

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI TP HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2023

NGHIÊN CỨU XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH
HỌC SÂU

13-SV-2023-TH2

Thuộc nhóm ngành khoa học: An toàn thông tin

TPHCM – 10/2023

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI TP HỒ CHÍ MINH

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2023

NGHIÊN CỨU XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH
HỌC SÂU

13-SV-2023-TH2

Thuộc nhóm ngành khoa học: An toàn thông tin

Sinh viên thực hiện: Lê Hoài Duy Bình Nam, Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: D20DCAT01-N, Khoa Công nghệ thông tin 02

Năm thứ: 4 / Số năm đào tạo: 4,5

Ngành học: An toàn thông tin

Sinh viên hỗ trợ: Võ Thanh Phong Nam, Nữ: Nam

Dân tộc: Kinh

Lớp, khoa: D20DCAT01-N, Khoa Công nghệ thông tin 02

Năm thứ: 4 / Số năm đào tạo: 4,5

Ngành học: An toàn thông tin

Người hướng dẫn: Thạc sĩ Đàm Minh Lịnh

TPHCM – 10/2023

Mục Lục

LỜI MỞ ĐẦU	3
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	4
1.1. Giới thiệu các cuộc tấn công mạng.....	4
1.1.1. Giới thiệu	4
1.1.2. Các cuộc tấn công mạng phổ biến	4
1.1.3. Phòng chống	4
1.2. Giới thiệu mô hình hồi quy Logistics.....	5
1.2.1. Ma trận nhầm lẫn	5
1.2.2. Nguyên lý MLE	6
1.2.3. Hàm lỗi NLL	7
1.2.4. Xuống đồi bằng đạo hàm	7
1.2.5. Các thông số hàm sigmoid, softmax	7
Thuật toán huấn luyện LR-GD	8
1.2.6. Mô hình Logistics phân lớp nhiều lớp	9
Phân bố nhóm	9
Mô hình xác suất	9
1.3. Giới thiệu các mô hình ML/DL	10
1.3.1. Mô hình MLP	10
Giới thiệu	10
Huấn luyện	10
1.3.2. Mô hình CNN	11
Giới thiệu	11
Huấn luyện	11
1.3.3. Mô hình RNN	11
Giới thiệu	11
Huấn luyện	12
1.3.4. Mô hình LSTM	12
Giới thiệu	12
Huấn luyện	12
1.4. Giới thiệu bài toán trích xuất đặc trưng.....	13
Khái Niệm.....	13
Một số kỹ thuật trích xuất đặc trưng.....	13
1.5. Thuật toán KNN.....	16
Khái niệm.....	16
Chi tiết thuật toán.....	16
CHƯƠNG 2: THU THẬP DỮ LIỆU & MÔ HÌNH.....	18
2.1 Thu thập dữ liệu.....	18
2.2 Xử lý dữ liệu.....	19
2.3 Thiết kế mô hình học sâu.....	19

CHƯƠNG 3: TRIỂN KHAI VÀ XÂY DỰNG MÔ HÌNH	20
3.1 Import các thư viện để sử dụng:	20
3.2 Mã nguồn	20
3.3 Tạo DataFrame với multiclass labels	21
3.4 Chuẩn hóa dữ liệu	21
3.5 Mã hóa các nhãn	21
3.6 Tiến hành thiết kế mô hình	22
3.7 Thiết lập quá trình huấn luyện	23
3.8 Training đánh giá độ chính xác model CNN	23
3.9 Nhận xét	24
3.10 Training đánh giá độ chính xác model MLP	26
3.11 Training đánh giá độ chính xác model LSTM-RNN	27
3.9 Kết luận:	28
CHƯƠNG 4: TRIỂN KHAI ỨNG DỤNG	3
4.1 Xây dựng một chương trình phát hiện tấn công mạng dựa trên model đã training... 3	
CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ	4
5.1. Chuẩn bị máy Attacker và Victim	4
5.2. Tiến hành quá trình kiểm thử	6
Tài liệu tham khảo	9

DANH MỤC HÌNH ẢNH

Hình 1 . Minh họa ma trận nhầm lẫn	5
Hình 2 . Minh họa Sigmoid	7
Hình 3 . Minh họa Softmax	8
Hình 4 . Mô hình phân lớp nhiều lớp với softmax	10
Hình 5 . Minh họa mô hình MLP	10
Hình 6 . Minh họa mô hình CNN	11
Hình 7 . Minh họa RNN	12
Hình 8 . Minh Họa LSTM	13
Hình 9 . Ví dụ về BoW	14
Hình 10 . Ví dụ bag-of-n-gram	15
Hình 11 . Ví dụ Word2vec	16
Hình 12 . Công thức khoảng cách	17
Hình 13 . Ví dụ KNN	17
Hình 14 . Dữ liệu Dataset NSL-KDD	18
Hình 15 . Dữ liệu Dataset NSL-KDD_2	18
Hình 16 . Mô hình training CNN	19
Hình 17 . Import thư viện cần thiết	20
Hình 18 . Danh sách đặc trưng trong dataset	20
Hình 19 . Các nhãn tấn công	21
Hình 20 . Chuyển đổi các nhãn thành các lớp	21
Hình 21 . Tạo DataFrame	21
Hình 22 . Chuẩn hóa dữ liệu	21
Hình 23 . Mã hóa các nhãn	21
Hình 24 . Thiết kế model CNN	22
Hình 25 . Model CNN dạng biểu mẫu	22
Hình 26 . Tóm tắt mô hình kiến trúc CNN	23
Hình 27 . Compile mô hình	23
Hình 28 . Training và đánh giá model	24
Hình 29 . Biểu đồ về accuracy	24
Hình 30 . Biểu đồ loss	25
Hình 31 . AUC	26
Hình 32 . Tóm tắt mô hình kiến trúc MLP	26
Hình 33 . Training đánh giá model MLP	27
Hình 34 . Tóm tắt mô hình kiến trúc LSTM&RNN	27
Hình 35 . Training và đánh giá model RNN&LSTM	28
Hình 36 . Lưu đồ thuật toán chương trình	3
Hình 37 . Máy victim trước khi bị tấn công	6
Hình 38 . Cú pháp tấn công hping3	6
Hình 39 . Máy victim đang bị tấn công DDoS	7
1.1 Hình 40 . Bắt gói tin bằng wireshark	7
Hình 41 . Phát hiện tấn công	7
Hình 42 . Thông báo về mail khi phát hiện tấn công	8

LỜI MỞ ĐẦU

Đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến tất cả thầy cô đã giảng dạy và truyền đạt những kiến thức bổ ích tại học viện Công Nghệ Bưu Chính Viễn Thông khi chúng em học tập tại đây, và đã trang bị đầy đủ kiến thức cũng như kỹ năng để chúng em có thể hoàn thành được bài nghiên cứu khoa học này.

Tiếp theo là em xin dành lời cảm ơn chân thành đến người thầy tâm huyết Ths. Đàm Minh Linh đã hỗ trợ chúng em hết sức trong đồ án lần này, cảm ơn thầy vì những kiến thức mà thầy đã truyền đạt cho chúng em để có thể hoàn thành được bài nghiên cứu lần này. Xin chúc thầy và gia đình có thật nhiều sức khỏe để tiếp tục truyền lửa cho thế hệ sau này.

Dù đã cố gắng và hoàn thành đúng tiến độ nhưng nhóm chúng em vẫn còn nhiều hạn chế và thiếu sót do chưa có nhiều kinh nghiệm. Mong được thầy cô chỉ bảo và đóng góp ý kiến cho nhóm em để khắc phục những điểm yếu của bản thân và trở nên tốt hơn. Một lần nữa nhóm em xin chân thành cảm ơn các thầy cô và các bạn nghiên cứu khác đã đóng góp ý kiến trong bài nghiên cứu lần này.

Nội dung của bài báo cáo bao gồm 3 phần chính:

Chương 1: Cơ Sở Lý Thuyết

Chương 2: Thiết Kế Và Xây Dựng Hệ Thống

Chương 3: Triển khai xây dựng mô hình

Chương 4: Triển khai ứng dụng

Chương 5: Đánh giá kết quả

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu các cuộc tấn công mạng

1.1.1. Giới thiệu

Trong các cuộc tấn công mạng, để khai thác thông tin mục tiêu. Đầu tiên tin tặc sẽ thăm dò hệ thống và tìm kiếm các thông tin có thể khai thác được. Thông qua các kỹ thuật trinh sát khác nhau, tin tặc sẽ cố gắng lấy được cấu hình và sơ đồ mạng của mục tiêu, cách triển khai máy chủ thông tin các port và vulnerability. Kỹ thuật trinh sát một mạng máy tính có thể được phân loại thành 3 nhóm chính, là sử dụng 3 gia thức TCP, UDP và ICMP. Phản hồi nhận từ phía nạn nhân cho các thông tin hữu ích và sau đó sẽ khởi động 1 cuộc tấn công từ thông tin các cổng mở, hoặc các dịch vụ đang chạy, hệ điều hành, vulnerability,...

1.1.2. Các cuộc tấn công mạng phổ biến

Ở đây ta sẽ nghiên cứu về bộ dữ liệu NSL-KDD nên sẽ nói về 4 lớp tấn công phổ biến gồm:

- Dos là kiểu tấn công làm gián đoạn lưu lượng mạng nó được gửi đến hoặc đi từ Attacker. IDS bị tấn công với một lưu lượng lớn làm đó bị ngưng hoạt động lập tức.
- Probe là kiểu tấn công cố gắng thu nhập các thông tin từ một mạng nào đó, sẽ đánh cắp các dữ liệu quan trọng của cá nhân kể cả thông tin tài khoản ngân hàng.
- U2R là kiểu tấn công bắt đầu từ một tài khoản người dùng thông thường và cố gắng chiếm quyền truy cập vào hệ thống hoặc chiếm cả quyền root.
- R2L là kiểu tấn công cố gắng chiếm quyền truy cập vào hệ thống từ một máy tính từ xa.

Ngoài ra còn các kiểu tấn công phổ biến khác như: Tấn công giả mạo (Phishing attack), Tấn công trung gian (Man-in-the-middle attack),...

1.1.3. Phòng chống

Đối với các nhân:

- Hạn chế truy cập wifi công cộng.
- Không sử dụng phần mềm crack.
- Cập nhật phần mềm, hệ điều hành lên phiên bản mới nhất.
- Cảnh trọng khi duyệt email, kiểm tra kỹ tên người gửi.
- Không tải file hoặc nhấp vào được link không rõ nguồn gốc.
- ...

Đối với tổ chức, doanh nghiệp:

- Tuyệt đối nói không với phần mềm crack
- Luôn cập nhật phần mềm, firmware lên phiên bản mới nhất
- Sử dụng các dịch vụ đám mây uy tín cho mục đích lưu trữ.
- Xây dựng một chính sách bảo mật với các điều khoản rõ ràng, minh bạch
- Tổ chức các buổi đào tạo, training kiến thức sử dụng internet an toàn cho nhân viên
- ...

1.2. Giới thiệu mô hình hồi quy Logistics

1.2.1. Ma trận nhầm lẫn

- Ma trận nhầm lẫn (confusion matrix) thường dùng để đánh giá hiệu suất của các thuật toán phân loại trong mô hình học máy.
- Ma trận nhầm lẫn thường là được thành 4 ô, trong đó các hàng sẽ chia ra làm 2 nhãn là nhãn thực tế và nhãn dự đoán. Các ô của ma trận nhầm lẫn sẽ biểu thị cho số lượng mẫu thuộc của mỗi nhóm, và sẽ được phân ra 2 loại là True hoặc False. Nó cho phép đánh giá được hiệu suất của mô hình là tốt hay không tốt.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Hình 1. Minh họa ma trận nhầm lẫn

Ở hình minh họa ta có thể thấy rằng bài toán chúng ta sẽ phân dữ liệu thành 2 lớp là **Positive** và **Negative**. Các ô của ma trận nhầm lẫn nó sẽ biểu thị số lượng mẫu cho

mỗi nhóm, sẽ phân ra là True hoặc False. Ví dụ bây giờ ta có 45 mẫu thuộc lớp Positive mà trong đó có 30 mẫu được phân loại là True và 15 mẫu được phân loại là False và sẽ vào lớp Negative.

