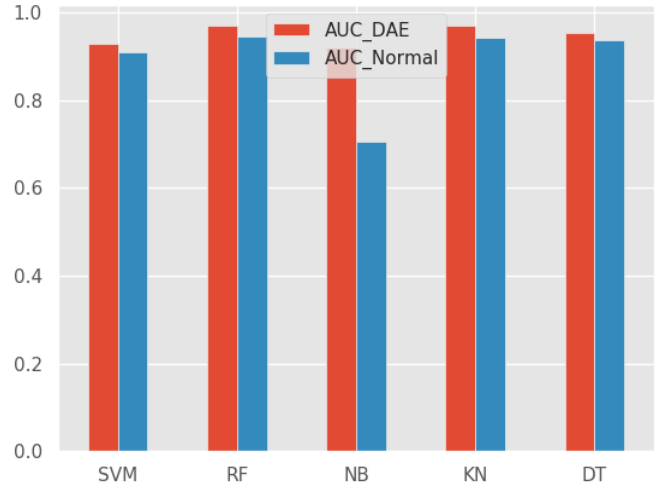
Đối với mạng SDAE, thuật toán Naive Bayes tăng khoảng 23% và các thuật toán khác cũng chỉ tăng từ 02 đến 03 %. Nhưng ta có thể nhận thấy rằng chỉ số AUC của mạng SDAE cao hơn mạng SAE trong bộ dữ liệu này.

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



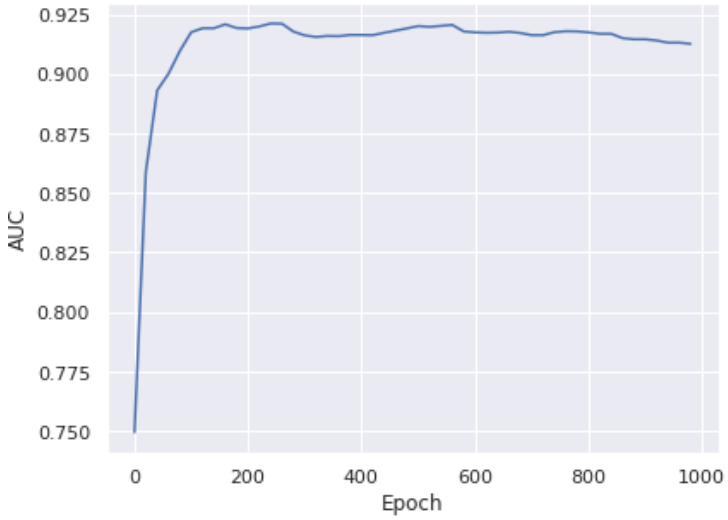
Hình 3.5 Biểu đồ so sánh AUC giữa sử dụng SAE và không sử dụng SAE đối với dữ liệu Phishing Data Website



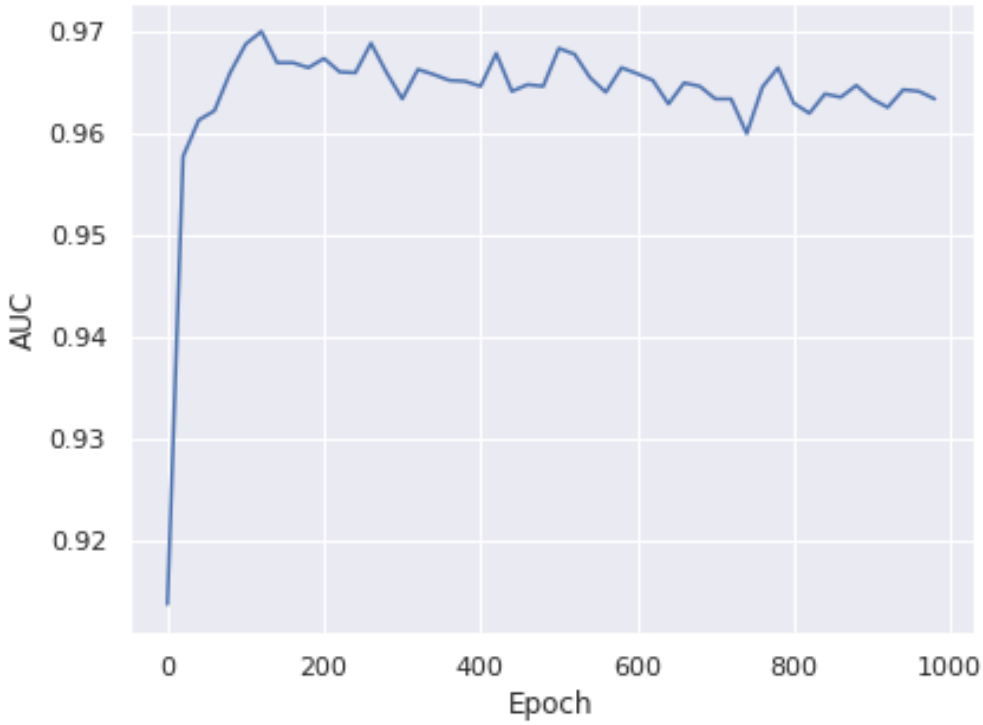
Hình 3.6 Biểu đồ so sánh AUC giữa sử dụng SDAE và không sử dụng SDAE đối với bộ dữ liệu Phishing Data Website

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 và được kết quả như những hình dưới.

