

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 2024

**KHẢO SÁT VÀ THỰC NGHIỆM MÔ HÌNH ENTC
DÙNG TRANSFORMER**

Mã số: 25-SV-2024-TH2

Thuộc nhóm ngành khoa học: Công nghệ thông tin.

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 2024

**KHẢO SÁT VÀ THỰC NGHIỆM MÔ HÌNH ENTC
DÙNG TRANSFORMER**

Mã số: 25-SV-2024-TH2

Thuộc nhóm ngành khoa học: Công nghệ thông tin.

Sinh viên thực hiện:

Kiều Tuấn Trung Anh	Giới tính: Nam	Dân tộc: Kinh
Huỳnh Chí Trung	Giới tính: Nam	Dân tộc: Kinh
Nguyễn Thúy Quỳnh	Giới tính: Nữ	Dân tộc: Kinh
Ngô Thị Duyên	Giới tính: Nữ	Dân tộc: Kinh

Lớp: E21CQCNTT01 – N

Khoa: Công nghệ thông tin.

Năm thứ: 4 / Số năm đào tạo: 4.5 năm.

Người hướng dẫn: TS.Nguyễn Hồng Sơn

TPHCM – 10/2024

MỤC LỤC

CHƯƠNG 1 TỔNG QUAN VỀ ENCRYPTION NETWORK TRAFFIC CLASSIFICATION (ENTC)	1
1.1 Giới thiệu về ENTC:	1
1.2 Khó khăn trong việc phân loại lưu lượng mã hóa:	1
1.3 Các yếu tố ảnh hưởng đến việc phân loại:	2
1.4 Phương pháp hiện tại trong ENTC:	3
1.4.1 Phương pháp thống kê (Statistical Methods):	3
1.4.2 Mô hình học sâu (Deep Learning Models):	4
1.5 Phương pháp tiên tiến trong ENTC:	4
CHƯƠNG 2 TỔNG QUAN VỀ TRANSFORMER	5
2.1 Giới thiệu về Transformer:	5
2.1.1 Nguồn gốc ra đời:	5
2.1.2 Lí do phát triển:	5
2.2 Nguyên lý hoạt động Transformer	6
2.2.1 Cấu trúc Encoder và Decoder:	7
2.2.2 Cơ chế Self-Attention:	7
2.2.3 Multi-Head Attention	9
2.3 Các thành phần chính của Transformer:	10
2.3.1 Input Embedding:	10
2.3.2 Positional encoding:	11
2.3.3 Kết nối Residual:	11
2.3.4 Khối Feed-Forward:	11
2.4 Ứng dụng Transformer trong phân loại lưu lượng mạng:	11
CHƯƠNG 3 MÔ HÌNH ET-BERT	13
3.1 Mô hình kiến trúc	13
3.2 Biểu diễn lưu lượng Datagram2Token	13
3.2.1 Bộ tạo BURST	14
3.2.2 BURST2Token	14
3.2.3 Token2Embedding	15
3.3 Pre-training ET-BERT	16
3.4 Fine-tuning ET-BERT	17
CHƯƠNG 4 KIỂM THỬ VÀ KẾT QUẢ	19

4.1 Yêu cầu chuẩn bị.....	19
4.2 Mô hình thực nghiệm	20
4.3 Kết quả và đánh giá.....	25
KẾT LUẬN VÀ KIẾN NGHỊ.....	26
PHỤ LỤC	27
DANH MỤC TÀI LIỆU THAM KHẢO	28

DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH

HÌNH 1.1: Bốn Phương pháp chính để phân loại lưu lượng được mã hóa: (a) So khớp dấu vân tay dựa trên đặc điểm văn bản thuần túy. (b) Học máy dựa trên đặc điểm thống kê. (c) ML dựa trên đặc điểm lưu lượng thô. (d) Đào tạo trước dựa trên lưu lượng thô.

HÌNH 2.1: Mô hình Transformer

HÌNH 2.2: Cơ chế Self-Attention

HÌNH 2.3: Cách hoạt động của bộ phận self-attention

HÌNH 2.4: Multi-Head Attention

HÌNH 2.5: Input Embedding

HÌNH 3.1: Tổng quan mô hình ET-BERT

HÌNH 4.1: Tổ chức file của model

HÌNH 4.2: Tập hợp file dataset

HÌNH 4.3: file chứa model đã được pretrain

HÌNH 4.4: Tập hợp 120 ứng dụng web sử dụng cho quá trình pretrain

HÌNH 4.5: Tổ chức của file dataset

HÌNH 4.6: Lệnh tiến hành quá trình fine-tune

HÌNH 4.7: Kết quả sau quá trình fine-tune

HÌNH 4.8: Độ chính xác của model sau quá trình fine-tune

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

CNN:	C onvolution N eural N etwork
LSTM:	L ong S hort T erm M emory
GRUS:	G ated R ecurrent U nits
ENTC:	E ncryted N etwork T raffic C lassification
DOM:	D ocument O bject M odel
SBP:	S ame-origin B urst P rediction
MBM:	M asked B urst M odel
CSTNET:	C hina S cience A nd T echnology N etwork
ETBERT:	E ncrypted T raffic B idirectional E ncoder R epresentations from T ransformer
TLS:	T ransport L ayer S ecurity
ENTC:	E ncrypted T raffic C lassification
IDS:	I ntrusion D etection S ystems
HTTPS:	H yper T ext T ransport P rotocol S ecurity
VPN:	V irtual P rivate N etwork
QoS:	Q uality O f S ervice
DPI:	D ep P acket I nspection
SSL:	S ecure S ocket L ayer

MỞ ĐẦU

CHƯƠNG 1 TỔNG QUAN VỀ ENCRYPTION NETWORK TRAFFIC CLASSIFICATION (ENTC) [1]

1.1 Giới thiệu về ENTC:

- Phân loại lưu lượng mạng được mã hóa (Encryption Network Traffic Classification - ENTC) là một lĩnh vực quan trọng trong an ninh mạng, tập trung vào việc nhận dạng và phân loại các loại lưu lượng dữ liệu trên mạng Internet, đặc biệt là khi dữ liệu đã được mã hóa. Với sự gia tăng mạnh mẽ của việc sử dụng các giao thức mã hóa như HTTPS, VPN, hay Tor, các phương pháp kiểm tra gói tin truyền thống không còn hiệu quả. Điều này đặt ra thách thức lớn cho việc giám sát, quản lý chất lượng dịch vụ (QoS), và phát hiện các mối đe dọa tiềm ẩn trên mạng.
- Các phương pháp hiện đại cho ENTC chủ yếu dựa trên máy học (machine learning) và học sâu (deep learning) để phân tích các đặc điểm không phải nội dung, như metadata, thời gian luồng dữ liệu, và các đặc điểm hành vi của mạng. Điều này cho phép phát hiện và phân loại chính xác loại lưu lượng mà không cần giải mã nội dung, giúp tăng cường hiệu quả bảo mật và quản lý mạng.
- Trong bối cảnh đó, **ET-BERT** là một mô hình tiêu biểu áp dụng kiến trúc Transformer để giải quyết các thách thức của ENTC. ET-BERT sử dụng phương pháp tiền huấn luyện (pre-training) trên dữ liệu quy mô lớn và không nhãn, nhằm học các biểu diễn ngữ cảnh hóa sâu (contextualized representation) của gói tin (datagram). Mô hình này cho phép nhận diện các loại lưu lượng khác nhau, kể cả khi dữ liệu được mã hóa, bằng cách dựa trên các mẫu lưu lượng, thời gian, và metadata thay vì nội dung của gói tin.
- ET-BERT đã chứng minh hiệu quả vượt trội trong việc phân loại các loại lưu lượng phức tạp như VPN hay Tor, thậm chí trên các nền tảng khác nhau như Android. Đặc biệt, mô hình có khả năng tổng quát hóa tốt trên dữ liệu mới chưa được huấn luyện, khắc phục được hạn chế của các phương pháp trước đó vốn phụ thuộc nhiều vào các tập dữ liệu có nhãn lớn. Nhờ đó, ET-BERT không chỉ là một giải pháp mạnh mẽ trong việc nâng cao an ninh mạng mà còn là một công cụ hữu ích trong quản lý lưu lượng và đảm bảo chất lượng dịch vụ.

1.2 Khó khăn trong việc phân loại lưu lượng mã hóa:

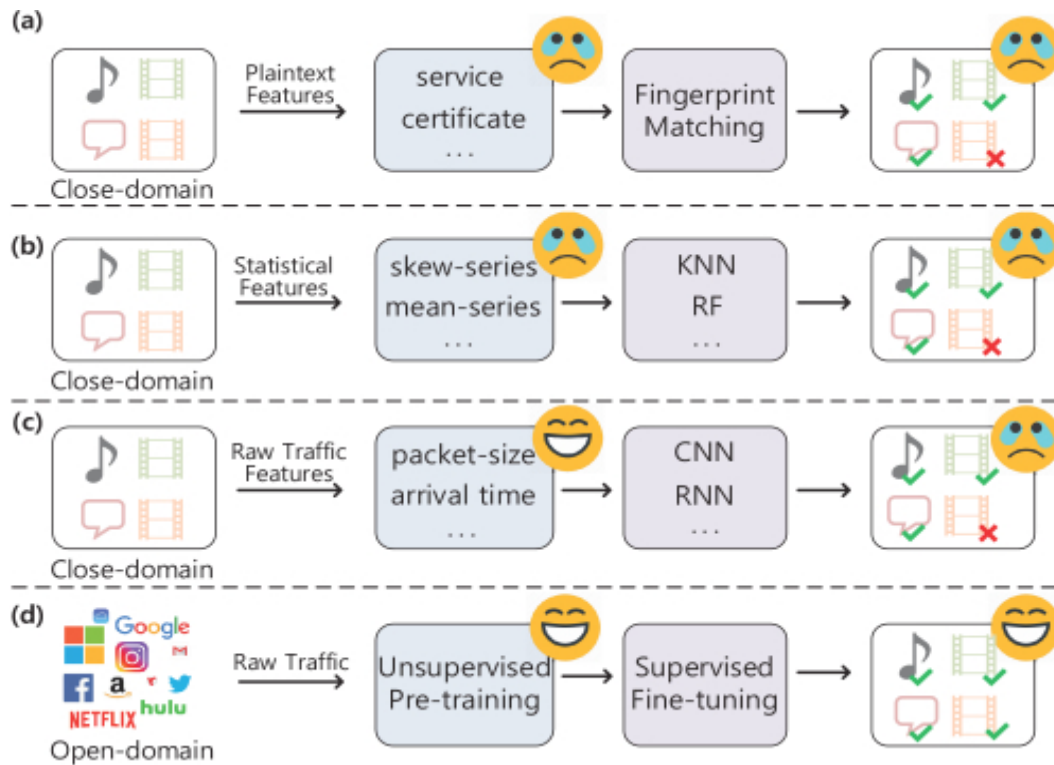
- **Thiếu thông tin chi tiết về payload:** Do lưu lượng được mã hóa, việc trích xuất thông tin ứng dụng từ payload trở nên khó khăn.
- **Sự đa dạng của giao thức và phương pháp mã hóa:** Các ứng dụng sử dụng nhiều giao thức và phương pháp mã hóa khác nhau, gây khó khăn cho việc phân loại thống nhất.

