HỌC VIỆN NGÂN HÀNG
KHOA CÔNG NGHỆ THÔNG TIN & KINH TẾ SỐ

BÀI TẬP LỚN
TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI
ỨNG DỤNG MÔ HÌNH TRANSFORMER TRONG PHÁT HIỆN HÀNH VI PHISHING EMAIL

GIẢNG VIÊN HƯỚNG DẪN	**:**	**TS. Vũ Trọng Sinh**
MÃ LỚP HỌC PHẦN	**:**	**242IS54A01**
NHÓM	**:**	**13**

HANOI – T6/2025

HỌC VIỆN NGÂN HÀNG
KHOA CÔNG NGHỆ THÔNG TIN & KINH TẾ SỐ

BÀI TẬP LỚN
TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI
ỨNG DỤNG MÔ HÌNH TRANSFORMER TRONG PHÁT HIỆN HÀNH VI PHISHING EMAIL

Giảng viên hướng dẫn : TS. Vũ Trọng Sinh

Nhóm lớp học phần : 242IS54A01

Nhóm : 13

STT	Mã sinh viên	Họ và Tên
1	25A4041921	Phạm Tiến Thắng (NT)
2	25A4041550	Nguyễn Việt Hưng
3	25A4041901	Nguyễn Thành Nhân
4	25A4041542	Võ Mạnh Hoàng

HANOI – T6/2025

DANH SÁCH THÀNH VIÊN NHÓM

MÃ SINH VIÊN	HỌ VÀ TÊN	NỘI DUNG CÔNG VIỆC	TỶ LỆ CÔNG VIỆC	KÝ XÁC NHẬN
25A4041921	Phạm Tiến Thắng (NT)			
25A4041550	Nguyễn Việt Hưng			
25A4041901	Nguyễn Thành Nhân			
25A4041542	Võ Mạnh Hoàng			

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC HÌNH ẢNH

LỜI CAM ĐOAN

Chúng em xin cam đoan rằng bài báo cáo với đề tài: **"ỨNG DỤNG MÔ HÌNH TRANSFORMER TRONG PHÁT HIỆN HÀNH VI PHISHING EMAIL"** là kết quả nghiên cứu độc lập của nhóm chúng em dưới sự hướng dẫn của giảng viên.

Toàn bộ nội dung trong bài báo cáo là sản phẩm do nhóm chúng em tự tìm hiểu và thực hiện, không sao chép từ bất kỳ cá nhân, nhóm, hay tài liệu nào mà không được trích dẫn. Các tài liệu tham khảo được sử dụng đều được trích dẫn và chú thích nguồn gốc rõ ràng.

Chúng em xin hoàn toàn chịu trách nhiệm về tính trung thực của bài báo cáo và sẵn sàng chấp nhận mọi hình thức xử lý nếu có bất kỳ sai phạm nào được phát hiện.

Nhóm sinh viên thực hiện

Nhóm 13

LỜI CẢM ƠN

Trước tiên, chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô Khoa Công nghệ Thông tin và Kinh tế Số, Học viện Ngân hàng, đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi cho chúng em trong suốt quá trình học tập và nghiên cứu.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên Vũ Trọng Sinh, người đã tận tâm hướng dẫn, chỉ bảo và đóng góp những ý kiến quý báu, giúp chúng em hoàn thiện bài báo cáo với đề tài "ỨNG DỤNG MÔ HÌNH TRANSFORMER TRONG PHÁT HIỆN HÀNH VI PHISHING EMAIL".

Nhờ sự hỗ trợ và động viên từ thầy, chúng em không chỉ hiểu rõ hơn về các kỹ năng lập trình, ngôn ngữ lập trình mà còn rèn luyện được nhiều kỹ năng, kiến thức thực tế quan trọng trong quá trình thực hiện đề tài.

Mặc dù đã cố gắng hết sức, song do hạn chế về thời gian và kinh nghiệm, bài báo cáo của chúng em chắc chắn không tránh khỏi những thiếu sót. Chúng em mong nhận được sự góp ý từ thầy để hoàn thiện hơn trong tương lai.

Cuối cùng, chúng em kính chúc thầy luôn mạnh khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy!

CHƯƠNG I. TỔNG QUAN ĐỀ TÀI

1.1. Đặt vấn đề

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ trên toàn cầu, email vẫn giữ vai trò là một trong những phương tiện trao đổi thông tin phổ biến và quan trọng nhất. Tuy nhiên, song song với sự phát triển đó là sự gia tăng nhanh chóng của các hành vi tấn công lừa đảo qua email (phishing email), gây ra nhiều rủi ro nghiêm trọng cho cả cá nhân và tổ chức, từ thất thoát tài chính cho đến mất mát dữ liệu, vi phạm bảo mật và ảnh hưởng đến uy tín doanh nghiệp [1]. Thống kê từ các tổ chức an ninh mạng cho thấy số lượng email phishing gia tăng theo từng năm với các kỹ thuật tấn công ngày càng tinh vi, vượt qua nhiều hệ thống phòng vệ truyền thống.

Cụ thể, số lượng email lừa đảo đã tăng 17,3% trong khoảng thời gian từ 15 tháng 9 năm 2024 đến 14 tháng 2 năm 2025 [2]. Đáng báo động hơn, các cuộc tấn công lừa đảo đã tăng 1.265% do sự phát triển của AI tạo sinh [3], với 82,6% email lừa đảo đã sử dụng AI [2]. Tội phạm mạng đang sử dụng trí tuệ nhân tạo để tạo ra các email được cá nhân hóa, bắt chước các thông tin liên lạc hợp pháp với độ chính xác đáng báo động [4]. Sự xuất hiện của các chiến dịch lừa đảo đa hình, với 76,4% các cuộc tấn công lừa đảo có ít nhất một tính năng đa hình trong năm 2024 [2], và việc sử dụng các ký tự vô hình để "phá vỡ" các hệ thống phát hiện đã làm tăng 47% số lượng email lừa đảo né tránh phát hiện [2]. Hơn nữa, lừa đảo đa kênh, nơi những kẻ tấn công dàn dựng các cuộc tấn công trên email, giọng nói và video, đang ngày càng gia tăng [4], khiến các biện pháp phòng thủ truyền thống trở nên lỗi thời.