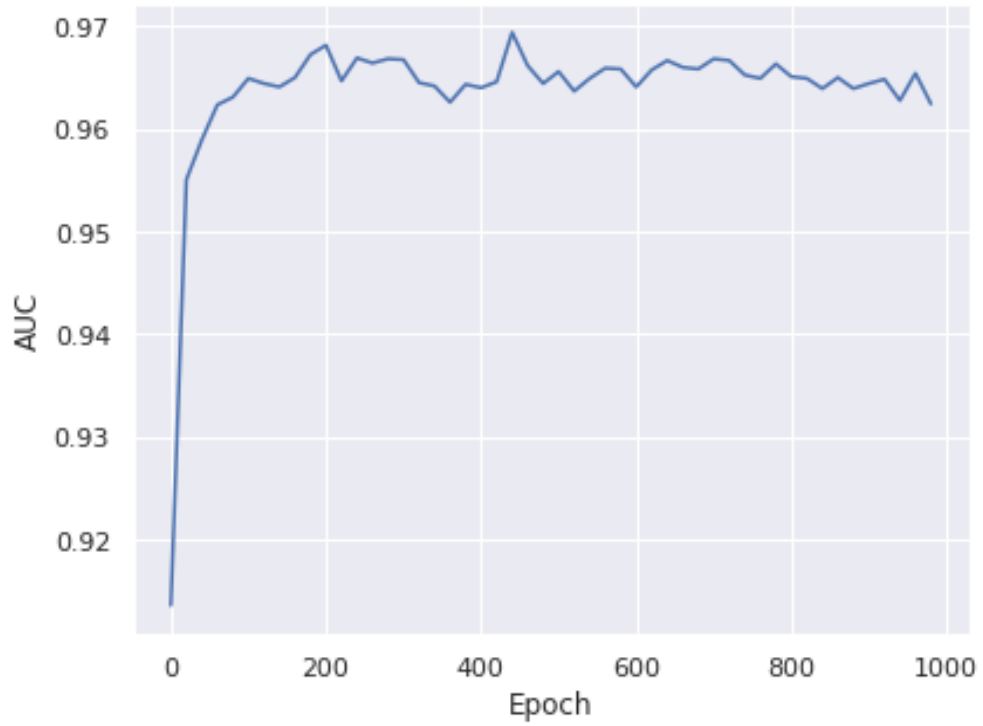
Ta nhận thấy quá trình phân loại cho kết quả tương đối tốt ngay từ những đợt huấn luyện đầu tiên. Với thuật toán NB và SVM chỉ số AUC của cả 02 mạng đều không biến động nhiều trong suốt quá trình nhưng ngược lại các thuật toán DT, KN và RF chỉ số AUC biến đổi liên tục không ổn định. Đặc biệt là thuật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



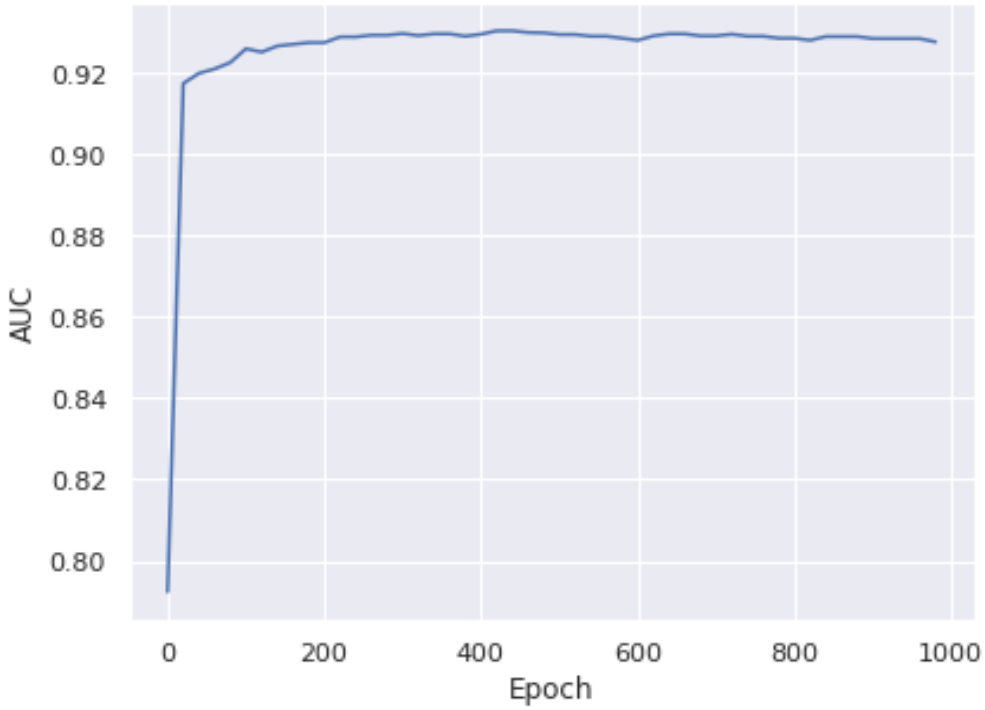
Hình 3.7 Biểu đồ AUC khi huấn luyện SAE sử dụng thuật toán NB đối với bộ dữ liệu Phishing Data Website



