ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**BÁO CÁO ĐỒ ÁN**

**Môn: Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập**

**Đề tài:**

**IoV-BERT-IDS: Hybrid Network Intrusion Detection System in IoV Using Large Language Models**

**GVHD: Đỗ Hoàng Hiển**

**Nhóm thực hiện: Nhóm 4**

**Sinh viên thực hiện:**

**Nguyễn Tài Hiếu - 22520442**

**Nguyễn Việt Hoàng - 22520471**

**Ngô Hồng Phúc - 22521124**

**Trần Hữu Hiếu - 22520444**

Tp. Hồ Chí Minh, tháng 5 năm 2024

**MỤC LỤC**

[**I. Giới thiệu đề tài 3**](#_hsmpwiplgps2)

[1. Bối cảnh nghiên cứu 3](#_h3m67f97gpkk)

[2. Vấn đề nghiên cứu 3](#_7y8ffdxur14n)

[3. Mục tiêu thực nghiệm 4](#_57w8hd7zkawb)

[**II. Cơ sở lý thuyết 4**](#_tgh7q24zlhug)

[1. Internet of Things 4](#_g3c1x87dumx5)

[2. Internet of Vehicles 5](#_2ugle3qtdmvs)

[2.1 Định nghĩa và tổng quan về thành phần 5](#_3gnb3zyvs6m7)

[2.2 Kiến trúc của IoV 6](#_4kl5ui2a8u5d)

[2.3 Bảo mật của IoV 7](#_sqeb52oh41ur)

[3. Hệ thống xâm nhập IDS 8](#_sdsw74enib9v)

[3.1 Định nghĩa 8](#_8xjr0fkka61b)

[3.2 Ứng dụng trong IoV 9](#_egvl45czxy0)

[3.3 Nghiên cứu liên quan 10](#_33pgth7fd1s9)

[4. Mô hình BERT 11](#_2i5hnynmk9id)

[4.1. Định nghĩa 11](#_bn3c6h68i59f)

[4.2 Kiến trúc mô hình 11](#_7pn0i92w3n1l)

[4.3 Ứng dụng trong IoV 12](#_oz5vnym04nyy)

[**III. Dữ liệu thực nghiệm 13**](#_vnrn62jamf2t)

[1. Tập dữ liệu 13](#_hxcjs0ephi6)

[1.1. Giai đoạn pretrain 13](#_lg2hexc10rhx)

[1.2. Giai đoạn fine tune 14](#_ffmnqyvjbx6o)

[2. Tiền xử lý dữ liệu 17](#_44v91j51lntc)

[2.1. Semantic Extractor 17](#_5klewt8nays2)

[2.2. Preprocess\_CAN 22](#_a9vl1he8ewky)

[**IV. Phương pháp thực nghiệm 25**](#_fvpwjom1yy9s)

[1. Tổng quan về IoV-BERT-IDS. 25](#_ohs39skq4kxf)

[2. Semantic Extractor 26](#_l4velgwmw1ed)

[2.1. Semantic List 26](#_vwzb9vojlkyc)

[2.2. Unidirectional Semantic Extractor 27](#_y0zfsq5xzthq)

[2.3. Bidirectional Semantic Extractor 27](#_apg6mjpzlfi7)

[3. Pre-training 28](#_z0ftzqlb2lxc)

[3.1. Masked Byte Word Model 28](#_dqbvkdmzozdq)

[3.2. Giải thích code pretrain.py 29](#_peat7tgb3r47)

[4. Fine-tuning 32](#_e846jen0dxgs)

[**V. Kết quả thực nghiệm 39**](#_1j31mmrbbnpj)

[1. Các tiêu chí đánh giá 39](#_r469ztjqkemx)

[2. Thực nghiệm 41](#_laqw99jl9z7a)

[**VI. Kết luận 43**](#_eb4uk1zareto)

[**VII. Tham khảo 45**](#_39ffknn4bye3)

**BÁO CÁO CHI TIẾT**

# Giới thiệu đề tài

## Bối cảnh nghiên cứu

Internet of Vehicles (IoV) là sự tiến hóa của mạng lưới xe cộ truyền thống (VANET), tích hợp các phương tiện giao thông với cơ sở hạ tầng mạng, cảm biến, và các thiết bị thông minh để tạo ra một hệ sinh thái kết nối thông minh. IoV hỗ trợ các ứng dụng như quản lý giao thông, lái xe tự động, và chia sẻ dữ liệu thời gian thực, nâng cao hiệu quả và an toàn giao thông. Tuy nhiên, môi trường mạng mở của IoV khiến nó trở thành mục tiêu của các cuộc tấn công mạng, như tấn công từ chối dịch vụ (DoS), giả mạo dữ liệu, hay xâm nhập hệ thống. Các hệ thống phát hiện xâm nhập (Intrusion Detection Systems - IDS) đóng vai trò quan trọng trong việc bảo vệ IoV, nhưng các phương pháp dựa trên học máy (ML) và học sâu (DL) hiện tại còn nhiều hạn chế về hiệu suất và khả năng tổng quát hóa.

Gần đây, các mô hình ngôn ngữ lớn (Large Language Models - LLMs), đặc biệt là BERT, đã chứng minh tiềm năng trong việc xử lý dữ liệu phức tạp, mở ra cơ hội cải thiện IDS trong IoV. Nghiên cứu này tập trung vào việc tái hiện IoV-BERT-IDS, một hệ thống IDS lai sử dụng BERT để phát hiện xâm nhập trong cả mạng trong xe (in-vehicle) và ngoài xe (extra-vehicle).

## Vấn đề nghiên cứu

Các IDS truyền thống dựa trên ML và DL trong IoV đối mặt với ba thách thức chính:

* Hiệu suất phân loại hạn chế: Các phương pháp này thường chỉ khai thác đặc trưng ngữ cảnh một chiều, dẫn đến hiệu quả phát hiện xâm nhập thấp, đặc biệt trong các kịch bản tấn công phức tạp.
* Thiếu khả năng tổng quát hóa: Hầu hết các IDS được huấn luyện và kiểm tra trên dữ liệu từ một mẫu xe cụ thể, không thể thích nghi với các loại xe hoặc giao thức mạng khác nhau, làm giảm tính ứng dụng thực tiễn.
* Thiếu IDS lai: Rất ít nghiên cứu phát triển IDS có khả năng phát hiện xâm nhập đồng thời cho cả mạng trong xe (dựa trên giao thức CAN) và mạng ngoài xe (dựa trên TCP/UDP), gây khó khăn trong việc bảo vệ toàn diện hệ thống IoV.

Để giải quyết các vấn đề này, bài báo gốc đề xuất IoV-BERT-IDS, sử dụng BERT và một bộ trích xuất ngữ nghĩa (Semantic Extractor) để chuyển đổi dữ liệu lưu lượng mạng thành định dạng ngữ nghĩa, kết hợp với các tác vụ huấn luyện trước sáng tạo. Thực nghiệm này nhằm tái hiện và đánh giá hiệu quả của phương pháp này.

## Mục tiêu thực nghiệm

Thực nghiệm được thực hiện với các mục tiêu sau:

* Tái hiện IoV-BERT-IDS: Xây dựng lại khung công tác IoV-BERT-IDS, bao gồm Semantic Extractor, các tác vụ huấn luyện trước (Masked Byte Word Model và Next Byte Sentence Prediction), và giai đoạn tinh chỉnh, để phát hiện xâm nhập trong IoV.
* Đánh giá hiệu suất: Kiểm tra hiệu quả của IoV-BERT-IDS trên các tập dữ liệu mạng ngoài xe (CICIDS, BoT-IoT) và mạng trong xe (Car-Hacking, IVN-IDS), sử dụng các chỉ số Precision, Recall, F1-score, và Accuracy.
* So sánh với các phương pháp khác: Đối chiếu kết quả của IoV-BERT-IDS với các phương pháp tham chiếu (ByteSGAN, VAE, AE) để xác minh ưu thế của mô hình trong các kịch bản IoV.

# Cơ sở lý thuyết

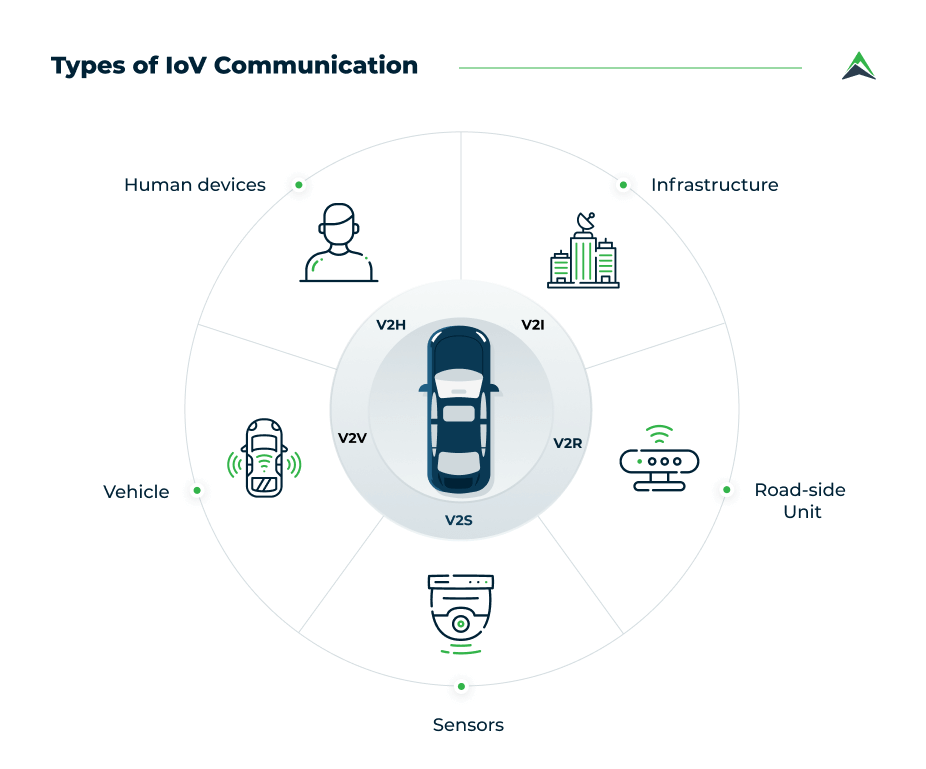
## Internet of Things

* Internet vạn vật (IoT) là một công nghệ tích hợp kết nối giữa thế giới số và thế giới thực. Ngoài ra, IoT hay Internet of Things còn là một tập hợp các thiết bị thông minh và công nghệ tạo điều kiện thuận lợi cho hoạt động giao tiếp giữa thiết bị và đám mây cũng như giữa các thiết bị với nhau [2]. Sự trao đổi thông tin qua mạng IoT sẽ ngày càng tiện lợi hơn, nhanh chóng và thông minh hơn với một định nghĩa mới về giao tiếp mọi lúc mọi nơi và mọi vật nhờ khả năng định vị thời gian thực, giao tiếp không dây nhanh chóng.
* Một vài ví dụ về hệ thống IoT được sử dụng trong phương tiện thông minh (smart vehicle), nhà thông minh (smart house), thành phố thông minh (smart city), …

## Internet of Vehicles

### 2.1 Định nghĩa và tổng quan về thành phần

* Internet of Vehicles (hay viết tắt là IoV) là một khái niệm mạng lưới thông tin đa chiều kết hợp IoT trong hệ thống giao thông thông minh - Intelligent Transportation Systems (ITS) với mục đích tăng cường khả năng giao tiếp và trao đổi thông tin giữa các phương tiện và cơ sở hạ tầng xung quanh - Vehicular Ad-Hoc Network (VANET), một module được tích hợp trong Internet of Things (IoT). Từ đó tạo ra một mạng lưới giao tiếp đa chiều giữa các thành phần mạng.
* Nhờ khả năng linh hoạt trong việc cấu trúc bản đồ, quy mô triển khai mạng lớn, kết nối mạng tin cậy, tương thích với các thiết bị các nhân và khả năng xử lý thông tin cao, IoV dần được xem là một hệ thống mạng phương tiện thiết yếu trong thực tiễn, bên cạnh VANETs.
* Mục tiêu chính của IoV là nâng cao khả năng trao đổi thông tin giữa các phương tiện với nhau và với môi trường xung quanh, từ đó hỗ trợ đưa ra các quyết định nhanh chóng và chính xác trong thời gian thực, góp phần cải thiện an toàn giao thông, tối ưu hóa lưu lượng và nâng cao trải nghiệm người dùng.
* Khác với các mô hình giao thông truyền thống hoặc đơn lẻ như VANETs, IoV cho thấy nhiều ưu điểm vượt trội như khả năng triển khai trên quy mô lớn, tính linh hoạt trong cấu trúc mạng, khả năng tích hợp với các thiết bị cá nhân, và hiệu suất xử lý dữ liệu cao nhờ vào công nghệ điện toán biên (Edge Computing) và trí tuệ nhân tạo (AI).
* IoV bao gồm ba loại giao tiếp chính:
* Giao tiếp giữa các phương tiện (Inter-vehicular communication)
* Giao tiếp bên trong phương tiện (Intra-vehicular communication)
* Giao tiếp di động giữa các phương tiện và thiết bị cá nhân hoặc môi trường xung quanh.
* Bên cạnh đó, mạng IoV có thể được phân loại thành năm loại giao tiếp cụ thể:
* Vehicle-to-Vehicle (V2V): phương tiện trao đổi thông tin với phương tiện khác.
* Vehicle-to-Roadside Unit (V2R): phương tiện giao tiếp với các thiết bị bên đường như đèn giao thông, camera.
* Vehicle-to-Personal Device (V2P): kết nối với điện thoại, đồng hồ thông minh.
* Vehicle-to-Sensor (V2S): tương tác với cảm biến môi trường hoặc cảm biến trong xe.
* Vehicle-to-Infrastructure of cellular networks (V2I): giao tiếp với các trạm cơ sở mạng di động hoặc máy chủ điều phối giao thông.