Những cuộc tấn công này gây ra gánh nặng tài chính đáng kể; chi phí trung bình toàn cầu của một vụ vi phạm dữ liệu là 4,88 triệu USD vào năm 2024, tăng 10% so với năm trước [5]. Hơn một nửa số tổ chức báo cáo lừa đảo là cuộc tấn công phổ biến nhất để đánh cắp thông tin xác thực bảo mật đám mây, cho thấy mối đe dọa này ảnh hưởng trực tiếp đến cơ sở hạ tầng quan trọng và bảo mật dữ liệu.

Chỉ số	Giá trị	Nguồn	Ý nghĩa
Tăng trưởng email lừa đảo (09/2024 - 02/2025)	17,3%	[6]	Cho thấy sự gia tăng liên tục của các cuộc tấn công.
Tăng trưởng cuộc tấn công lừa đảo do Gen AI	1.265%	[7]	Phản ánh sự bùng nổ của các cuộc tấn công được tăng cường bởi AI.
Tỷ lệ email lừa đảo sử dụng AI	82,6%	[6]	AI là công cụ chính trong việc tạo ra các cuộc tấn công.
Tỷ lệ tấn công đa hình	76,4%	[6]	Cho thấy sự phức tạp và khả năng né tránh của các cuộc tấn công hiện đại.
Tăng trưởng email lừa đảo né tránh phát hiện	47%	[6]	Thách thức ngày càng tăng đối với các hệ thống phòng thủ.
Chi phí trung bình toàn cầu vụ vi phạm dữ liệu (2024)	4,88 triệu USD	[7]	Định lượng tác động tài chính nghiêm trọng của các cuộc tấn công.
Lừa đảo là cuộc tấn công phổ biến nhất để đánh cắp thông tin xác thực đám mây	Hơn 50% tổ chức	[7]	Liên kết lừa đảo với các mối đe dọa cơ sở hạ tầng quan trọng.
Xu hướng tấn công đa kênh (email, giọng nói, video)	Đang gia tăng	[7]	Cho thấy sự đa dạng hóa chiến thuật tấn công, vượt ra ngoài email truyền thống.

Bảng 1. Các số liệu thống kê và xu hướng lừa đảo chính (2024-2025)

Trước những thách thức đó, nhu cầu nghiên cứu và ứng dụng các phương pháp trí tuệ nhân tạo (AI), đặc biệt là các mô hình học sâu (Deep Learning) vào bài toán phát hiện và ngăn chặn email phishing ngày càng trở nên cấp thiết. Trong số các mô hình tiên tiến, kiến trúc Transformer nổi bật với khả năng xử lý ngôn ngữ tự nhiên (NLP) hiệu quả, cho phép mô hình hiểu sâu sắc mối quan hệ giữa các thành phần trong văn bản mà không cần đến quy trình trích xuất đặc trưng phức tạp. Nhờ sự xuất hiện của Transformer và các biến thể như BERT, GPT, lĩnh vực phát hiện email phishing có cơ hội tiếp cận những công cụ mạnh mẽ và hiệu quả hơn hẳn so với các phương pháp truyền thống.

Xuất phát từ thực tiễn và tiềm năng đó, đề tài nghiên cứu "Ứng dụng mô hình Transformer trong phát hiện hành vi phishing email" được thực hiện nhằm tìm hiểu, đánh giá và thử nghiệm khả năng của kiến trúc Transformer trong việc phát hiện các email lừa đảo, từ đó góp phần tăng cường an ninh mạng và bảo vệ người dùng khỏi các mối đe dọa ngày càng phức tạp.

1.2. Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là nghiên cứu ứng dụng kiến trúc Transformer trong phát hiện email phishing, từ đó xây dựng mô hình có khả năng nhận diện chính xác các email có dấu hiệu lừa đảo. Cụ thể, nghiên cứu tập trung vào:

- Phân tích sâu các đặc điểm và cơ chế hoạt động của các hành vi phishing email hiện đại, đặc biệt là các biến thể tinh vi do Trí tuệ Nhân tạo (AI) tạo ra, các chiến thuật đa kênh mới nổi, và các kỹ thuật né tránh phát hiện.
- Phân tích kiến trúc, nguyên lý hoạt động và ưu điểm vượt trội của mô hình Transformer trong xử lý ngôn ngữ tự nhiên, đặc biệt là khả năng của nó trong việc nắm bắt các mối quan hệ ngữ nghĩa phức tạp và thích ứng với các kỹ thuật tấn công tinh vi.
- Áp dụng Transformer vào bài toán phát hiện email phishing thông qua quá trình huấn luyện và đánh giá mô hình trên các dạng tấn công phishing tinh vi và đa dạng trên các bộ dữ liệu thực tế.
- So sánh hiệu quả của mô hình Transformer với các phương pháp truyền thống và các mô hình học sâu khác nhằm đánh giá tính ưu việt của Transformer trong việc xử lý các kỹ thuật lừa đảo hiện đại, né tránh, và đề xuất hướng cải tiến mô hình nhằm nâng cao chất lượng nhận diện trong thực tế.