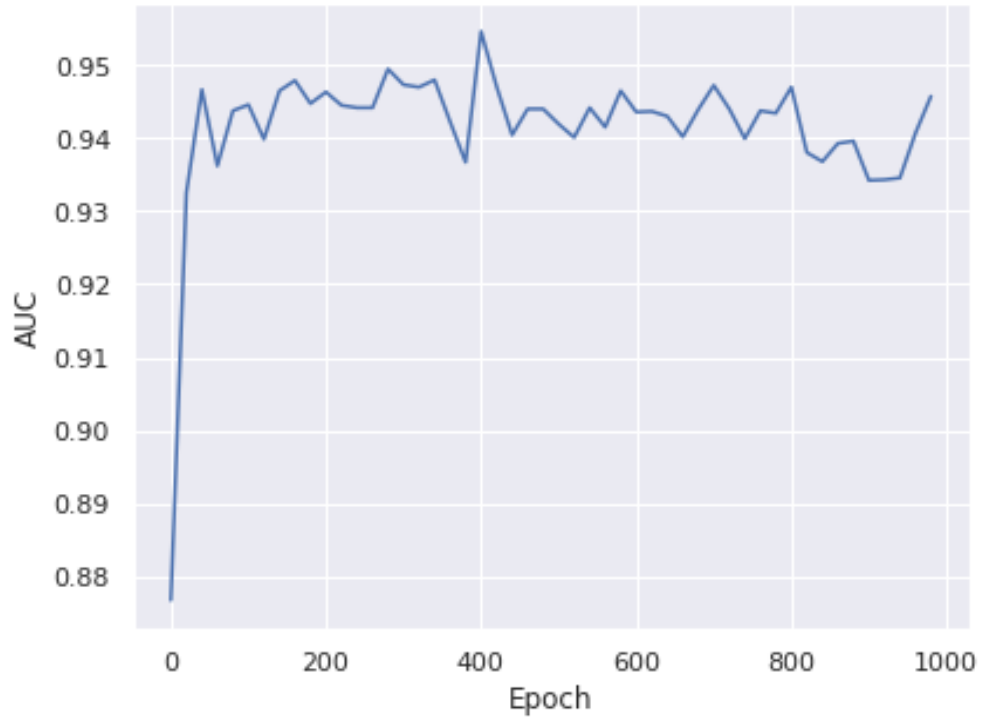
Hình 3.8 Biểu đồ AUC khi huấn luyện SAE sử dụng thuật toán KN đối với bộ dữ liệu Phishing Data Website



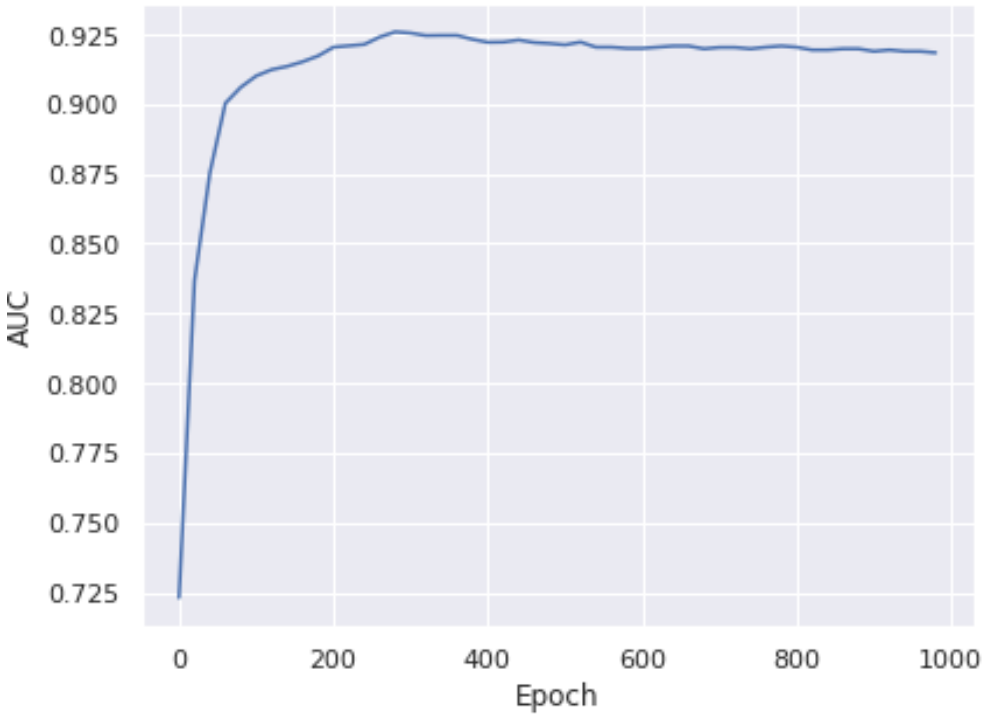
Hình 3.9 Biểu đồ AUC khi huấn luyện SAE sử dụng thuật toán RF đối với bộ dữ liệu Phishing Data Website



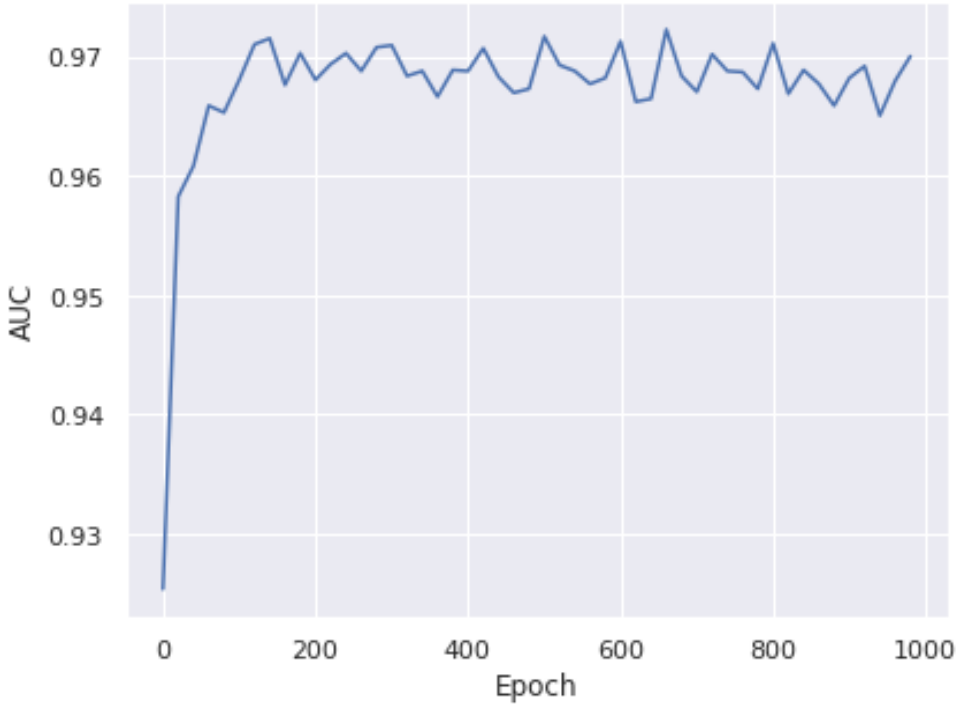
Hình 3.10 Biểu đồ AUC khi huấn luyện SAE sử dụng thuật toán SVM đối với bộ dữ liệu Phishing Data Website