- **Tính phức tạp của lưu lượng mạng:** Lưu lượng mạng có thể mang nhiều đặc điểm phức tạp như kích thước gói tin, thời gian giữa các gói tin, mô hình lưu lượng, v.v., ảnh hưởng đến hiệu quả phân loại.

1.3 Các yếu tố ảnh hưởng đến việc phân loại:

- **Sự phức tạp của lưu lượng:** Một số giao thức mạng có cấu trúc phức tạp, khiến việc phân loại trở nên khó khăn hơn, ví dụ: Giao thức, mã hóa, kích thước gói tin, v.v.
- **Các phương pháp mã hóa khác nhau:** Có nhiều loại mã hóa khác nhau, mỗi loại có đặc điểm riêng, ảnh hưởng đến khả năng phân loại, ví dụ: VPN, SSL/TLS, Tor, v.v.
- **Phân loại lưu lượng mã hóa:** Khi lưu lượng được mã hóa, các phương pháp truyền thống dựa vào nội dung gói tin trở nên không hiệu quả:
 - Các phương pháp truyền thống như kiểm tra gói tin sâu (DPI) không thể áp dụng cho lưu lượng mã hóa vì dữ liệu trong các gói tin đã được mã hóa và không thể đọc được các mẫu và từ khóa.
- **Các phương pháp mã hóa khác nhau:** Các kỹ thuật mã hóa mới nổi như TLS 1.3 làm cho việc phân loại lưu lượng trở nên phức tạp hơn vì thông tin plaintext ngày càng ít hoặc bị làm mờ:
 - Các công trình ban đầu sử dụng thông tin plaintext còn lại trong lưu lượng mã hóa (ví dụ như chứng chỉ) để xây dựng dấu vân tay và thực hiện phân loại bằng cách khớp dấu vân tay. Tuy nhiên, các phương pháp này không áp dụng được cho các kỹ thuật mã hóa mới nổi (ví dụ: TLS 1.3) vì thông tin plaintext ngày càng ít hoặc bị làm mờ
- **Khả năng thích ứng:** Phương pháp phân loại cho một loại lưu lượng mã hóa cụ thể không thể thích ứng tốt với môi trường mới hoặc các chiến lược mã hóa chưa thấy trước:
 - Hơn nữa, do sự phát triển nhanh chóng của công nghệ mã hóa, các phương pháp phân loại lưu lượng cho một loại lưu lượng mã hóa cụ thể không thể thích ứng tốt với môi trường mới hoặc các chiến lược mã hóa chưa thấy trước.
- **Sự phức tạp của lưu lượng:** Lưu lượng ngày càng phức tạp do sự phát triển nhanh chóng của các ứng dụng và trang web:
 - Rất khó để thiết kế các đặc trưng thống kê chung để đối phó với lượng lớn các ứng dụng và trang web ngày càng trở nên phức tạp.

1.4 Phương pháp hiện tại trong ENTC:



HÌNH 1.1: Bốn Phương pháp chính để phân loại lưu lượng được mã hóa: (a) So khớp dấu vân tay dựa trên đặc điểm văn bản thuần túy. (b) Học máy dựa trên đặc điểm thống kê. (c) ML dựa trên đặc điểm lưu lượng thô. (d) Đào tạo trước dựa trên lưu lượng thô.

- Một số nghiên cứu đề xuất sử dụng thông tin trường giao thức không mã hóa. Ví dụ, FlowPrint trích xuất các đặc trưng về thiết bị, chứng chỉ, kích thước và thời gian để đại diện cho mỗi luồng và xây dựng thư viện dấu vân tay bằng cách phân cụm và tương quan chéo. Tuy nhiên, những dấu vân tay này dễ bị thay đổi trong các mạng giao tiếp ảo và mất đi ý nghĩa đúng đắn của chúng, trong khi mô hình của chúng tôi không phụ thuộc vào bất kỳ thông tin plaintext nào.

1.4.1 Phương pháp thống kê (Statistical Methods):

- Các nghiên cứu về lưu lượng mã hóa chủ yếu khai thác các thuộc tính thống kê của lưu lượng để không phụ thuộc vào mã hóa. Ví dụ, AppScanner sử dụng các đặc trưng thống kê của kích thước gói tin để huấn luyện các bộ phân loại rừng ngẫu nhiên, trong khi BIND [2] cũng khai thác các đặc trưng thống kê về tính thời gian. Tuy nhiên, việc thiết kế các đặc trưng thống kê chung để đối phó với các ứng dụng và trang web ngày càng phức tạp là rất khó khăn, trong khi mô hình của chúng tôi không cần phải dựa vào các đặc trưng do con người thiết kế.

1.4.2 Mô hình học sâu (Deep Learning Models):

- Việc phân loại lưu lượng mã hóa bằng cách sử dụng học sâu có giám sát đã trở thành một phương pháp phổ biến, tự động trích xuất các đặc trưng phân biệt thay vì phụ thuộc vào thiết kế thủ công. DF [3] sử dụng mạng nơ-ron tích chập (CNNs) và FS-Net [4] sử dụng mạng nơ-ron hồi quy (RNNs) để tự động trích xuất các đại diện từ chuỗi kích thước gói tin thô của lưu lượng mã hóa, trong khi Deeppacket và TSCRNN đang mô tả các payload thô. Tuy nhiên, phương pháp này phụ thuộc vào một lượng lớn dữ liệu huấn luyện có giám sát để nắm bắt các đặc trưng hợp lệ, do đó học các đại diện bị thiên lệch trong dữ liệu không cân bằng, trong khi mô hình của chúng tôi không phụ thuộc vào dữ liệu được gán nhãn lớn.

1.5 Phương pháp tiên tiến trong ENTC:

- **Mô hình tiền huấn luyện (Pre-training Models):** Các mô hình tiền huấn luyện đã có bước đột phá lớn trong các lĩnh vực khác nhau như xử lý ngôn ngữ tự nhiên và thị giác máy tính. Các phương pháp dựa trên tiền huấn luyện sử dụng dữ liệu không nhãn lớn để học các biểu diễn dữ liệu không thiên vị, sau đó có thể dễ dàng chuyển giao cho các nhiệm vụ hậu kỳ.
- **ET-BERT:** Mô hình mã hóa lưu lượng Đa chiều từ Bộ mã hóa Transformer (ET-BERT) là một mô hình tiền huấn luyện mới để phân loại lưu lượng mã hóa. Nó học các biểu diễn lưu lượng chung từ lưu lượng mã hóa không nhãn quy mô lớn và có khả năng tổng quát hóa lớn, đạt hiệu suất tốt nhất trên nhiều nhiệm vụ phân loại lưu lượng mã hóa.

CHƯƠNG 2 TỔNG QUAN VỀ TRANSFORMER

2.1 Giới thiệu về Transformer:

2.1.1 Nguồn gốc ra đời:

- Các RNN và biến thể của chúng đã là các phương pháp tiên tiến cho các bài toán mô hình hóa và chuyển dịch chuỗi, như mô hình ngôn ngữ và dịch máy. Tuy nhiên, các khó khăn cố hữu của việc tính toán tuần tự đã thúc đẩy nhu cầu phát triển một kiến trúc mới. Transformer được phát triển nhằm đáp ứng các giới hạn và thách thức của mạng nơ-ron hồi quy (RNN) và các biến thể của nó, chẳng hạn như long short-term memory (LSTM) và gated recurrent units (GRUs), vốn là các phương pháp tiếp cận tiên tiến cho các bài toán mô hình hóa và chuyển dịch chuỗi như mô hình ngôn ngữ và dịch máy. Việc phát triển này có sự đóng góp của nhiều nhà nghiên cứu, những người đã đề xuất và tinh chỉnh ý tưởng thay thế RNN bằng các cơ chế self-attention.

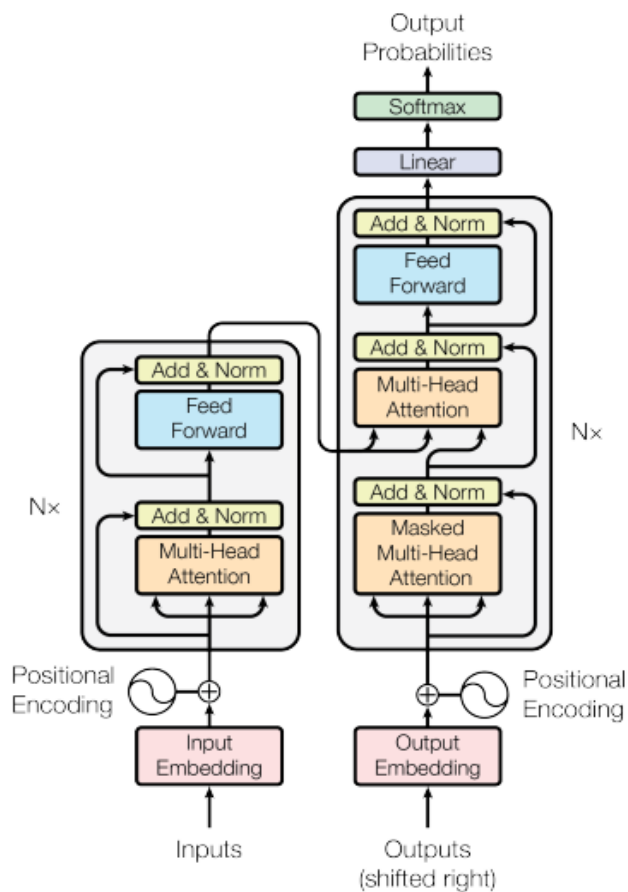
2.1.2 Lí do phát triển:

- **Giới hạn của tính toán tuần tự:** RNN và các biến thể của nó tính toán các chuỗi một cách tuần tự, sắp xếp các vị trí ký hiệu với các bước tính toán, điều này ngăn cản việc song song hóa trong các ví dụ huấn luyện. Bản chất tuần tự này trở thành một vấn đề nan giải, đặc biệt là đối với các chuỗi dài hơn, do các giới hạn về bộ nhớ hạn chế việc xử lý theo lô.
- **Hiệu quả và khả năng mở rộng:** Các nỗ lực cải thiện hiệu quả của RNN thông qua các thủ thuật phân tích và tính toán điều kiện đã mang lại những cải tiến đáng kể, nhưng giới hạn vốn có của tính toán tuần tự vẫn còn. Nhu cầu về các mô hình có thể song song hóa các tính toán để xử lý các chuỗi dài hơn hiệu quả hơn là một động lực chính.
- **Cơ chế Attention:** Cơ chế Attention đã chứng tỏ hiệu quả trong việc nâng cao mô hình hóa chuỗi bằng cách cho phép các mô hình xem xét các phụ thuộc giữa bất kỳ vị trí nào trong các chuỗi đầu vào hoặc đầu ra mà không cần quan tâm đến khoảng cách của chúng. Tuy nhiên, chúng thường được sử dụng cùng với các mạng hồi quy, vốn vẫn gặp phải các giới hạn của tính toán tuần tự.
- **Các phụ thuộc toàn cầu và song song hóa:** Kiến trúc Transformer được đề xuất để tận dụng hoàn toàn các cơ chế tự chú ý, loại bỏ hồi quy và cho phép mô hình tạo ra các phụ thuộc toàn cầu giữa các chuỗi đầu vào và đầu ra một cách hiệu quả hơn. Kiến trúc này cải thiện đáng kể khả năng song song hóa, giúp đạt được hiệu suất hàng đầu trong chất lượng dịch với thời gian huấn luyện ngắn hơn nhiều so với các mô hình dựa trên RNN.
- **Công việc trước đó và các đổi mới:** Các mô hình trước đó như Extended Neural GPU, ByteNet, và ConvS2S đã sử dụng mạng nơ-ron tích chập để song song hóa các tính toán, nhưng chúng vẫn gặp khó khăn trong việc học các phụ thuộc giữa các vị trí xa. Việc sử

dụng tự chú ý của Transformer giảm số lượng phép tính cần thiết để liên kết tín hiệu giữa các vị trí thành một số không đổi, do đó giải quyết được vấn đề này.