- Độ chính xác toàn thể: $(\text{TN} + \text{TP}) / N$
- Sai số toàn thể: $(\text{FP} + \text{FN}) / N$
- Độ nhạy (Sensitivity):
 - + Độ nhạy: $\text{TP} / (\text{TP} + \text{FN})$
 - + Sai số FN: $1 - \text{Độ nhạy} = \text{FN} / (\text{TP} + \text{FN})$
- Độ đặc hiệu (Specificity):
 - + Độ đặc hiệu: $\text{TN} / (\text{TN} + \text{FP})$
 - + Sai số FP: $1 - \text{Độ đặc hiệu} = \text{FP} / (\text{TN} + \text{FP})$

1.2.2. Nguyên lý MLE

MLE là viết tắt của Maximum Likelihood Estimation là 1 dạng kỹ thuật ước lượng tham số thống kê. MLE nó là dạng mô hình xác suất, các tham số nên được ước lượng để sao cho điều kiện dữ liệu được quan sát là lớn nhất.

Nguyên lý MLE là mô hình xác suất và hàm likelihood. Hàm likelihood là hàm biểu thị xác suất của các tham số trong mô hình dựa trên dữ liệu quan sát được.

- Công thức likelihood là: $L(\theta|X) = P(X|\theta)$
- Công thức của hàm likelihood trong mô hình logistic regression:

$$L(w, w_0) = P(D) = \prod_{i=1}^n P(y_i|x_i) = \prod_{i=1}^n \mu_i^{y_i} (1 - \mu_i)^{1-y_i}$$

- Công thức biểu diễn hàm mất mát (loss function) trong mô hình logistic

Regression nhằm giải quyết bài toán tối ưu cho mô hình:

$$\ell(w, w_0) = -\log L(w, w_0) = \sum_{i=1}^n -y_i \log \mu_i - (1 - y_i) \log(1 - \mu_i)$$

1.2.3. Hàm lỗi NLL

$$\ell(\mathbf{W}) = -\log L(\mathbf{W}) = -\sum_{i=1}^n \sum_{c=1}^C y_{ic} \log \mu_{ic}$$

Gọi tên là hàm lỗi **entropy chéo** (cross-entropy loss - CE loss) đo khoảng cách Kullback-Leibler (KL distance) giữa hai phân bố μ_{ic} và y_{ic} (trong các thư viện học máy, tham số *reduce* = "sum", còn nếu dùng "mean" thì sẽ lấy trung bình).

1.2.4. Xuống đồi bằng đạo hàm

Là phương pháp tối ưu hóa được sử dụng để tìm giá trị nhỏ nhất của 1 hàm số. Phương pháp này hoạt động bằng cách lặp đi lặp lại các bước cập nhật giá trị của biến số dựa trên giá trị đạo hàm của hàm số tại điểm đó, với mục tiêu tìm được điểm cực tiểu của hàm số

- Tính đạo hàm của hàm NLL mát đối với hàng thứ c của ma trận trọng số W

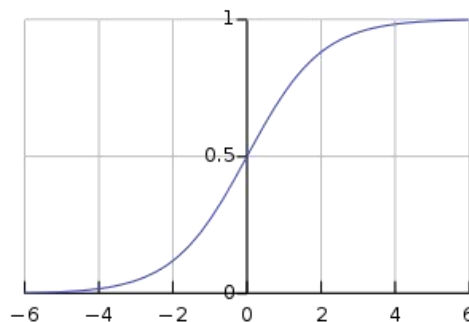
$$\nabla_{w_c} \ell(\mathbf{W}) = \sum_{i=1}^n (\mu_{ic} - y_{ic}) x_i$$

- Tính đạo hàm của hàm NLL đối với hệ số điều chỉnh (bias term) của lớp thứ c .

$$\nabla_{w_{c0}} \ell(\mathbf{W}) = \sum_{i=1}^n (\mu_{ic} - y_{ic})$$

1.2.5. Các thông số hàm sigmoid, softmax

Hàm sigmoid là hàm số có dạng đường cong chữ S sẽ thường được gọi là đường cong sigmoid.

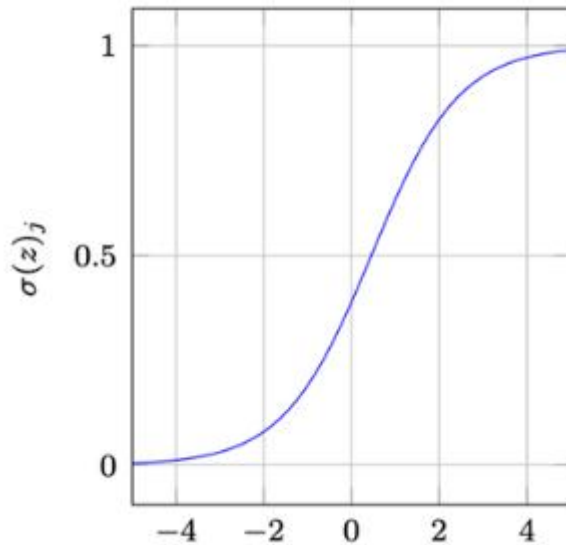


Hình 2. Minh họa Sigmoid

- Công thức hàm sigmoid được thể hiện như sau:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$

Hàm softmax được sử dụng rộng rãi trong mô hình học máy để tính xác suất của các lớp đầu ra trong một bài toán phân loại



Hình 3. Minh họa Softmax

- Công thức hàm softmax được thể hiện như sau:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Thuật toán huấn luyện LR-GD

Train LR-GD $(D, \alpha) \rightarrow w, w_0$

- Ban đầu ta sẽ khởi tạo $w = 0$ thuộc R^d , $w_0 = 0$

- Tiếp theo lặp epoch = 1, 2, ...

+ Tính $l(w, w_0)$

+ Tính đạo hàm $\nabla_w \ell, \nabla_{w_0} \ell$

+ Cập nhật tham số:

$$\begin{aligned} w &\leftarrow w - \lambda \nabla_w \ell(w, w_0) \\ w_0 &\leftarrow w_0 - \lambda \nabla_{w_0} \ell(w, w_0) \end{aligned}$$

- Dừng lại khi:

- + Khi epoch đủ lớn
- + Hàm lỗi giảm không đáng kể
- + Đạo hàm đủ nhỏ $\|\nabla_w \ell\|, \nabla_{w_0} \ell$

1.2.6. Mô hình Logistics phân lớp nhiều lớp

Tập nhãn $\mathcal{Y} = \{1, 2, \dots, C\}$

Phân bố nhóm

Biến ngẫu nhiên $Y \sim \text{Cat}(y|\theta_1, \theta_2, \dots, \theta_C)$ tức là

$$P(Y = c) = \theta_c, c = 1, 2, \dots, C$$

Ví dụ: xúc sắc thì $C = 6, \theta_c = 1/6, \forall c$

$$P(Y = y) = \prod_{c=1}^C \theta_c^{\mathbb{I}(c=y)}$$

Mô hình

Có C lớp \rightarrow có C bộ trọng số $w_c \in \mathbb{R}^d, w_{c0} \in \mathbb{R}, c = 1, 2, \dots, C$, tính hàm tuyến tính

$$f_c(x) = w_c^T x + w_{c0}$$

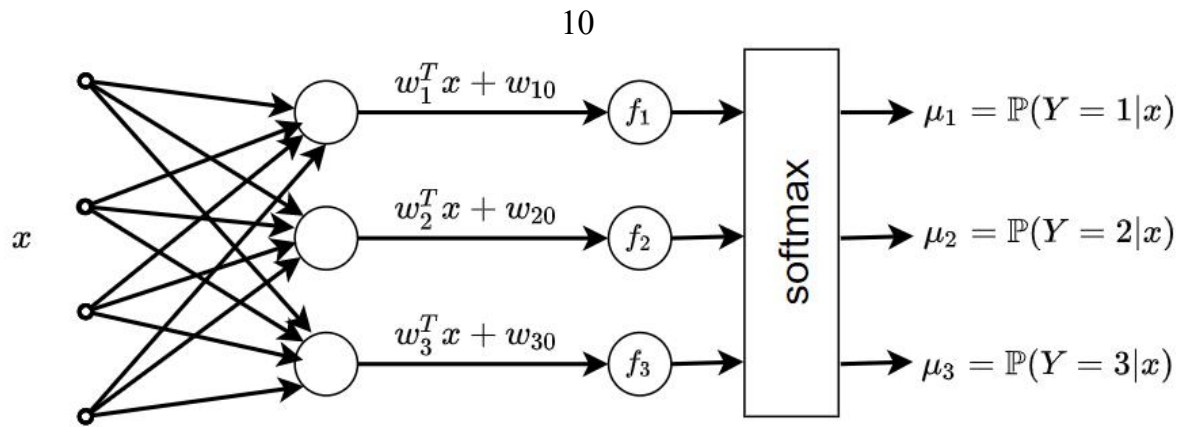
Sử dụng hàm truyền softmax để chuyển thành xác suất

$$\mathcal{S}(z_1, z_2, \dots, z_C) = \left[\frac{e^{z_c}}{\sum_{c'=1}^C e^{z_{c'}}} \right]_{c=1,2,\dots,C}$$

Mô hình xác suất

$$Y|X = x \sim \text{Cat}(y|\mathcal{S}(f_1(x), f_2(x), \dots, f_C(x)))$$

Suy luận: $h^{LR}(x)$: chọn c cực đại hóa $f_c(x) \rightarrow$ cực đại hóa $\mathbb{P}(Y = c|x)$



Hình 4. Mô hình phân lớp nhiều lớp với softmax

1.3. Giới thiệu các mô hình ML/DL

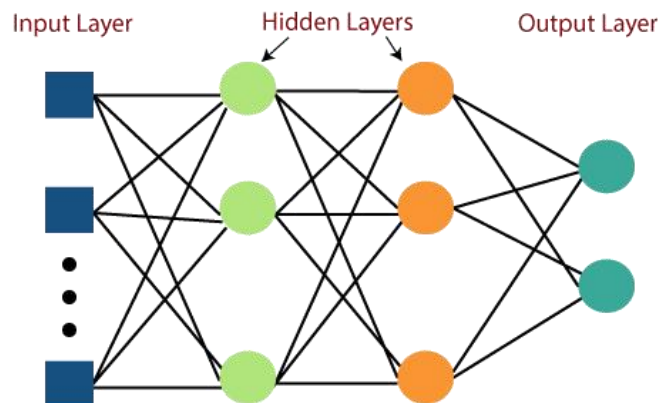
1.3.1. Mô hình MLP

Giới thiệu

Mô hình MLP (Multi-Layer Perceptron) là 1 dạng mạng nơ ron phổ biến tổng ngành Machine Learning và Data Mining. Mô hình này được thiết kế lên nhằm để giải quyết các vấn đề phân loại và dự đoán, bao gồm các bài toán không gian đa chiều, phân loại hình ảnh,... MLP được hình thành bởi nhiều lớp (layer) nơ ron, bao gồm lớp đầu vào (input layer) giữa hai lớp đầu vào và ra.

Huấn luyện

Mô hình MLP được huấn luyện thông qua việc tối ưu hóa và hàm NLL bằng các phương pháp đặc trưng như xuống đồi bằng đạo hàm, và còn được áp dụng cho các bài toán phân loại và dự đoán trên dữ liệu có cấu trúc và không cấu trúc.