*Hình 1 - Hình ảnh giao tiếp trong mạng IoV*

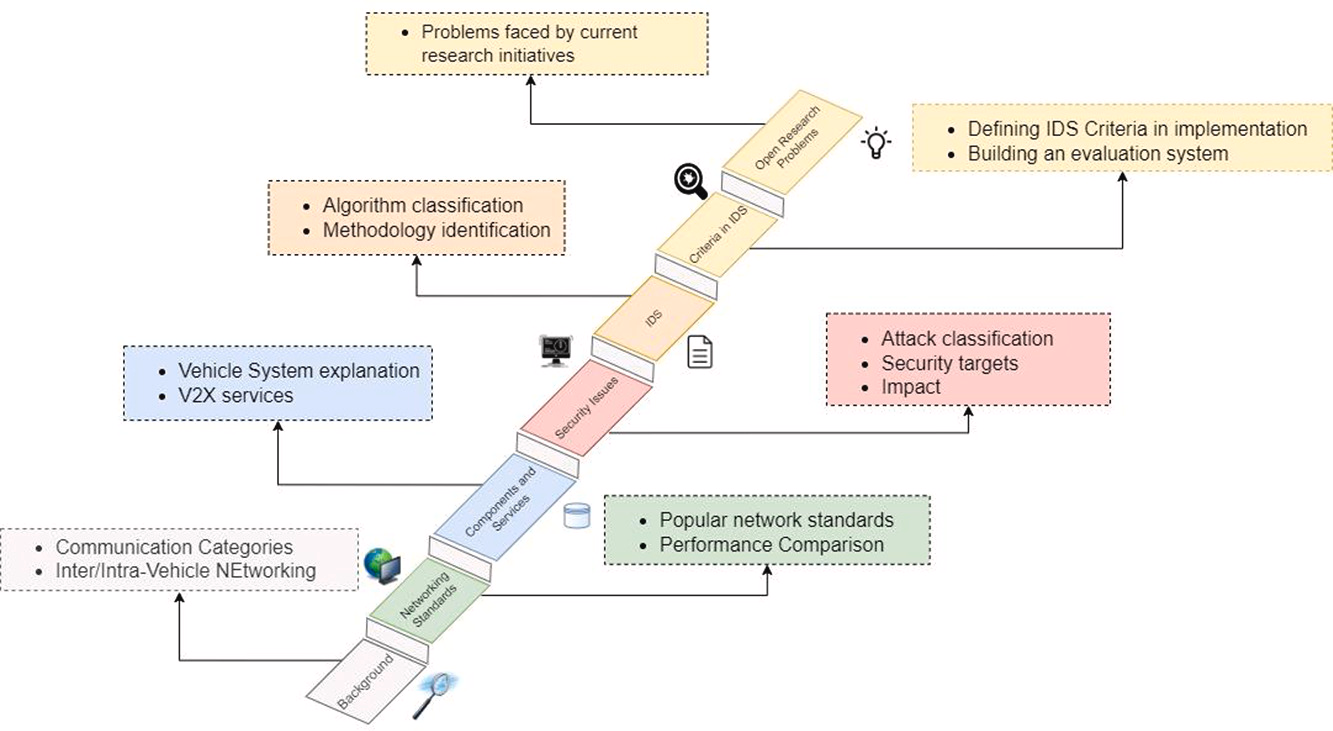
### 2.2 Kiến trúc của IoV

IoV được chia thành bốn lớp:

* Lớp điều khiển và cảm biến môi trường (Environment sensing and control layer): đóng vai trò quan trọng vì lớp này thu thập thông tin linh hoạt của môi trường, con người và các phương tiện thông qua công nghệ cảm biến cũng như dựa vào việc sử dụng cảm biến nhiệt.
* Lớp truy cập mạng và vận chuyển (Network access and transport layer): lớp này có chức năng quản lý nút, xử lý dữ liệu, giám sát từ xa và phân tích dữ liệu; đồng thời nó cũng xem xét và xử lý các ràng buộc về mạng và khả năng truy cập mạng không đồng thời cho các phương tiện trong mạng.
* Lớp tính toán phối hợp (Coordinative computing layer): lớp này hỗ trợ tương tác trong việc tính toán nhận thức; khả năng tính toán thông minh theo nhóm sử dụng cloud computing và edge computing ; xử lý dữ liệu và phân bố tài nguyên.
* Lớp ứng dụng (Application layer): lớp này cung cấp hai loại là dịch vụ đóng tập trung vào ứng dụng cụ thể như nền tảng điều khiển và xử lý giao thông và dịch vụ mở liên quan đến xử lý thông tin giao thông theo thời gian thực.

### 2.3 Bảo mật của IoV

* Mặc dù mạng IoV mang lại nhiều lợi ích to lớn trong việc vận hành và xử lý thông tin giao tiếp các phương tiện, nhưng chính việc giao tiếp qua mạng này lại gây ra nhiều nguy cơ bảo mật cho chính hệ thống.
* Các rủi ro bật mật trong mạng IoV và cả VANETs đều liên quan đến tính chất không dây và phân tán của nó[5], ví dụ như:
* Sybil Attack: hacker tạo nhiều danh tính giả để truy cập mạng, sau đó phát tán thông tin sai lệch hoặc gây ra rối loạn thông tin cho hệ thống.
* DoS Attack: hacker làm quá tải mạng bằng cách gửi nhiều dữ liệu giả, từ đó làm gián đoạn giao tiếp giữa các phương tiện hoặc cơ sở hạ tầng.
* Impersonation Attack: hacker giả mạo danh tính của một phương tiện hoặc cơ sở hạ tầng (như Road Side Unit) để gửi thông điệp sai lệch hoặc đánh cắp dữ liệu nhạy cảm.
* Location tracking: hacker theo dõi vị trí của phương tiện thông qua việc tiết lộ thông tin giao tiếp giữa các thành phần trong mạng .
* Man-in-the-Middle Attack: chặn và sửa đổi dữ liệu truyền giữa các thành phần và thiết bị trong mạng.
* Replay Attack: ghi lại và phát lại dữ liệu để gây nhiễu hệ thống.
* Hệ quả:
* Mất an toàn giao thông.
* Ảnh hưởng đến quyền riêng tư của người dùng.
* Nguy cơ về an toàn tính mạng con người.



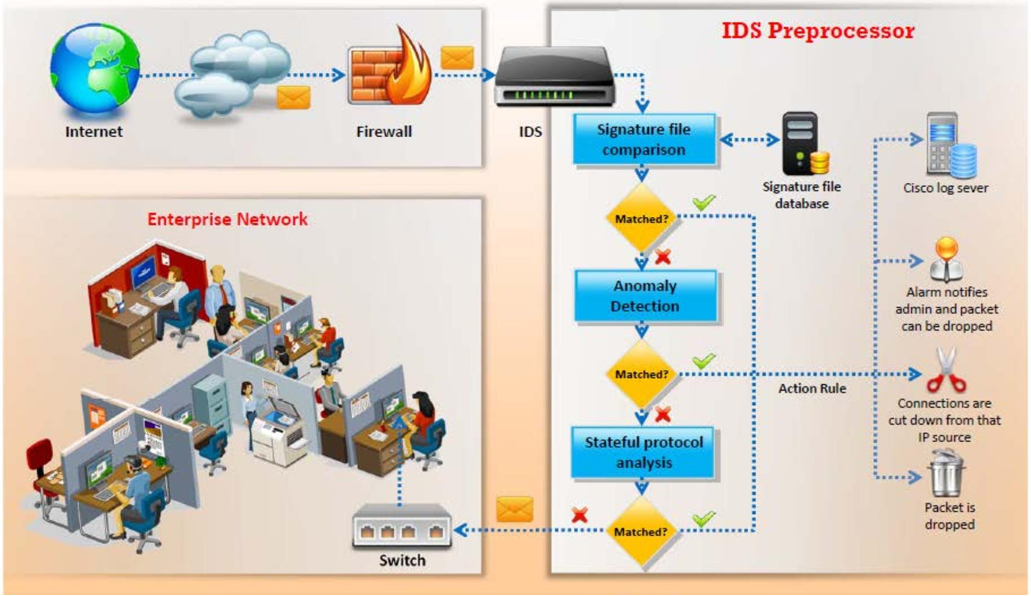
*Hình 2 - tổng hợp các thách thức thành phần kỹ thuật cốt lõi của IoV về*

*bảo mật, kiến trúc mạng*

## 3. Hệ thống xâm nhập IDS

### 3.1 Định nghĩa

* Hệ thống phát hiện xâm nhập - Intrusion Detection System (IDS) là một hệ thống phần mềm hoặc phần cứng thực hiện chức năng tự động phát hiện xâm nhập, ví dụ như các giao tiếp, hành vi độc hại trong mạng.
* Chức năng: IDS có thể liên tục theo dõi các gói tin trong mạng và hiểu thông tin ở dạng nhị phân, sau đó cảnh báo nếu lưu lượng được coi là bất thường khi khớp với một signature đã biết. Và IDS hỗ trợ các cơ chế phòng thủ mạng khác nhau, được triển khai trong chiến lược phòng thủ theo chiều sâu (defense-in-depth).
* Cơ chế hoạt động đơn giản của IDS: Hệ thống phát hiện xâm nhập là một hệ thống giám sát cảnh giác liên tục giám sát lưu lượng mạng để tìm bất kỳ dấu hiệu truy cập trái phép hoặc hoạt động độc hại nào. Khi phát hiện các hoạt động độc hại như vậy, IDS sẽ hành động bằng cách cảnh báo cho quản trị viên.
* Cụ thể như sau:
* Giám sát và phân tích: IDS liên tục kiểm tra luồng lưu lượng mạng trong khi xem xét kỹ lưỡng mọi hoạt động để tìm ra bất kỳ điều gì đáng ngờ.
* So sánh quy tắc và mẫu: Sử dụng cơ sở dữ liệu các quy tắc và mẫu được xác định trước, đóng vai trò là tiêu chí của IDS để xác định hành vi có khả năng đáng ngờ hoặc độc hại.
* Tạo cảnh báo: Khi hoạt động mạng đáp ứng bất kỳ tiêu chí nào đã thiết lập ở trên, IDS sẽ đưa ra cảnh báo bằng cách cảnh báo cho quản trị viên hệ thống hoặc cơ quan có thẩm quyền.



*Hình 3 - mô tả về cơ chế hoạt động của một hệ thống IDS*

### 3.2 Ứng dụng trong IoV

Hệ thống phát hiện xâm nhập ứng dụng mạng lưới Internet of Vehicles dùng để giám sát trạng thái theo thời gian thực của mạng ô tô. Bên cạnh đó, IDS còn có khả năng nhận diện các nguy cơ bảo mật chẳng hạn như bị tấn công bởi hacker, lây nhiễm virus; và từ đó cài đặt và thiết lập các biện pháp phòng thủ bảo vệ cần thiết để bảo đảm sự an toàn của phương tiện.

### 3.3 Nghiên cứu liên quan

* *Yang, Li & Shami, Abdallah. (2022). A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles. 10.48550/arXiv.2201.11812. [6]*

→ Trong bài báo này, tác giả đề xuất một hệ thống IDS dựa trên học chuyển giao và học tổ hợp, sử dụng mạng nơ-ron tích chập (CNN) cùng với kỹ thuật tối ưu hóa siêu tham số cho hệ thống IoV. Trong các thí nghiệm, hệ thống IDS được đề xuất đã đạt tỷ lệ phát hiện và điểm F1 (F1-score) vượt mức 99,25% trên hai bộ dữ liệu chuẩn nổi tiếng về an ninh mạng trong IoV: bộ dữ liệu Car-Hacking và CICIDS2017. Kết quả này chứng minh tính hiệu quả của hệ thống IDS được đề xuất trong việc phát hiện tấn công mạng cả trong mạng nội bộ của phương tiện và mạng kết nối bên ngoài.

* *L. Yang, A. Moubayed, and A. Shami, “MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles,” IEEE Internet Things J., vol. 9, no. 1, pp. 616–632, Jan. 2022. [7]*

→ Ở bài báo này, tác giả đã đề xuất một hệ thống IDS lai đa tầng (Multitiered Hybrid IDS - MTH-IDS), kết hợp giữa IDS dựa trên chữ ký và IDS dựa trên bất thường, đạt được điểm F1 cao trong việc phát hiện các cuộc tấn công chưa từng thấy (zero-day) trong mạng lưới phương tiện.

* *Yu, Tianqi & Hua, Guodong & Wang, Huaisheng & Yang, Jianfeng & Hu, Jianling. (2022). Federated-LSTM based Network Intrusion Detection Method for Intelligent Connected Vehicles. 10.1109/ICC45855.2022.9838655. [8]*

*→* Tác giả đã xây dựng một hệ thống IDS dựa trên mô hình Long Short-Term Memory liên kết liên bang (Federated-LSTM), khai thác tính chu kỳ của chuỗi ID trong các gói tin mạng nội bộ của phương tiện.

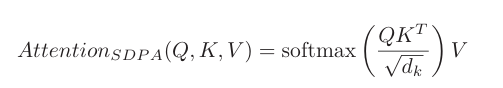
## 4. Mô hình BERT

### 4.1. Định nghĩa

* Mô hình biểu diễn ngôn ngữ tự nhiên hai chiều ứng dụng kỹ thuật Transformer (Bidirectional Encoder Representation from Transformer - BERT) là một mô hình mạng nơ-ron dựa vào cơ chế attention để thực hiện tính toán song song.
* Ngoài ra, việc cho phép mô hình tính toán song song các biểu diễn đặc trưng (feature representations) của các vị trí khác nhau giúp BERT xử lý tốt hơn các chuỗi dài và cải thiện hiệu quả huấn luyện so với các mô hình xử lý tuần tự như Long-Short Term Memory.

### 4.2 Kiến trúc mô hình

* Mỗi bộ encoder transformer hai chiều đều gồm:
* Module Multiple Self-Attention Mechanism: đây là nơi transformer tìm hiểu mối liên kết giữa các token trong chuỗi dữ liệu. BERT sẽ thực hiện tính toán liên hệ Query (Q), Key (K) và Value (V) bởi công thức sau:



* Module Feed-Forward Neural Network: sau khi qua bước attention, module này thực hiện biến đổi phi tuyến tính trên dữ liệu đầu vào tại từng vị trí bằng cách đưa mỗi token qua một mạng nơ-ron nhỏ nhằm trích xuất các đặc trưng phức tạp hơn, tăng khả năng biểu diễn dựa trên việc kết hợp đặc trưng vừa học với ngữ cảnh hoạt động; được mô tả bởi công thức sau:



* Module Residual Connection and Layer Normalization: gồm hai bộ phận

++ Residual Connection ( Kết nối dữ liệu dư thừa): là một kỹ thuật trong mạng nơ-ron sâu (deep neural network) để giúp giảm hiện tượng mất mát thông tin và gradient vanish khi mạng có quá nhiều lớp. Cách hoạt động: đầu vào ban đầu xxx được cộng trực tiếp với đầu ra của lớp xử lý F(x), tạo thành x+F(x). Kết quả này sau đó được đưa vào lớp tiếp theo.