1.3. Phạm vi nghiên cứu

Nội dung nghiên cứu tập trung vào việc phát hiện hành vi phishing email dựa trên phân tích nội dung email văn bản và các liên kết URL đi kèm. Dữ liệu nghiên cứu được thu thập từ các nguồn uy tín như PhishTank, IP2Location và Kaggle, bao gồm các tập dữ liệu "DataFromPhishingTank.xlsx" chứa URL độc hại và tập dữ liệu "Phishing Email Detection" với nội dung email đầy đủ.

Quá trình nghiên cứu bao gồm các bước tiền xử lý dữ liệu như làm sạch văn bản, chuẩn hóa văn bản về chữ thường, loại bỏ URL và ký tự đặc biệt, mã hóa văn bản bằng tokenizer của BERT, và xử lý cân bằng dữ liệu bằng kỹ thuật ADASYN. Mô hình được xây dựng dựa trên kiến trúc Transformer cùng các thành phần quan trọng như attention, multi-head attention, position encoding, feed-forward network, residual connection và layer normalization. Phạm vi nghiên cứu chủ yếu tập trung vào dữ liệu văn bản email. Mặc dù nhận thức được sự gia tăng của các chiến thuật tấn công đa kênh (như qua tin nhắn, mạng xã hội, hoặc cuộc gọi) trong bối cảnh hiện nay, đề tài này chưa mở rộng đến việc phân tích các yếu tố kỹ thuật hạ tầng mạng hay hành vi người dùng bên ngoài hệ thống email.

1.4. Phương pháp nghiên cứu

Phương pháp nghiên cứu của đề tài bao gồm các bước chính như sau: thu thập dữ liệu từ các nguồn dữ liệu mở, đảm bảo tính đa dạng và đầy đủ cho bài toán huấn luyện mô hình; tiến hành tiền xử lý dữ liệu văn bản để loại bỏ nhiễu và chuẩn hóa dữ liệu đầu vào; cân bằng dữ liệu bằng kỹ thuật tổng hợp dữ liệu thiểu số ADASYN nhằm khắc phục tình trạng mất cân bằng tập huấn luyện; mã hóa dữ liệu bằng tokenizer của BERT để đưa dữ liệu về dạng phù hợp cho mô hình Transformer.

Việc lựa chọn kiến trúc Transformer cho đề tài này được dựa trên khả năng vượt trội của nó trong việc xử lý các phụ thuộc dài hạn và nắm bắt ngữ cảnh phức tạp trong văn bản, điều mà các mô hình truyền thống (như SVM, Naive Bayes) và một số kiến trúc học sâu khác (như RNN, LSTM) thường gặp khó khăn khi đối phó với các email phishing ngày càng tinh vi và đa dạng về cấu trúc, bao gồm cả các kỹ thuật né tránh phát hiện.

Sau đó, mô hình Transformer Encoder được xây dựng và huấn luyện trên dữ liệu đã xử lý, bao gồm các thành phần encoder, self-attention, multi-head attention, feed-forward network và residual connection. Cuối cùng, mô hình được đánh giá hiệu quả bằng các chỉ số chính như Accuracy, Precision, Recall và F1-score. Kết quả mô hình Transformer cũng được so sánh với các mô hình truyền thống như SVM, Naive Bayes, Decision Tree, Random Forest và các mô hình học sâu khác như Bi-GRU, CNN, LSTM nhằm đánh giá sự ưu việt của kiến trúc Transformer trong bài toán phát hiện hành vi phishing email.

1.5. Ý nghĩa khoa học và thực tiễn

Về mặt khoa học, đề tài đóng góp vào lĩnh vực nghiên cứu ứng dụng học sâu và trí tuệ nhân tạo trong lĩnh vực an toàn thông tin, đặc biệt là bài toán nhận diện hành vi lừa đảo qua email. Việc áp dụng thành công mô hình Transformer vào bài toán này mở rộng khả năng ứng dụng các mô hình NLP tiên tiến vào các lĩnh vực an ninh mạng, xử lý văn bản khó, đặc biệt là trong việc đối phó với các kỹ thuật tấn công tinh vi do AI tạo ra và các biến thể đa hình nhằm né tránh các hệ thống phòng thủ truyền thống, góp phần tạo ra các hướng nghiên cứu mới trong tương lai về phòng chống tấn công mạng dựa trên AI.

Về mặt thực tiễn, kết quả nghiên cứu giúp xây dựng hệ thống phát hiện email phishing với độ chính xác cao hơn, hỗ trợ doanh nghiệp và cá nhân giảm thiểu nguy cơ rủi ro từ các cuộc tấn công mạng thông qua email. Đặc biệt, việc triển khai các giải pháp AI và tự động hóa trong an ninh mạng có thể giúp các tổ chức tiết kiệm trung bình 2,22 triệu USD hàng năm trong việc ngăn chặn vi phạm dữ liệu.2 Bên cạnh đó, hệ thống còn có thể được tích hợp vào các phần mềm quản lý email hiện có hoặc triển khai dưới dạng dịch vụ API, góp phần nâng cao mức độ an toàn thông tin cho người dùng và tổ chức trong kỷ nguyên số hóa ngày nay.

CHƯƠNG II. CƠ SỞ LÝ THUYẾT VÀ TOÁN HỌC TRONG MÔ HÌNH TRANSFORMER VÀ NHỮNG MÔ HÌNH CÙNG BÀI TOÁN

2.1. Tổng quan về Deep Learning

Học sâu (Deep Learning) là một nhánh tiên tiến của học máy (Machine Learning), tập trung vào việc thiết kế và huấn luyện các mạng nơ-ron nhân tạo với cấu trúc nhiều lớp ẩn (hidden layers). Mục tiêu chính của học sâu là tự động học và biểu diễn các mối quan hệ phức tạp, phi tuyến tính từ dữ liệu thô [8]. Điểm khác biệt cốt lõi so với các phương pháp học máy truyền thống là khả năng tự động trích xuất đặc trưng (feature extraction) mà không cần sự can thiệp thủ công. Điều này cho phép các mô hình học sâu khám phá các mẫu ẩn và cấu trúc phức tạp trong dữ liệu, từ đó cải thiện đáng kể hiệu quả và khả năng tổng quát hóa trên nhiều bài toán thực tế.