Hình 5. Minh họa mô hình MLP

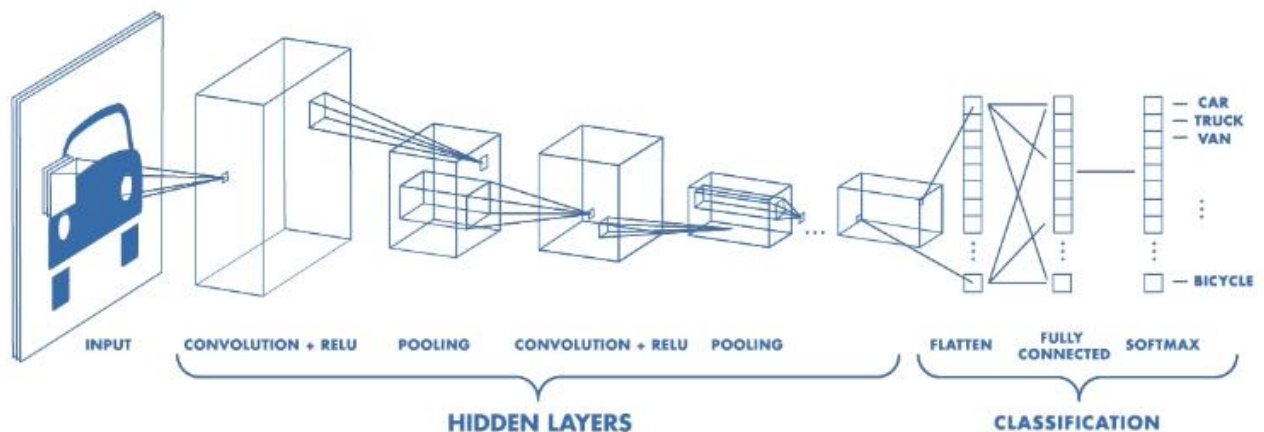
1.3.2. Mô hình CNN

Giới thiệu

Mô hình CNN (Convolutional Neural Network) cũng là một loại mạng nơ-ron phổ biến với lĩnh vực thị giác máy tính và xử lý các hình ảnh. CNN được thiết kế nhằm để có thể giải quyết các vấn đề liên quan đến học sâu và xử lý ảnh gồm phân loại ảnh, nhận dạng khuôn mặt,... Mô hình gồm có các lớp layer đặc biệt cần chú ý ở đây là phần lớp tích chập(convolutional layer) gồm có tích chập 1 chiều(conv1D) và tích chập 2 chiều(conv2D), và lớp gộp(pooling layer). Các lớp này có chức năng lọc và giữ vai trò giảm chiều dữ liệu đầu vào bằng cách áp dụng các bộ lọc filter để tìm ra các đặc trưng của ảnh.

Huấn luyện

Quá trình huấn luyện, các tham số của các bộ lọc được điều chỉnh sao cho các đặc trưng quan trọng của ảnh được trích xuất và đưa vào mô hình để phân loại. CNN còn có các lớp kết nối đầy đủ(fully connected layer) ở cuối để nó kết hợp các đặc trưng đã được trích xuất từ lớp tích chập và lớp gộp, tạo ra dự đoán cho bài toán phân loại.



Hình 6. Minh họa mô hình CNN

1.3.3. Mô hình RNN

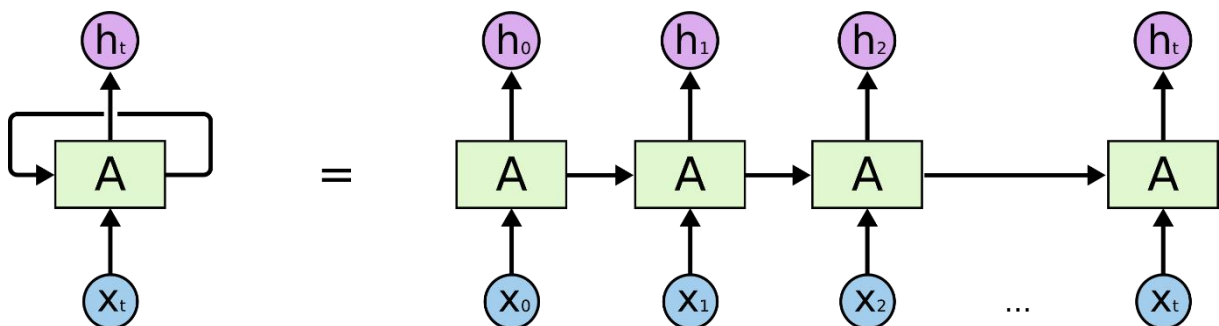
Giới thiệu

Ngữ nghĩa của một từ, câu được tạo thành từ mối liên kết với các từ, câu khác trong văn bản theo một cấu trúc ngữ pháp xác định. Nếu xét từng từ, câu một đứng riêng lẻ sẽ không thể hiểu nội dung ý nghĩa trọn vẹn của từ, câu đó do vậy ta không thể sử

dụng các mô hình truyền thống để giải quyết các bài toán về ngữ nghĩa của một từ, câu mà ta cần các mô hình có cấu trúc đặc biệt hơn để giải quyết các bài toán này. Mô hình RNN (Recurrent Neural Network) là mô hình được sinh ra để giải quyết các bài toán như thế.

Huấn luyện

Quá trình huấn luyện RNN cũng gần như tương tự với các mạng nơ ron thông thường chỉ có vòng lặp ở thân mạng nơ ron là có sự khác biệt. Vòng lặp này là chuỗi sao chép nhiều lần của cùng một kiến trúc nhằm cho phép các thành phần có thể kết nối liền mạch với nhau theo hình chuỗi, đầu ra của vòng lặp trước chính là đầu vào của vòng lặp sau.



Hình 7. Minh họa RNN

1.3.4. Mô hình LSTM

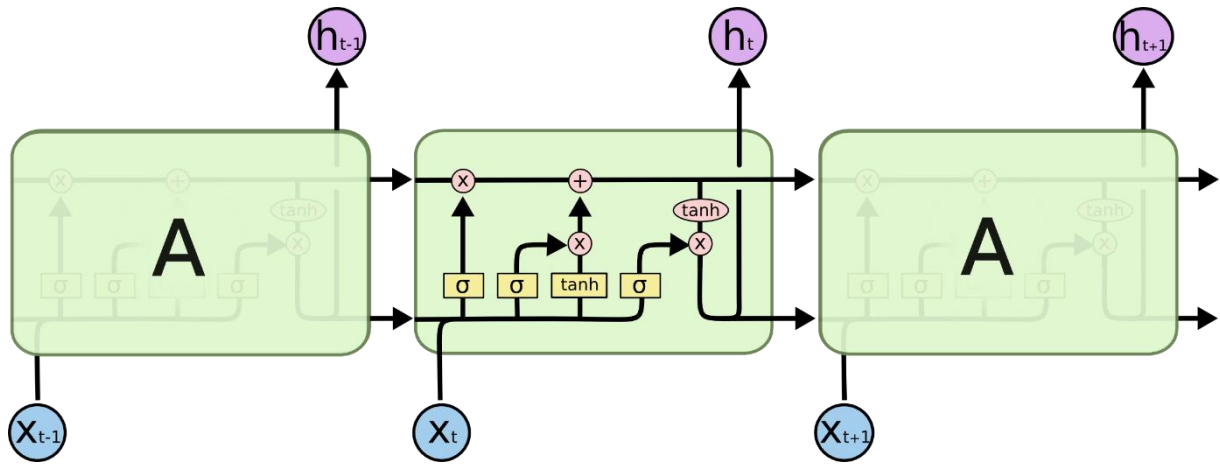
Giới thiệu

Mô hình LSTM (Long Short Term Memory) là một kiến trúc đặc biệt của RNN có khả năng học được sự phụ thuộc trong dài hạn (long-term dependencies). LSTM có các kết nối phản hồi (feedback connection) nghĩa là nó có khả năng xử lý toàn bộ chuỗi dữ liệu, ngoại trừ các điểm dữ liệu đơn lẻ như hình ảnh. LSTM đã khắc phục được rất nhiều những hạn chế của RNN trước đây về triệt tiêu đạo hàm. Hiện nay kiến trúc này được sử dụng phổ biến và rộng rãi như được ứng dụng vào nhận dạng chữ viết tay, nhận dạng hành động của con người ..v.v..

Huấn luyện

Vai trò trung tâm của mô hình LSTM được nắm giữ bởi một ô nhớ được gọi là “trạng thái tế bào (cell state). Trạng thái tế bào là một dạng giống như băng truyền nó chạy xuyên suốt thông qua tất cả các mắc xích, thông tin khi được truyền trên băng truyền

này sẽ không thay đổi trong suốt quá trình. LSTM có khả năng bỏ đi hoặc thêm thông tin cho trạng thái tế bào thông qua 3 cổng mỗi cổng có cấu tạo gồm một lớp mạng sigmoid và một phép nhân.



Hình 8. Minh Họa LSTM

1.4. Giới thiệu bài toán trích xuất đặc trưng

Khái Niệm

Trích suất đặc trưng (feature extraction) là quá trình biến các dữ liệu đầu vào chưa được mã hóa như hình ảnh, âm thanh, văn bản ...v.v. Thành các dữ liệu số để có thể đưa vào và huấn luyện các mô hình học máy.

Một số kỹ thuật trích xuất đặc trưng

+Phương pháp bag-of-words (BoW): là phương pháp mã hóa các từ trong câu thành một vector có độ dài bằng số lượng các từ trong từ điển và đếm tần suất xuất hiện của các từ. Tần suất xuất hiện của từ thứ i trong từ điển sẽ chính bằng phần tử thứ i trong vector.

Bag of Words Example

Document 1

The quick brown
fox jumped over
the lazy dog's
back.

Document 2

Now is the time
for all good men
to come to the
aid of their party.

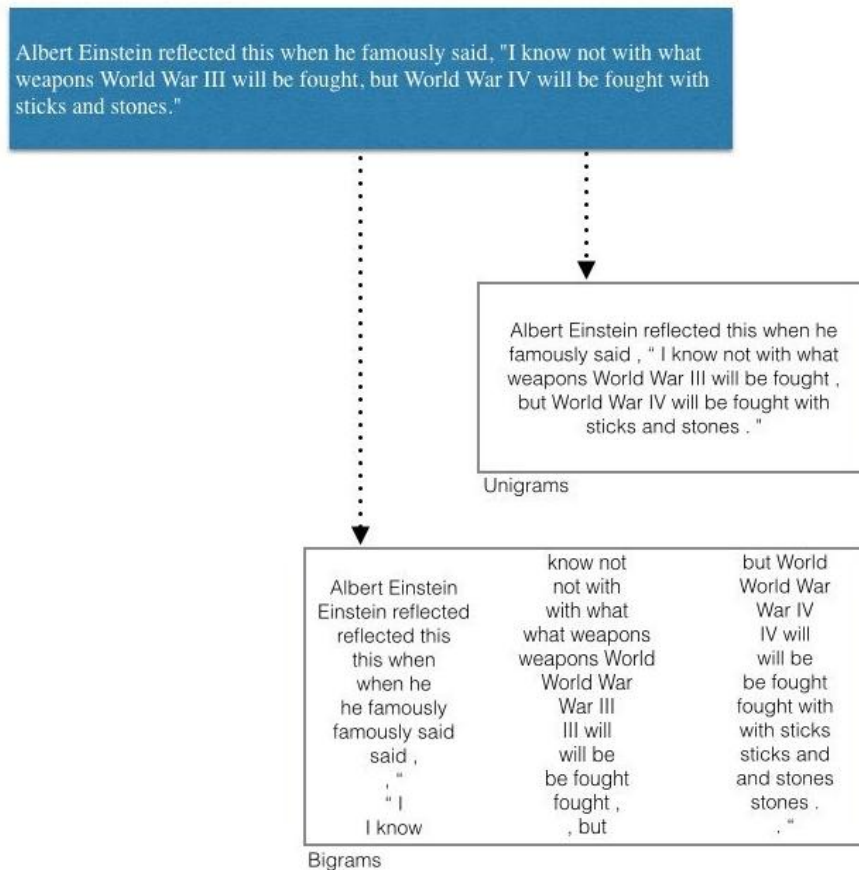
Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Hình 9. Ví dụ về BoW

+Phương pháp bag-of-n-gram: là phương pháp mở rộng của BoW. n-gram là một chuỗi bao gồm n token. Nếu n=1 ta gọi là unigram, n=2 là bigram và n=3 là trigram. Khi thực hiện tokenization với n-grams thì trong từ điển sẽ xuất hiện những cụm n-grams từ nếu chúng xuất hiện trong văn bản.