→ Giữ lại thông tin gốc (tránh mất mát qua nhiều lớp). Tập trung học phần sai lệch (residual) giữa đầu vào và đầu ra giúp huấn luyện nhanh và ổn định hơn.

++ Layer Normalization: Là kỹ thuật chuẩn hóa giá trị của đầu vào một cách độc lập tại mỗi lớp nhằm tăng tốc độ hội tụ của mô hình (convergence), Tăng khả năng tổng quát hóa (generalization). Cụ thể:



### 4.3 Ứng dụng trong IoV

Tính ứng dụng của BERT trong mạng IoV được thể hiện trong các bài báo có liên quan đến xử lý thông tin, bảo mật dữ liệu, tính kết hợp với Artificial Intelligence và Machine Learning. Cụ thể:

* Controller Area Network Intrusion Detection System based on BERT Language Model.[4]

→ Bài báo đề xuất "CAN-BERT", một hệ thống phát hiện xâm nhập mạng dựa trên học sâu, sử dụng BERT để phát hiện các cuộc tấn công mạng trên giao thức CAN bus. Mô hình BERT học chuỗi các định danh (IDs) trong CAN bus để phát hiện bất thường bằng cách sử dụng mô hình ngôn ngữ bị che (masked language model).

* A Method for Network Intrusion Detection Using Flow Sequence and BERT Framework.[1]

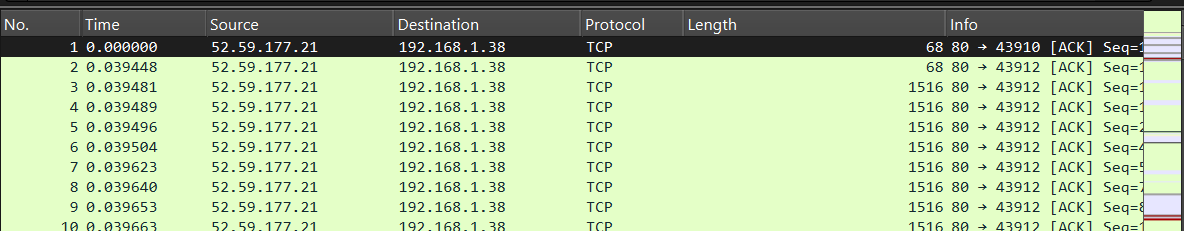
→ Bài báo đề xuất phương pháp sử dụng chuỗi lưu lượng mạng và mô hình BERT để phát hiện xâm nhập mạng. BERT được sử dụng để biểu diễn chuỗi lưu lượng mạng, sau đó được xử lý bởi mạng Multi-layer Perceptron (MLP) để phân loại lưu lượng.

# Dữ liệu thực nghiệm

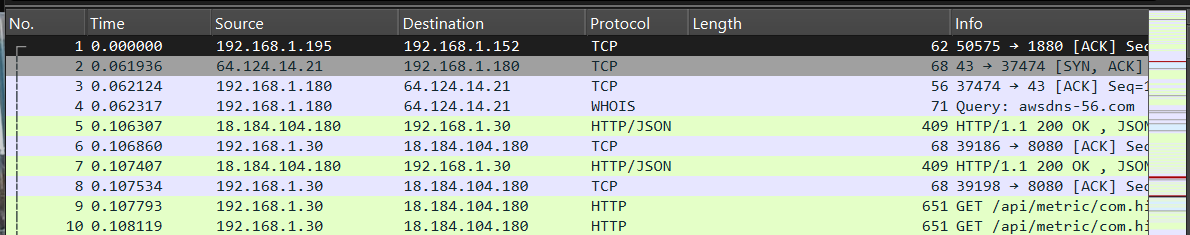
## Tập dữ liệu

### Giai đoạn pretrain

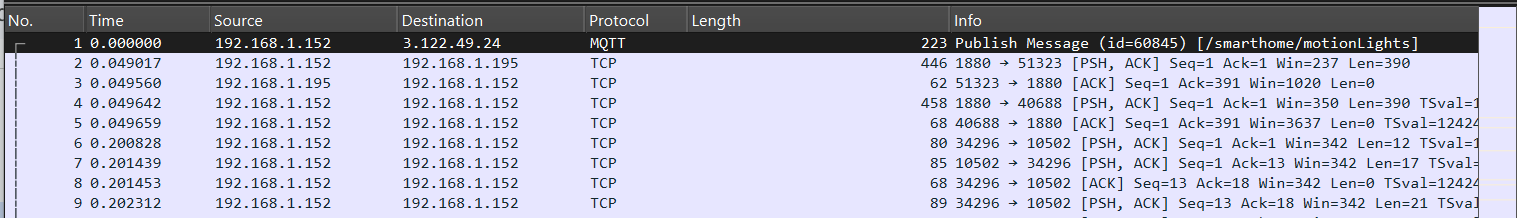
* Các tập dữ liệu trong giai đoạn này gồm: injection\_normal4.pcap, normal\_DDos\_1.pcap, mormal\_IOT\_2.pcap.
* Mô tả tập dữ liệu thô (raw dataset): Các tệp PCAP (packet capture) chứa lưu lượng mạng từ các giao thức như TCP, UDP, ICMP, được thu thập từ mạng thử nghiệm tại Cyber Range và IoT Labs, UNSW Canberra. Đây là dữ liệu chính được sử dụng trong giai đoạn pre-training, vì IoV-BERT-IDS yêu cầu dữ liệu mạng ngoài xe (tương tự CICIDS và BoT-IoT trong bài báo gốc).



*Hình 4 - Injection\_normal4.pcap*

****

*Hình 5 - normal\_DDos\_1.pcap*

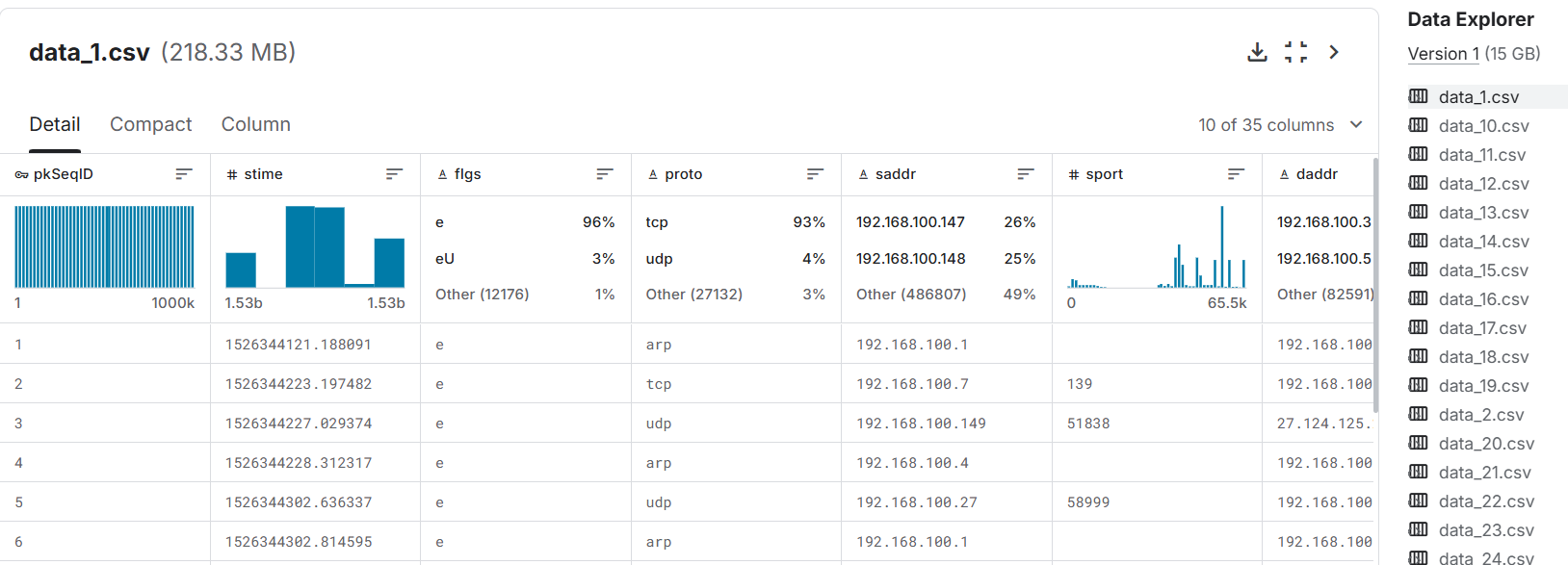
******

*Hình 6 - mormal\_IOT\_2.pcap*

* Các tệp PCAP chứa các gói tin (packets) với thông tin như:
* 5-tuple: Địa chỉ IP nguồn/đích, cổng nguồn/đích, giao thức (TCP/UDP/ICMP).
* Payload: Dữ liệu thô của gói tin, biểu diễn dưới dạng chuỗi thập lục phân (hexadecimal strings).
* Timestamp: Thời gian ghi nhận gói tin.

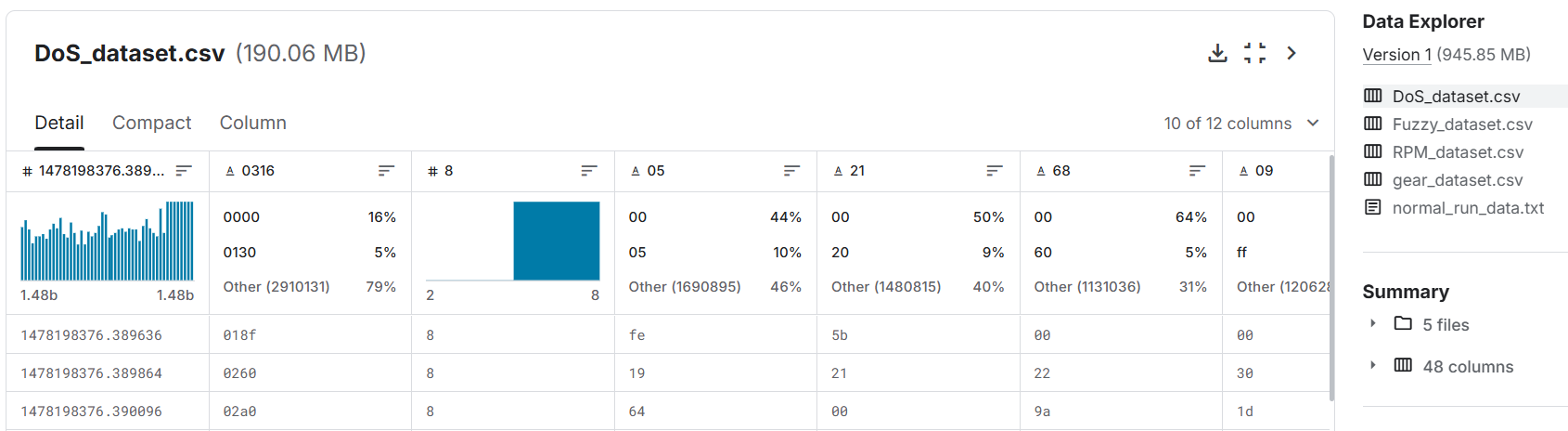
### Giai đoạn fine tune

* Các tập dữ liệu trong giai đoạn này gồm: Bot\_IOT, Car\_Hacking, CICIDS-2017, IVN-IDS.
* Mô tả tập dữ liệu:
* Bot\_IOT: bộ dataset gồm các lưu lượng mạng bình thường và tấn công (botnet), thu thập từ các thiết bị Iot trong môi trường mô phỏng. Kích thước lớn chứa nhiều gói tin, phù hợp với mô hình học sâu BERT. Dữ liệu được thu thập từ nhiều giao thức mạng (TCP,UDP,HTTP,v.v), phản ánh các kịch bản Iot thực tế. Lưu lượng mạng được trích xuất dưới dạng CSV có dung lượng 16,7GB.



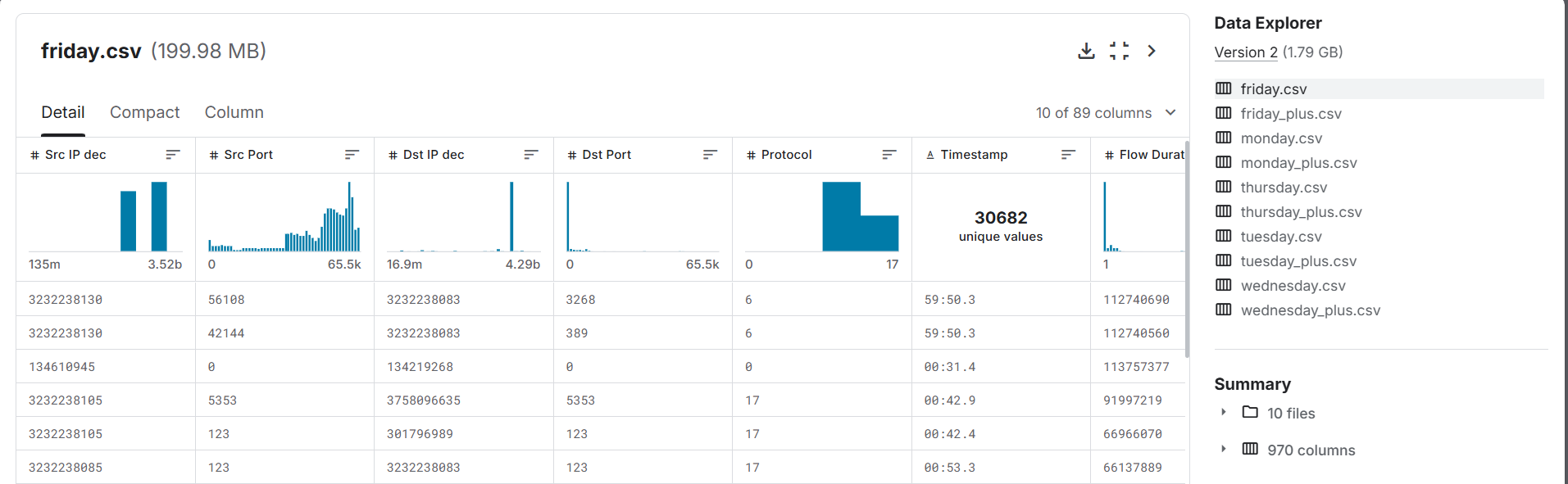
*Hình 7 - Bot\_IOT trên kaggle*

* Car\_Hacking: là một bộ dữ liệu ghi lại lưu lượng mạng trên giao thức Controller Area Network (CAN) bus từ một phương tiện thực tế. Bộ dữ liệu này được thiết kế để mô phỏng các kích bản lái xe thông thường và bốn loại hình tấn công nhằm hỗ trợ nghiên cứu phát hiện xâm nhập trong các hệ thống xe hơi. Bộ dữ liệu ghi lại lưu lượng CAN bus từ một chiếc xe thực tế, bao gồm cả dữ liệu từ các kịch bản lái xe thông thường và các cuộc tấn công mạng được mô phỏng. Các loại tấn công bao gồm:
* Tấn công DoS (Denial of Service): Làm gián đoạn hoạt động bình thường của mạng CAN.
* Tấn công Fuzzy: Gửi các gói tin ngẫu nhiên hoặc không hợp lệ để gây rối loạn.
* Tấn công giả mạo số hộp số (Spoofing the Drive Gear): Giả mạo dữ liệu liên quan đến hộp số của xe.
* Tấn công giả mạo đồng hồ đo vòng tua máy (Spoofing the RPM Gauge): Giả mạo dữ liệu liên quan đến vòng tua động cơ.