Một mạng nơ-ron sâu điển hình bao gồm ba thành phần chính:

- *Lớp đầu vào (Input Layer):* Chịu trách nhiệm tiếp nhận dữ liệu thô ban đầu, có thể là văn bản, hình ảnh, hoặc các dạng dữ liệu khác.
- *Lớp ẩn (Hidden Layers):* Đây là trái tim của mạng nơ-ron sâu, nơi các phép biến đổi phi tuyến tính được thực hiện qua nhiều tầng. Các lớp này cho phép mô hình học được các đặc trưng trừu tượng và phân cấp, từ những đặc trưng cơ bản đến những đặc trưng phức tạp hơn, đại diện cho dữ liệu.
- *Lớp đầu ra (Output Layer):* Cung cấp kết quả dự đoán cuối cùng của mô hình, thường là nhãn phân loại (đối với bài toán phân loại) hoặc giá trị số (đối với bài toán hồi quy).

Các kiến trúc học sâu đã phát triển đa dạng để phù hợp với nhiều loại dữ liệu và bài toán khác nhau, bao gồm:

- *Multilayer Perceptron (MLP):* Là dạng mạng nơ-ron cơ bản nhất, với các lớp kết nối đầy đủ, thích hợp cho các bài toán phân loại và hồi quy đơn giản.
- *Convolutional Neural Network (CNN):* Đặc biệt hiệu quả trong việc xử lý dữ liệu có cấu trúc không gian như hình ảnh, nhưng cũng được ứng dụng trong xử lý văn bản ngắn nhờ khả năng nhận diện các mẫu cục bộ.
- *Recurrent Neural Network (RNN):* Được thiết kế để xử lý dữ liệu tuần tự, nơi thứ tự của các phần tử có ý nghĩa, như chuỗi ký tự, từ trong câu hoặc tín hiệu âm thanh.
- *Long Short-Term Memory (LSTM):* Một biến thể tiên tiến của RNN, được phát triển để khắc phục vấn đề biến mất gradient và khả năng ghi nhớ phụ thuộc dài hạn trong các chuỗi dữ liệu rất dài.

- *Transformer:* Là một kiến trúc đột phá dựa trên cơ chế attention, đã cách mạng hóa lĩnh vực xử lý ngôn ngữ tự nhiên. Transformer cho phép xử lý hiệu quả các chuỗi dữ liệu dài bằng cách nắm bắt mối quan hệ giữa các phần tử không phụ thuộc vào khoảng cách vật lý, đồng thời hỗ trợ song song hóa quá trình huấn luyện, trở thành nền tảng cho nhiều mô hình ngôn ngữ lớn hiện nay.

Trong bối cảnh an ninh mạng, đặc biệt là các bài toán liên quan đến phát hiện hành vi lừa đảo qua email (phishing), học sâu đã chứng minh hiệu quả vượt trội so với các phương pháp truyền thống. Khả năng tự động học các mẫu dữ liệu tinh vi và thích ứng nhanh chóng với những thay đổi không ngừng trong chiến thuật tấn công của tội phạm mạng đã làm cho học sâu trở thành công cụ không thể thiếu [9]. Việc áp dụng kiến trúc Transformer, với khả năng xử lý ngữ cảnh và phụ thuộc phức tạp trong văn bản, vào bài toán phân loại email lừa đảo là một minh chứng tiêu biểu cho tiềm năng mạnh mẽ của học sâu trong việc tăng cường an ninh mạng.

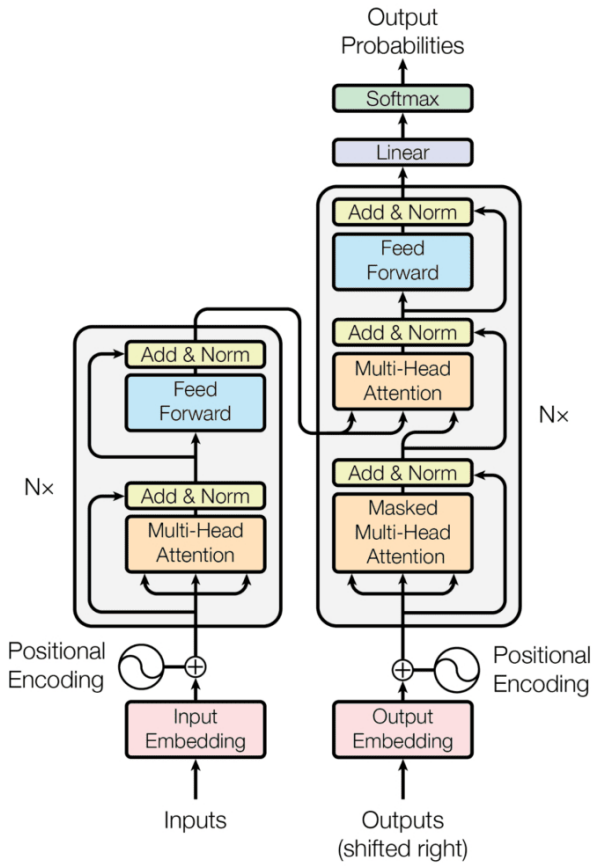
2.2. Giới thiệu về mô hình Transformer

Transformer là một kiến trúc mạng nơ-ron sâu được thiết kế chuyên biệt cho việc xử lý dữ liệu tuần tự, đặc biệt hiệu quả trong các bài toán xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) [10]. Mô hình này được giới thiệu lần đầu vào năm 2017 trong công trình khoa học "Attention Is All You Need" của Vaswani và các cộng sự tại Google, đánh dấu một bước ngoặt quan trọng trong lĩnh vực học sâu.