Hình 10. Ví dụ bag-of-n-gram

+Phương pháp TF-IDF: là phương pháp đánh trọng số cho các từ xuất hiện trong một vài văn bản cụ thể lớn hơn thông qua công thức:

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D; t \in d\}| + 1} = \log \frac{|D|}{\text{df}(d, t) + 1}$$

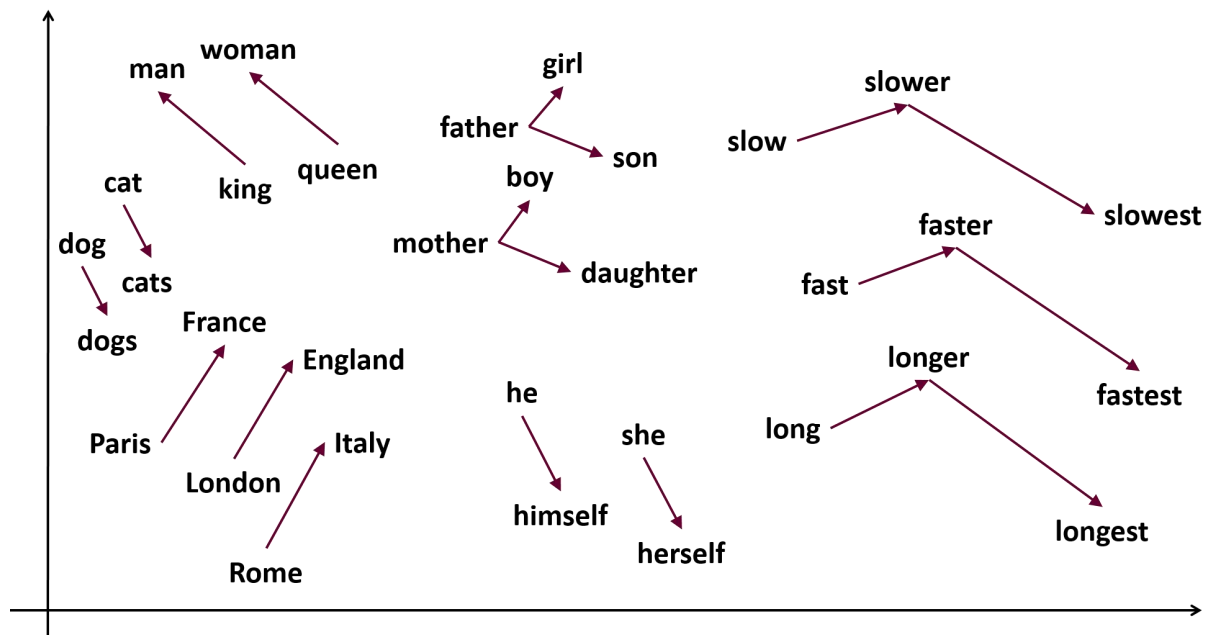
$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

trong đó:

- $|D|$ là số lượng các văn bản trong bộ văn bản.
- $\text{df}(d, t) = |\{d \in D; t \in d\}|$ là tần suất các văn bản $d \in D$ mà từ t xuất hiện.
- $\text{tf}(t, d)$ là tần suất xuất hiện của từ t trong văn bản d .

+Phương pháp Word2vec: là một nhóm các mô hình sử dụng để tạo ra biểu diễn nhúng cho từ. Những mô hình này tương đối nông, chỉ bao gồm những mạng nơ ron 2 lớp được huấn luyện để tái tạo lại bối cảnh ngôn ngữ từ. Thông qua mô hình word2vec

mỗi một từ trong một bộ văn bản được biểu diễn thông qua một véc tơ trong không gian cao chiều, có thể lên tới hàng trăm chiều, sao cho các từ có chung ngữ cảnh sẽ được đặt gần nhau hơn trong không gian.



Hình 11. Ví dụ Word2vec

1.5. Thuật toán KNN

Khái niệm

Thuật toán KNN (K-Nearest Neighbors) là một thuật toán học máy có giám sát được sử dụng trong các bài toán phân loại và dự đoán. Thuật toán này hoạt động dựa trên nguyên tắc các điểm dữ liệu có thuộc tính tương tự nhau thì thường nằm gần nhau trong không gian đặc trưng.

Chi tiết thuật toán

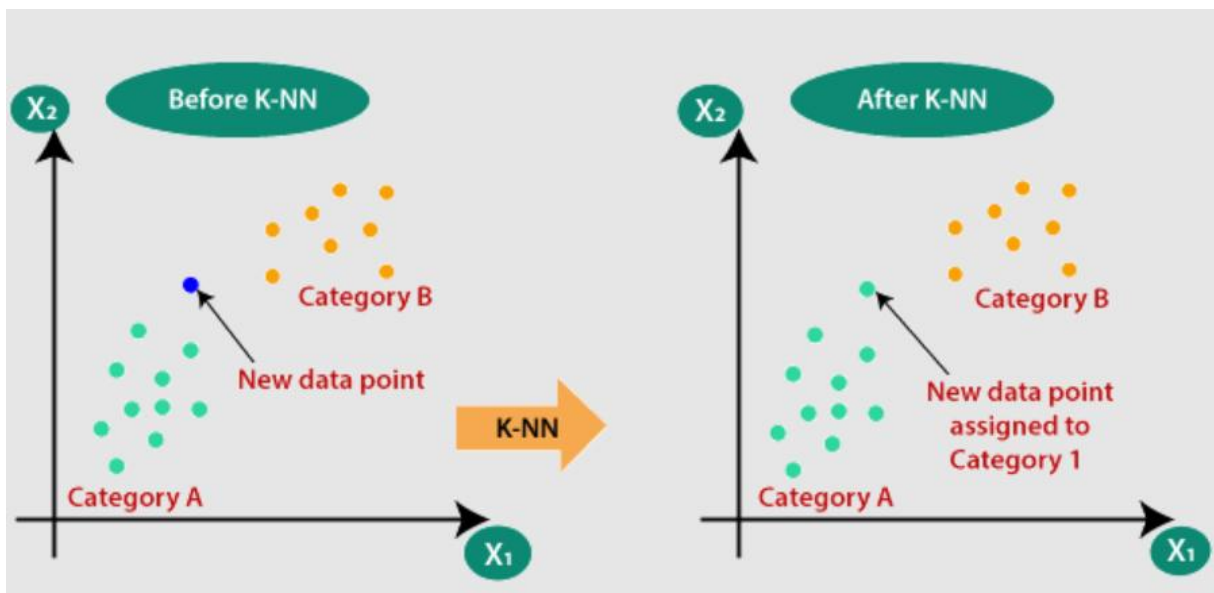
- Chọn số láng giềng K
- Tính khoảng cách giữa điểm dữ liệu cần phân loại hoặc dự đoán với tất cả các điểm dữ liệu trong tập dữ liệu đã biết.

Distance functions

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k x_i - y_i $
Minkowski	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

Hình 12. Công thức khoảng cách

- Chọn K điểm dữ liệu có khoảng cách gần nhất đến điểm dữ liệu cần phân loại hoặc dự đoán.
- Dựa trên nhãn của K điểm dữ liệu gần nhất, quyết định lớp hoặc nhãn của điểm dữ liệu mới. Có thể phân loại bằng cách đếm số lượng điểm dữ liệu thuộc từng lớp trong K lân cận và chọn lớp có số lượng nhiều nhất hoặc dựa trên trọng số khoảng cách giữa các điểm dữ liệu.



Hình 13. Ví dụ KNN

CHƯƠNG 2: THU THẬP DỮ LIỆU & MÔ HÌNH

Trong chương này nhóm sẽ tập trung vào thiết kế và xây dựng dựa trên nghiên cứu đã trình bày trong phần cơ sở lý thuyết của chương 1. Quá trình thiết kế và xây dựng hệ thống sẽ được trình bày chi tiết các bước ở bên dưới.

2.1 Thu thập dữ liệu

- Trước khi đi vào việc xây dựng hệ thống, thì nhóm đã thu thập các dữ liệu từ bộ dữ liệu NSL-KDD, đây là bộ dữ liệu phân loại tấn công mạng. Quá trình thu thập dữ liệu được nhóm tiến hành theo quy chuẩn nên đã đảm bảo được tính đúng đắn và đáng tin cậy của dữ liệu

In [5]: train_data

Out[5]:

ame_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	label	difficulty
0.17	0.00	0.00	0.00	0.05	0.00	normal	20
0.88	0.00	0.00	0.00	0.00	0.00	normal	15
0.00	0.00	1.00	1.00	0.00	0.00	neptune	19
0.03	0.04	0.03	0.01	0.00	0.01	normal	21
0.00	0.00	0.00	0.00	0.00	0.00	normal	21
...
0.00	0.00	1.00	1.00	0.00	0.00	neptune	20
0.01	0.00	0.00	0.00	0.00	0.00	normal	21
0.00	0.00	0.72	0.00	0.01	0.00	normal	18
0.00	0.00	1.00	1.00	0.00	0.00	neptune	20
0.30	0.00	0.00	0.00	0.00	0.00	normal	21

Hình 14. Dữ liệu Dataset NSL-KDD

In [8]: # Number of attack Label
train_data['label'].value_counts()

Out[8]:

normal	67343
neptune	41214
satan	3633
ipsweep	3599
portsweep	2931
smurf	2646
nmap	1493
back	956
teardrop	892
warezclient	890
pod	201
guess_passwd	53
buffer_overflow	30
warezmaster	20
land	18
imap	11
rootkit	10
loadmodule	9
ftp_write	8
multihop	7
phf	4
perl	3
spy	2

Name: label, dtype: int64

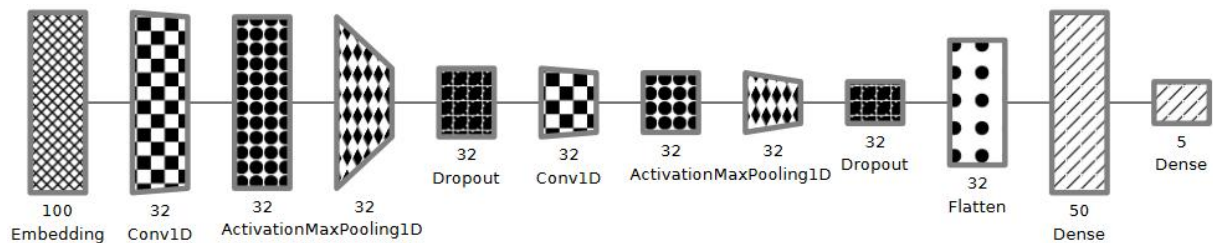
Hình 15. Dữ liệu Dataset NSL-KDD_2

2.2 Xử lý dữ liệu

- Sau khi đã có dữ liệu, nhóm sẽ tiến hành xử lý dữ liệu nhằm chuẩn bị cho quá trình xây dựng và đánh giá độ chính xác của mô hình. Ở phân khúc này sẽ loại bỏ đi các dữ liệu dư thừa không cần thiết, và bổ sung các dữ liệu bị thiếu, mã hóa dữ liệu và chuẩn hóa cho dữ liệu.

2.3 Thiết kế mô hình học sâu

- Dựa trên phần giới thiệu và nghiên cứu ở phần cơ sở lý thuyết, tiến hành thiết kế mô hình học sâu cho bài toán phân loại tấn công mạng, gồm các mô hình như: MLP, CNN, RNN, LSTM khám phá đặc trưng của dữ liệu và phân loại tấn công mạng. Ở đây thì nhóm lựa chọn thuật toán CNN để tiến hành thiết kế mô hình học sâu cho bộ dữ liệu NSL-KDD



Hình 16. Mô hình training CNN

CHƯƠNG 3: TRIỂN KHAI VÀ XÂY DỰNG MÔ HÌNH

3.1 Import các thư viện để sử dụng:

- Trước khi đi vào xây dựng model thì cần import các thư viện cần thiết để tiến hành training mô hình và đưa ra kết quả mà model đạt được.