2.2 Nguyên lý hoạt động Transformer

- Đa số các mô hình chuyển đổi chuỗi (neural sequence transduction) đều có cấu trúc mã hóa-giải mã (encoder-decoder). Ở đây, bộ mã hóa sẽ ánh xạ một chuỗi đầu vào gồm các biểu diễn ký hiệu (x_1, \dots, x_n) thành một chuỗi các biểu diễn liên tục $z = (z_1, \dots, z_n)$. Với z đã có, bộ giải mã sẽ tạo ra một chuỗi đầu ra (y_1, \dots, y_m) gồm các ký hiệu, từng phần tử một. Tại mỗi bước, mô hình hoạt động theo cách tự hồi quy (auto-regressive), nghĩa là sử dụng các ký hiệu đã được sinh ra trước đó làm đầu vào bổ sung khi tạo ra ký hiệu tiếp theo.



HÌNH 2.1: Mô hình Transformer

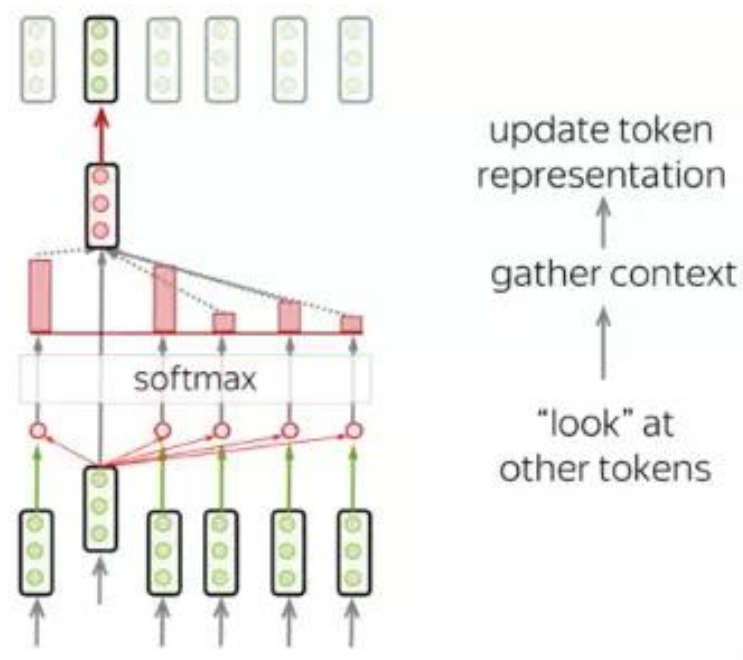
- Transformer tuân theo kiến trúc tổng thể này bằng cách sử dụng các lớp self-attention xếp chồng lên nhau và các lớp kết nối hoàn toàn theo từng điểm (point-wise, fully connected) cho cả bộ mã hóa và bộ giải mã, như được hiển thị ở nửa bên trái và bên phải của HÌNH 2.1 tương ứng.

2.2.1 Cấu trúc Encoder và Decoder:

- **Encoder:** được cấu thành từ một stack gồm $N = 6$ lớp giống hệt nhau. Mỗi lớp có lớp con (sub-layer). Lớp con đầu tiên là multi-head self-attention, và lớp con thứ hai là một mạng nơ-ron tích chập đơn giản theo vị trí (position-wise fully connected feed-forward network). Sử dụng một kết nối dư (residual connection) xung quanh mỗi lớp con, sau đó là chuẩn hóa lớp (layer normalization). Điều này có nghĩa là, đầu ra của mỗi lớp con là $\text{LayerNorm}(x + \text{Sublayer}(x))$, trong đó $\text{Sublayer}(x)$ là hàm được thực hiện bởi chính tiểu lớp đó. Để tạo điều kiện cho các kết nối dư này, tất cả các lớp con trong mô hình, cũng như các lớp nhúng (embedding layer), đều tạo ra đầu ra có kích thước là $d_{\text{model}} = 512$.
- **Decoder:** Decoder cũng được cấu thành từ một stack gồm $N = 6$ lớp giống hệt nhau. Ngoài hai lớp con trong mỗi lớp của encoder, decoder chèn thêm một lớp con thứ ba, thực hiện multi-head attention qua đầu ra của encoder stack. Tương tự như encoder, sử dụng các kết nối dư xung quanh mỗi lớp con, sau đó là chuẩn hóa lớp. Chúng tôi cũng điều chỉnh self-attention sub-layer trong decoder stack để ngăn các vị trí không chú ý đến các vị trí tiếp theo. Việc che (masking) này, kết hợp với thực tế rằng các nhúng đầu ra được bù một vị trí, đảm bảo rằng các dự đoán cho vị trí i chỉ có thể phụ thuộc vào các đầu ra đã biết ở các vị trí nhỏ hơn i .

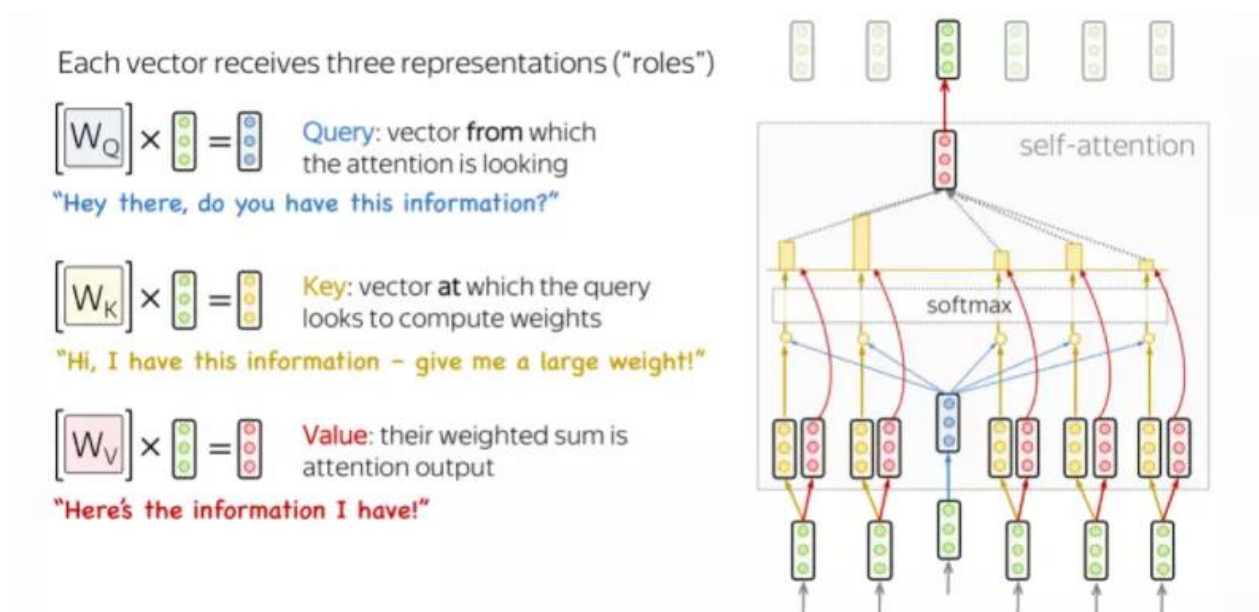
2.2.2 Cơ chế Self-Attention:

- Có thể khẳng định rằng self-attention chính là thành phần quan trọng nhất của transformer. Sự khác biệt là, trong khi cơ chế attention sẽ tính toán dựa trên trạng thái của decoder ở time-step hiện tại và tất cả các trạng thái ẩn của encoder. Còn self-attention có thể hiểu là attention trong một câu, khi từng thành phần trong câu sẽ tương tác với nhau. Từng token sẽ "quan sát" các tokens còn lại trong, thu thập ngữ cảnh của câu và cập nhật vector biểu diễn.



HÌNH 2.2: Cơ chế Self-Attention

- Để xây dựng cơ chế self-attention ta cần chú ý đến hoạt động của 3 vector biểu diễn cho mỗi từ lần lượt là:
 - **Query:** hỏi thông tin
 - **Key:** trả lời rằng nó có một số thông tin
 - **Value:** trả về thông tin đó
- Query được sử dụng khi một token "quan sát" những tokens còn lại, nó sẽ tìm kiếm thông tin xung quanh để hiểu được ngữ cảnh và mối quan hệ của nó với các tokens còn lại. Key sẽ phản hồi yêu cầu của Query và được sử dụng để tính trọng số attention. Cuối cùng, Value được sử dụng trọng số attention vừa rồi để tính ra vector đại diện (attention vector). Trong ảnh 3 ma trận W_Q , W_K và W_V chính là các hệ số mà mô hình cần huấn luyện.