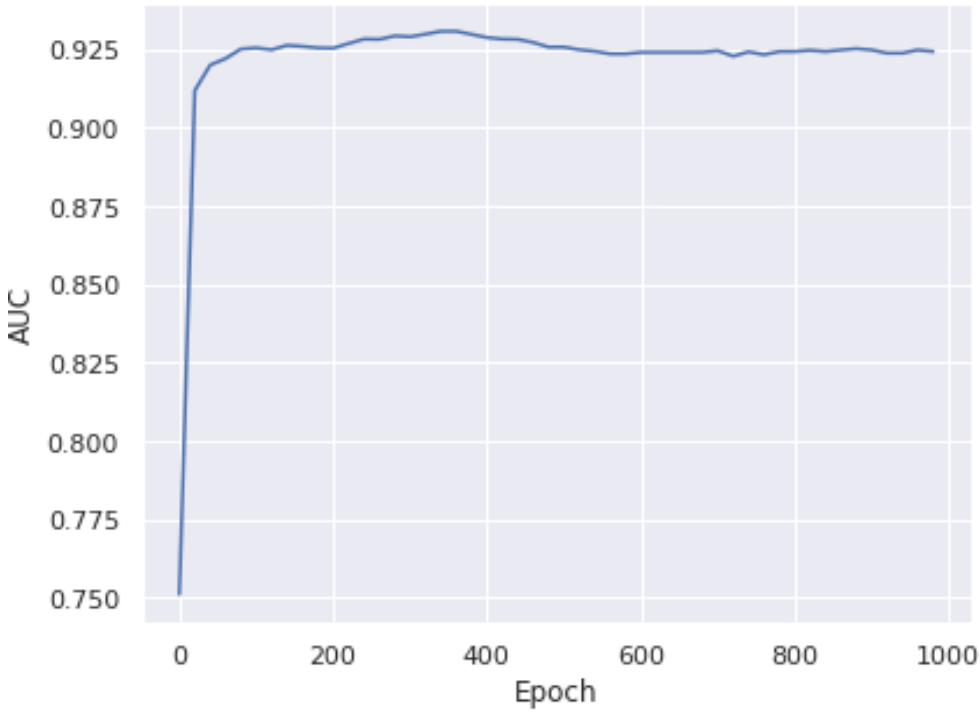
Hình 3.11 Biểu đồ AUC khi huấn luyện SAE sử dụng thuật toán DT đối với bộ dữ liệu Phishing Data Website



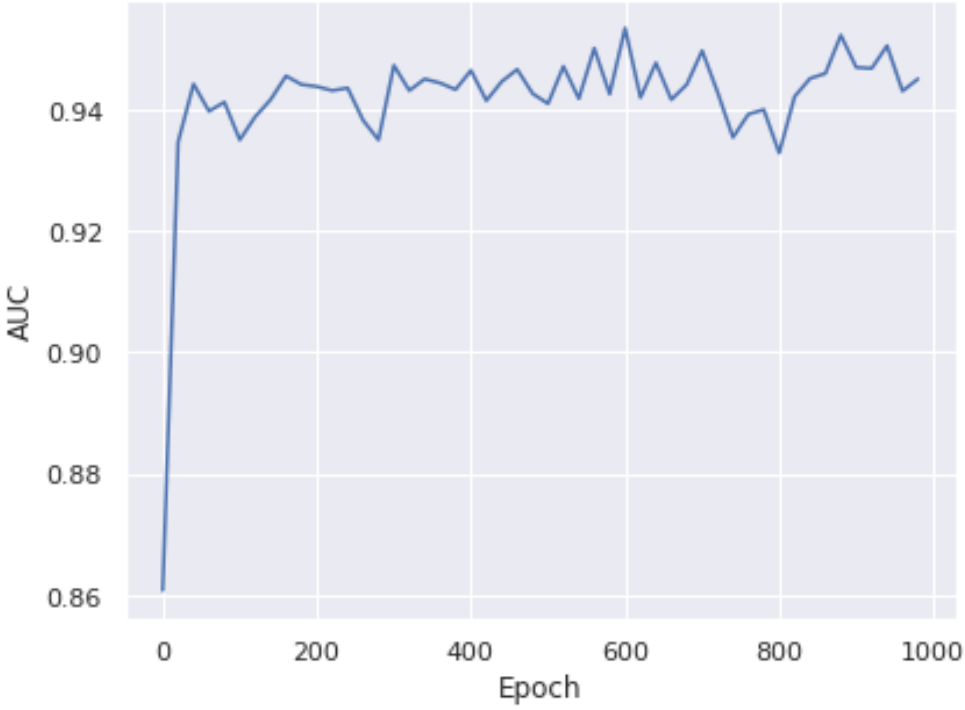
Hình 3.12 Biểu đồ AUC khi huấn luyện SDAE sử dụng thuật toán NB đối với bộ dữ liệu Phishing Data Website