Import library

```
In [2]: # importing required Libraries
import numpy as np
import pandas as pd
import pickle # saving and loading trained model
from os import path

# importing required Libraries for normalizing data
from sklearn import preprocessing
from sklearn.preprocessing import (StandardScaler, OrdinalEncoder, LabelEncoder, MinMaxScaler, OneHotEncoder)
from sklearn.preprocessing import Normalizer, MaxAbsScaler, RobustScaler, PowerTransformer
from sklearn.metrics import confusion_matrix

# importing library for plotting
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import accuracy_score # for calculating accuracy of model
from sklearn.model_selection import train_test_split # for splitting the dataset for training and testing
from sklearn.metrics import classification_report # for generating a classification report of model
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import itertools
from itertools import cycle

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

import tensorflow as tf
from tensorflow.keras.utils import to_categorical

from keras.layers import Dense, Conv1D, MaxPool1D, Flatten, Dropout, LSTM # importing dense Layer
from keras.models import Sequential #importing Sequential Layer
from keras.layers import Input
from keras.models import Model
# representation of model layers
from keras.utils.vis_utils import plot_model
from sklearn.preprocessing import LabelBinarizer
```

Hình 17. Import thư viện cần thiết

3.2 Mã nguồn

- Ở phần này, nhóm xem xét các mã nguồn có liên quan đến tấn công mạng, nó sẽ có thể nhận biết được các hình thức tấn công mạng trong bộ dataset NSL-KDD

+ Đầu tiên nhóm sẽ định nghĩa các danh sách đặc trưng trong dataset

+ Tiếp theo là định nghĩa các nhãn tấn công trong dataset

+ Cuối cùng nhóm đã viết thêm 1 hàm để chuyển đổi các nhãn tấn công thành các lớp tấn công

```
feature=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot",
        "num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root","num_file_creations","num
        "num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate","srv_se
        "rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_si
        "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst
        "dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate","label","difficulty"]
```

Hình 18. Danh sách đặc trưng trong dataset


```
attack_labels = ['Normal', 'DoS', 'Probe', 'U2R', 'R2L']
```

Hình 19. Các nhãn tấn công

```
def change_label(df):
    df.label.replace(['apache2', 'back', 'land', 'neptune', 'mailbomb', 'pod', 'processtable', 'smurf', 'teardrop', 'udpstorm', 'worm'],
    df.label.replace(['ftp_write', 'guess_passwd', 'httptunnel', 'imap', 'multihop', 'named', 'phf', 'sendmail', 'snmpgetattack', 'snmp
    df.label.replace(['ipsweep', 'mscan', 'nmap', 'portsweep', 'saint', 'satan'], 'Probe', inplace=True)
    df.label.replace(['buffer_overflow', 'loadmodule', 'perl', 'ps', 'rootkit', 'sqlattack', 'xterm'], 'U2R', inplace=True)
```

Hình 20. Chuyển đổi các nhãn thành các lớp

3.3 Tạo DataFrame với multiclass labels

- Ở đây nhóm sẽ tạo DataFrame giúp sao chép lại dữ liệu của train_data nó giúp đảm bảo không bị thay đổi dữ liệu gốc trong quá trình xử lý.

```
# creating a dataframe with multi-class labels (Dos,Probe,R2L,U2R,normal)
multi_data = train_data.copy()
multi_label = pd.DataFrame(multi_data.label)
```

Hình 21. Tạo DataFrame

3.4 Chuẩn hóa dữ liệu

- Nhóm sẽ sử dụng StandardScaler nhằm chuẩn hóa dữ liệu đặc trưng dựa trên các giá trị trung bình và độ lệch chuẩn, và nó còn giúp cải thiện hiệu suất của mô hình.

```
std_scaler = StandardScaler()
def standardization(df,col):
    for i in col:
        arr = df[i]
        arr = np.array(arr)
        df[i] = std_scaler.fit_transform(arr.reshape(len(arr),1))
    return df

numeric_col = multi_data.select_dtypes(include='number').columns
data = standardization(multi_data,numeric_col)
```

Hình 22. Chuẩn hóa dữ liệu

3.5 Mã hóa các nhãn

- Nhóm đã sử dụng LabelEncoder từ thư viện có sẵn nhằm mục đích ở đây là chuyển đổi các nhãn tấn công thành các giá trị tương ứng kiểu số như (0, 1, 2, 3, 4).

```
le2 = preprocessing.LabelEncoder()
enc_label = multi_label.apply(le2.fit_transform)
multi_data['intrusion'] = enc_label
#y_mul = multi_data['intrusion']
multi_data
```

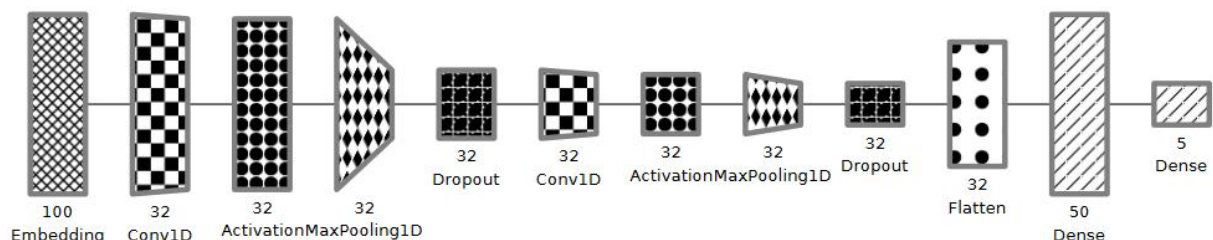
Hình 23. Mã hóa các nhãn

3.6 Tiến hành thiết kế mô hình

- Mô hình thiết kế của nhóm sẽ gồm 7 tầng:
- + Đầu tiên sẽ là tầng Conv1D với tầng này sẽ gồm có 32 bộ lọc, kích thước sẽ là 3, chế độ “same” cho việc padding và phải đảm bảo rằng chiều ra sẽ có kích thước độ dài với đầu vào, và có thêm hàm kích ReLU
- + Tiếp theo sẽ thêm tầng MaxPooling1D sẽ có pool_size là 4 nhằm giảm kích thước đầu ra
- + Rồi tiếp theo thêm tầng Dropout nhằm giúp ngăn chặn trường hợp bị overfitting
- + Sau đó lại tiếp tục thêm tầng Dropout và Conv1D tương tự như trên
- + Tiếp theo là tầng Flatten nó sẽ giúp chuyển đổi dữ liệu từ dạng ma trận sang vector tuyến tính.
- + Tiếp theo đó là tầng Dense(fully connected) với 50 neurons.
- + Cuối cùng là tầng Dense(output) Ở tầng này sẽ có 5 neurons nó tương đương với 5 dạng tấn công khác nhau trong dataset là Normal, DoS, Probe, U2R, R2L. Ở đây nhóm dùng hàm kích hoạt softmax để biểu thị lên xác suất của mỗi lớp.

```
model = Sequential() # initializing model
# # input layer and first layer with 50 neurons
model.add(Conv1D(32, 3, padding="same", input_shape = (X_train.shape[1], 1), activation='relu'))
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Conv1D(32, 3, padding="same", activation='relu'))
model.add(MaxPool1D(pool_size=(4)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=50))
# output layer with softmax activation
model.add(Dense(units=5, activation='softmax'))
```

Hình 24. Thiết kế model CNN



Hình 25. Model CNN dạng biểu mẫu

Tóm tắt mô hình kiến trúc của nhóm:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 123, 32)	128
max_pooling1d (MaxPooling1D)	(None, 30, 32)	0
dropout (Dropout)	(None, 30, 32)	0
conv1d_1 (Conv1D)	(None, 30, 32)	3104
max_pooling1d_1 (MaxPooling1D)	(None, 7, 32)	0
dropout_1 (Dropout)	(None, 7, 32)	0
flatten (Flatten)	(None, 224)	0
dense (Dense)	(None, 50)	11250
dense_1 (Dense)	(None, 5)	255

=====
 Total params: 14,737
 Trainable params: 14,737
 Non-trainable params: 0

Hình 26. Tóm tắt mô hình kiến trúc CNN

3.7 Thiết lập quá trình huấn luyện

- Ở đây nhóm sẽ compile mô hình với các hàm mất mát “categorical_crossentropy”, hàm tối ưu hóa “adam”, và đánh giá accuracy

```
# Compiling the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Hình 27. Compile mô hình

3.8 Training đánh giá độ chính xác model CNN

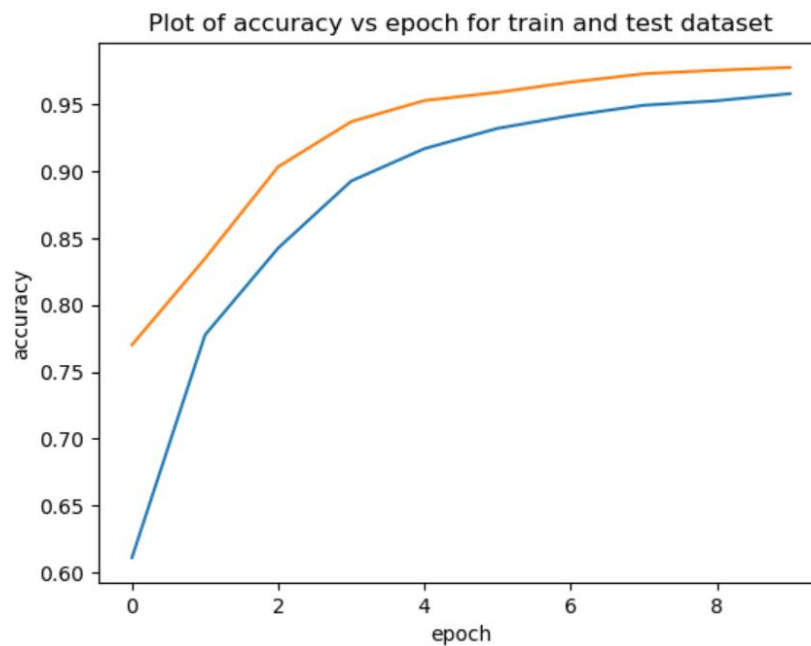
- Ở đây nhóm sẽ training mô phỏng tầm 10 epochs để xem thử độ chính xác của thuật toán, và độ chính xác của thuật toán khá tốt đạt tới tầm gần 99%

```
# training the model on training dataset
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