HÌNH 2.3: Cách hoạt động của bộ phận self-attention

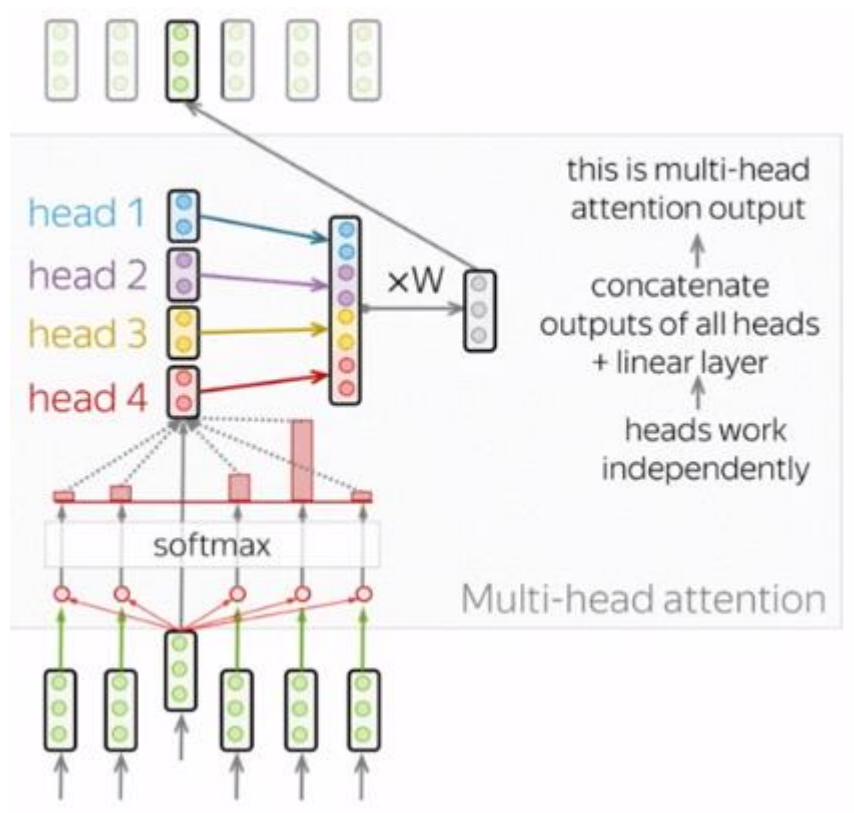
- Biểu thức để tính attention vector như sau:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

- Với d_k là số chiều của vector Key với mục đích tránh tràn luồng!
- Dựa vào ý nghĩa của q, k, v có thể giải thích được công thức của Transformers Attention:
 - $q.k$ qua hàm softmax để đưa ra xác suất của từ liên quan nhất với từng từ được hỏi tương ứng.
 - Sau đó nhân với v để đưa ra giá trị dựa vào sự tương quan đó.

2.2.3 Multi-Head Attention

- Thông thường, để có thể hiểu được vai trò của một từ trong một câu ta cần hiểu được sự liên quan giữa từ đó và các thành phần còn lại trong câu. Điều này rất quan trọng trong quá trình xử lý câu đầu vào ở ngôn ngữ A và cả trong quá trình tạo ra câu ở ngôn ngữ B. Vì vậy, mô hình cần phải tập trung vào nhiều thứ khác nhau, cụ thể là thay vì chỉ có một cơ chế self-attention như đã giới thiệu hay còn gọi là 1 "head" thì mô hình sẽ có nhiều "heads" mỗi head sẽ tập trung vào khía cạnh về sự liên quan giữa từ và các thành phần còn lại. Đó chính là multi-head attention.



HÌNH 2.4: Multi-Head Attention

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_n)W$$

$$head_i = Attention(Q_i, K_i, V_i)$$

- Khi triển khai, ta cần dựa vào query, key và value để tính toán cho từng head. Sau đó ta sẽ concat các ma trận thu được để thu được ma trận của multi-head attention. Để có đầu ra có cùng kích thước của đầu vào thì cần nhân với ma trận W.

2.3 Các thành phần chính của Transformer:

2.3.1 Input Embedding:

- Máy tính không hiểu câu chữ mà chỉ đọc được số, vector, ma trận; vì vậy ta phải biểu diễn câu chữ dưới dạng vector, gọi là input embedding. Điều này đảm bảo các từ gần nghĩa có vector gần giống nhau. Hiện đã có khá nhiều pretrained word embeddings như GloVe, Fasttext, gensim Word2Vec,... cho bạn lựa chọn.

Input Embedding



HÌNH 2.5: Input Embedding

2.3.2 Positional encoding:

- Bởi vì transformer không có các mạng hồi tiếp hay mạng tích chập nên nó sẽ không biết được thứ tự của các token đầu vào. Vì vậy, cần phải có cách nào đó để cho mô hình biết được thông tin này. Đó chính là nhiệm vụ của positional encoding. Như vậy, sau bước nhúng từ (embedding layers) để thu được các tokens thì ta sẽ cộng nó với các vector thể hiện vị trí của từ trong câu.

2.3.3 Kết nối Residual:

- Kết nối residual bản chất rất đơn giản: thêm đầu vào của một khối vào đầu ra của nó. Với kết nối này giúp mạng có thể chồng được nhiều layers. Như trên hình, kết nối residual sẽ được sử dụng sau các khối FFN và khối attention. Như trên hình từ "Add" trong "Add & Norm" sẽ thể hiện cho kết nối residual.

2.3.4 Khối Feed-Forward:

- Đây là khối cơ bản, sau khi thực hiện tính toán ở khối attention ở mỗi lớp thì khối tiếp theo là FFN. Có thể hiểu là cơ chế attention giúp thu thập thông tin từ những tokens đầu vào thì FFN là khối xử lý những thông tin đó.

2.4 Ứng dụng Transformer trong phân loại lưu lượng mạng:

- Transformer, với cơ chế tự chú ý (self-attention) mạnh mẽ và khả năng học các phụ thuộc toàn cục, đã được ứng dụng thành công trong lĩnh vực phân loại lưu lượng mạng (Network Traffic Classification), đặc biệt là phân loại lưu lượng mạng mã hóa (Encrypted Network Traffic Classification - ENTC).

- Đây là một số ứng dụng cụ thể của Transformer trong phân loại lưu lượng mạng:
 - **Phân Loại Lưu Lượng Mã Hóa:** Trong các mạng hiện đại, một tỷ lệ lớn lưu lượng được mã hóa để bảo vệ tính riêng tư và bảo mật. Điều này làm cho các phương pháp truyền thống khó phân loại được các loại lưu lượng khác nhau như video, âm thanh, hoặc ứng dụng dữ liệu. Transformer đã được sử dụng để phân loại các dạng lưu lượng mã hóa nhờ khả năng học các phụ thuộc dài hạn và khai thác các thông tin tinh vi từ chuỗi lưu lượng. Ví dụ, mô hình **ET-BERT** sử dụng kiến trúc Transformer để học các đặc trưng của lưu lượng mạng mã hóa, đặc biệt là các thông tin thời gian và gói dữ liệu (packet).
 - **Chuyển Đổi Dữ Liệu Dạng Chuỗi:** Dữ liệu lưu lượng mạng thường được biểu diễn dưới dạng chuỗi gói tin, có cấu trúc không đồng nhất về thời gian và kích thước. Transformer có thể được áp dụng để mô hình hóa các chuỗi này, với self-attention giúp xác định các mẫu và đặc trưng trong các giai đoạn khác nhau của kết nối mạng.
 - **Giảm Thiểu Tính Toán Tuần Tự:** So với các mô hình trước đây như RNN hay LSTM, vốn gặp khó khăn trong việc xử lý chuỗi dữ liệu dài, Transformer có khả năng xử lý song song các gói dữ liệu mạng, giúp tăng tốc độ phân loại và giảm độ trễ trong các hệ thống mạng lớn.

CHƯƠNG 3 MÔ HÌNH ET-BERT

3.1 Mô hình kiến trúc

- Trong bài báo cáo này, chúng tôi nhằm mục đích học các biểu diễn lưu lượng mã hóa tổng quát và phân loại chúng trong các kịch bản khác nhau (ví dụ như ứng dụng, giao thức mã hóa, hoặc dịch vụ). Để đạt được mục tiêu này, chiến lược tiền huấn luyện (pre-training) mà chúng tôi đề xuất bao gồm hai giai đoạn chính: tiền huấn luyện để học các biểu diễn lưu lượng mã hóa tổng quát với dữ liệu không có nhãn quy mô lớn và tinh chỉnh để điều chỉnh mô hình đã được tiền huấn luyện cho nhiệm vụ hạ nguồn cụ thể. Trong giai đoạn tiền huấn luyện, với các luồng lưu lượng không có nhãn, mô hình tiền huấn luyện sẽ xuất ra các biểu diễn lưu lượng ở mức datagram tổng quát. Trong giai đoạn tinh chỉnh, với các gói tin hoặc luồng có nhãn cụ thể, mô hình đã tinh chỉnh sẽ dự đoán danh mục của chúng.
- Lưu lượng mã hóa khác biệt rất nhiều so với ngôn ngữ tự nhiên và hình ảnh không chứa nội dung mà con người có thể hiểu và các đơn vị ngữ nghĩa rõ ràng. Để tận dụng hiệu quả kỹ thuật tiền huấn luyện cho việc phân loại lưu lượng mã hóa, chúng tôi chủ yếu đề xuất ba thành phần chính trong ET-BERT như được hiển thị trong **Hình 2**: (1) Chúng tôi đề xuất phương pháp Datagram2Token để chuyển đổi lưu lượng mã hóa thành đơn vị token bảo tồn mẫu cho tiền huấn luyện; (2) Sau đó, hai nhiệm vụ tiền huấn luyện, ví dụ như Mô hình Masked BURST và Dự đoán BURST Cùng Nguồn, được đề xuất để học các biểu diễn datagram ngữ cảnh hóa từ ngữ cảnh chuyển tiếp thay vì ngữ cảnh ngữ nghĩa; (3) Để thích ứng với các kịch bản phân loại lưu lượng khác nhau, chúng tôi tiếp tục đề xuất hai chiến lược tinh chỉnh, ví dụ như tinh chỉnh mức gói tin cho phân loại gói tin đơn và tinh chỉnh mức luồng cho phân loại luồng đơn.
- Kiến trúc mạng chính của ET-BERT bao gồm các khối Transformer hai chiều nhiều lớp. Mỗi khối bao gồm các lớp tự chú ý nhiều đầu, giúp nắm bắt các mối quan hệ ẩn giữa các đơn vị lưu lượng được mã hóa trong các datagram. Trong công việc này, kiến trúc mạng bao gồm 12 khối transformer với 12 đầu chú ý trong mỗi lớp tự chú ý. Kích thước của mỗi token đầu vào H được đặt là 768 và số lượng token đầu vào là 512.