*Hình 8 - Bộ dữ liệu Car Hacking trên kaggle*

* CICIDS-2017: là một bộ dữ liệu đánh giá phát hiện xâm nhập mạng, được phát triển bởi Trung tâm Hợp tác An ninh mạng Canada (Canadian Institute for Cybersecurity - CIC). Bộ dữ liệu này được thiết kế để mô phỏng lưu lượng mạng thực tế, bao gồm cả lưu lượng bình thường và các cuộc tấn công mạng phổ biến, nhằm hỗ trợ nghiên cứu về phát hiện xâm nhập trong an ninh mạng.Bộ dữ liệu CICIDS-2017 được tạo ra để cung cấp một tập hợp dữ liệu đáng tin cậy, phản ánh các kịch bản mạng thực tế, bao gồm lưu lượng mạng lành tính (benign) và các loại tấn công mạng phổ biến, cập nhật đến năm 2017. Các cuộc tấn công bao gồm:
* DoS/DDoS: Các cuộc tấn công từ chối dịch vụ, chẳng hạn như Hulk, GoldenEye, Slowloris.
* Web Attacks: Bao gồm Brute Force, XSS (Cross-Site Scripting), và SQL Injection.
* Botnet: Các hoạt động liên quan đến mạng botnet.
* Infiltration: Các cuộc tấn công xâm nhập vào hệ thống mạng.
* Port Scan: Quét cổng để tìm kiếm lỗ hổng.
* Brute Force: Các cuộc tấn công thử mật khẩu (FTP, SSH).



*Hình 9 - Bộ dữ liệu CICIDS-2017 trên Kaggle*

* IVN-IDS: là một bộ dữ liệu được thiết kế để hỗ trợ nghiên cứu về phát hiện xâm nhập trong mạng nội bộ của xe hơi (In-Vehicle Network), cụ thể là mạng CAN (Controller Area Network). Bộ dữ liệu này chứa các bản ghi lưu lượng mạng từ mạng CAN bus, nhằm mô phỏng các hoạt động bình thường và các cuộc tấn công mạng trong hệ thống xe hơi. Bộ dữ liệu IVN-IDS được tạo ra để mô phỏng lưu lượng mạng CAN bus trong các kịch bản thực tế, bao gồm cả lưu lượng bình thường và các cuộc tấn công mạng. Các loại tấn công được mô phỏng có thể bao gồm các dạng như:
* DoS (Denial of Service): Gửi lượng lớn dữ liệu để làm gián đoạn hoạt động của mạng CAN.
* Fuzzy Attacks: Gửi các gói tin ngẫu nhiên hoặc không hợp lệ để gây rối loạn.
* Spoofing Attacks: Giả mạo dữ liệu liên quan đến các thành phần của xe, chẳng hạn như số hộp số hoặc đồng hồ đo vòng tua máy (RPM).

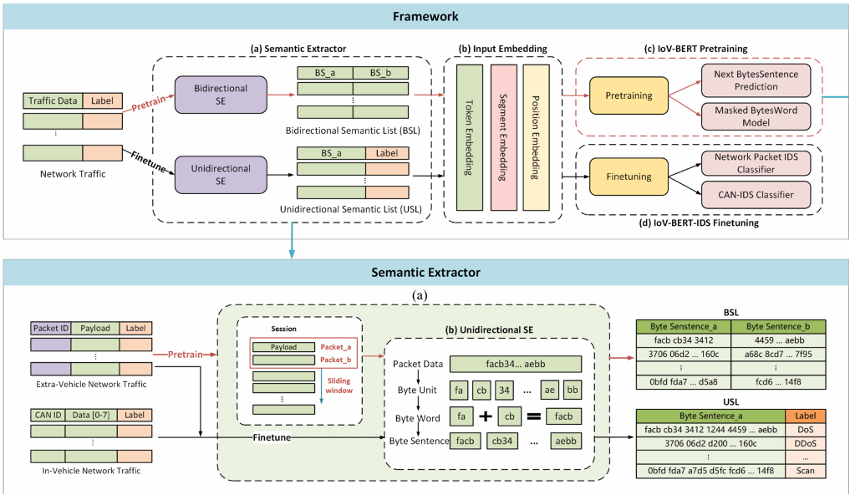


*Hình 10 - Bộ dữ liệu IVN-IDS trên Kaggle*

## Tiền xử lý dữ liệu

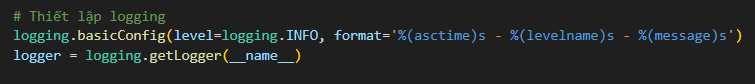
### Semantic Extractor

* Giai đoạn tiền xử lý được thực hiện thông qua Semantic Extractor (SE), bao gồm hai thành phần chính Unidirectional Semantic Extractor (USE) và Bidirectional Semantic Extractor (BSE).
* Dữ liệu đầu vào của mô-đun là các tệp PCAP (chứa dữ liệu giao thông mạng) để tạo ra danh sách BSL, trong đó mỗi mục là mỗi cặp câu byte (byte sentences) từ các gói tin liên tiếp trong cùng một phiên (session). Các câu byte này được lưu vào một tệp văn bản để sử dụng trong giai đoạn pre-training của mô hình IoV-BERT-IDS.
* **Quy trình chính:**
* Đọc các tệp PCAP.
* Xử lý song song các tệp PCAP bằng cách sử dụng multiprocessing.
* Trích xuất payload từ các gói tin TCP/UDP, chuyển đổi thành chuỗi thập lục phân để tạo byte sentences.
* Sử dụng cơ chế sliding window để tạo các cặp câu byte (BSL).
* Lưu kết quả vào một tệp văn bản.

****

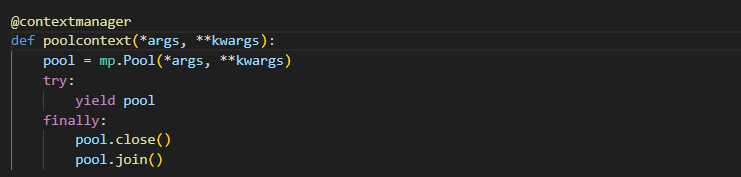
*Hình 11 - Giai đoạn Semantic Extractor*

* **Thư viện sử dụng:**
* Scapy: đọc và phân tích các tệp PCAP.
* Multiprocessing: xử lý song song nhiều tệp PCAP, tối ưu hóa hiệu suất.
* Logging: ghi lại thông tin xử lý và lỗi.
* Contextlib: quản lý tài nguyên (multiprocessing pool).
* **Coding:**

****

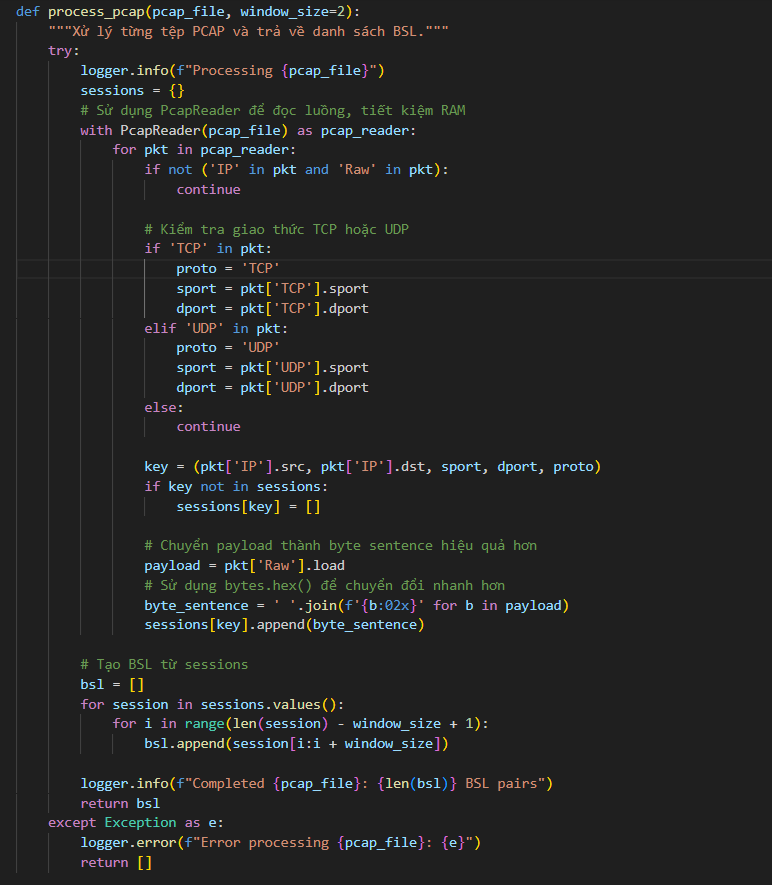
*Hình 12 - Thiết lập logging*

* Thiết lập hệ thống ghi log với mức độ INFO, định dạng bao gồm thời gian, mức độ log (INFO, ERROR,..)và thông điệp. Nhằm theo dõi tiến trình xử lý, ghi lại thông tin về các tệp PCAP được xử lý và bất kỳ lỗi nào xảy ra.



*Hình 13 - Hàm tạo context manager quản lý tài nguyên*

* Đây là một context manager để quản lý tài nguyên của multiprocessing.Pool. Đảm bảo pool được đóng và giải phóng tài nguyên sau khi hoàn thành xử lý. Tăng tính an toàn và hiệu quả khi sử dụng đa xử lý, tránh rò rỉ tài nguyên.



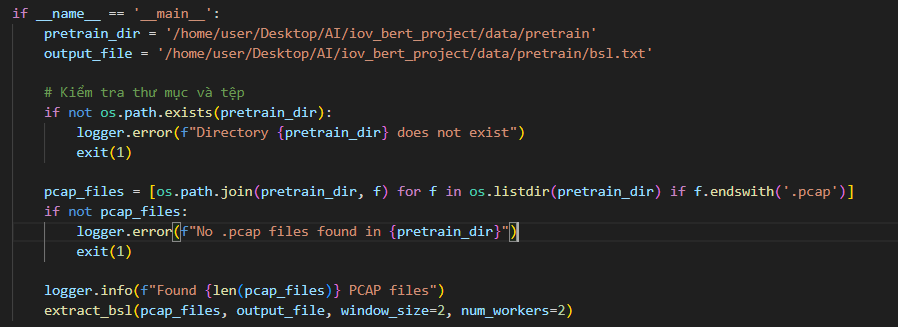
*HÌnh 14 - Hàm xử lý tệp PCAP và trả về dữ liệu BSL*

* Chức năng: xử lý một tệp PCAP để tạo danh sách BSL. Tạo BSL từ dữ liệu thô, đại diện cho các cặp câu byte liên tiếp, phù hợp với mô tả của BSE trong bài báo.
* Quy trình:
* Đọc tệp PCAP: sử dụng PcapReader từ Scap để đọc từng gói tin (packet) một cách hiệu suất.
* Kiểm tra gói tin: Chỉ xử lý các gói tin có lớp IP và Raw (chứa payload). Các gói tin không thuộc TCP hoặc UDP bị bỏ qua.
* Xác định phiên (session): Mỗi gói tin được gán vào một phiên dựa trên bộ năm thông tin (source IP, destination IP, source port, destination port, protocol).
* Chuyển đổi payload: Payload của gói tin được chuyển thành chuỗi thập lục phân (hex string), với mỗi byte được biểu diễn bằng hai ký tự (vd: 0a), các byte được nối bằng dấu cách để tạo thành byte sentence.
* Tạo BSL: Sử dụng cửa sổ trượt với kích thước window\_size (mặc định là 2) để lấy các cặp câu byte liên tiếp trong cùng một phiên. Ví dụ, với window\_size=2, nếu một phiên có 3 câu byte [s1, s2, s3], BSL sẽ chứa [ [s1, s2], [s2, s3] ].
* Xử lý lỗi: Nếu có lỗi trong quá trình xử lý, trả về danh sách rỗng và ghi log lỗi.



*Hình 15 - Xử lý song song các tệp PCAP và lưu BSL*

* Điều phối việc xử lý song song nhiều tệp PCAP và lưu kết quả BSL vào một tệp văn bản. Tối ưu hóa việc xử lý khối lượng lớn tệp PCAP và lưu trữ BSL theo định dạng phù hợp với mô hình BERT.
* Quy trình:
* Xử lý song song: Sử dụng multiprocessing.Pool với số lượng worker được chỉ định (num\_workers=2) để gọi hàm process\_pcap cho từng tệp PCAP.
* Gộp kết quả: Kết hợp các danh sách BSL từ tất cả các tệp PCAP vào một danh sách tổng.
* Lưu kết quả: Ghi các cặp câu byte vào tệp văn bản (output\_file), với mỗi cặp được phân tách bằng ký hiệu [SEP]. Sử dụng bộ đệm (buffering=8192) để tối ưu hóa hiệu suất ghi file.