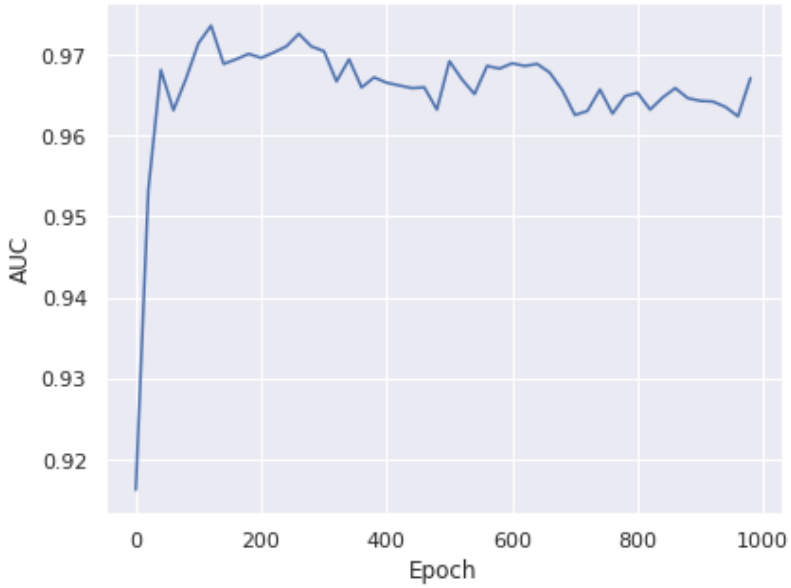
Hình 3.13 Biểu đồ AUC khi huấn luyện SDAE sử dụng thuật toán RF đối với bộ dữ liệu Phishing Data Website



Hình 3.14 Biểu đồ AUC khi huấn luyện SDAE sử dụng thuật toán SVM đối với bộ dữ liệu Phishing Data Website

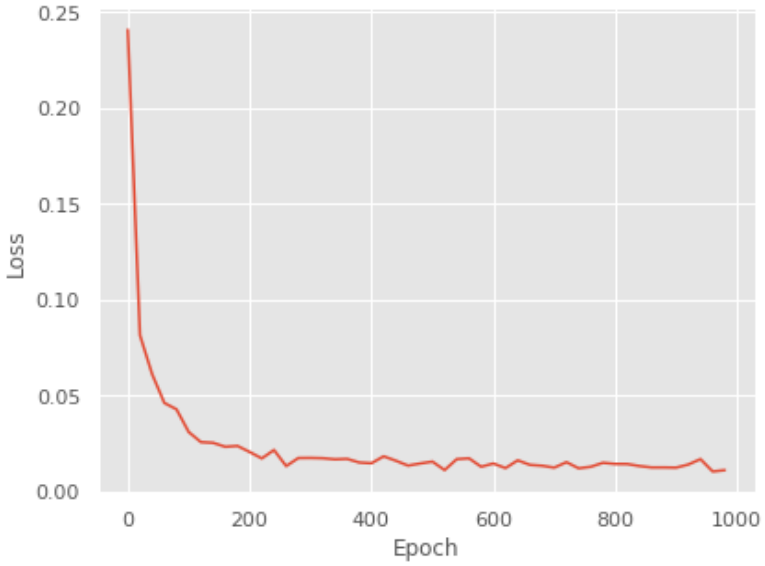


Hình 3.15 Biểu đồ AUC khi huấn luyện SDAE sử dụng thuật toán DT đối với bộ dữ liệu Phishing Data Website

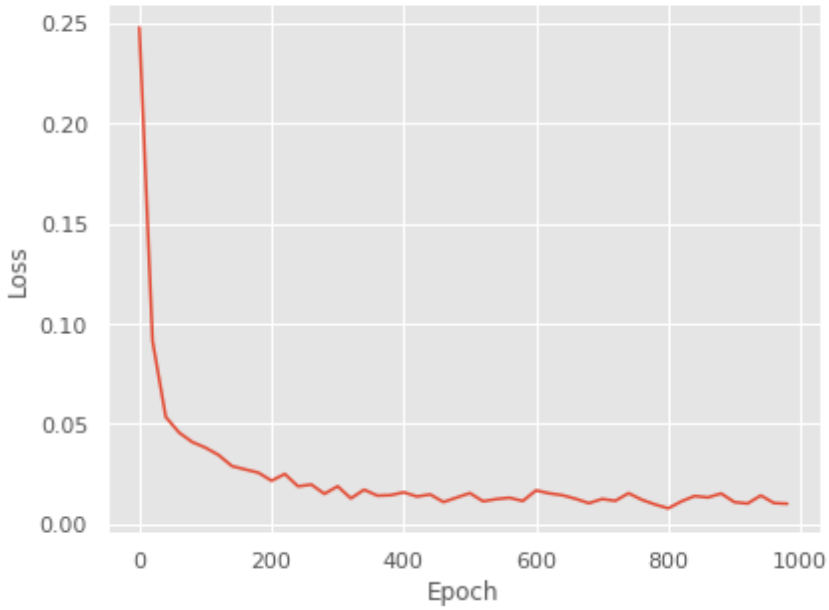


Hình 3.16 Biểu đồ AUC khi huấn luyện SDAE sử dụng thuật toán KN đối với bộ dữ liệu Phishing Data Website

Qua đồ thị của 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 xảy ra do ta đã áp dụng thuật toán tối ưu SGD giúp hàm loss hội tụ nhanh chóng về điểm cực tiểu. Điều này có nghĩa 02 mạng SAE và DAE đã học các đặc trưng rất tốt ngay từ những đợt huấn luyện đầu tiên. Chính vì vậy, chỉ số AUC tăng nhanh và ta thấy rằng loss function của 02 mạng gần giống nhau.