3.2 Biểu diễn lưu lượng Datagram2Token

- Trong môi trường mạng thực tế, một lượng lớn lưu lượng chứa nhiều luồng từ nhiều loại khác nhau (ví dụ: các ứng dụng, giao thức hoặc dịch vụ khác nhau), điều này làm cho việc học các biểu diễn ổn định và phân biệt của một loại lưu lượng cụ thể trở nên khó khăn. Do đó, trước khi biểu diễn lưu lượng, chúng tôi tách các luồng có cùng IP, cổng và giao thức từ các trace. Kết quả là, mỗi luồng tách ra thuộc về cùng một loại lưu lượng, chứa một phiên luồng hoàn chỉnh. Để chuyển đổi một luồng thành các token giống như từ ngữ trong ngôn ngữ tự nhiên, chúng tôi đề xuất một mô-đun

Datagram2Token bao gồm ba bước: (1) Bộ tạo BURST sẽ trích xuất các gói tin liên tục từ máy chủ đến máy khách hoặc ngược lại trong một phiên luồng, gọi là BURST, để đại diện cho thông tin một phần của phiên. (2) Sau đó, quy trình BURST2Token chuyển đổi datagram trong mỗi BURST thành các token embedding qua mô hình bi-gram. Đồng thời, quy trình này cũng tách BURST thành hai đoạn để chuẩn bị cho các nhiệm vụ tiền huấn luyện. (3) Cuối cùng, Token2Embedding nối các token embedding, embedding vị trí và embedding phân đoạn của mỗi token để làm đầu vào cho việc tiền huấn luyện.

3.2.1 Bộ tạo BURST.

- Một BURST được định nghĩa là tập hợp các gói tin liên kế về mặt thời gian được phát sinh từ yêu cầu hoặc phản hồi trong một phiên luồng. Một chuỗi các BURST thể hiện mẫu của luồng mạng từ quan điểm lớp ứng dụng. Ở lớp ứng dụng, cây DOM giữa các trang web trở nên đa dạng do sự cá nhân hóa các dịch vụ web. Khi quá trình render phía máy khách chia nhỏ dữ liệu web thành các đối tượng khác nhau (ví dụ: văn bản và hình ảnh), cấu trúc DOM tạo ra các đoạn có ý thức ngữ nghĩa và ngầm ảnh hưởng đến các yêu cầu tài nguyên của máy khách. Mỗi đoạn được tạo thành một BURST trong mạng, chứa một phần nội dung hoàn chỉnh với loại cụ thể từ cấu trúc DOM. Chúng tôi trích xuất các BURST để làm đầu vào cho mô hình tiền huấn luyện.
- Với BURST, chúng tôi quan tâm đến nguồn gốc và đích của gói tin. Khi nhận được dấu vết từ tệp chụp gói tin dưới dạng chuỗi $Trace = \{flow_i, i \in N^+\}$, trong đó $flow = \{p_j, j \in N^+\}$ là một phiên luồng bao gồm các gói tin từ nguồn đến đích p được xác định bởi năm cặp (IPsrc, IPdst, Protocol). BURST được định nghĩa như sau:

$$BURST = \begin{cases} B^{src} = \{p_m^{src}, m \in N^+\} \\ B^{dst} = \{p_n^{dst}, n \in N^+\} \end{cases} \quad (1)$$

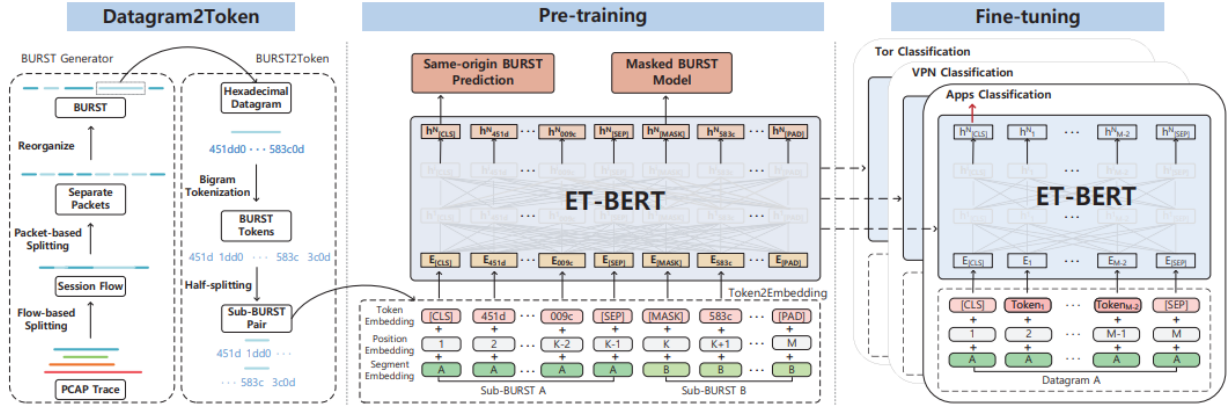
- trong đó m, n đại diện cho số lượng gói tin đơn hướng tối đa từ nguồn đến đích và từ đích đến nguồn tương ứng.

3.2.2 BURST2Token.

- Để chuyển đổi biểu diễn BURST thành biểu diễn token cho quá trình tiền huấn luyện, chúng tôi phân giải BURST dạng hex thành một chuỗi các đơn vị. Để làm điều này, chúng tôi sử dụng mô hình bi-gram để mã hóa chuỗi hex, mỗi đơn vị bao gồm hai byte liên kế. Sau đó, chúng tôi sử dụng mã hóa Byte-Pair cho biểu diễn token, trong đó mỗi đơn vị token có phạm vi từ 0 đến 65535, kích thước từ điển $|V|$ tối đa là 65536. Ngoài ra, chúng tôi còn thêm các token đặc biệt [CLS], [SEP], [PAD] và [MASK] cho các nhiệm vụ huấn luyện. Token đầu tiên của mỗi chuỗi luôn là [CLS], và trạng thái lớp ẩn cuối cùng liên kết với token này được sử dụng để biểu diễn chuỗi hoàn chỉnh cho các nhiệm vụ phân loại. Token [PAD] là ký hiệu đệm để đáp ứng yêu cầu về độ dài tối thiểu.

Cặp sub-BURST của một BURST sẽ được phân tách bởi [SEP]. Token [MASK] xuất hiện trong quá trình tiền huấn luyện để học ngữ cảnh của lưu lượng.

- Như minh họa trong hình bên dưới, chúng tôi chia đều một BURST thành hai sub-BURST cho nhiệm vụ SBP. Chúng tôi phân biệt các sub-BURST bằng token đặc biệt [SEP] và embedding phân đoạn cho biết liệu nó thuộc về đoạn A hay đoạn B. Chúng tôi ký hiệu đoạn A là sub-BURST_A và đoạn B là sub-BURST_B.



HÌNH 3.1: Tổng quan mô hình ET-BERT

3.2.3 Token2Embedding.

- Chúng tôi đại diện mỗi token thu được từ BURST2Token bằng ba loại embedding: token embedding, position embedding và segment embedding. Một biểu diễn token hoàn chỉnh được xây dựng bằng cách cộng ba loại embedding này lại. Trong nghiên cứu này, chúng tôi sử dụng các datagram đã được phân tích token đầy đủ làm đầu vào gốc. Nhóm vector embedding đầu tiên được khởi tạo ngẫu nhiên, với kích thước embedding $D = 768$. Sau N lần mã hóa bằng Transformer, chúng tôi thu được embedding token cuối cùng.
- Token Embedding. Như minh họa trong Hình 2, biểu diễn của token học được từ bảng tra cứu trong Mục 3.2.2 được gọi là token embedding E_{token} . Vector ẩn cuối cùng của token đầu vào được ký hiệu là $E_{token} \in \mathbb{R}^H$, trong đó kích thước embedding H được đặt là 768.
- Position Embedding. Vì việc truyền dữ liệu lưu lượng có liên quan chặt chẽ đến thứ tự, chúng tôi sử dụng position embedding để đảm bảo mô hình học cách tập trung vào mối quan hệ tạm thời của các token qua các vị trí tương đối. Chúng tôi gán một vector H -dimension cho mỗi token đầu vào để đại diện cho thông tin vị trí của nó trong chuỗi. Chúng tôi ký hiệu position embedding là $E_{pos} \in \mathbb{R}^H$, trong đó kích thước embedding H được đặt là 768.

- Segment Embedding của sub-BURST được ký hiệu là $Eseg \in \mathbb{R}^H$, trong đó kích thước embedding H được đặt là 768. Ở giai đoạn tinh chỉnh, chúng tôi đại diện cho một gói tin hoặc một luồng như một đoạn cho nhiệm vụ phân loại.

3.3 Pre-training ET-BERT

- Các nhiệm vụ tiền huấn luyện mà chúng tôi đề xuất nhằm nắm bắt mối quan hệ ngữ cảnh giữa các byte lưu lượng bằng cách dự đoán token bị ẩn cũng như thứ tự truyền tải chính xác bằng cách dự đoán Same-origin BURST. **Quy trình chi tiết được minh họa ở giữa Hình 2.**
- Mô hình Masked BURST. Nhiệm vụ này tương tự như Mô hình Ngôn Ngữ Mặt Nạ (Masked Language Model) được sử dụng bởi BERT [6]. Sự khác biệt chính là các token lưu lượng không có ngữ nghĩa rõ ràng được tích hợp vào ET-BERT để nắm bắt các phụ thuộc giữa các byte datagram. Trong quá trình tiền huấn luyện, mỗi token trong chuỗi đầu vào được ngẫu nhiên mặt nạ với xác suất 15%. Đối với token đã chọn, chúng tôi thay thế nó bằng [MASK] với xác suất 80%, hoặc chọn một token ngẫu nhiên để thay thế hoặc để nguyên với xác suất 10%, tương ứng.
- Đối với các masked token được thay thế bằng token đặc biệt [MASK], ET-BERT được huấn luyện để dự đoán các token tại các vị trí bị ẩn dựa trên ngữ cảnh. Nhờ vào biểu diễn hai chiều sâu mà nhiệm vụ này mang lại, chúng tôi ngẫu nhiên mặt nạ k token cho chuỗi đầu vào X . Chúng tôi sử dụng hàm mất mát là log-likelihood âm như là hàm mất mát của chúng tôi và định nghĩa chính thức như sau:

$$L_{MBM} = - \sum_{i=1}^k \log(P(MASK_i = token | \bar{X}; \theta)) \quad (2)$$

- trong đó θ đại diện cho tập hợp các tham số có thể huấn luyện của ET-BERT. Xác suất P được mô hình hóa bởi bộ mã hóa Transformer với θ . \bar{X} là biểu diễn của X sau khi đã mặt nạ và $MASK_i$ đại diện cho token bị ẩn tại vị trí i trong chuỗi token.
- Dự đoán Same-origin BURST. Tầm quan trọng của BURSTs trong lưu lượng mạng đã được nêu ra trong phần trước, và mục tiêu của chúng tôi là học các biểu diễn lưu lượng tốt hơn bằng cách nắm bắt mối tương quan giữa các gói tin trong BURSTs. Hơn nữa, chúng tôi xem xét mối quan hệ chặt chẽ giữa cấu trúc BURST và nội dung web, điều này có khả năng truyền tải sự khác biệt giữa các BURST được tạo ra từ các loại lưu lượng khác nhau. Ví dụ, có sự phân biệt trong lưu lượng khi tải nội dung riêng biệt cho các trang mạng xã hội với các cấu trúc DOM khác nhau, ví dụ: theo thứ tự văn bản, hình ảnh, video và theo thứ tự hình ảnh, văn bản, video. Hiện tượng này cũng được nghiên cứu xác nhận trong nghiên cứu [39] về fingerprinting trong miền.