Khác với các kiến trúc truyền thống như mạng nơ-ron hồi quy (RNN) hay mạng tích chập (CNN), Transformer loại bỏ hoàn toàn các cơ chế xử lý tuần tự, thay vào đó áp dụng cơ chế attention, cụ thể là self-attention, để xác định và học mối quan hệ giữa các phần tử trong chuỗi dữ liệu đầu vào. Cách tiếp cận này giúp mô hình có khả năng nắm bắt các phụ thuộc xa trong dữ liệu một cách hiệu quả, đồng thời cho phép tăng tốc quá trình huấn luyện nhờ khả năng song song hóa trên toàn bộ chuỗi [10].

Kể từ khi ra đời, Transformer đã trở thành nền tảng cho hàng loạt mô hình ngôn ngữ hiện đại như BERT, GPT, T5 cùng nhiều biến thể khác. Những mô hình này đã tạo ra bước tiến vượt bậc trong các tác vụ như phân loại văn bản, dịch máy, sinh ngôn ngữ tự nhiên, và hệ thống trả lời câu hỏi tự động [10].

2.3. Kiến trúc mô hình Transformer



Hình 1. Cấu trúc encoder-decoder của kiến trúc Transformer "Attention is All You Need"

Mô hình Transformer được xây dựng dựa trên cấu trúc bộ mã hóa – bộ giải mã (encoder–decoder) [10]. Trong đó, thành phần bộ mã hóa (encoder) đóng vai trò xử lý chuỗi dữ liệu đầu vào, sau đó ánh xạ thành một chuỗi biểu diễn liên tục có kích thước cố định, chứa thông tin ngữ nghĩa cần thiết.

Tiếp theo, thành phần bộ giải mã (decoder) nhận đầu ra từ bộ mã hóa cùng với đầu ra của các bước giải mã trước đó, để dần tạo thành chuỗi dữ liệu đầu ra hoàn chỉnh. Nhờ vào thiết kế này, mô hình có khả năng xử lý hiệu quả các bài toán yêu cầu dự đoán tuần tự như dịch máy, sinh văn bản và các tác vụ xử lý ngôn ngữ tự nhiên khác [10].

2.3.1. Bộ mã hóa(Encoder)

Thành phần bộ mã hóa (encoder) của Transformer được cấu tạo từ N = 6 lớp giống hệt nhau xếp chồng lên nhau [10]. Mỗi lớp trong đó bao gồm hai thành phần chính:

- *Cơ chế chú ý đa đầu (Multi-head Attention):* cho phép mô hình tập trung vào các vị trí khác nhau trong chuỗi đầu vào, từ đó nắm bắt tốt hơn các mối quan hệ giữa các phần tử.
- *Mạng truyền tiếp theo vị trí (Position-wise Feed-Forward Network):* thực hiện phép biến đổi phi tuyến tính độc lập tại từng vị trí của chuỗi biểu diễn trung gian.

Ngoài ra, mỗi lớp đều được bổ sung kết nối tắt (residual connection) và lớp chuẩn hóa theo lớp (Layer Normalization) nhằm tăng độ ổn định khi huấn luyện và giúp quá trình lan truyền ngược hiệu quả hơn.

Công thức tổng quát được biểu diễn như sau:

$$\text{LayerNorm}(x+\text{Sublayer}(x))$$

Trong đó Sublayer(x) là chức năng được thực hiện bởi lớp con. Để tạo điều kiện thuận lợi cho các kết nối còn lại này, tất cả các lớp trong mô hình cũng như lớp nhúng (embedding layers) đều tạo đầu ra có kích thước

$$d_{model}=512.$$

2.3.2. Bộ giải mã(Decoder)

Tương tự như bộ mã hóa, thành phần bộ giải mã (decoder) của Transformer cũng được cấu tạo từ N = 6 lớp giống hệt nhau xếp chồng lên nhau. Tuy nhiên, mỗi lớp trong bộ giải mã bao gồm ba thành phần chính [10]:

- *Cơ chế Self-Attention với masking:* cho phép mô hình chỉ tập trung vào các vị trí trước đó trong chuỗi đầu ra đã sinh, nhằm dự đoán từ kế tiếp. Cơ chế masking này đảm bảo rằng, tại thời điểm dự đoán một từ, mô hình không thể truy cập vào thông tin của các từ ở vị trí sau, từ đó duy trì tính nhân quả cần thiết trong các bài toán sinh chuỗi.
- *Attention đến đầu ra của bộ mã hóa (Encoder-Decoder Attention):* giúp bộ giải mã tham chiếu và khai thác thông tin ngữ nghĩa từ chuỗi biểu diễn đầu ra của bộ mã hóa để hỗ trợ quá trình sinh dữ liệu đầu ra.
- *Mạng truyền tiếp theo vị trí (Position-wise Feed-Forward Network):* tương tự như trong encoder, thực hiện các phép biến đổi phi tuyến tính độc lập tại từng vị trí của chuỗi trung gian.

Giống với bộ mã hóa, mỗi lớp trong bộ giải mã cũng được trang bị kết nối tắt (residual connection) và lớp chuẩn hóa (Layer Normalization) nhằm đảm bảo quá trình huấn luyện ổn định và hiệu quả.

2.4. Cơ chế Attention

Attention là một hàm ánh xạ giữa một truy vấn (Query) và một tập hợp các cặp khóa-giá trị (Key-Value) để tạo ra đầu ra tương ứng. Trong đó, query, key, value và đầu ra đều được biểu diễn dưới dạng vector.