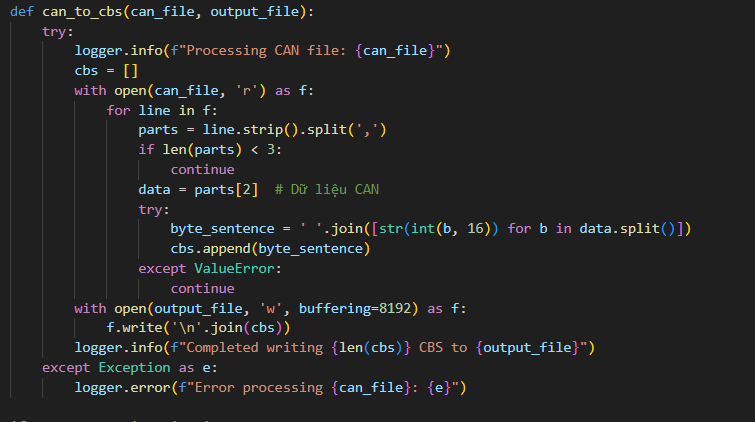


*Hình 16 - Hàm main của chương trình semantic extractor*

* Khởi tạo và chạy quy trình chính. Đảm bảo chương trình chạy đúng với các tham số được cấu hình và xử lý tất cả các tệp PCAP trong thư mục.
* Quy trình:
* Kiểm tra thư mục: Xác minh sự tồn tại của thư mục chứa tệp PCAP (pretrain\_dir).
* Tìm tệp PCAP: Lấy danh sách các tệp có đuôi .pcap trong thư mục.
* Kiểm tra tệp: Nếu không tìm thấy tệp PCAP, ghi log lỗi và thoát.
* Gọi hàm extract\_bsl: Xử lý các tệp PCAP với kích thước cửa sổ là 2 và sử dụng 2 worker.

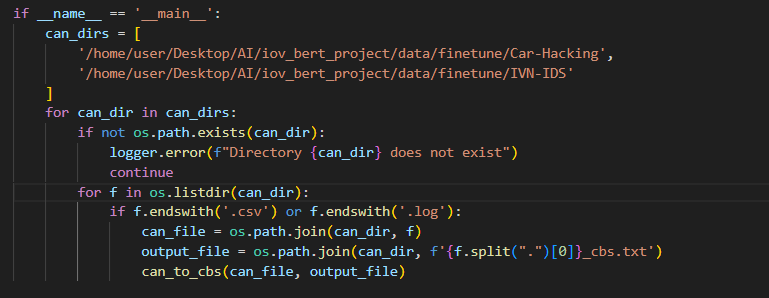
### Preprocess\_CAN

* Chương trình sau được dùng để xử lý các tệp dữ liệu CAN (Controller Area Network) để tạo ra CAN Byte Sentences (CBS), một dạng biểu diễn ngữ nghĩa của dữ liệu CAN trong xe. Chương trình thuộc giai đoạn Unidirectional Semantic Extractor (USE) trong mô-đun Semantic Extractor (SE), được sử dụng để tiền xử lý dữ liệu CAN trong giai đoạn fine-tuning.
* Chương trình đọc các tệp CAN (định dạng .csv), trích xuất trường dữ liệu CAN (payload), chuyển đổi thành CAN Byte Sentences (CBS) và lưu kết quả vào các tệp văn bản.
* **Quy trình chính:**
* Duyệt qua các thư mục chứa tệp CAN (Car-Hacking và IVN-IDS).
* Đọc từng tệp CAN, trích xuất trường dữ liệu (DATA[0-7]).
* Chuyển đổi dữ liệu thành chuỗi thập lục phân (hexadecimal string) với các byte được phân tách bằng dấu cách, tạo thành CBS.
* Lưu các CBS vào tệp văn bản để sử dụng trong giai đoạn tinh chỉnh.
* **Thư viện sử dụng:**
* Os: để thao tác với hệ thống tệp (duyệt thư mục, kiểm tra tệp).
* logging:Ghi lại thông tin xử lý và lỗi để theo dõi quá trình.
* **Coding:**

****

*Hình 17 - Hàm xử lý dữ liệu*

* Chuyển đổi dữ liệu CAN từ một tập đầu vào thành định dạng CBS và lưu vào tệp đầu ra. Tạo danh sách CBS từ dữ liệu CAN thô, biểu diễn dữ liệu dạng chuỗi số phù hợp để làm đầu vào cho mô hình trong giai đoạn fine-tuning.
* Quy trình:
* Đọc tệp CAN: Mở tệp CAN (định dạng .csv) và đọc từng dòng.
* Kiểm tra định dạng: Mỗi dòng được tách thành các phần bằng dấu phẩy (,). Nếu dòng có ít hơn 3 phần (thiếu dữ liệu cần thiết), bỏ qua.
* Trích xuất dữ liệu CAN: Lấy trường dữ liệu CAN (thường là DATA[0-7]) từ phần thứ ba (parts[2]).
* Chuyển đổi thành CBS:
* Trường dữ liệu CAN (chuỗi thập lục phân, ví dụ: 1a 2b 3c) được tách thành các byte riêng lẻ (data.split()).
* Mỗi byte (chuỗi hex) được chuyển thành số nguyên (int(b, 16)) và sau đó thành chuỗi số (str(int(b, 16))), tạo thành byte sentence với các giá trị được phân tách bằng dấu cách.
* Ví dụ: Chuỗi hex 1a 2b 3c trở thành 26 43 60.
* Xử lý lỗi: Nếu có lỗi trong quá trình chuyển đổi bỏ qua dòng đó.
* Lưu kết quả: Ghi các CBS vào tệp đầu ra, mỗi CBS trên một dòng, sử dụng bộ đệm để tối ưu hóa hiệu suất ghi.



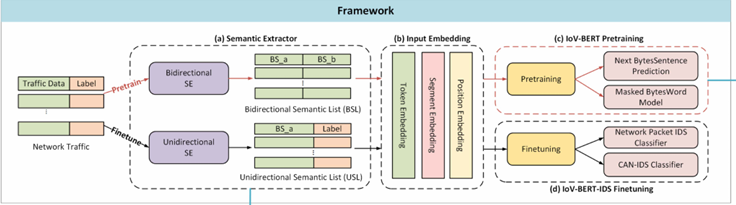
*Hình 18 - Hàm main của giai đoạn preprocess\_CAN*

* Điều phối việc xử lý các tệp CAN trong các thư mục được chỉ định. Tự động hóa việc xử lý nhiều tệp CAN từ các tập dữ liệu khác nhau, đảm bảo tất cả dữ liệu CAN được chuyển đổi thành CBS.
* Quy trình:
* Xác định thư mục: Liệt kê hai thư mục chứa dữ liệu CAN (Car-Hacking và IVN-IDS), tương ứng với các tập dữ liệu được sử dụng trong bài báo
* Kiểm tra thư mục: Nếu thư mục không tồn tại, ghi log lỗi và bỏ qua.
* Duyệt tệp: Tìm tất cả các tệp có đuôi .csv trong thư mục.
* Xử lý từng tệp: Gọi hàm can\_to\_cbs để chuyển đổi từng tệp CAN thành CBS và lưu vào tệp đầu ra với tên được tạo từ tên tệp gốc (thêm hậu tố \_cbs.txt).

# Phương pháp thực nghiệm

## Tổng quan về IoV-BERT-IDS.

* IoV-BERT-IDS là một framework kết hợp giữa mô hình xử lý ngôn ngữ hai chiều - BERT (Bidirectional Encoder Representation from Transformer) và hệ thống phát hiện xâm nhập - IDS (Intrusion Detection System) ứng dụng cho mạng phương tiện - IoV (Internet of Vehicles). Và mô hình hệ thống này được thiết kế để phát hiện và dự đoán về các mối đe dọa, nguy cơ tấn công mạng và các rủi ro trong giao tiếp giữa các thành phần trong môi trường IoV, dựa trên khả năng xử lý ngôn ngữ tự nhiên (Native Language Processing) của BERT để phân tích các luồng dữ liệu trong mạng.
* Cấu trúc thành phần của IoV-BERT-IDS Framework:
* Semantic Extractor: module dùng để chuyển đổi dữ liệu mạng thành một danh sách ngữ nghĩa (semantic list). Phần này sử dụng extra-vehicle network data, tạo dữ liệu cho giai đoạn pre-training. Sau giai đoạn fine-tuning, thì SE dùng USE xử lý gói tin tạo ra USL làm đầu vào cho mô hình.
* Input Embedding: module này việc kết hợp token embedding, segment embedding và position embedding để tạo thành một biểu diễn số học toàn diện của chuỗi đầu vào. Đồng thời, module dùng để chuyển đổi chuỗi dạng byte sang dạng mà hệ thống có thể hiểu và xử lý hiệu quả.
* IoV-BERT-IDS Pre-Training: module này thực hiện Masked Byte Word Model và Next Byte Sentence Prediction để tạo một biểu diễn tổng quát lưu lượng IoV nhằm xử lý dữ liệu Byte Sentence Level.
* IoV-BERT-IDS Fine-Tuning: Module xử lý dữ liệu đầu vào Universal Sentence Representation để gán nhãn các lưu lượng mạng tấn công vào môi trường IoV bằng cách học và phân loại các vector normal và malicious, từ đó dùng để huấn luyện phát hiện xâm nhập.



*Hình 19 - Cấu trúc tổng quan của IoV-BERT-IDS.*

## Semantic Extractor

Semantic Extractor (SE) là phương pháp tiền xử lý dữ liệu được đề xuất trong mô hình IoV-BERT-IDS nhằm giải quyết thách thức khi xử lý dữ liệu lưu lượng mạng có ngữ nghĩa mơ hồ. Mục tiêu của SE là biến đổi dữ liệu thô thành các cặp lưu lượng mang ngữ cảnh rõ ràng để phục vụ cho quá trình huấn luyện mô hình BERT.

Phương pháp này bao gồm hai thành phần: Unidirectional Semantic Extractor (USE) và Bidirectional Semantic Extractor (BSE), tương ứng phục vụ cho hai giai đoạn fine-tuning và pre-training.

### Semantic List

Semantic Extractor tạo ra một tập danh sách ngữ nghĩa - Semantic List (SL), được chia thành 2 loại:

* BSL (Bidirectional Semantic List): Là danh sách ngữ nghĩa hai chiều, chưa được gán nhãn. Gồm 2 trường *byte\_sentence\_a* và *byte\_sentence\_b* tương ứng với *text\_a* và *text\_b* trong mô hình BERT.
* USL (Unidirectional Semantic List): Là danh sách ngữ nghĩa một chiều, đã được gán nhãn, gồm hai trường *byte\_sentence\_a* và *label*.

Trong đó, Byte Sentence là đơn vị cơ bản cấu thành nên SL. Dựa vào loại mạng trong hệ thống phương tiện, Byte Sentence có thể được chia thành:

* CBS (CAN Byte Sentences): Dữ liệu mạng nội bộ trong xe.
* TBS (Traffic Byte Sentences): Dữ liệu mạng bên ngoài xe.

### Unidirectional Semantic Extractor

USE chịu trách nhiệm xử lý dữ liệu đã gán nhãn (cho giai đoạn fine-tuning) và chuyển chúng thành USL:

* Dữ liệu đầu vào là chuỗi hexadecimal của payload từ các gói tin mạng.
* Mỗi chuỗi được chia nhỏ thành các Byte Unit (BU) – mỗi BU gồm 2 ký tự (tương ứng 1 byte).
* Các BU tiếp theo được ghép lại thành Byte Word (BW).
* Tập hợp các BW tạo thành Byte Sentence (BS).

Đối với dữ liệu:

* Trong xe (CAN): USE trích xuất *CAN ID* và *DATA[0-7]*, sau đó tạo thành CBS.
* Ngoài xe (network): USE trích xuất payload từ gói tin mạng và chuyển đổi thành TBS. Đại diện cho *byte\_sentence\_a*.

Đầu ra của USE là một Semantic List dạng USL gồm: *byte\_sentence\_a* thu được từ dữ liệu gói tin. *label*: Nhãn tương ứng của gói tin (VD: benign/malicious).

### Bidirectional Semantic Extractor

BSE mở rộng chức năng của USE để xử lý dữ liệu chưa gán nhãn (cho giai đoạn pre-training):

* Sử dụng cơ chế sliding window để chọn các cặp gói tin liên tiếp trong cùng một phiên mạng (session).
* Với mỗi cặp, payload được trích xuất và đưa qua USE để tạo ra *byte\_sentence\_a* và *byte\_sentence\_b*.
* Các cặp byte sentence này tạo thành Semantic List dạng BSL, dùng làm đầu vào cho pre-training.

Quy trình hoạt động của BSE có thể được tóm tắt như sau:

* Duyệt các tệp pcap để lấy thông tin gói tin như: thời gian, IP nguồn/đích, cổng, giao thức, và payload.
* Phân nhóm các gói tin theo phiên (dựa vào 5-tuple).
* Duyệt theo sliding window (length 2, step size 1) để tạo cặp liên tiếp.
* Trích xuất *byte\_sentence\_a* và *byte\_sentence\_b* từ USE.
* Kết quả là danh sách BSL phục vụ cho huấn luyện mô hình.
  1. **Sliding window**
* Trong mô hình, kỹ thuật sliding window được sử dụng trong giai đoạn Session của Sematric Extractor để xử lý dữ liệu gói tin từ lưu lượng mạng.
* Cách hoạt động: Sliding window di chuyển một cửa sổ cố định qua các chuỗi các gói tin để trích xuất các đoạn dữ liệu liên tiếp. Cửa sổ này chứa một số gói tin nhất định và di chuyển dần để tạo ra các tập hợp dữ liệu từ lưu lượng mạng. Giúp mô hình phần tích các mẫu hoặc các mối quan hệ tạm thời giữa các gói tin.
* Mục đích:
* Bắt các mẫu tạm thời: Sliding window cho phép mô hình nắm bắt các đặc trưng động hoặc chuỗi sự kiện trong dữ liệu mạng, điều quan trọng để phát hiện các hành vi bất thường (như DDoS)
* Chuẩn bị dữ liệu cho huấn luyện: Dữ liệu từ cửa sổ trượt được sử dụng trong quá trình pretrain và finetune để tạo ra các Byte Sentence (chuỗi byte), sau đó được xử lý thêm để trích xuất ý nghĩa ngữ nghĩa (semantic extraction).
* Tăng hiệu quả: Bằng cách chia nhỏ dữ liệu thành các cửa sổ, mô hình có thể xử lý hiệu quả hơn thay vì phân tích toàn bộ lưu lượng mạng cùng lúc.