- Chúng tôi học các phụ thuộc giữa các gói tin bên trong BURST thông qua nhiệm vụ Dự đoán Same-origin BURST (SBP). Đối với nhiệm vụ này, một bộ phân loại nhị phân được sử dụng để dự đoán liệu hai sub-BURST có cùng nguồn gốc từ cùng một BURST hay không. Cụ thể, khi chọn sub-BURST_A và sub-BURST_B cho mỗi cặp sub-BURST, 50% thời gian sub-BURST_B là sub-BURST kế tiếp thực sự theo sau sub-BURST_A, và 50% thời gian còn lại là một sub-BURST ngẫu nhiên từ các BURST khác. Đối với đầu vào chứa cặp sub-BURST $B_j = (sub-B_j^A, sub-B_j^B)$ và nhãn thực tế của nó $y_j \in [0, 1]$ (0 đại diện cho việc hai sub-BURST không cùng nguồn gốc và 1 đại diện cho việc hai sub-BURST có cùng nguồn gốc).

$$L_{SBP} = - \sum_{j=1}^k \log(P(y_j|B_j; \theta)) \quad (3)$$

- Tóm lại, mục tiêu cuối cùng của tiền huấn luyện là tổng của hai hàm mất mát trên, được định nghĩa như sau:

$$L = L_{MBM} + L_{SBP} \quad (4)$$

- **Tập dữ liệu tiền huấn luyện.** Trong nghiên cứu này, khoảng 30GB dữ liệu lưu lượng chưa gán nhãn được sử dụng cho việc tiền huấn luyện. Tập dữ liệu này bao gồm hai phần: (1) khoảng 15GB lưu lượng từ các tập dữ liệu công khai [9, 32]; (2) khoảng 15GB lưu lượng từ dữ liệu được thu thập thụ động dưới mạng Khoa học và Công nghệ Trung Quốc (CSTNET). Hơn nữa, tập dữ liệu chứa các giao thức mạng phong phú, như giao thức mã hóa mới dựa trên giao thức UDP QUIC, Bảo mật lớp vận chuyển (Transport Layer Security), Giao thức truyền tệp (File Transfer Protocol), Giao thức truyền văn bản siêu liên kết (Hyper Text Transfer Protocol), Shell an toàn (Secure Shell), v.v., mà đều là các giao thức mạng phổ biến.

3.4 Fine-tuning ET-BERT

- Việc tinh chỉnh có thể phục vụ tốt cho các nhiệm vụ phân loại hạ nguồn vì:
 - Biểu diễn tiền huấn luyện không phụ thuộc vào lớp lưu lượng và có thể áp dụng cho bất kỳ lớp biểu diễn lưu lượng nào; (2) Vì đầu vào của mô hình tiền huấn luyện là ở cấp độ byte datagram, các nhiệm vụ hạ nguồn cần phân loại gói tin và luồng có thể được chuyển đổi thành token byte datagram tương ứng để được phân loại bởi mô hình; (3) Token đặc biệt [CLS] của đầu ra mô hình tiền huấn luyện mô hình hóa biểu diễn của toàn bộ lưu lượng đầu vào và có thể được sử dụng trực tiếp cho phân loại.

- Cấu trúc của việc tinh chỉnh và tiền huấn luyện về cơ bản là giống nhau, chúng tôi đưa các biểu diễn gói tin hoặc luồng đặc thù của nhiệm vụ vào ET-BERT đã được tiền huấn luyện và tinh chỉnh tất cả các tham số trong một mô hình end-to-end. Ở lớp đầu ra, biểu diễn [CLS] được đưa vào một bộ phân loại đa lớp để dự đoán. Chúng tôi đề xuất hai chiến lược tinh chỉnh để điều chỉnh phân loại cho các kịch bản khác nhau: (1) cấp độ gói tin như đầu vào để thử nghiệm liệu ET-BERT có thể điều chỉnh cho dữ liệu lưu lượng chi tiết hơn, gọi là ET-BERT(packet); (2) cấp độ luồng như đầu vào để so sánh công bằng và khách quan ET-BERT với các phương pháp khác, gọi là ET-BERT(flow). Sự khác biệt chính giữa hai mô hình tinh chỉnh là lượng thông tin của lưu lượng đầu vào. Chúng tôi sử dụng một datagram ghép của M gói tin liên tiếp trong một luồng làm dữ liệu đầu vào, trong đó M được đặt là 5 trong phương pháp của chúng tôi. Việc xử lý dữ liệu lưu lượng được mô tả chi tiết trong Mục 4.1.
- Chi phí của việc tinh chỉnh tương đối rẻ so với tiền huấn luyện, và một GPU đơn là đủ cho một nhiệm vụ tinh chỉnh.

CHƯƠNG 4 KIỂM THỬ VÀ KẾT QUẢ

4.1 Yêu cầu chuẩn bị

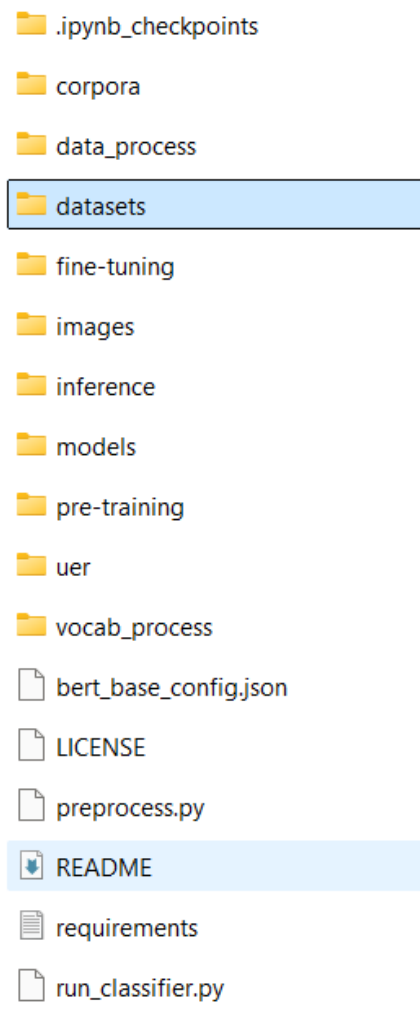
- Python (≥ 3.6): Ngôn ngữ lập trình cơ bản cần thiết để chạy tất cả các tập lệnh.
- CUDA 11.4: Nền tảng và API để sử dụng GPU nhằm tăng tốc quá trình huấn luyện mạng nơ-ron, đặc biệt cho các framework như PyTorch và TensorFlow.
- GPU Tesla V100S: GPU mạnh mẽ có thể xử lý các tính toán nặng, thường được sử dụng cho các mô hình học sâu, đặc biệt trong các tác vụ xử lý dữ liệu quy mô lớn.
- Torch (≥ 1.1): Framework PyTorch để xây dựng và huấn luyện mạng nơ-ron.
- Six ($\geq 1.12.0$): Thư viện Python hỗ trợ tương thích giữa Python 2 và 3, hữu ích khi chạy các tập lệnh cần tương thích ngược.
- Scapy ($= 2.4.4$): Thư viện Python để thao tác gói tin, thường được sử dụng trong phân tích hoặc thử nghiệm lưu lượng mạng.
- Numpy ($= 1.19.2$): Thư viện Python được sử dụng cho tính toán số học, thường dùng trong các thao tác tensor của các mô hình học máy.
- Shutil: Thư viện Python chuẩn cho các thao tác tập tin ở cấp cao, như sao chép và xóa tập tin.
- Random: Thư viện Python chuẩn để tạo số ngẫu nhiên, hữu ích trong các tác vụ như chia tập dữ liệu hoặc tăng cường dữ liệu.
- Json: Thư viện Python chuẩn để phân tích và thao tác các tệp JSON, có thể dùng cho cấu hình hoặc lưu trữ kết quả trung gian.
- Pickle: Thư viện Python chuẩn dùng để tuần tự hóa và giải tuần tự hóa đối tượng Python, giúp lưu trữ các mô hình, tham số hoặc dữ liệu lớn.
- Binscii: Một module dùng để chuyển đổi giữa nhị phân và ASCII, thường hữu ích trong các tác vụ liên quan đến mạng.
- Flowcontainer: Được sử dụng để container hóa luồng mạng, có lẽ dùng để tổ chức dữ liệu lưu lượng mạng.
- Argparse: Thư viện chuẩn để phân tích các đối số dòng lệnh.
- Packaging: Thư viện giúp xử lý phiên bản gói và phụ thuộc.
- Tshark: Trình phân tích giao thức mạng dòng lệnh, thường được sử dụng cùng với Scapy để phân tích lưu lượng mạng.
- SplitCap: Công cụ chia nhỏ các tệp bắt gói tin lớn (pcap) thành các phần nhỏ hơn, hữu ích trong xử lý các tập dữ liệu lớn.
- Scikit-learn: Thư viện học máy cung cấp các tiện ích cho phân loại, hồi quy, phân cụm.
- NVIDIA Apex: Thư viện từ NVIDIA cho huấn luyện với độ chính xác hỗn hợp nhằm tối ưu hóa hiệu suất mô hình và giảm sử dụng bộ nhớ.

- TensorFlow: Framework học máy cần thiết để chuyển đổi mô hình được huấn luyện trước, đặc biệt nếu một số mô hình đã được huấn luyện bằng TensorFlow và cần chuyển sang PyTorch.
- WordPiece: Phương pháp phân tích từ dùng trong xử lý ngôn ngữ tự nhiên, cần thiết để phân tích từ văn bản cho các mô hình như BERT.
- Pytorch-crf: Thư viện cho Mô hình Trường Ngẫu nhiên Có Điều Kiện (CRF) áp dụng vào các tác vụ gán nhãn theo chuỗi, như nhận diện thực thể có tên.

4.2 Mô hình thực nghiệm

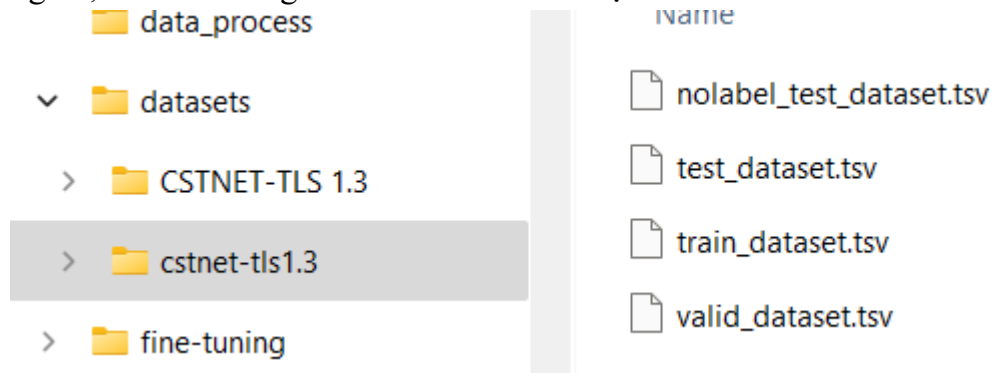
- Đầu tiên huấn luyện trước mô hình trên một lượng lớn dữ liệu không dán nhãn từ các tập dữ liệu lưu lượng mã hóa khác nhau. Sau đó, tinh chỉnh mô hình trên một lượng nhỏ dữ liệu có nhãn cho từng nhiệm vụ phân loại cụ thể.