Hình 3.17 Biểu đồ loss function khi huấn luyện SAE đối với bộ dữ liệu Phishing Website Data



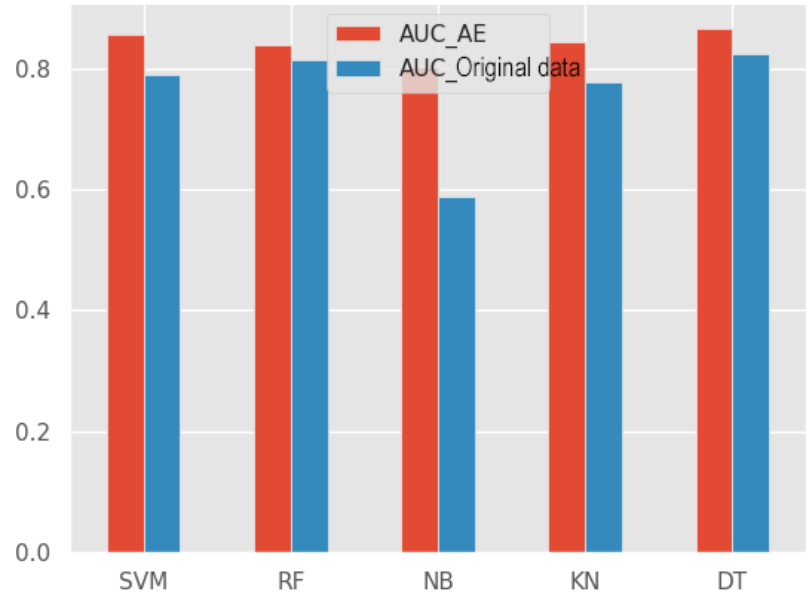
Hình 3.18 Biểu đồ loss function khi huấn luyện SDAE đối với bộ dữ liệu Phishing Website Data

3.5.2 Kết quả của bộ dữ liệu NSL-KDD

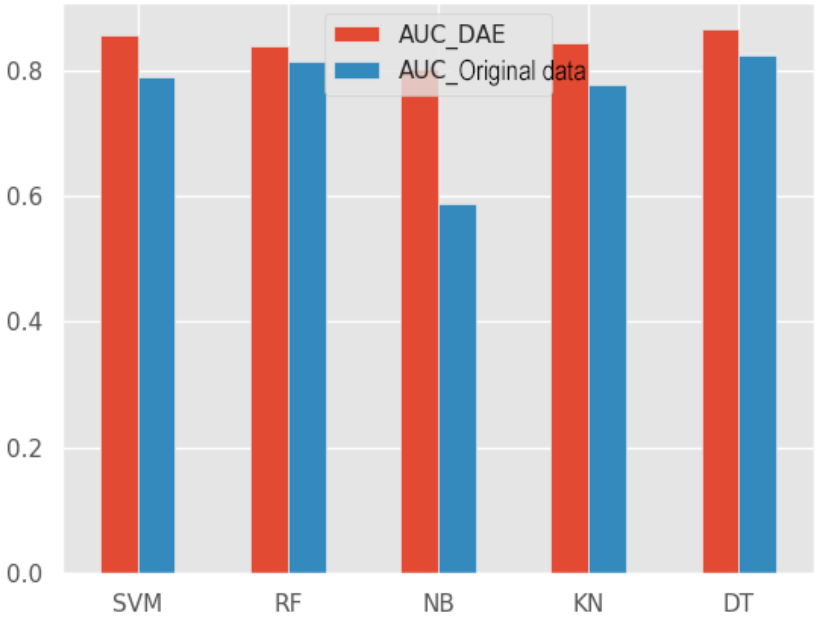
Bộ dữ liệu NSL-KDD bao gồm 41 feature nên trong quá trình huấn luyện, tôi đã xây dựng mạng SAE và SDAE với cấu trúc 02 tầng ẩn (30 và 15 nút). Bộ dữ liệu được dán 02 loại dãn nhán là tấn công và không tấn công. Thuật toán NB khi sử dụng mạng SAE và SDAE với bộ dữ liệu NSL-KDD tăng rất nhiều khoảng 30% 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 3.19 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