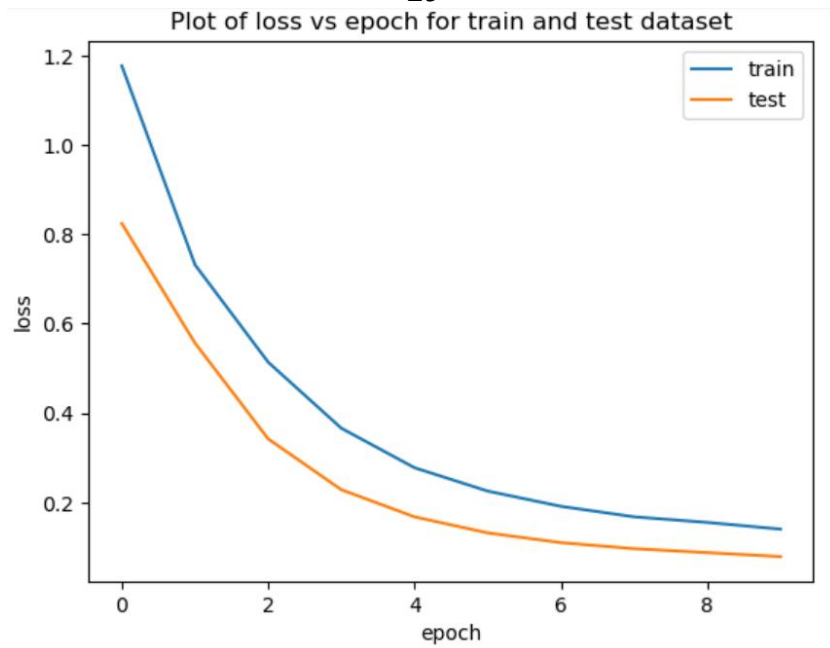
Epoch 1/10
2520/2520 [=====] - 10s 4ms/step - loss: 0.1365 - accuracy: 0.9589 - val_loss: 0.0347 - val_accuracy: 0.9883
Epoch 2/10
2520/2520 [=====] - 9s 4ms/step - loss: 0.0617 - accuracy: 0.9812 - val_loss: 0.0301 - val_accuracy: 0.9917
Epoch 3/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0511 - accuracy: 0.9844 - val_loss: 0.0299 - val_accuracy: 0.9901
Epoch 4/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0441 - accuracy: 0.9867 - val_loss: 0.0255 - val_accuracy: 0.9923
Epoch 5/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0402 - accuracy: 0.9880 - val_loss: 0.0213 - val_accuracy: 0.9924
Epoch 6/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0368 - accuracy: 0.9887 - val_loss: 0.0212 - val_accuracy: 0.9939
Epoch 7/10
2520/2520 [=====] - 9s 4ms/step - loss: 0.0368 - accuracy: 0.9883 - val_loss: 0.0237 - val_accuracy: 0.9937
Epoch 8/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0333 - accuracy: 0.9893 - val_loss: 0.0209 - val_accuracy: 0.9945
Epoch 9/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0352 - accuracy: 0.9895 - val_loss: 0.0178 - val_accuracy: 0.9945
Epoch 10/10
2520/2520 [=====] - 8s 3ms/step - loss: 0.0328 - accuracy: 0.9901 - val_loss: 0.0187 - val_accuracy: 0.9932

Hình 28. Training và đánh giá model

3.9 Nhận xét

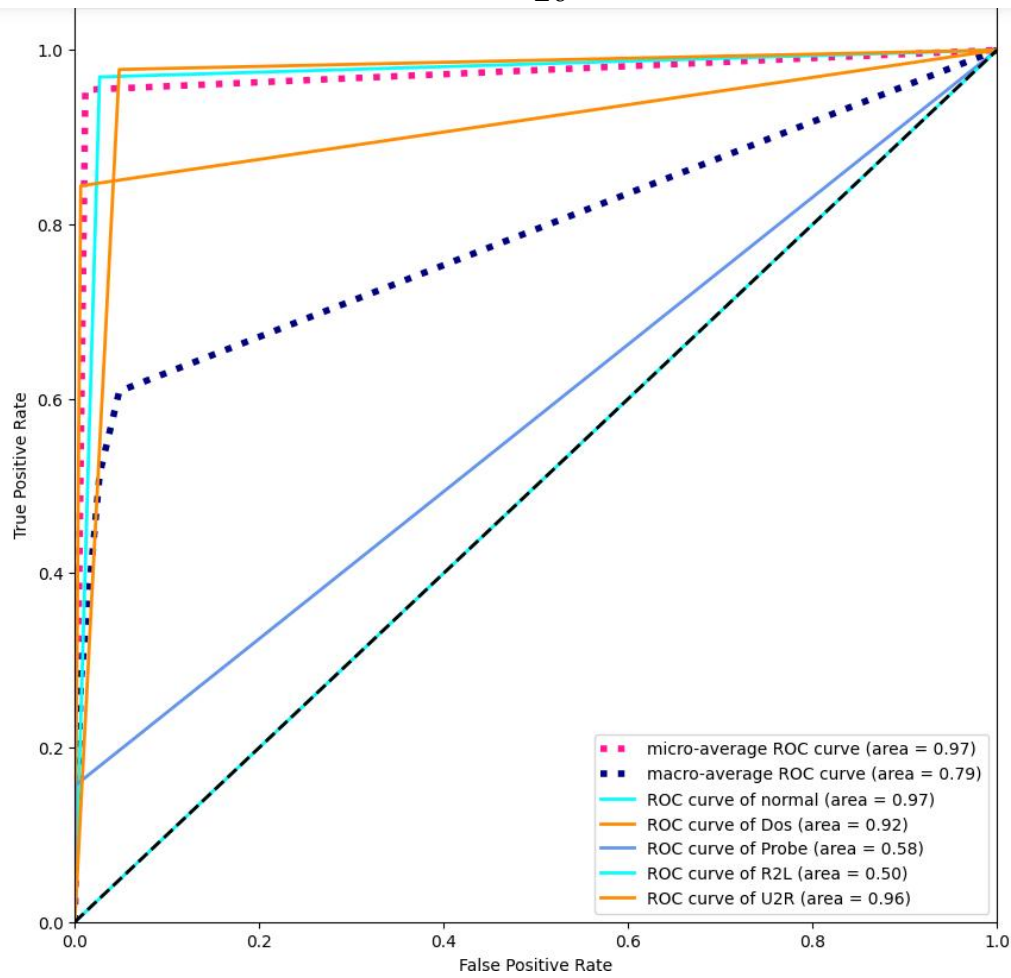


Hình 29. Biểu đồ về accuracy



Hình 30. Biểu đồ loss

- Tổng quan nhìn được rằng mặc dù chỉ vs training có tầm 10 epochs nhưng độ chính xác mà mô hình chúng tôi mang lại được tầm hơn 97% gần 98%=> Đây là 1 model rất tốt.
- Đánh giá hiệu suất mô hình phân loại dựa trên đường cong ROC và AUC



Hình 31. AUC

3.10 Training đánh giá độ chính xác model MLP

- Tóm tắt mô hình kiến trúc của nhóm:

```
# Printing the summary of the model
mlp_model.summary()
```

Model: "sequential_13"

Layer (type)	Output Shape	Param #
dense_38 (Dense)	(None, 50)	6200
dropout_12 (Dropout)	(None, 50)	0
dense_39 (Dense)	(None, 50)	2550
dense_40 (Dense)	(None, 5)	255
Total params: 9,005		
Trainable params: 9,005		
Non-trainable params: 0		

Hình 32. Tóm tắt mô hình kiến trúc MLP

- Ở đây nhóm sẽ training mô phỏng tầm 10 epochs để xem thử độ chính xác của thuật toán, và độ chính xác đạt 99%

```
# training the model on training dataset
history = mlp_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

Epoch 1/10
2520/2520 [=====] - 4s 1ms/step - loss: 0.1471 - accuracy: 0.9547 - val_loss: 0.0383 - val_accuracy: 0.9913
Epoch 2/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0628 - accuracy: 0.9813 - val_loss: 0.0279 - val_accuracy: 0.9948
Epoch 3/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0509 - accuracy: 0.9844 - val_loss: 0.0222 - val_accuracy: 0.9952
Epoch 4/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0530 - accuracy: 0.9853 - val_loss: 0.0245 - val_accuracy: 0.9957
Epoch 5/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0420 - accuracy: 0.9860 - val_loss: 0.0222 - val_accuracy: 0.9928
Epoch 6/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0421 - accuracy: 0.9864 - val_loss: 0.0197 - val_accuracy: 0.9958
Epoch 7/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0392 - accuracy: 0.9878 - val_loss: 0.0160 - val_accuracy: 0.9960
Epoch 8/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0343 - accuracy: 0.9882 - val_loss: 0.0176 - val_accuracy: 0.9960
Epoch 9/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0361 - accuracy: 0.9881 - val_loss: 0.0236 - val_accuracy: 0.9959
Epoch 10/10
2520/2520 [=====] - 3s 1ms/step - loss: 0.0334 - accuracy: 0.9889 - val_loss: 0.0190 - val_accuracy: 0.9959
```

Hình 33. Training đánh giá model MLP

3.11 Training đánh giá độ chính xác model LSTM-RNN

- Tóm tắt mô hình kiến trúc của nhóm:

```
# Printing the summary of the model
lstm_rnn_model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10400
dense_3 (Dense)	(None, 50)	2550
dense_4 (Dense)	(None, 5)	255
Total params: 13,205		
Trainable params: 13,205		
Non-trainable params: 0		

Hình 34. Tóm tắt mô hình kiến trúc LSTM&RNN

- Ở đây nhóm sẽ training mô phỏng tầm 10 epochs để xem thử độ chính xác của thuật toán, và độ chính xác đạt 99%


```
# training the model on training dataset
history = lstm_rnn_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