Giá trị đầu ra của hàm attention được tính toán dưới dạng tổng có trọng số của các vector value, với trọng số được xác định thông qua mức độ tương thích giữa truy vấn và từng khóa. Cụ thể, mức độ tương thích này được đo bằng một hàm tính điểm (score function), phản ánh mức liên quan giữa truy vấn và khóa tương ứng, sau đó được chuẩn hóa (thường bằng softmax) để tạo thành các trọng số.

Trong cơ chế Self-Attention, truy vấn, khóa và giá trị đều được lấy từ cùng một chuỗi đầu vào, cho phép mô hình tự học được mối quan hệ giữa các phần tử tại các vị trí khác nhau trong chuỗi.

Hàm chú ý (Scaled dot-product attention): Ánh xạ một truy vấn (Query) và một tập hợp các cặp khóa-giá trị (key-value) thành một đầu ra. Trong đó, đầu vào (input) bao gồm các truy vấn và khóa có chiều dk và các giá trị có chiều dv. Với Q là ma trận truy vấn, K là ma trận khóa, V là ma trận giá trị, hàm chú chú được tính như sau:

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

Hình 2. Hàm chú ý (Scaled dot-product attention)

2.5. Mã hóa vị trí (Position Encoding)

Mã hóa vị trí cung cấp thông tin về vị trí của mỗi từ trong chuỗi đầu vào, sử dụng các hàm sin và cos với các tần số khác nhau. Cụ thể, với mỗi từ tại vị trí i, vector biểu diễn của từ sẽ được cộng với một ma trận biểu diễn PEi:

$$\text{PE}_{(\text{pos},2i)} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{PE}_{(\text{pos},2i+1)} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

Hình 3. Với mỗi từ tại vị trí i, vector biểu diễn của từ sẽ được cộng với một ma trận biểu diễn PEi

2.6. Chú ý đa đầu (Multi-head attention)

Cơ chế chú ý đa đầu (Multi-Head Attention) cho phép mô hình đồng thời tập trung vào thông tin tại nhiều vị trí khác nhau trong chuỗi dữ liệu. Thay vì chỉ sử dụng một hàm attention duy nhất, cơ chế này triển khai song song nhiều đầu chú ý (attention heads), mỗi đầu sử dụng các tập khóa–giá trị–truy vấn riêng biệt với các tham số khác nhau. Cụ thể, với mỗi đầu i, truy vấn Qi, khóa Ki, và giá trị Visẽ được tính toán chú ý. Đầu ra từ mỗi đầu sẽ được ghép lại và nhân với trọng số WO.

Công thức của multi-head attention được tính như sau:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

Hình 4. Công thức của multi-head attention

trong đó mỗi đầu (head) được tính như sau:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Hình 5. Công thức tính mỗi đầu (head)

2.7. Mạng chuyển tiếp nguồn cấp dữ liệu theo vị trí

Mạng chuyển tiếp nguồn cấp dữ liệu theo vị trí (Position-wise Feed-Forward Networks) là một thành phần quan trọng trong kiến trúc Transformer. Thành phần này được thiết kế dưới dạng một mạng nơ-ron gồm hai lớp tuyến tính kết nối đầy đủ (fully connected layers). Với một chuỗi đầu vào có chiều dmodel, mạng sẽ ánh xạ mỗi từ tại vị trí i, thành một vector có chiều dff bằng cách sử dụng hai ma trận trọng số W1và W2 và hàm kích hoạt ReLU:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Hình 6. Sử dụng hai ma trận trọng số W1và W2 và hàm kích hoạt ReLU

2.8. Kết nối tắt và Chuẩn hóa theo lớp (Residual Connection and Layer Normalization)

Trong quá trình huấn luyện các mô hình mạng nơ-ron sâu, một trong những thách thức lớn là hiện tượng biến mất gradient (vanishing gradient), khiến việc cập nhật trọng số trở nên khó khăn khi mạng có quá nhiều lớp. Để khắc phục vấn đề này, Transformer áp dụng cơ chế kết nối tắt (Residual Connection) cho từng lớp trong mô hình. Cụ thể, đầu vào của mỗi lớp được cộng trực tiếp với đầu ra của lớp đó trước khi chuyển sang bước xử lý tiếp theo. Sau khi thực hiện kết nối tắt, kết quả sẽ được chuẩn hóa bằng chuẩn hóa theo lớp (Layer Normalization) nhằm ổn định giá trị các vector biểu diễn, giúp tăng tốc quá trình huấn luyện và cải thiện khả năng hội tụ của mô hình:

LayerNorm(x+Sublayer(x))

Cơ chế này đã được chứng minh giúp Transformer huấn luyện hiệu quả hơn và đạt được độ hội tụ tốt hơn so với các kiến trúc không sử dụng kết nối tắt.