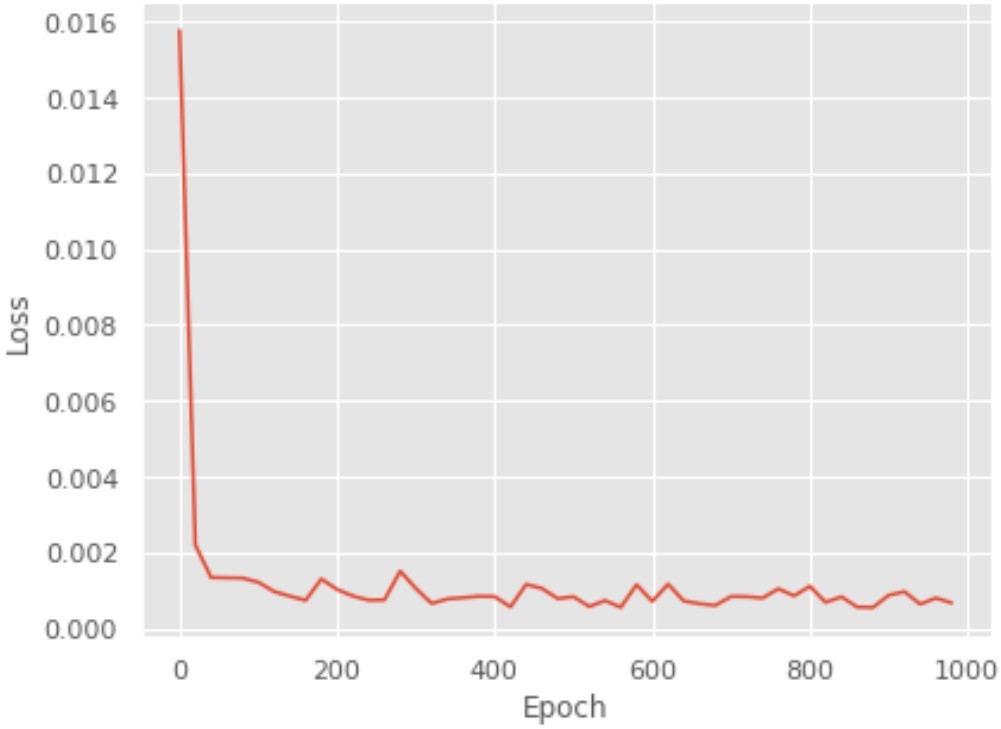


Hình 3.20 Biểu đồ so sánh AUC giữa sử dụng SAE và không sử dụng SAE đối với bộ dữ liệu NSL-KDD

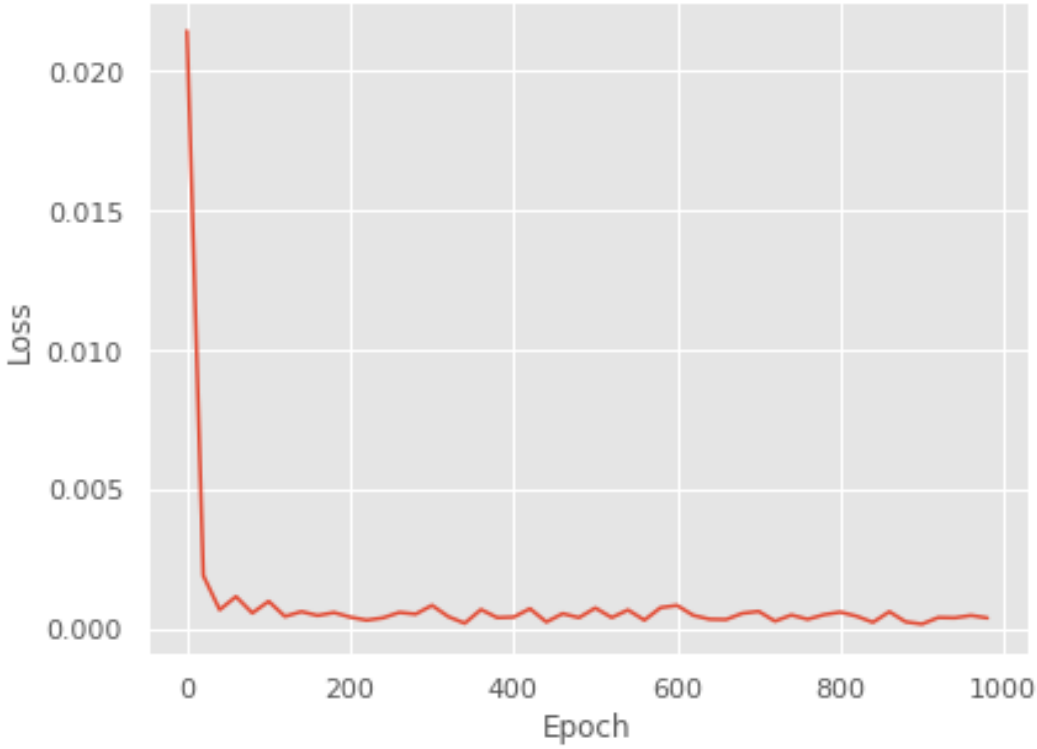


Hình 3.21 Biểu đồ so sánh AUC giữa sử dụng SDAE và không sử dụng SDAE đối với bộ dữ liệu NSL-KDD

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.



Hình 3.22 Biểu đồ loss function khi huấn luyện SAE đối với bộ dữ liệu NSL-KDD



Hình 3.23 Biểu đồ loss function khi huấn luyện SDAE đối với bộ dữ liệu *NSL-KDD*

3.6 Kết luận chương

Trong chương 3 của luận văn đã mô tả chi tiết dữ liệu được sử dụng cho mô hình phát hiện tấn công xâm nhập mạng sử dụng học máy, mô tả chi tiết phương pháp huấn luyện và phát hiện xâm nhập và thống kê các kết quả đạt được bằng nhiều kịch bản thử nghiệm khác nhau từ đó rút ra được nhận xét ưu điểm và những hạn chế của phương pháp học máy sử dụng AE.