Epoch 1/10
2520/2520 [=====] - 79s 31ms/step - loss: 0.4109 - accuracy: 0.8463 - val_loss:
racy: 0.9062
Epoch 2/10
2520/2520 [=====] - 80s 32ms/step - loss: 0.2434 - accuracy: 0.9114 - val_loss:
racy: 0.8909
Epoch 3/10
2520/2520 [=====] - 79s 31ms/step - loss: 0.1733 - accuracy: 0.9393 - val_loss:
racy: 0.9415
Epoch 4/10
2520/2520 [=====] - 77s 30ms/step - loss: 0.1470 - accuracy: 0.9508 - val_loss:
racy: 0.9340
Epoch 5/10
2520/2520 [=====] - 78s 31ms/step - loss: 0.1111 - accuracy: 0.9631 - val_loss:
racy: 0.9771
Epoch 6/10
2520/2520 [=====] - 77s 31ms/step - loss: 0.0877 - accuracy: 0.9730 - val_loss:
racy: 0.9644
Epoch 7/10
2520/2520 [=====] - 78s 31ms/step - loss: 0.0789 - accuracy: 0.9760 - val_loss:
racy: 0.9838
Epoch 8/10
2520/2520 [=====] - 78s 31ms/step - loss: 0.0612 - accuracy: 0.9815 - val_loss:
racy: 0.9802
Epoch 9/10
2520/2520 [=====] - 78s 31ms/step - loss: 0.0532 - accuracy: 0.9840 - val_loss:
racy: 0.9892
Epoch 10/10
2520/2520 [=====] - 78s 31ms/step - loss: 0.0506 - accuracy: 0.9850 - val_loss:
racy: 0.9918
```

Hình 35. Training và đánh giá model RNN&LSTM

3.9 Kết luận:

- So sánh về kích thước thông số của các mô hình, chúng ta thấy rằng mô hình MLP có số lượng tham số nhỏ hơn so với CNN và LSTM-RNN. Do đó, quá trình phân tích dữ liệu của nó có thể không hiệu quả bằng CNN và LSTM-RNN.
- Tuy nhiên, mô hình LSTM-RNN có thể gặp vấn đề về tốc độ huấn luyện khi sử dụng một số epoch cao, điều này có thể dẫn đến việc mất nhiều thời gian và tiềm ẩn nguy cơ overfitting.
- Trong khi đó, mô hình CNN có một số ưu điểm quan trọng:
 - + Trước hết, nó phù hợp với việc xử lý dữ liệu phức tạp. Dữ liệu từ bộ NSL-KDD có thể chứa các mẫu phức tạp và mối quan hệ không tuyến tính. Mô hình CNN thường có khả năng học các mẫu không gian và phức tạp hơn, đặc biệt là khi dữ liệu có cấu trúc không gian. Điều này giúp mô hình CNN trích xuất các đặc trưng quan trọng hơn từ dữ liệu mạng.
 - + Ngoài ra, sự đa dạng của tập dữ liệu NSL-KDD cũng là một yếu tố quan trọng. Nếu tập dữ liệu chứa nhiều loại tấn công khác nhau hoặc biến thể của chúng, mô hình CNN có khả năng tổng quát hóa tốt hơn, đặc biệt là khi bạn sử dụng các lớp tích chập để trích xuất đặc trưng không gian từ dữ liệu.

+ Ngoài ra, mô hình CNN được sử dụng trong nhiều ứng dụng thực tế, chẳng hạn như phát hiện xâm nhập trong mạng. Điều này làm cho việc sử dụng mô hình CNN trở nên hợp lý và tích hợp vào hệ thống mạng.

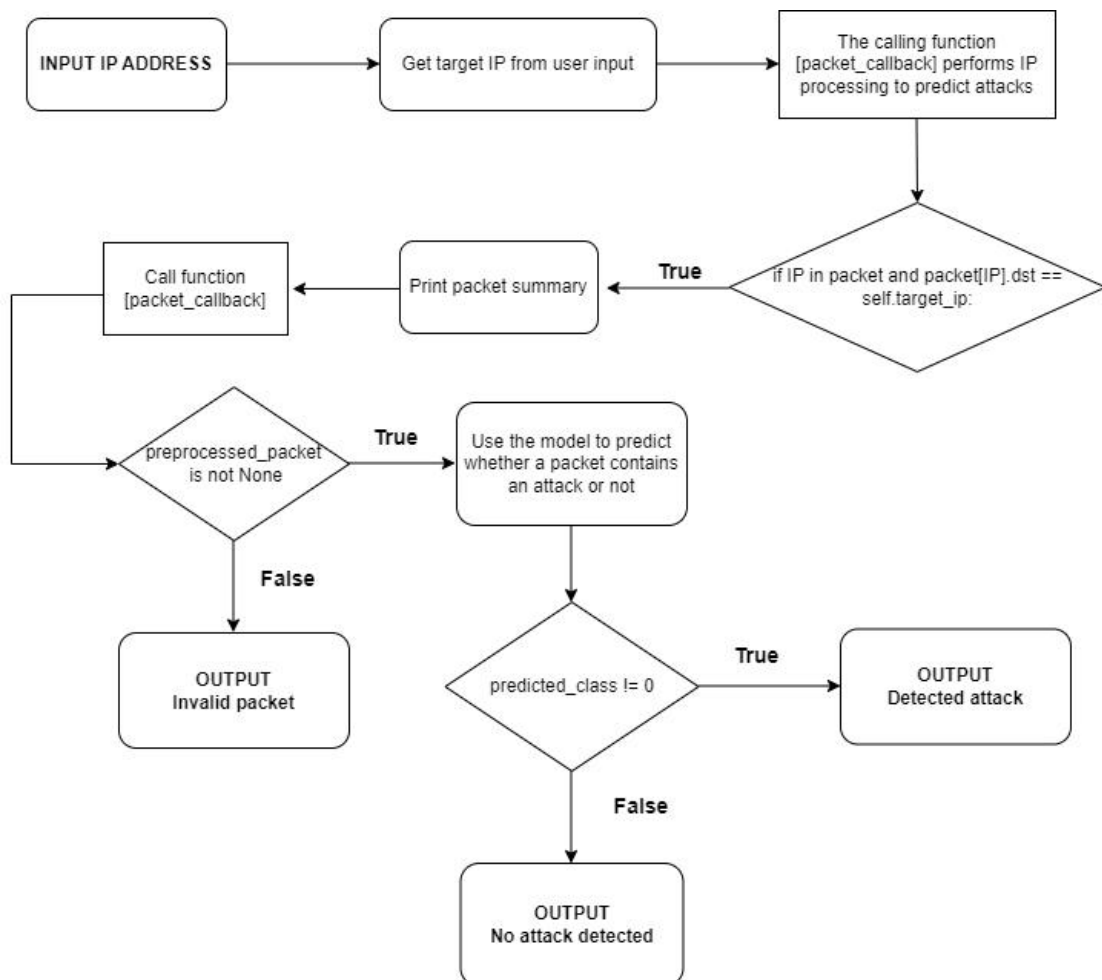
+ Cuối cùng, khả năng mở rộng của mô hình CNN cũng là một điểm mạnh. Bạn có thể dễ dàng mở rộng mô hình để xử lý dữ liệu lớn hơn hoặc thêm các lớp tích chập và kết nối, điều này có ích khi bạn có kế hoạch mở rộng dự án trong tương lai.

=> Dựa trên những điểm này, nhóm quyết định lựa chọn mô hình CNN để thực hiện dự đoán về các cuộc tấn công.

CHƯƠNG 4: TRIỂN KHAI ỨNG DỤNG

4.1 Xây dựng một chương trình phát hiện tấn công mạng dựa trên model đã training

- Mô hình phát hiện tấn công mạng dựa trên CNN hoạt động bằng cách theo dõi các lưu lượng gói tin gửi đến máy tính. Cụ thể, chương trình sẽ sử dụng thư viện socket để lấy IP của máy tính và sau đó sử dụng thư viện scapy để bắt các gói tin. Các gói tin này sau đó sẽ được phân tích bởi mô hình CNN đã được đào tạo trước đó. Mô hình sẽ đưa ra dự đoán dựa trên các đặc trưng của các gói tin. Nếu dự đoán cho thấy máy tính đang bị tấn công, mô hình sẽ gửi dialog ra màn hình là máy đang bị tấn công, và chương trình còn dùng thư viện smtplib để phòng ngừa khi victim không ngồi trực tiếp trên máy thì nó sẽ gửi mail thông báo.

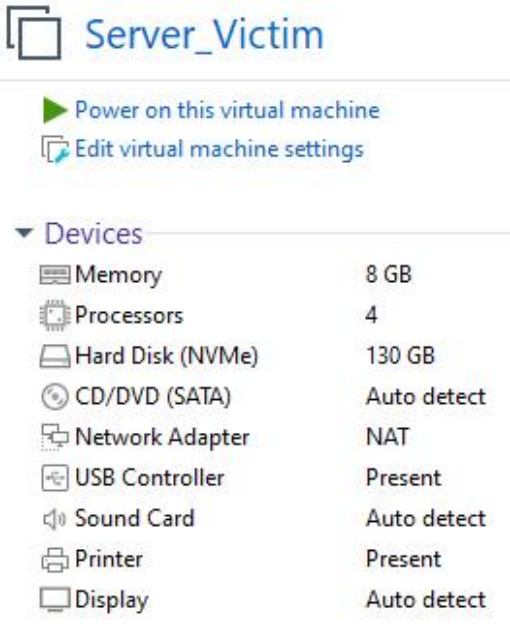


Hình 36. Lưu đồ thuật toán chương trình

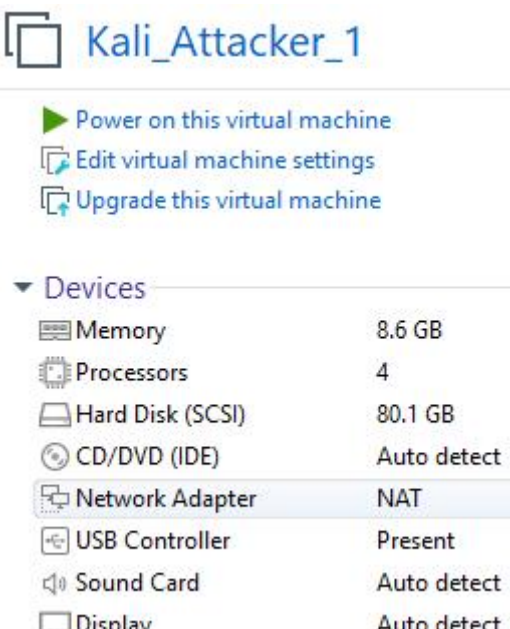
CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ

5.1. Chuẩn bị máy Attacker và Victim

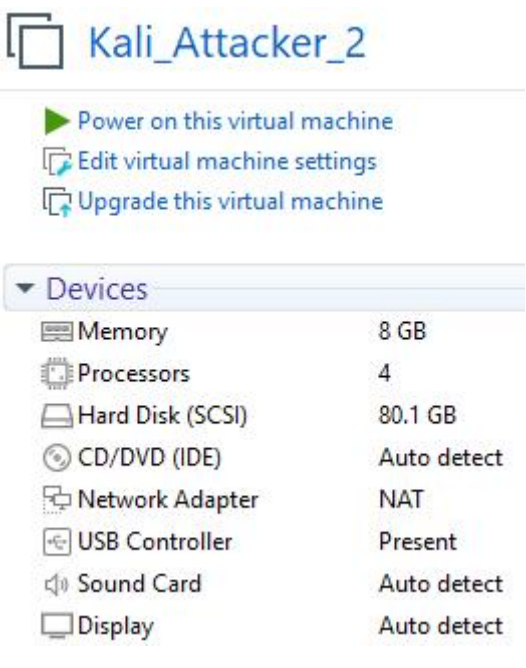
- Máy Victim (Windows 10 pro)

 <p>The screenshot shows the 'Server_Victim' virtual machine settings. It includes buttons for 'Power on this virtual machine' and 'Edit virtual machine settings'. Under the 'Devices' section, the following configurations are listed:</p> <ul style="list-style-type: none"> Memory: 8 GB Processors: 4 Hard Disk (NVMe): 130 GB CD/DVD (SATA): Auto detect Network Adapter: NAT USB Controller: Present Sound Card: Auto detect Printer: Present Display: Auto detect 	<ul style="list-style-type: none"> - IP : 192.168.81.128 - Gateway: 192.168.81.2 - RAM: 8GB - Processors: 4 - Network Adapter: NAT
---	---

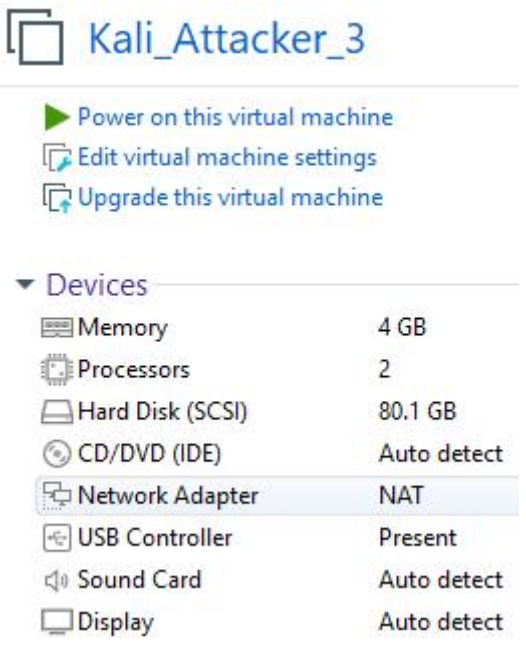
- Máy Kali_Attacker_1:

 <p>The screenshot shows the 'Kali_Attacker_1' virtual machine settings. It includes buttons for 'Power on this virtual machine', 'Edit virtual machine settings', and 'Upgrade this virtual machine'. Under the 'Devices' section, the following configurations are listed:</p> <ul style="list-style-type: none"> Memory: 8.6 GB Processors: 4 Hard Disk (SCSI): 80.1 GB CD/DVD (IDE): Auto detect Network Adapter: NAT USB Controller: Present Sound Card: Auto detect Display: Auto detect 	<ul style="list-style-type: none"> - IP: 192.168.81.129 - Gateway: 192.168.81.2 - RAM: 8GB - Processors: 4 - Network Adapter: NAT
---	--

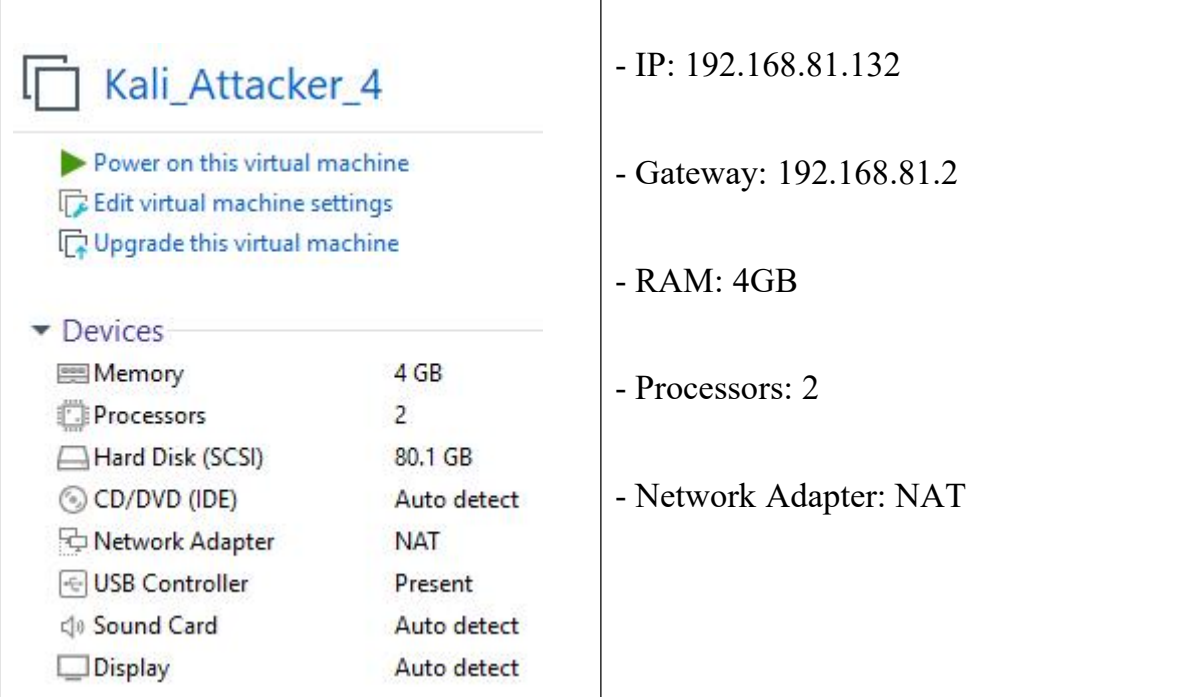
- Máý Kali_Attacker_2:

	<ul style="list-style-type: none"> - IP: 192.168.81.130 - Gateway: 192.168.81.2 - RAM: 8GB - Processors: 4 - Network Adapter: NAT
---	--

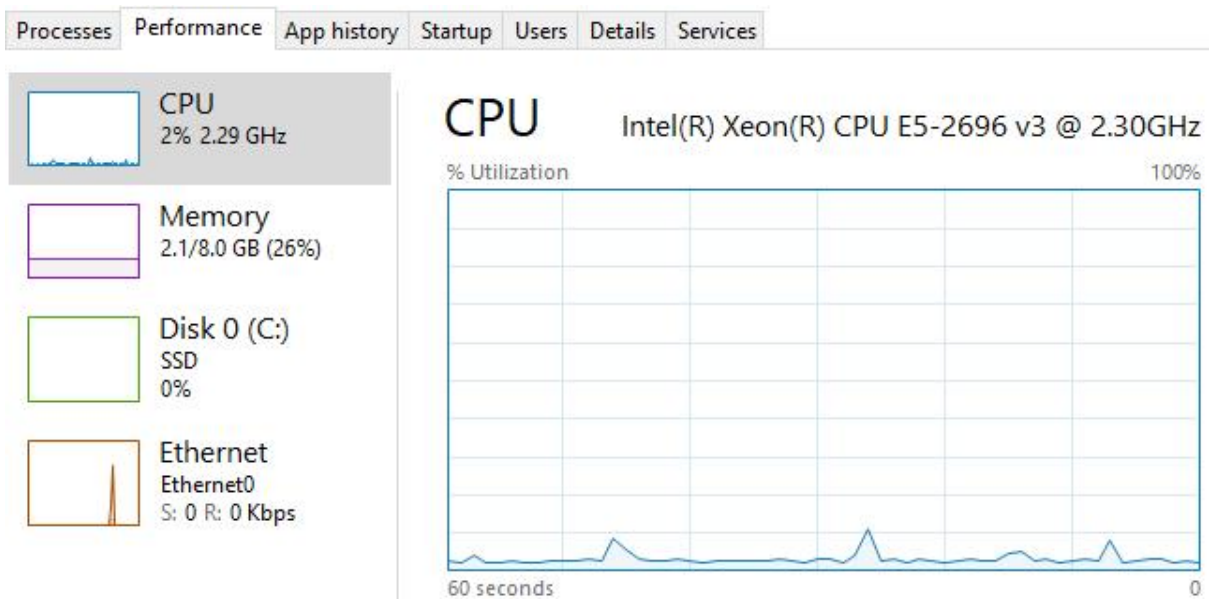
- Máý Kali_Attacker_3:

	<ul style="list-style-type: none"> - IP: 192.168.81.131 - Gateway: 192.168.81.2 - RAM: 4GB - Processors: 2 - Network Adapter: NAT
---	--

- Máy Kali_Attacker_4:

	<p>- IP: 192.168.81.132</p> <p>- Gateway: 192.168.81.2</p> <p>- RAM: 4GB</p> <p>- Processors: 2</p> <p>- Network Adapter: NAT</p>
--	---

5.2. Tiến hành quá trình kiểm thử

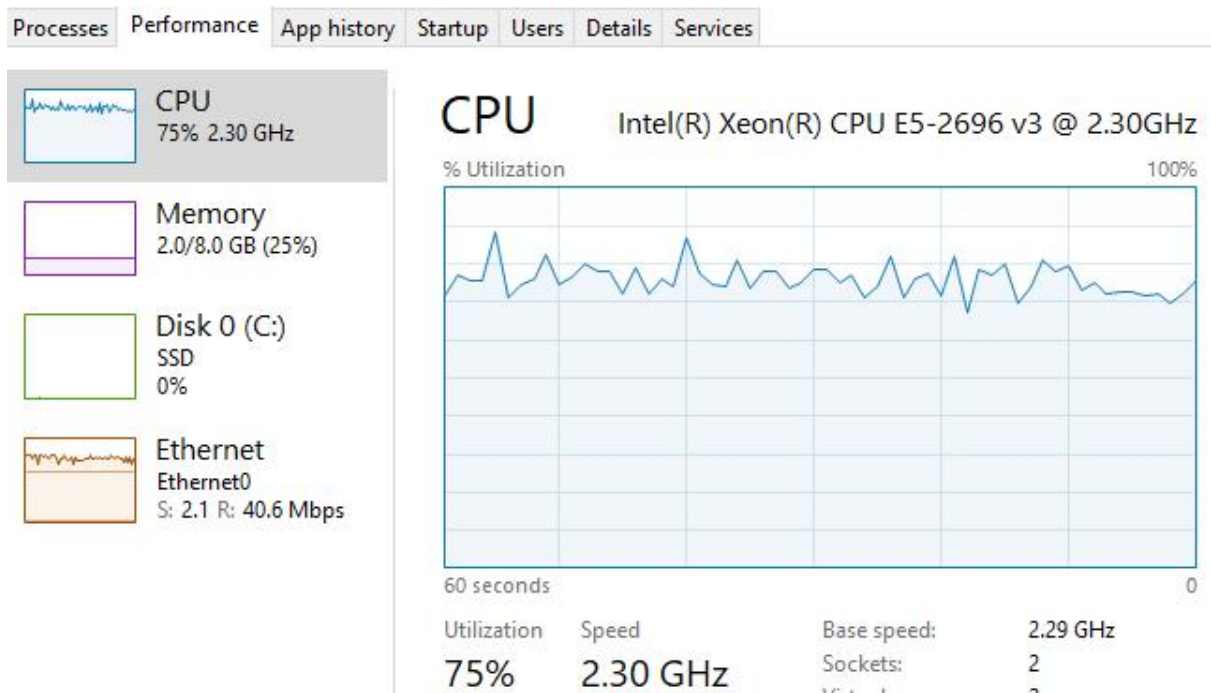


Hình 37. Máy victim trước khi bị tấn công

- Bắt đầu tiến hành dùng 4 máy Kali tấn công DDoS vào máy victim