2.9. Những mô hình cùng bài toán

Trong lĩnh vực phát hiện hành vi phishing email, đã có nhiều phương pháp và mô hình được đề xuất và ứng dụng. Có thể chia thành hai nhóm chính:

a, Các mô hình học máy truyền thống (Machine Learning)

Các mô hình truyền thống dựa trên việc trích xuất đặc trưng (feature extraction) từ nội dung email (như độ dài tiêu đề, số lượng liên kết, từ khóa đáng ngờ…) và sử dụng các thuật toán học máy để phân loại:

- Naive Bayes (NB)
- Support Vector Machine (SVM)
- Random Forest (RF) và Decision Tree (DT)

b, Các mô hình học sâu (Deep Learning)

Nhằm khắc phục việc phải trích xuất đặc trưng thủ công, các mô hình học sâu có khả năng tự học đặc trưng từ dữ liệu văn bản gốc:

- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)
- Convolutional Neural Networks (CNN)

Dưới đây là bảng so sánh các mô hình cùng bài toán Transformer:

Tên mô hình	Tác giả	Accuracy	Precisiom	Recall	F1-score
Advanced 1D-CNNPD + Bi-GRU	Najwa Altwaijry	99.68 %	100 %	99.32 %	99.66 %
BERT-LSTM hybrid	M. Mullangi	99.61 %	99.87%	99.23%	99.55%
Bi-GRU + 2D-CNN	Remmide	98.14%	98.84%	99.44%	99.14%
Mạng nơ-ron nhân tạo	Ana Bezerra, Ivo Pereira, Miguel Â. Rebelo,..	99.18%	96%		

Bảng 2. Bảng so sánh các mô hình cùng bài toán Transformer

CHƯƠNG III. TIỀN XỬ LÝ DỮ LIỆU EMAIL

3.1. Chuẩn bị cơ sở dữ liệu

Nhóm đã thu thập dữ liệu từ các nguồn public đáng tin cậy và phổ biến trong lĩnh vực an ninh mạng và xử lý ngôn ngữ tự nhiên. Cụ thể, dữ liệu được lấy từ các nền tảng như PhishTank, IP2Location, và Kaggle, với mức độ liên quan cao đến bài toán phân loại email lừa đảo (phishing). Dữ liệu cụ thể bao gồm:

- Email chứa liên kết phishing được thu thập từ tập dữ liệu "Phishing URL Dataset" do người dùng ESDAUNG chia sẻ, tổng hợp từ IP2Location và PhishTank – những nguồn dữ liệu uy tín chuyên phát hiện các URL độc hại (2024).
- Email văn bản đầy đủ (subject + body) được lấy từ bộ dữ liệu "Phishing Email Detection", cung cấp bởi cộng đồng nghiên cứu trên Kaggle. Bộ dữ liệu này bao gồm các email hợp lệ (safe) và email lừa đảo (phishing), đã được gán nhãn đầy đủ.

Nhóm đã thực hiện việc gán nhãn dữ liệu bằng cách dựa vào cột Email Type trong bộ dữ liệu, trong đó mỗi email được xác định là Phishing Email hoặc Safe Email. Đồng thời, nhóm đã chuyển các nhãn này sang dạng nhị phân (0 cho Safe, 1 cho Phishing) để phù hợp với yêu cầu của các mô hình học máy. Phương pháp này đảm bảo độ chính xác, nhất quán và thuận tiện trong quá trình huấn luyện mô hình, đồng thời tiết kiệm đáng kể thời gian so với việc gán nhãn thủ công.

Dưới đây là đoạn mã ví dụ cho việc dán nhãn của nhóm:

```
# Giữ lại các nhãn hợp lệ
df = df[df['Email Type'].isin(['Phishing Email', 'Safe Email'])].copy()
df['label'] = df['Email Type'].map({'Phishing Email': 1, 'Safe Email': 0})
```

Hình 7. Dán nhãn dữ liệu từ kaggle

```
[ ]  df = pd.read_excel('/content/drive/MyDrive/DataFromPhishingTank.xlsx')
     df = df.dropna(subset=['URLs', 'Labels'])
     df = df[df['URLs'].str.strip() != '']
     df = df[df['Labels'].isin([0, 1])]
     urls = df['URLs'].values
     labels = df['Labels'].astype(np.float32).values
     print(f"Loaded {len(urls)} URLs.")
```

Hình 8. Dán nhãn dữ liệu từ PhishTank

Trong đề tài, nhóm sử dụng 2 bộ dữ liệu được lấy từ trang PhishTank và trang kaggle dưới dạng file xlsx và csv có tên DataFromPhishingEmail.xlsx bao gồm các URL bị nghi ngờ là phishing kèm thông tin xác thực. Còn với bộ dữ liệu Phishing email bao gồm nội dung email. Dữ liệu từ PhishTank chứa các cột:

- URL: Đường dẫn bị nghi ngờ là phishing.
- Label: Nhãn dữ liệu.

Số lượng mẫu: 55000 email, trong đó:

Loại email	**Số lượng**
Legitimate Email	50000
Phishing Email	5000

Và bộ dữ liệu từ kaggle chứa các cột:

- Email Text: Chứa nội dung email
- Email Type: Nhãn dữ liệu.

Số lượng mẫu: 18650 email, trong đó:

Loại email	**Số lượng**
Safe Email	7328
Phishing Email	11322

Dữ liệu được chia theo 80/20. Trong đó 80% cho tập train và 20% cho tập kiểm thử

```
# Chia tập train/test
X_train, X_test, y_train, y_test = train_test_split(
    X_pad, y, test_size=0.2, random_state=42, stratify=y
)
```

Hình 9. Chia tập train/test

Việc chia dữ liệu theo tỷ lệ hợp lý giữa hai nhãn giúp đảm bảo mô hình Transformer có thể học hiệu quả, không bị lệch lớp, đồng thời đánh giá chính xác khả năng tổng quát hóa trên dữ liệu chưa từng thấy. Bên cạnh đó, sự đồng đều về số lượng email trong từng tập và từng nhãn cũng giúp hạn chế tình trạng mất cân bằng dữ liệu – một trong những nguyên nhân phổ biến dẫn đến sai lệch kết quả trong bài toán phân loại văn bản.