## Pre-training

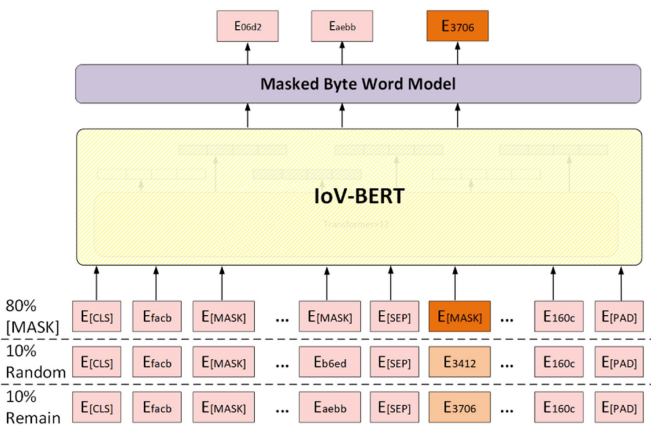
Với đầu vào là danh sách BSL chưa được gán nhãn, mô hình tiền huấn luyện được yêu cầu học một biểu diễn tổng quát về lưu lượng mạng, hoàn thành mức độ "đào tạo cơ bản". Để học được các biểu diễn lưu lượng mang tính ngữ nghĩa tương đối, nhiệm vụ được đề xuất trong giai đoạn này là: Masked Byte Word Model (MBWM)

### Masked Byte Word Model

Nhiệm vụ MBWM tương tự với Masked Language Modeling (MLM) trong BERT, điểm khác biệt duy nhất là MBWM được huấn luyện trên các Byte Word (BW) với tính ngữ nghĩa tương đối, nhằm mục tiêu nắm bắt sự phụ thuộc giữa các BW. Thiết kế này cho phép việc dự đoán mỗi BW dựa trên ngữ cảnh của toàn bộ Byte Sentence (BS), từ đó mang lại khả năng mã hóa hai chiều sâu (deep bidirectional encoding).

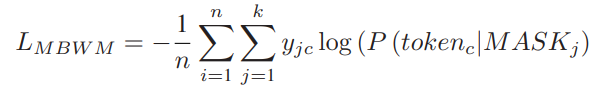
Trong quá trình tiền huấn luyện, mỗi token trong chuỗi đầu vào sẽ có xác suất 15% bị che ngẫu nhiên. Để mô phỏng gần hơn với giai đoạn fine-tuning của các tác vụ downstream sau này, các thao tác sau được áp dụng đối với 15% token bị chọn:

* Với xác suất 80%, token được thay bằng token [MASK].
* Với xác suất 10%, token được thay bằng một token khác ngẫu nhiên.
* Với xác suất 10% còn lại, token được giữ nguyên.



*Hình 20 - Sơ đồ của nhiệm vụ tiền đào tạo MBWM*

Nói một cách đơn giản, nhiệm vụ MBWM dự đoán token tại vị trí bị che dựa vào ngữ cảnh. Cross-entropy loss của MBWM (ký hiệu là *​LMBWM*), được định nghĩa như sau:

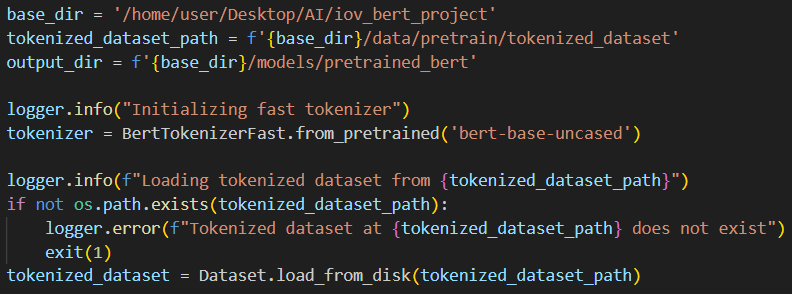


Trong đó:

* *k* là tổng số token bị che trong chuỗi đầu vào,
* *MASKj* là token bị che thứ *j*,
* *tokenc* là token thứ 𝑐 trong từ điển,
* *n* là kích thước batch,
* *P(tokenc | MASKj)* là xác suất token bị che thứ 𝑗 được mô hình dự đoán là token thứ 𝑐 trong từ điển,
* yjc là hàm nhị phân, bằng 1 nếu dự đoán đúng token, ngược lại bằng 0.

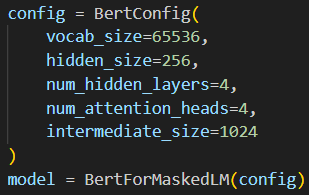
### Giải thích code pretrain.py

Để huấn luyện mô hình BERT cho nhiệm vụ tiền huấn luyện trong hệ thống IoV-BERT-IDS, đoạn code dưới đây triển khai quá trình huấn luyện mô hình BERTForMaskedLM từ thư viện Hugging Face Transformers với nhiệm vụ Masked Language Modeling - tương đương với nhiệm vụ Masked Byte Word Model (MBWM).



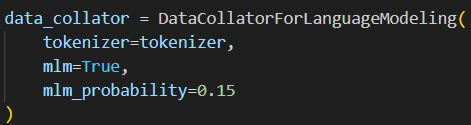
*Hình 21 - Cấu hình ban đầu và tải dữ liệu*

* Tokenizer loại BertTokenizerFast được sử dụng để xử lý các byte sentence đã được mã hóa thành token.
* Dữ liệu huấn luyện là tokenized\_dataset đã được xử lý sẵn và lưu dưới định dạng của thư viện datasets.



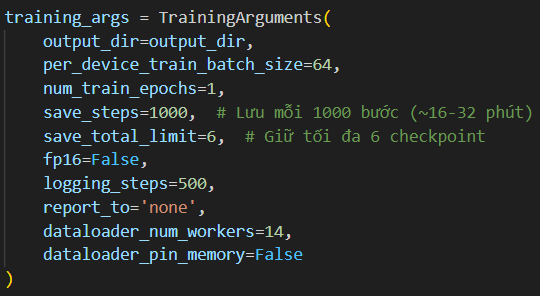
*Hình 22 - Khởi tạo mô hình BERT*

* vocab\_size = 65536: kích thước từ điển, tương ứng với toàn bộ không gian byte pair (2^16).
* hidden\_size = 256: kích thước vector ẩn cho mỗi token.
* num\_hidden\_layers = 4: số lớp transformer encoder.
* num\_attention\_heads = 4: số "đầu" attention trong mỗi lớp.
* intermediate\_size = 1024: kích thước lớp trung gian trong feed-forward.

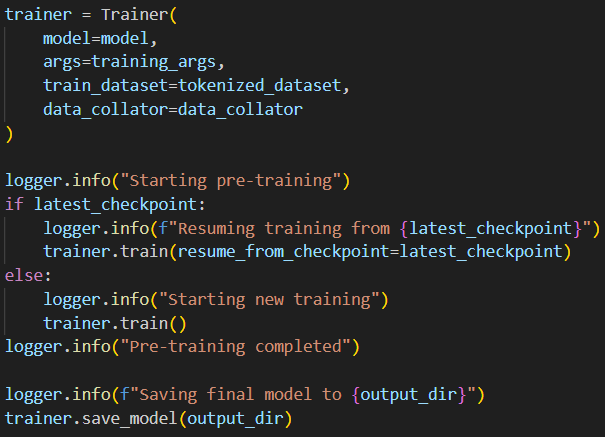


*Hình 23 - Cấu hình collator cho MBWM*

* DataCollatorForLanguageModeling tự động che 15% token đầu vào:
* 80% thay bằng [MASK].
* 10% thay bằng token ngẫu nhiên.
* 10% giữ nguyên.



*Hình 24 - Cấu hình tham số huấn luyện*

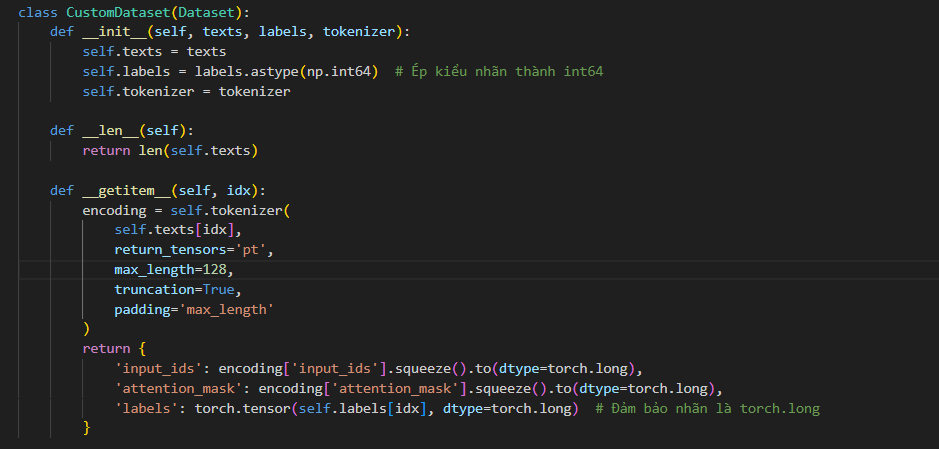


*Hình 25 - Huấn luyện mô hình*

* Sử dụng Trainer của Hugging Face để quản lý toàn bộ quá trình huấn luyện.
* Tự động xử lý việc nạp dữ liệu, tính toán loss, và cập nhật mô hình.

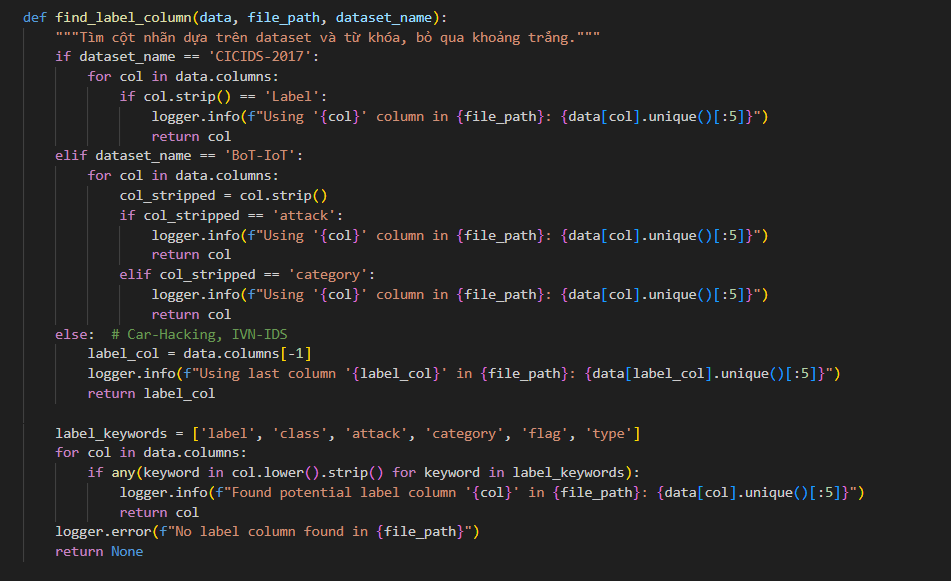
## Fine-tuning

* Là giai đoạn điều chỉnh mô hình BERT đã được huấn luyện trước (pre-trained) để phù hợp với các tác vụ cụ thể là phát hiện xâm nhập IoV. Quá trình này giúp mô hình học các đặc trưng chuyên biệt từ dữ liệu có nhãn của mạng trong xe (CAN) và ngoài xe (TCP/UDP).
* **Dữ liệu đầu vào:**
* Dữ liệu CAN thô được tiền xử lý bằng Unidirectional Semantic Extractor (USE) để tạo thành CAN Byte Sentences (CBS), tức là chuỗi các byte thập lục phân (hoặc thập phân, tùy cách triển khai) biểu diễn trường dữ liệu CAN (DATA[0-7]).
* Dữ liệu giao thông ngoài xe (TCP/UDP) được chuyển thành Traffic Byte Sentences (TBS), cũng ở dạng chuỗi thập lục phân.
* Cả CBS và TBS đều được chuyển đổi thông qua Semantic Extractor (SE) để đảm bảo biểu diễn token phù hợp với đầu vào của BERT.
* **Quy trình chính:**
* Tải và tiền xử lý dữ liệu từ các tệp .csv (CICIDS-2017, BoT-IoT) hoặc kết hợp .cbs.txt và .csv (Car-Hacking, IVN-IDS).
* Chuẩn hóa đặc trưng (features) và nhãn (labels), giới hạn số mẫu để cân bằng.
* Mã hóa văn bản bằng BertTokenizerFast và chia dữ liệu train/test.
* Tinh chỉnh mô hình BertForSequenceClassification với lớp phân loại đa lớp.
* Lưu checkpoint và đánh giá mô hình dựa trên độ chính xác (accuracy).
* **Thư viện sử dụng:**
* PyTorch: Quản lý tensor và huấn luyện mô hình.
* Transformers: Cung cấp mô hình BERT, tokenizer, và Trainer.
* scikit-learn: Chia dữ liệu và mã hóa nhãn.
* pandas/numpy: Xử lý dữ liệu.
* logging: Ghi log quá trình xử lý.
* glob/os: Quản lý tệp và thư mục.
* **Coding:**



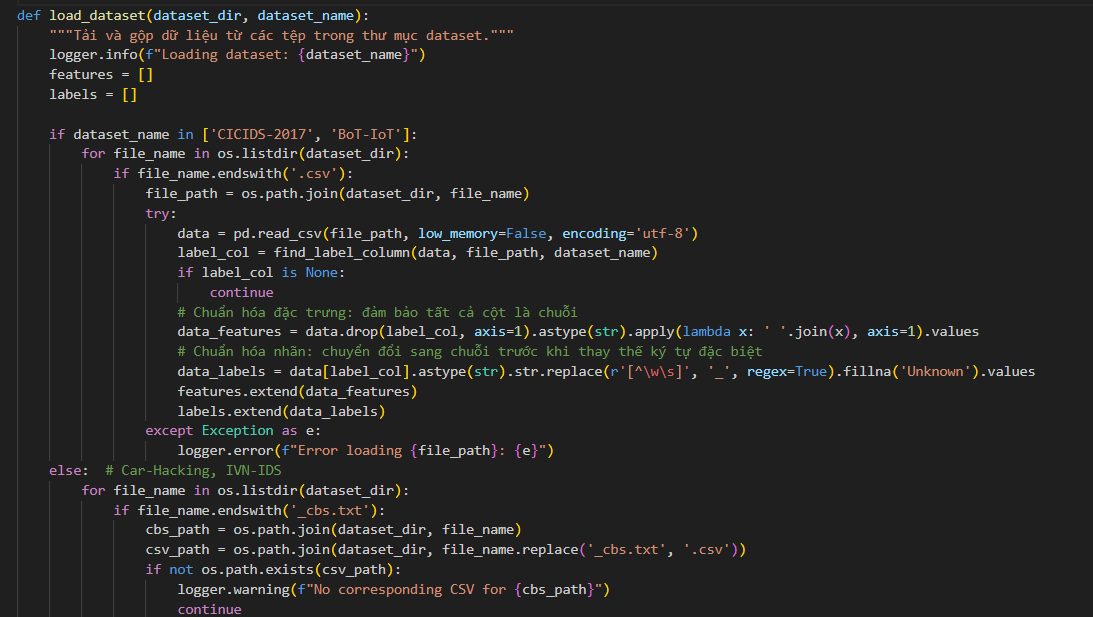
*Hình 26 - Lớp CustomDataset cung cấp cho mô hình BERT*