```
(root@kali)-[/home/kali]
# hping3 192.168.81.128 --flood
HPING 192.168.81.128 (eth0 192.168.81.128): NO FLAGS are set, 40 headers
hping in flood mode, no replies will be shown
```

Hình 38. Cú pháp tấn công hping3



Hình 39. Máy victim đang bị tấn công DDoS

Dùng Wireshark tiến hành bắt các gói tin từ các máy Kali tấn công vào máy victim

2179695	24.043599	192.168.81.128	192.168.81.131	TCP	54	0 → 65431	[RST, ACK]	Seq=1	Ack=3979890588	Win=0
2179696	24.043609	192.168.81.128	192.168.81.132	TCP	54	0 → 61889	[RST, ACK]	Seq=1	Ack=4181030850	Win=0
2179697	24.043612	192.168.81.129	192.168.81.128	TCP	60	30437 → 0	[<None>]	Seq=3988543812	Win=512	Len=0
2179698	24.043612	192.168.81.130	192.168.81.128	TCP	60	10098 → 0	[<None>]	Seq=4028426433	Win=512	Len=0
2179699	24.043618	192.168.81.129	192.168.81.128	TCP	60	30438 → 0	[<None>]	Seq=3752902996	Win=512	Len=0
2179700	24.043619	192.168.81.130	192.168.81.128	TCP	60	10099 → 0	[<None>]	Seq=3541269965	Win=512	Len=0
2179701	24.043654	192.168.81.132	192.168.81.128	TCP	60	61970 → 0	[<None>]	Seq=836249649	Win=512	Len=0
2179702	24.043655	192.168.81.131	192.168.81.128	TCP	60	65508 → 0	[<None>]	Seq=3437524248	Win=512	Len=0

1.1 Hình 40. Bắt gói tin bằng wireshark

Chương trình của máy victim đã bắt các gói tin tấn công từ 4 máy kali và đưa ra dialog thông báo



Hình 41. Phát hiện tấn công

```

Ether / IP / TCP 192.168.81.2:37134 > 192.168.81.128:epmap S / Padding
1/1 [=====] - 0s 37ms/step
Dos
Ether / IP / TCP 192.168.81.2:37135 > 192.168.81.128:epmap S / Padding
1/1 [=====] - 0s 58ms/step
Dos

```



Hình 42. Thông báo về mail khi phát hiện tấn công

Kết luận:

Model training đã dự đoán và phát hiện tấn công từ các gói tin vô cùng chính xác. Ngoài ra chương trình có thể chạy realtime để bắt được cái gói tấn công và hiển thị thông báo giúp cho máy victim có thể nhận biết được tấn công mọi lúc khi khởi chạy chương trình lên và còn gửi mail phòng trường hợp victim không ở trên máy.

Hướng phát triển thêm sau này: Không những thông báo dạng tấn công mà có thể giúp tích hợp thêm các tính năng bảo mật, kiểm tra danh tính và mã hóa dữ liệu để tăng cường hệ thống, mở rộng thêm khả năng phân tích tấn công, tổng hợp dữ liệu và cung cấp báo cáo chi tiết cho người dùng.

Tài liệu tham khảo

- [1] Kingda Young, "CNN-LSTM, NSL-KDD, Binary-class, 2022" Kaggle, Available: <https://www.kaggle.com/code/kingdayoung/cnn-lstm-nsf-kdd-binary-class-2022>, [28 08 2023].
- [2] FarazFatahnaie, "Attack Detection,", Kaggle, Available: https://www.kaggle.com/code/farazfatahnaie/attackdetection/notebook?fbclid=IwAR2abzi_1xR86SoypmlZ1GVqi_N74QbruT3clVJ_9JqhGwYPWic2IsKb4GQ, [02 09 2023].
- [3] Omer Rosenbaum, "How to Use Scapy – Python Networking Tool Explained," FreeCodeCamp, 21 12 2022, Available: <https://www.freecodecamp.org/news/how-to-use-scapy-python-networking/>, [10 09 2023].
- [4] Joska de Langen, "Sending Emails With Python," RealPython, Available: <https://realpython.com/python-send-email/>, [21 09 2023].
- [5] Omer Rosenbaum, "How to Use Scapy – Python Networking Tool Explained," FreeCodeCamp, 21 12 2022. Available: <https://www.freecodecamp.org/news/how-to-use-scapy-python-networking/>, [18 09 2023].
- [6] Nathan Jennings, "Socket Programming in Python (Guide)", RealPython, Available: <https://realpython.com/python-sockets/>, [05 10 2023]