- Mô hình gồm các file



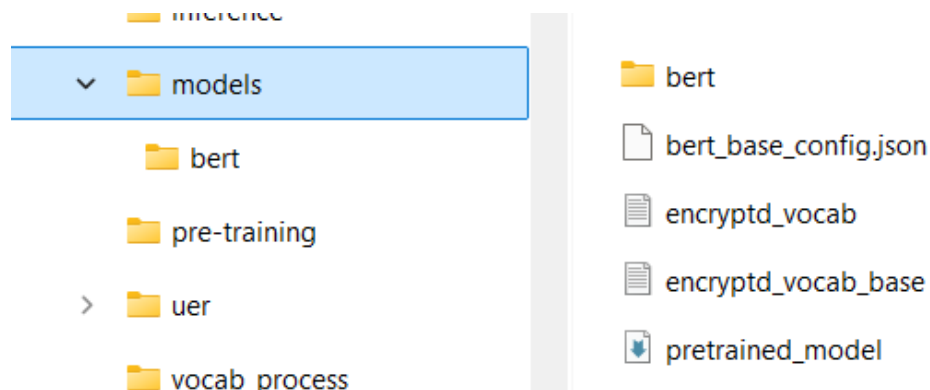
HÌNH 4.1: Tổ chức file của model

- Trong đó, file datasets gồm các file chứa các bộ test khác nhau



HÌNH 4.2: Tập hợp file dataset

- Và file models chứa các mô hình đã được huấn luyện trước và tinh chỉnh cho việc phân loại lưu lượng mạng đã mã hóa



HÌNH 4.3: file chứa model đã được pretrain

- Bộ dataset được dùng là tập CSTNET-TLS 1.3 được phát triển bởi một nhóm các nhà nghiên cứu từ China Science and Technology Network (CSTNet) nhằm cung cấp một nguồn tài nguyên cho việc nghiên cứu và phân tích lưu lượng mạng mã hóa sử dụng giao thức TLS 1.3. Mục đích chính của bộ dữ liệu này là hỗ trợ việc phân loại và phát hiện các hoạt động trong lưu lượng mạng mã hóa mà không cần giải mã dữ liệu, phục vụ cho các nghiên cứu về bảo mật mạng và học máy.
- Dưới đây là 120 ứng dụng được gán nhãn của bộ dữ liệu:



HÌNH 4.4: Tập hợp 120 ứng dụng web sử dụng cho quá trình pretrain

- Đầu vào của dataset cho fine-tuning sẽ gồm 2 cột, cột đầu tiên sẽ được đánh số từ 1 đến 119 đại diện cho 120 ứng dụng, cột thứ 2 tên là text_a với khoảng hơn 58000 dòng dữ liệu(được viết dưới dạng chuỗi mã hexa(hexadecimal) được biểu diễn dưới dạng nhị phân, với mỗi kí tự đại diện cho 4 bit thông tin) tương ứng với 120 ứng dụng này.

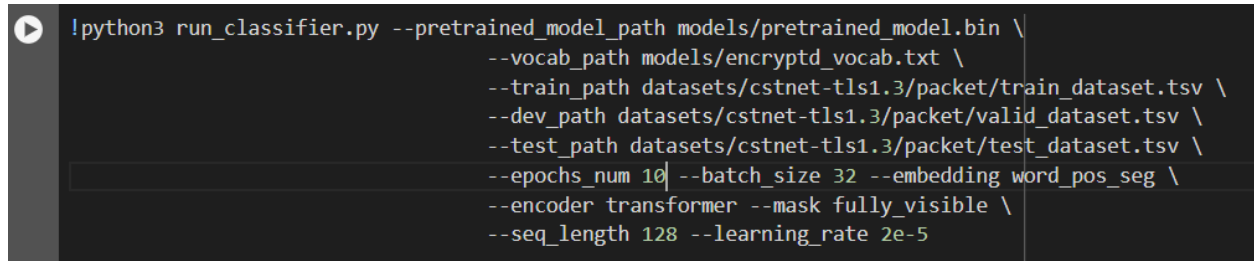
1	label	text_a
2		61 cbb8 b854 5421 21c3 c3de de8d 8d32 3264 64e1 e180 8010 1001 01f9 f9d5 d582 8200 0000 0001 0101 0108 080a 0a3d 3dfe fe4b 4b40 4
3	114	e2b3 b3af af8a 8a62 6263 6361 61cc cc73 734a 4a50 5018 1803 03aa aafd fddb db00 0000 0017 1703 0303 0302 02d7 d727 2777 775c 5
4	110	01bb bbfc fc8e 8e3c 3c82 8250 50de de59 59f7 f7f0 f010 1010 1021 2181 810d 0d00 0000 0001 0101 0108 080a 0a00 00da da88 881d 1d
5	21	480b 0b21 211f 1f37 374d 4d2f 2fb3 b341 41e6 e6b0 b010 1004 0400 00eb eb0a 0a00 0000 0001 0101 0108 080a 0a50 5084 8438 3800 c
6	4	01bb bb55 5529 294f 4f3f 3f77 7787 8744 4402 0250 5018 1803 03ff ff36 36af af00 0000 0017 1703 0303 0301 0188 883b 3b5e 5ea9 a92f
7	95	9a52 52ad adcb cb96 9690 90b8 b8ce ceb1 b1cf cf80 8018 1801 01e6 e670 7014 1400 0000 0001 0101 0108 080a 0a07 07a6 a695 9580
8	38	9262 62e3 e3c1 c1f3 f3b8 b8cc cc9c 9c87 87d3 d350 5018 1800 001f 1f06 065e 5e00 0000 0017 1703 0303 0300 00fc fc74 7437 373f 3f8a
9	54	e69e 9ead adad ad32 3242 4279 7998 981e 1ec4 c450 5010 1002 0250 5033 3348 4800 0000 00a1 a1d3 d3d9 d996 962e 2ea1 a113 13a1
10	85	f4d5 d57f 7f05 05de de74 7409 090c 0c5f 5fa4 a450 5010 1000 009d 9db7 b7ec ec00 0000 00b8 b8d0 d00e 0ebd bd5e 5e25 25aa ae83 83
11	52	ceda da4d 4da0 a0ef ef3f 3f66 6695 9567 675b 5b50 5010 1001 01f5 f5e5 e564 6400 0000 0017 1703 0303 031f 1f7c 7cf1 f1fa fa2b 2b09
12	17	01bb bbc0 c0a2 a2b2 b2ce ce04 04c3 c3de de5e 5e80 8010 1008 0800 00ab ab35 3500 0000 0001 0101 0108 080a 0a32 3217 17d0 d0ac
13	26	c5dc dc93 93e5 e50b 0b57 57d6 d6f2 f2e4 e42f 2f50 5010 1000 0047 479d 9de6 e600 0000 0020 208e 8e00 00fd fd2b 2bfe fe76 769f 9fa7
14	111	cad0 d0d7 d78a 8a18 180a 0ac8 c8c9 c9a7 a773 7350 5010 1003 03af af13 130b 0b00 0000 0005 05ef ef9a 9a0e 0e7b 7bd0 d091 91ff ff1
15	68	d4b4 b476 760c 0c99 9927 2706 06e9 e96f 6fa6 a680 8010 1002 029c 9cfa fa85 8500 0000 0001 0101 0108 080a 0ae4 e406 06cb cbce c
16	55	abd9 d900 00a6 a658 5835 35ad ad5f 5f28 2869 6980 8018 1802 0243 4362 6250 5000 0000 0001 0101 0108 080a 0a04 04b5 b599 9985
17	57	dc42 42a4 a46d 6d81 8196 9610 10b4 b4e8 e889 8950 5010 10ad ad13 137d 7d08 0800 0000 0053 5386 8648 4883 836b 6b24 24da da2
18	110	f1d2 d22a 2a3e 3e0d 0d0f 0d08 d89b 9b0f 0f92 9280 8010 1002 025c 5ce1 e138 3800 0000 0001 0101 0108 080a 0a0b 0bd6 d6a3 a39b 9b
19	107	84bc bcdf df61 6181 816d 6ddf fd3d 3d99 994f 4f80 8010 1001 01f5 f5e0 e034 3400 0000 0001 0101 0108 080a 0a38 38f5 f5d9 d9a6 a648
20	79	da39 39f2 f239 3951 5106 0696 9691 91fd fd77 7750 5018 1800 0085 85e5 e5ca ca00 0000 0017 1703 0303 0300 0013 138a 8a68 6847 4
21	43	f541 41b7 b729 2927 272e 2e32 329b 9b9d 9d7b 7b80 8010 1000 0089 898f 8f29 2900 0000 0001 0101 0108 080a 0a4e 4e24 244c 4c30
22	24	01bb bb18 1869 697b 7be0 e057 5744 448b 8baa aa50 5018 1802 0205 05f7 f725 2500 0000 0017 1703 0303 0300 001a 1a51 519d 9d1a
23	64	cd14 147b 7bce ce27 27c1 c190 9040 40c9 c9ea ea50 5010 1001 0111 11f7 f774 7400 0000 009a 9a45 45c0 c0fc fc05 0511 1187 8764 64
24	31	d8bd bdc6 c633 338c 8c2d 2d58 5840 4020 20ae ae50 5010 1000 0043 4319 19d4 d400 0000 00cf cfd9 df9c 9c2e 2ef7 f7c9 c9dd dd75 75
25	104	01bb bbba bac3 c3ec ec9b b9df dfb5 b5f3 f3d3 d3e0 e010 101d 1dec ec2f 2fda da00 0000 0001 0101 0105 0522 22df dfb7 b7f5 f51b 1bdf
26	63	d8c6 c68d 8ddf dfa6 a63a 3ae1 e1e5 e518 1862 6250 5010 1001 01f5 f539 393d 3d00 0000 00de def2 f2ba baa1 a1c4 c42a 2a79 792d 2d
27	17	c8ea ea23 239e 9e44 444c 4c46 467d 7d02 02dc dc50 5018 1800 0042 4254 5406 0600 0000 0017 1703 0303 0300 003e 3eb9 b956 561f
28	67	01bb bbdd dd45 4529 29da da9b 9bca ca77 777e 7e80 8018 1803 0373 735a 5ae8 e800 0000 0001 0101 0108 080a 0ae4 e450 5014 147

...