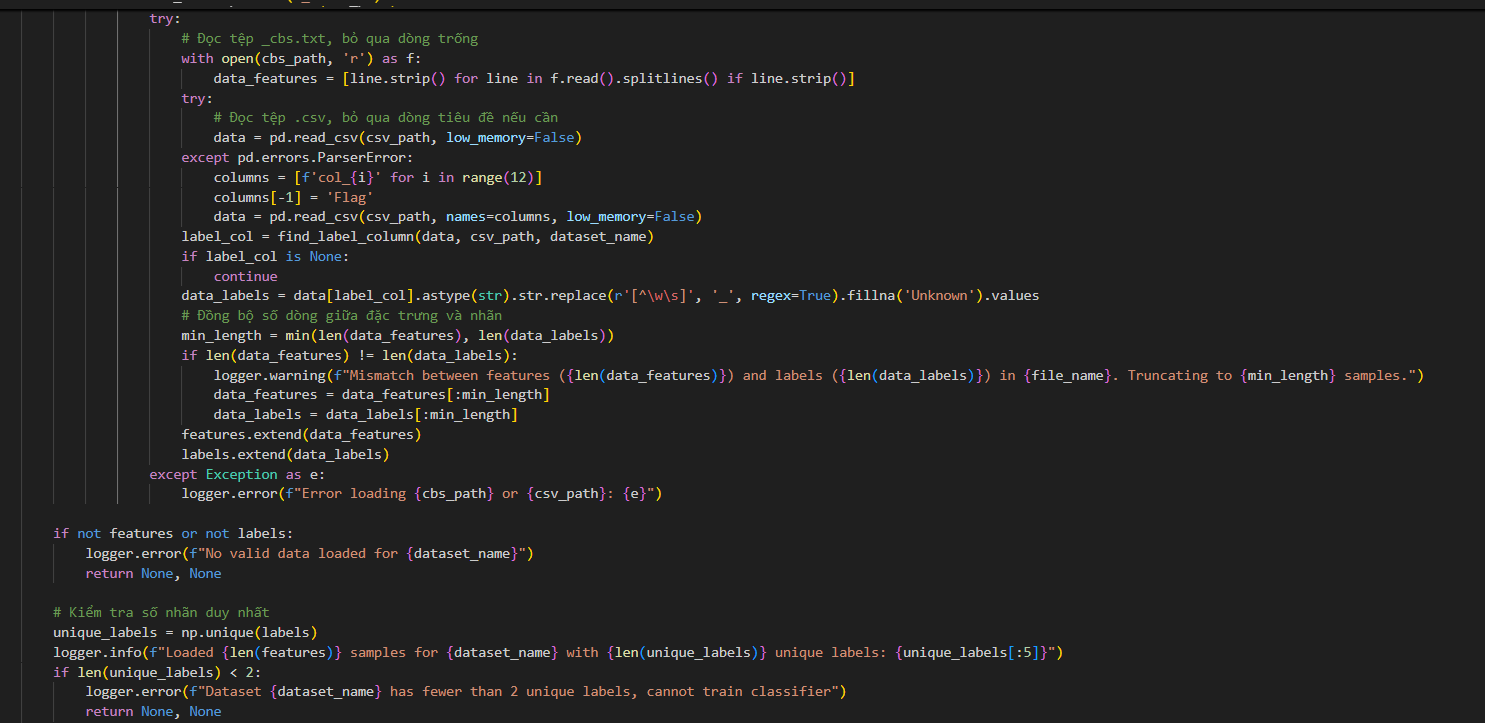
* Tạo dataset tùy chỉnh để cung cấp dữ liệu cho mô hình BERT. Chuẩn bị dữ liệu theo định dạng yêu cầu của BERT, đảm bảo tính tương thích với Trainer.
* Quy trình:
* Khởi tạo: Nhận danh sách văn bản (texts, ví dụ: CBS hoặc TBS), nhãn (labels), và tokenizer.
* Mã hóa văn bản: Sử dụng BertTokenizerFast để chuyển văn bản thành input\_ids và attention\_mask, với:
* max\_length=128: Giới hạn độ dài chuỗi tối đa 128 token.
* truncation=True: Cắt bớt chuỗi dài hơn 128 token.
* padding='max\_length': Đệm chuỗi ngắn hơn 128 token bằng 0.
* Đầu ra: Trả về dictionary chứa input\_ids, attention\_mask, và labels dưới dạng tensor torch.long.



*Hình 27 - Hàm find\_label\_column xác định cột chứa nhãn*

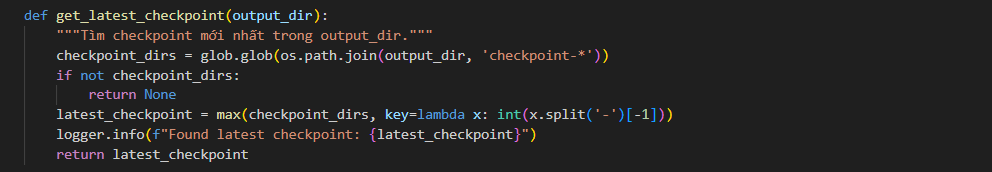
* Xác định cột chứa nhãn trong dữ liệu dựa trên tên tập dữ liệu hoặc từ khóa. Đảm bảo tìm đúng cột nhãn trong các tập dữ liệu có định dạng khác nhau.
* Quy trình:
* CICIDS-2017: Tìm cột có tên Label.
* BoT-IoT: Tìm cột attack hoặc category.
* Car-Hacking/IVN-IDS: Mặc định lấy cột cuối cùng.
* Dự phòng: Tìm cột có từ khóa như label, class, attack, …
* Ghi log giá trị nhãn duy nhất để kiểm tra.





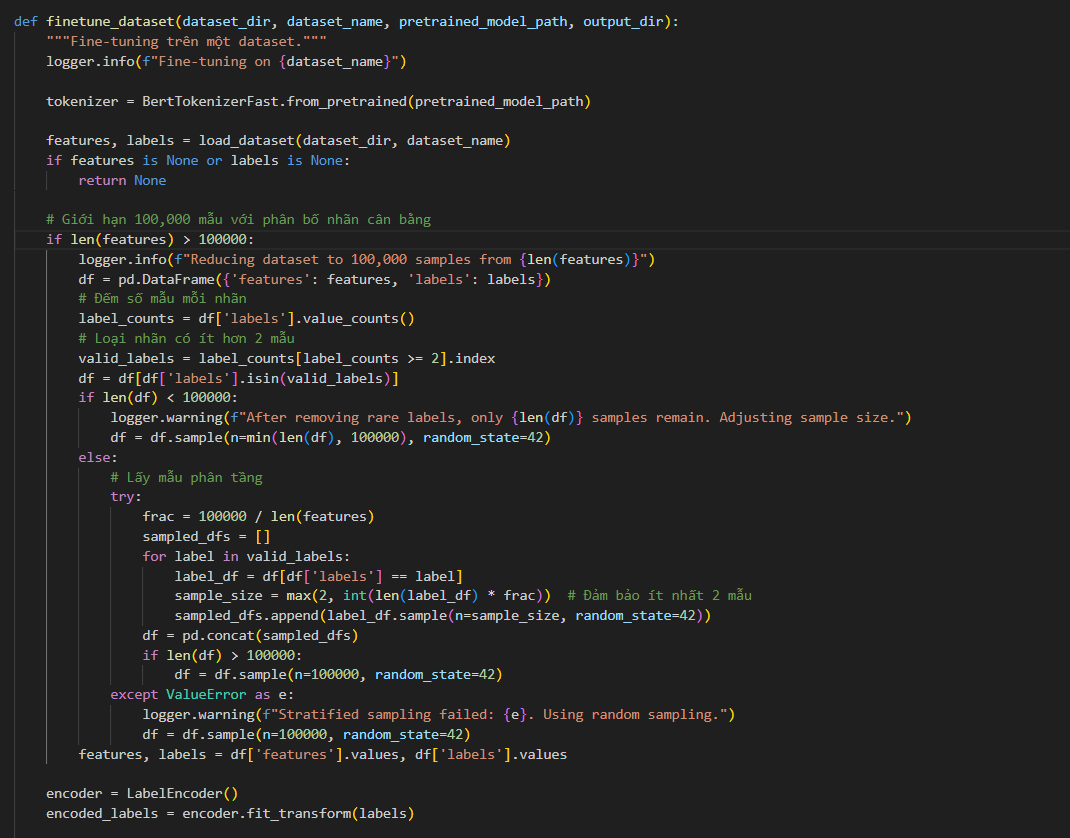
*Hình 28 - Hàm load\_dataset*

* Tải và gộp dữ liệu từ thư mục dataset. Chuẩn bị đặc trưng (CBS/TBS) và nhãn cho tinh chỉnh, hỗ trợ các định dạng dữ liệu khác nhau.
* Quy trình:
* CICIDS-2017/BoT-IoT:
* Đọc tệp .csv, lấy đặc trưng bằng cách nối tất cả cột (trừ cột nhãn) thành chuỗi, phân tách bằng dấu cách (tạo TBS).
* Chuẩn hóa nhãn: Chuyển thành chuỗi, thay ký tự đặc biệt bằng \_, điền Unknown cho giá trị thiếu.
* Car-Hacking/IVN-IDS:
* Đọc đặc trưng từ tệp \_cbs.txt (CBS, được tạo bởi USE)
* Đọc nhãn từ tệp .csv tương ứng.
* Đồng bộ số dòng giữa đặc trưng và nhãn bằng cách cắt bớt nếu không khớp.
* Kiểm tra: Đảm bảo có ít nhất 2 nhãn duy nhất để phân loại.



*Hình 29 - Hàm get\_lastest\_checkpoint*

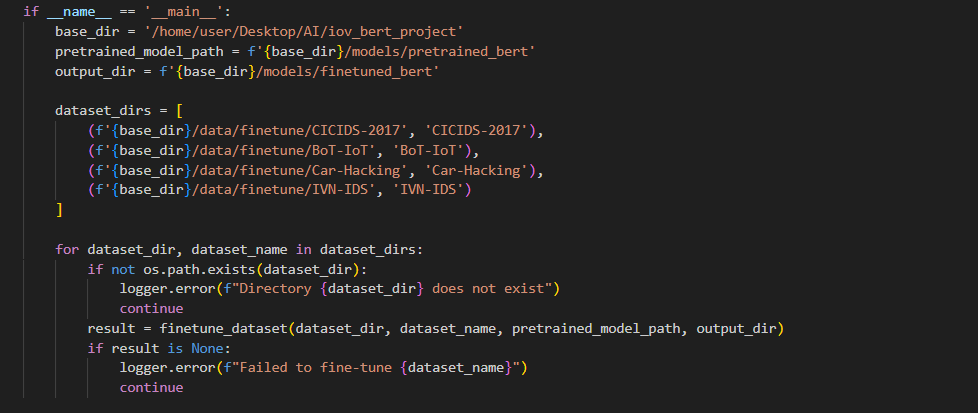
* Tìm checkpoint mới nhất trong thư mục đầu ra. Cho phép tiếp tục huấn luyện từ checkpoint trước đó, tiết kiệm thời gian nếu quá trình bị gián đoạn.





*Hình 30 - Hàm finetune\_dataset*

* Tinh chỉnh mô hình BERT trên một tập dữ liệu cụ thể. Tinh chỉnh mô hình để phân loại đa lớp, phát hiện xâm nhập.
* Quy trình:
* Tải tokenizer: Sử dụng BertTokenizerFast từ mô hình đã huấn luyện trước.
* Tải dữ liệu: Gọi load\_dataset để lấy đặc trưng và nhãn.
* Giới hạn mẫu:
* Nếu số mẫu > 100,000, giảm xuống 100,000 với phân bố nhãn cân bằng (lấy mẫu phân tầng theo nhãn, đảm bảo mỗi nhãn có ít nhất 2 mẫu).
* Mã hóa nhãn: Sử dụng LabelEncoder để chuyển nhãn thành số nguyên.
* Chia dữ liệu: Chia train/test (80/20) với ưu tiên phân tầng (stratified split)
* Tạo dataset: Khởi tạo CustomDataset cho train và test.
* Tải mô hình: Khởi tạo BertForSequenceClassification với số lớp nhãn (num\_labels).
* Cấu hình huấn luyện:
* Batch size: 32 (train/test).
* Epochs: 2.
* Lưu/evaluate mỗi 500 bước, giữ tối đa 3 checkpoint.
* Tối ưu dựa trên độ chính xác (accuracy).
* Sử dụng 12 worker cho dataloader.
* Huấn luyện: Sử dụng Trainer để huấn luyện, hỗ trợ tiếp tục từ checkpoint.
* Kết quả: Trả về trainer, dataset test, và nhãn test.



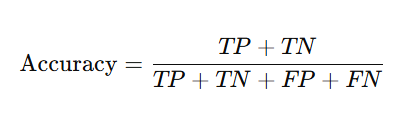
*Hình 31 - Hàm main của giai đoạn fine-tuning*

* Điều phối tinh chỉnh cho từng tập dữ liệu. Duyệt qua các thư mục dataset, kiểm tra tồn tại, và gọi finetune\_dataset. Tự động hóa tinh chỉnh trên nhiều tập dữ liệu.

# Kết quả thực nghiệm

## Các tiêu chí đánh giá

* Để đánh giá hiệu quả của hệ thống IoV-BERT-IDS trong việc phát hiện các cuộc tấn công mạng, nhóm nghiên cứu sử dụng các tiêu chí phổ biến trong lĩnh vực phân loại bao gồm Precision, Recall, F1-score và Accuracy, được tính toán dựa trên bốn tham số TP, TN, FP và FN :
* **Accuracy**: Tỷ lệ các mẫu được phân loại đúng trên tổng số mẫu. Đây là chỉ số đánh giá tổng thể độ chính xác của mô hình. Được tính bằng công thức:



Trong đó:

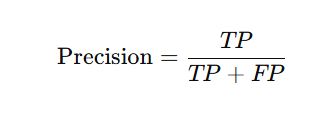
TP (True Positive): dự đoán đúng mẫu là tấn công.

TN (True Negative): dự đoán đúng mẫu là bình thường.

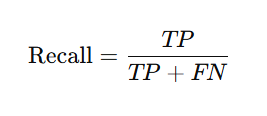
FP (False Positive): dự đoán nhầm mẫu bình thường là tấn công.