KẾT LUẬN

Kết quả đạt được:

Từ nội dung của 3 chương, luận văn đã đạt được những kết quả sau:

- Trình bày khái quát lý thuyết về xâm nhập mạng, phát hiện xâm nhập mạng và một số phương pháp phát hiện xâm nhập mạng.
- Trình bày khái quát về học máy và học sâu.
- Nghiên cứu về Autoencoder và sử dụng để trích chọn đặc trưng dữ liệu giám sát xâm nhập mạng, ứng dụng trong tiền xử lý dữ liệu.
- Đưa ra mô hình phát hiện tấn công xâm nhập mạng và nguyên lý hoạt động của mô hình. Trình bày quá trình xử lý dữ liệu sử dụng phương pháp trích chọn đặc trưng AutoEncoder và đưa dữ liệu vào huấn luyện, phát hiện tấn công sử dụng một số thuật toán học máy có giám sát (*SVM - Support Vector Machine, DT - Decision Tree, RF - Random Forest, NB - Naive Baves, K Neighbors*).
- Thử nghiệm mô hình phát hiện tấn công xâm nhập mạng đã được xây dựng. Kết quả: Tính hiệu quả của các thuật toán NB tốt hơn rất nhiều so với việc không sử dụng Autoencoder. Tuy nhiên, với các thuật toán SVM, DT, RF, KNN, tính hiệu quả của việc sử dụng mạng Autoencoder không khác nhiều so với không sử dụng.

Hướng phát triển trong tương lai

- Do hạn chế về thời gian và khả năng, luận văn mới chỉ xây dựng 01 mạng Nơron cho cả 05 thuật toán học máy có giám sát nên tính hiệu quả của các thuật toán ngoại trừ thuật toán NB tăng rất ít so với việc không sử dụng Autoencoder. Trong thời gian tới, tôi sẽ tiếp tục nghiên cứu, tìm hiểu thêm để cải tiến, tối ưu mạng với mục tiêu tăng mức độ hiệu quả hơn nữa đối với từng thuật toán.
- Cập nhật thêm dữ liệu để phát hiện được các loại tấn công xâm nhập mới hiện nay.

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

[1] Hoàng Xuân Dậu (2017), Bài giảng an toàn bảo mật hệ thống thông tin, Học viện Công nghệ Bưu chính Viễn thông, 151.

[2]. Kendall, Kristopher (1999), "A Database of Computer Attacks for the Evaluation of Intrusion Detection System", Massachusetts Institute of Technology, 124.

[3] Carl Endorf (2003), Intrusion Detection and Prevention, 500.

[4] Meera Gandhi, S.K.Srivatsa (2008), Detecting and Preventing Attacks Using Network Intrusion Detection Systems, International Journal of Computer Science and Security (IJCSS), 13.

[5] Lê Minh Trung (biên dịch) (1999), Giáo trình mạng nơron nhân tạo, Nhà XB Thống kê, 1999.

[6]. Ian Goodfellow, Yoshua Bengio, Aaron Courville, (2017), Deep learning, 800.

[7]. Vũ Hữu Tiệp (2017), Machine Learning cơ bản, 31.

[8]. Ankit, S. (2017), *TensorFlow Tutorial: 10 minutes Practical TensorFlow Lesson for Quick Learners*, http://cv-tricks.com/artificial-intelligence/deeplearning/deep-learningframeworks/tensorflow-tutorial/, 26/10/2016.

[9]. Mor Geva (2018), Introduction to TensorFlow, https://www.cs.tau.ac.il/~joberant/teaching/advanced_nlp_spring_2018/files/tensorflow_tutorial.pdf, 04/2018

[10]. Mohammed Gharib (2019), *AutoIDS: Auto-encoder Based Method for Intrusion Detection System*, https://www.researchgate.net/publication/337157749_AutoIDS_Auto-encoder_Based_Method_for_Intrusion_Detection_System, 14/04/2020.

[11]. Fahimeh Farahnakian (2018), A deep auto-encoder based approach for intrusion detection system, 20th International Conference on Advanced Communication Technology (ICACT), Gang'weondo - South Korea, 11-14/02/2018.

[12] J. Ling and C. Wu, "Feature selection and deep learning based approach for network intrusion detection," in 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019), Atlantis Press, 2019

[13] J. Ling and C. Wu (2019), "Feature selection and deep learning based approach for network intrusion detection" 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019), Atlantis Press, 29-30/03/2019

[14] Zolzaya Kherlenchimeg (2018), "Network Intrusion Classifier Using Autoencoder with Recurrent Neural Network" The Fourth International Conference on Electronics and Software Science (ICESS2018), Takamatsu, Japan, 05-07/11/2018

[15] Shahadate Rezvy (2018), “Intrusion Detection and Classification with Autoencoded Deep Neural Network” 11th International Conference, SecITC 2018, Bucharest, Romania, 08–09/11/2018

[16] Sun, X.; Dai, J.; Liu, P.; Singhal, A.; Yen, J. “Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths”. IEEE Transaction on Information Forensics and Securirty. 2018, 29/3/2018.

[17] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning” MIT Press: Cambridge, MA, USA, 2016, 07/10/2016.

[18] Hinton, G.E.; Salakhutdinov, R.R. “Reducing the Dimensionality of Data with Neural Networks.” Science 2006, 28/7/2006.

[19] Zabalza, J.; Ren, J.; Zheng, J.; Zhao, H.; Qing, C.; Yang, Z.; Du, P.; Marshall, S. “Novel Segmented Stacked Autoencoder for Effective Dimensionality Reduction and Feature Extraction in Hyperspectral Imaging.” Neurocomputing 2016, 26/5/2016.

[20] Zhou, C.; Paffenroth, R.C. “Anomaly Detection with Robust Deep Autoencoders.” In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17/8/2017

[21] TS. Hoàng Xuân Dậu và Nguyễn Trọng Hưng, “Phát Hiện Tấn Công Web Thường Gặp Dựa Trên Học Máy Sử Dụng Web Log”, Kỷ yếu Hội nghị Quốc gia lần thứ XIII về Nghiên cứu cơ bản và ứng dụng Công Nghệ thông tin (FAIR), Nha Trang Việt Nam, 08-09/10/2020./.