58162	105	b4dc dc38 388f 8f96 967e 7ed0 d000 00b6 b630 3050 5010 1000 003e 3e04 0412 1200 0000 0029 29ad ad5c 5cf2 f244 44c4 c497 973d 3d13
58163	26	c2f1 f1c0 c0e1 e185 8522 229b 9b8d 8df0 f099 9950 5018 1800 0047 47f3 f332 3200 0000 0017 1703 0303 0304 0429 29e7 e7f2 f21e 1ede de
58164	81	eb92 920a 0a80 8043 438e 8e89 89d2 d203 030c 0c50 5010 1000 0087 8718 18d2 d200 0000 008e 8e34 3489 89c4 c40b 0bdd dde2 e2e8 e8
58165	42	03e1 e160 60a6 a6b0 b04d 4d89 899b 9b19 199b 9b80 8018 1801 0172 72f1 f11f 1f00 0000 0001 0101 0108 080a 0a00 009a 9ae1 e1d0 d00b
58166	56	fed4 d401 0187 8789 89d3 d30d 0d67 67ee ee69 6950 5010 1000 0049 49a9 a943 4300 0000 0054 54ca cac4 c4d8 d896 9689 89ef ef20 202d
58167	22	ceee ee60 6083 8300 005b 5b31 31bd bdb7 b767 6750 5010 1001 01f5 f5ce ce71 7100 0000 0091 9107 0754 54fc fc82 823c 3c4c 4c53 5374
58168	51	01bb bbe5 e5f2 f279 790f 0fdb db23 232a 2a83 8380 8010 1008 0800 0093 93a4 a400 0000 0001 0101 0108 080a 0a09 0947 4794 94c1 c1c6
58169	38	daca ca36 3638 38b5 b528 2850 5013 13ac acdc dc50 5018 1800 003c 3ce0 e09a 9a00 0000 0020 20cc cc99 99da da21 21e2 e2e0 e084 841
58170	81	eb92 920b 0b05 05e2 e2e6 e689 89d2 d203 030c 0c50 5010 1000 0087 8732 3242 4200 0000 000e 0e80 80f7 f79c 9c5e 5ec0 c0c5 c527 27a3
58171	47	d035 35c2 c2d3 d30b 0ba3 a31d 1d88 8843 43cb cb50 5018 1800 007b 7bfb fbda da00 0000 003a 3a5b 5b7e 7e49 490e 0e66 66e7 e7a6 a6b1
58172	108	01bb bb00 0082 82ff ffe6 6eb8 b8de dec0 c079 7950 5018 1802 0205 0568 68a2 a200 0000 0017 1703 0303 0300 0057 5771 7175 75b7 b76d

HÌNH 4.5: Tổ chức của file dataset

- Sau khi có được mô hình pre-trained, ET-BERT có thể được áp dụng cho tác vụ spetic bằng cách fine-tuning ở cấp độ gói với lưu lượng mạng được gắn nhãn bằng lệnh sau:



```
!python3 run_classifier.py --pretrained_model_path models/pretrained_model.bin \
--vocab_path models/encryptd_vocab.txt \
--train_path datasets/cstnet-tls1.3/packet/train_dataset.tsv \
--dev_path datasets/cstnet-tls1.3/packet/valid_dataset.tsv \
--test_path datasets/cstnet-tls1.3/packet/test_dataset.tsv \
--epochs_num 10 --batch_size 32 --embedding word_pos_seg \
--encoder transformer --mask fully_visible \
--seq_length 128 --learning_rate 2e-5
```

HÌNH 4.6: Lệnh tiến hành quá trình fine-tune

- Với epochs_num 10: Đây là số lần (epoch) toàn bộ dữ liệu huấn luyện được đưa qua mô hình để học. Mỗi epoch có nghĩa là mô hình sẽ xem xét tất cả các mẫu trong tập huấn luyện một lần. Trong trường hợp này, mô hình sẽ trải qua 10 epoch. Và batch_size 32: Đây là số lượng mẫu dữ liệu sẽ được xử lý cùng một lúc trước khi mô hình cập nhật các trọng số của nó. Batch size là 32 nghĩa là 32 mẫu được xử lý cùng lúc trong mỗi bước huấn luyện (iteration).

4.3 Kết quả và đánh giá

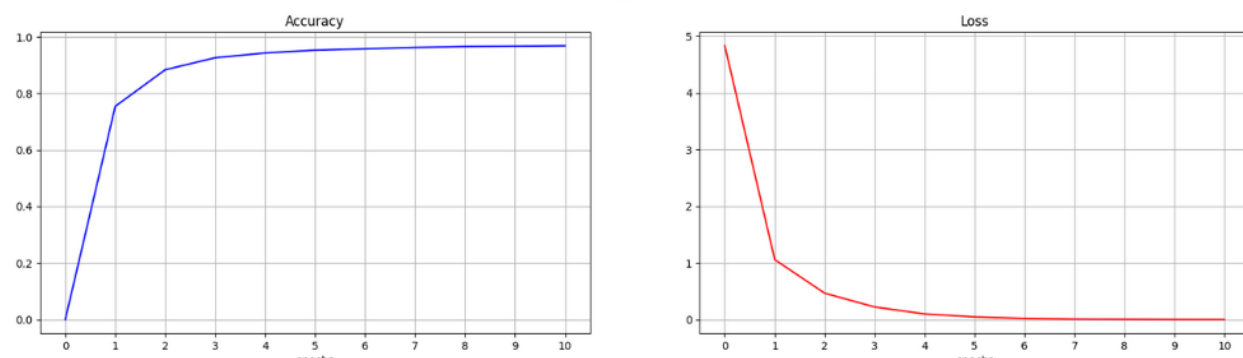
- Sau khi fine-tuning xong ta được confusion_matrix với đường chéo của ma trận là các trường hợp dự đoán chính xác và biểu đồ độ chính xác như các hình bên dưới:

```
⇒ Batch size: 32
The number of training instances: 465367
Start training.
 0% 0/10 [00:00<?, ?it/s]Epoch id: 1, Training steps: 100, Avg loss: 4.828
Epoch id: 1, Training steps: 200, Avg loss: 4.815
Epoch id: 1, Training steps: 300, Avg loss: 4.808

...

Epoch id: 10, Training steps: 14400, Avg loss: 0.001
Epoch id: 10, Training steps: 14500, Avg loss: 0.001
Acc. (Correct/Total): 0.9683 (56327/58171)
100% 10/10 [16:21:32<00:00, 5889.28s/it]
Test set evaluation.
Confusion matrix:
tensor([[479,  0,  0, ...,  0,  0,  0],
        [  0, 216,  0, ...,  0,  0,  0],
        [  1,  0, 491, ...,  0,  0,  0],
        ...,
        [  0,  0,  0, ..., 482,  0,  0],
        [  1,  0,  0, ...,  0, 480,  0],
        [  0,  0,  0, ...,  0,  0, 466]])
```

HÌNH 4.7: Kết quả sau quá trình fine-tune



HÌNH 4.8: Độ chính xác của model sau quá trình fine-tune

- Kết quả này cho thấy độ chính xác của mô hình ET-BERT chính xác tổng quan cao đến 96,83%. Điều này cho thấy mô hình có thể học các biểu diễn mạnh mẽ từ lưu lượng mã hóa TLS 1.3 và phân loại chính xác các luồng lưu lượng này.

KẾT LUẬN VÀ KIẾN NGHỊ

- Nghiên cứu này đã triển khai và thử nghiệm mô hình ET-BERT trong việc phân loại lưu lượng mạng mã hóa (encrypted traffic classification - ENTC). Kết quả cho thấy rằng ET-BERT không chỉ đạt được độ chính xác cao trong việc phân loại, mà còn vượt trội hơn so với các phương pháp truyền thống nhờ vào khả năng nắm bắt thông tin ngữ cảnh sâu hơn trong các gói tin mạng. Mô hình sử dụng kiến trúc Transformer cùng với cơ chế attention cho phép xử lý tốt các chuỗi dữ liệu dài và phức tạp, vốn thường gặp trong lưu lượng mạng mã hóa.
- Ngoài ra, ET-BERT thể hiện khả năng tổng quát tốt, cho phép phân loại chính xác không chỉ giữa các loại lưu lượng mã hóa và không mã hóa, mà còn giữa các ứng dụng và giao thức mạng khác nhau, bất kể sự biến đổi trong các phương thức mã hóa. Điều này chứng tỏ tiềm năng của mô hình trong việc ứng dụng thực tế, giúp phát hiện và quản lý lưu lượng mạng một cách hiệu quả hơn.
- Để cải thiện tính đại diện và hiệu suất của mô hình ET-BERT, các nghiên cứu tương lai nên thu thập thêm nhiều tập dữ liệu đa dạng hơn, bao gồm cả các dạng mã hóa phức tạp và đa dạng về giao thức mạng. Điều này sẽ giúp mô hình xử lý tốt hơn các tình huống thực tế trong các mạng lưới lớn.
- Tiếp tục tối ưu hóa kiến trúc ET-BERT, đặc biệt là giảm độ phức tạp tính toán và yêu cầu tài nguyên GPU, để phù hợp hơn cho các hệ thống mạng quy mô lớn. Các kỹ thuật như huấn luyện với độ chính xác hỗn hợp (mixed precision training) hoặc nén mô hình có thể được xem xét để giảm tải bộ nhớ và tăng tốc độ xử lý.
- Khuyến khích triển khai mô hình ET-BERT vào các hệ thống phát hiện xâm nhập (Intrusion Detection Systems - IDS) và quản lý lưu lượng mạng trong các môi trường doanh nghiệp và trung tâm dữ liệu. Điều này sẽ cung cấp thông tin chi tiết và chính xác hơn về hành vi lưu lượng mạng, từ đó cải thiện khả năng phát hiện các mối đe dọa tiềm ẩn.
- Cần có thêm các nghiên cứu về bảo mật để đánh giá mức độ an toàn của mô hình khi đối mặt với các cuộc tấn công gây nhiễu hoặc thay đổi gói tin. Khả năng của ET-BERT trong việc nhận biết và phân loại các dạng lưu lượng độc hại cũng là một lĩnh vực cần được khám phá sâu hơn.

PHỤ LỤC

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, Jing Yu, "ET-BERT: A Contextualized Datagram Representation with Pretraining Transformers for Encrypted Traffic Classification," in *Proceedings of the ACM Web Conference 2022 (WWW '22)*, New York, NY, USA, 2022.
- [2] Khaled Al-Naami, Swarup Chandra, Ahmad Mustafa , "Adaptive Encrypted Traffic Fingerprinting with Bi-Directional Dependence," in *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16)*, 2016.
- [3] Payap Sirinam, Mohsen Imani, Marc Juárez, "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*, Toronto, ON, Canada, 2018.
- [4] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li, " FS-Net: A Flow Sequence Network For Encrypted Traffic Classification," in *IEEE Conference on Computer Communications, INFOCOM 2019*, Paris, 2019.