FN (False Negative): dự đoán nhầm mẫu tấn công là bình thường.

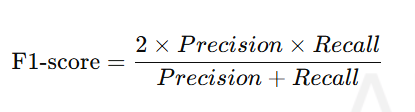
* **Precision**: Tỷ lệ các mẫu được mô hình dự đoán là tấn công mà thật sự là tấn công. Được tính bằng công thức:



* **Recall** (độ bao phủ) : Tỷ lệ các mẫu thật sự là tấn công mà mô hình dự đoán đúng. Được tính bằng công thức



* **F1-score**: là chỉ số cân bằng giữa Precision và Recall. Được tính bằng công thức:



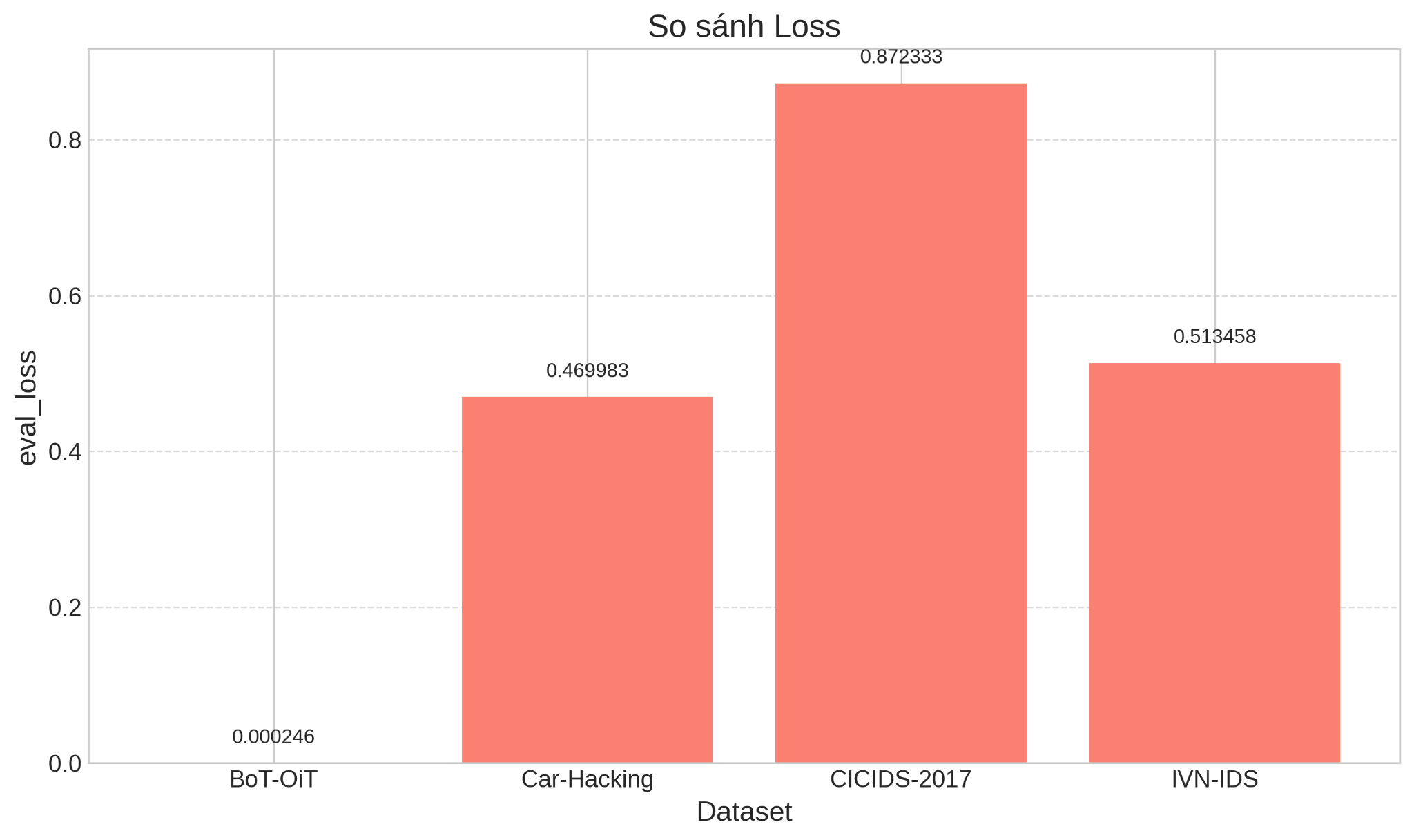
## Thực nghiệm

* Bảng kết quả đánh giá mô hình trên các tập dataset:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Loss | Accuracy | Precision | Recall | F1-score |
| BoT-IoT | 0.0002 | 0.99996625 | 0.99996617 | 0.99996625 | 0.9999661 |
| Car-Hacking | 0.4700 | 0.8472 | 0.7177 | 0.8472 | 0.7771 |
| CICIDS-2017 | 0.8723 | 0.8857 | 0.8875 | 0.8857 | 0.8866 |
| IVN-IDS | 0.5135 | 0.7817 | 0.6111 | 0.7817 | 0.6860 |



*Hình 32 - Bảng so sánh các chỉ số đánh giá*



*Hình 33 - Bảng so sánh tỉ lệ loss giữa các dataset*

* Mô hình đạt hiệu suất hoàn hảo trên tập dữ liệu BoT-IoT, với tất cả các chỉ số đạt ~100% và hàm mất mát cực kỳ thấp (~0.0002). Điều này cho thấy mô hình dự đoán chính xác gần như mọi mẫu trong tập kiểm tra (~800,000 mẫu, 20% của 4M mẫu).
* Nguyên nhân đạt kết quả như vậy:
* Dữ liệu BoT-IoT có thể có đặc điểm rõ ràng, giúp mô hình dễ phân loại.
* Quá trình fine-tuning trên BoT-IoT được thực hiện tốt, lượng dữ liệu đầu vào nhỏ.
* Nhưng hiệu suất ~100% chỉ ra tập dữ liệu quá đơn giản.
* Trên tập dữ liệu Car-Hacking: Accuracy và Recall (~84.72%) cho thấy mô hình dự đoán đúng ~84.72% mẫu và phát hiện được ~84.72% mẫu dương tính thực sự. Tuy nhiên, Precision thấp (~71.77%) chỉ ra rằng mô hình tạo ra nhiều dự đoán dương tính sai (false positives), làm giảm điểm F1 (~77.71%). Car-Hacking có dữ liệu phức tạp hơn, với các lớp như "R" (bình thường) và "T" (tấn công) hoặc nhiều lớp hơn.
* CICIDS-2017: Accuracy, Precision, Recall, và F1-score đều ~88-89%, cho thấy mô hình hoạt động ổn nhưng chưa xuất sắc. Các chỉ số này cân bằng, nghĩa là không có vấn đề nghiêm trọng về false positives hoặc false negatives. Loss cao (~0.8723) chỉ ra rằng mô hình gặp khó khăn trong việc khớp với nhãn thực tế, do dữ liệu phức tạp.
* Hiệu suất trên IVN-IDS là thấp nhất, Accuracy và Recall (~78.17%) cho thấy mô hình dự đoán đúng ~78.17% mẫu và phát hiện ~78.17% mẫu dương tính thực sự. Tuy nhiên, Precision rất thấp (~61.11%) chỉ ra rằng mô hình tạo ra nhiều dự đoán dương tính sai, kéo điểm F1 xuống (~68.60%).

# Kết luận

* Bài báo “IoV-BERT-IDS: Hybrid Network Intrusion Detection System in IoV Using Large Language Models” đưa ra giải pháp cho những hạn chế của các mô hình bảo mật truyền thống khi đối mặt với các kết nối phức tạp trong Internet of Vehicles (IoV). Từ đó, mô hình IoV-BERT-IDS được đề xuất sử dụng với việc triển khai mô hình BERT để thu thập biểu diễn tổng quát của raw data trong IoV. Nhờ cách tiếp cận này, độ chính xác và độ tinh chỉnh của mô hình tích hợp càng được nâng cao và uy tín hơn.
* Nhằm giải quyết thách thức về ngữ nghĩa không rõ ràng trong dữ liệu lưu lượng, hệ thống bộ trích xuất ngữ nghĩa sẽ chuyển đổi thông tin thô thành các cặp lưu lượng có ngữ nghĩa ngữ cảnh. Cách tiếp cận toàn diện này giúp mô hình hiểu rõ hơn về ngữ nghĩa ngữ cảnh tiềm ẩn trong lưu lượng IoV.
* Hơn nữa, việc đưa vào hai tác vụ tiền huấn luyện đã giúp IoV-BERT-IDS học được các đặc trưng ngữ cảnh hai chiều, góp phần đáng kể vào khả năng học và nắm bắt các mẫu và đặc trưng ngữ cảnh.

→ Hiệu quả của IoV-BERT-IDS được chứng minh thông qua quá trình kiểm thử toàn diện trên nhiều bộ dữ liệu: CICIDS, BoT-IoT, Car-Hacking, IVN-IDS.

* **Đánh giá kết quả chung:**
* **Hiệu suất:**

IoV-BERT-IDS được đánh giá cao với độ chính xác gần như tuyệt đối (0.9999) ở các tiêu chí đánh giá trên tập dữ liệu BoT-IoT, nhưng hiệu suất giảm rõ trên hai tập dữ liệu Car-Hacking và IVN-IDS, với Accuracy chỉ khoảng 0.88857 và 0.7817. Điều này cho thấy mô hình có sự không đồng đều trong hiệu suất khi xử lý các tập dữ liệu khác nhau, đặc biệt là trong các kịch bản liên quan đến mạng trong xe (in-vehicle networks).

CICIDS-2017 có hiệu năng ổn định, với các chỉ số dao động quanh mức 0.87-0.89, cho thấy mô hình phù hợp với các tập dữ liệu mạng tổng quát hơn.

* **Khả năng phát hiện:**

Mô hình hoạt động tốt nhất trong việc phát hiện các cuộc tấn công botnet trên BoT-IoT, nhưng khả năng phát hiện giảm đáng kể trên Car-Hacking và IVN-IDS, đặc biệt là Recall thấp, cho thấy mô hình gặp khó khăn trong việc phát hiện đầy đủ các mẫu tấn công trong môi trường mạng xe (in-vehicle networks). Đối với CICIDS-2017, khả năng phát hiện ổn định, phù hợp với các kịch bản mạng tổng quát, nhưng vẫn chưa đạt được hiệu suất tối ưu như BoT-IoT.

* **Ưu điểm:**

Hiệu năng vượt trội trên BoT-IoT: IoV-BERT-IDS đạt gần 100% trên tất cả các chỉ số, cho thấy khả năng xử lý mạnh mẽ các cuộc tấn công botnet trong môi trường IoT. Điều này có thể là do mô hình tận dụng tốt các đặc điểm ngữ nghĩa (semantic features) được chuyển đổi từ dữ liệu mạng thông qua Semantic Extractor (SE) và các giai đoạn pre-training như Masked Byte Word Model (MBWM) và Next Byte Sentence Prediction (NBSP)

Khả năng xử lý cả in-vehicle và extra-vehicle networks: IoV-BERT-IDS được thiết kế để phát hiện tấn công cả trong mạng nội bộ xe (IVN) và mạng bên ngoài, làm tăng tính linh hoạt của mô hình trong môi trường IoV.

* **Nhược điểm:**

Hiệu suất thấp trên Car-Hacking và IVN-IDS: Accuracy và Recall thấp trên các tập dữ liệu này cho thấy mô hình gặp khó khăn trong việc xử lý các cuộc tấn công trong mạng nội bộ xe (CAN bus). Điều này có thể do dữ liệu CAN bus có cấu trúc đơn giản hơn và ít ngữ cảnh hơn so với dữ liệu mạng thông thường, khiến mô hình BERT khó tận dụng được lợi thế của nó.

Phức tạp tính toán: Việc sử dụng BERT, một mô hình ngôn ngữ lớn, có thể làm tăng độ phức tạp tính toán, gây khó khăn khi triển khai trên các thiết bị IoV có tài nguyên hạn chế.

Chưa có đánh giá về các cuộc tấn công zero-day: một thách thức lớn đối với hệ thống mạng phương tiện.

* **Hướng phát triển trong tương lai:**
* Tối ưu triển khai mô hình BERT: BERT có yêu cầu cao về phần cứng như RAM/CPU/GPU - chưa thực tế để triển khai IoV-BERT-IDS trên phương tiện giao thông. Dùng MobileBERT (https://arxiv.org/abs/2004.02984).
* Tích hợp với hệ thống xe tự hành: triển khai hệ thống IDS theo thời gian thực trên IoV (CANBERT: A Language-based Intrusion Detection Model for In-vehicle Networks).
* Tích hợp với hệ sinh thái V2X/5G-Slicing: chia bbăngthoong cho các lưu lượng khác nhau trong IoV (Empowering the Internet of Vehicles with Multi-RAT5G Network Slicing) .

# Tham khảo

[1] Nguyen, Loc Gia, and Kohei Watabe. "A Method for Network Intrusion Detection Using Flow Sequence and BERT Framework." In *ICC 2023-IEEE International Conference on Communications*, pp. 3006-3011. IEEE, 2023.

[2] Berte, Dan-Radu. "Defining the iot." In *Proceedings of the international conference on business excellence*, vol. 12, no. 1, pp. 118-128. Pearson Educational, 2018.

[3] M. A. Ferrag et al., “Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices,” IEEE Access, vol. 12, pp. 23733–23750, Feb. 2024.

[4] Jeong, Woojin, Eunmin Choi, Hoseung Song, Minji Cho, and Ji-Woong Choi. "Adaptive controller area network intrusion detection system considering temperature variations." *IEEE Transactions on Information Forensics and Security* 17 (2022): 3925-3933.

[5] Caballero-Gil, Pino, and X. Wang. "Security issues in vehicular ad hoc networks." *Mobile ad-hoc networks: applications* (2011): 67-88.

[6] Yang, Li, and Abdallah Shami. "A transfer learning and optimized CNN based intrusion detection system for Internet of Vehicles." In ICC 2022-IEEE International Conference on Communications, pp. 2774-2779. IEEE, 2022.

[7] L. Yang, A. Moubayed, and A. Shami, “MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles,” IEEE Internet Things J., vol. 9, no. 1, pp. 616–632, Jan. 2022.

[8] Yu, Tianqi, Guodong Hua, Huaisheng Wang, Jianfeng Yang, and Jianling Hu. "Federated-lstm based network intrusion detection method for intelligent connected vehicles." In ICC 2022-IEEE International Conference on Communications, pp. 4324-4329. IEEE, 2022.