3.2. Tiền xử lý dữ liệu

Nhận dạng và phân loại email lừa đảo là một bài toán phức tạp trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đòi hỏi phải trải qua nhiều bước tiền xử lý nhằm chuyển đổi văn bản thô thành dạng dữ liệu có thể hiểu và học được bởi mô hình học sâu. Việc chuẩn bị dữ liệu đầu vào thông qua các bước tiền xử lý văn bản là rất quan trọng, vì nó giúp loại bỏ nhiễu, chuẩn hóa ngôn ngữ và giữ lại các đặc trưng hữu ích, từ đó nâng cao hiệu quả và độ chính xác của mô hình.

Quá trình tiền xử lý không chỉ giúp mô hình giảm độ phức tạp trong việc hiểu ngôn ngữ tự nhiên, mà còn tăng cường khả năng học và phân biệt giữa email hợp lệ và email phishing.

a, Mục tiêu chính của tiền xử lý dữ liệu:

- Tải và đọc dữ liệu email từ tệp CSV hoặc cơ sở dữ liệu đã thu thập.
- Làm sạch nội dung email, loại bỏ ký tự đặc biệt, đường dẫn URL, HTML tags và đưa về chữ thường.
- Chuẩn hóa văn bản để phù hợp với các mô hình học sâu như BERT hoặc các kiến trúc Transformer khác.
- Mã hóa văn bản thành chuỗi token dạng số bằng tokenizer của mô hình (ví dụ: BERT tokenizer).
- Cắt/padding văn bản về độ dài cố định để đảm bảo dữ liệu đầu vào nhất quán.
- Lưu trữ các văn bản đã xử lý cùng với nhãn tương ứng (0 cho email hợp lệ, 1 cho email phishing) vào các tập X_train, y_train... để chuẩn bị cho quá trình huấn luyện.

b, Phương pháp:

Đoạn mã tiền xử lý văn bản trong bài toán này được thực hiện qua các bước sau:

- Đọc dữ liệu và gán nhãn nhị phân từ cột Email Type.
- Làm sạch văn bản bằng cách xóa URL, ký tự không cần thiết, chuẩn hóa về chữ thường.
- Sử dụng tokenizer để chuyển văn bản thành chuỗi token và mã hóa về định dạng số.
- Cắt/padding để chuẩn hóa độ dài văn bản.
- Lưu trữ dữ liệu đầu vào (input_ids) và nhãn (labels) để huấn luyện mô hình Transformer.

Bước 1: Khởi tạo và nhập các thư viện cần thiết

Bước 1.1: Khởi tạo và nhập các thư viện cần thiết Data Phishing Tank

```
# Mount Drive
from google.colab import drive
drive.mount('/content/drive')

# Import các thư viện cần thiết
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.over_sampling import ADASYN
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import (
    Input, Embedding, LayerNormalization, Dropout, MultiHeadAttention, Dense,
    GlobalAveragePooling1D
)
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import Precision, Recall
from tensorflow.keras import regularizers
import pickle
import os
```

Hình 10. Khởi tạo và nhập các thư viện cần thiết Data Phishing Tank

Bước 1.2: Khởi tạo và nhập các thư viện cần thiết Data Phishing Email

```
import pandas as pd
import re
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from imblearn.over_sampling import ADASYN
from tensorflow.keras.layers import MultiHeadAttention, Dropout, LayerNormalization, Dense, Input, Embedding, GlobalAveragePooling1D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.metrics import Precision, Recall
from sklearn.metrics import classification_report
from google.colab import drive
drive.mount('/content/drive')
```

Hình 11. Khởi tạo và nhập các thư viện cần thiết Data Phishing Email

Các thư viện cần thiết bao gồm:

- Xử lý dữ liệu
  - pandas: xử lý dữ liệu dạng bảng (CSV, DataFrame).
  - numpy: hỗ trợ toán học và xử lý mảng.
  - os: thao tác với file và đường dẫn.
- Chia dữ liệu và đánh giá mô hình
  - train_test_split (từ sklearn.model_selection): chia tập train/test.
  - classification_report (từ sklearn.metrics): đánh giá độ chính xác, recall, F1-score.
- Xử lý dữ liệu mất cân bằng
  - ADASYN (từ imblearn.over_sampling): tạo dữ liệu mới cho lớp thiểu số để cân bằng.
- Tiền xử lý văn bản (NLP)
  - Tokenizer (từ tensorflow.keras.preprocessing.text): chuyển văn bản thành chuỗi số (tokens).
  - pad_sequences (từ tensorflow.keras.preprocessing.sequence): cắt/padding độ dài chuỗi token.
- Xây dựng mô hình Transformer
  - tensorflow.keras.layers:
    - Input, Embedding: định nghĩa đầu vào và ánh xạ từ token → vector.
    - LayerNormalization, Dropout: ổn định mô hình và chống overfitting.
    - MultiHeadAttention: lớp Attention trong Transformer.
    - Dense, GlobalAveragePooling1D: các lớp fully connected và tổng hợp đặc trưng.
  - tensorflow.keras.models.Model: định nghĩa mô hình học sâu tổng thể.
- Tối ưu hóa và huấn luyện mô hình
  - EarlyStopping, ReduceLROnPlateau (từ callbacks): điều khiển quá trình học.
  - Adam (từ optimizers): thuật toán tối ưu hóa.
  - Precision, Recall (từ metrics): đo lường hiệu suất mô hình.
  - regularizers: thêm ràng buộc để tránh overfitting.
- Lưu trữ và tương tác với hệ thống
  - pickle: lưu trữ mô hình hoặc dữ liệu tạm.
  - google.colab.drive: gắn kết Google Drive trong Google Colab.

Bước 2: Làm sạch nội dung văn bản

Bước 2.1: Làm sạch nội dung văn bản Data Phishing Email

```
import re

def clean_text(text):
    text = text.lower()  # Chuyển về chữ thường
    text = re.sub(r"http\S+", "", text)  # Xóa URL
    text = re.sub(r"[^a-z\s]", "", text)  # Xóa ký tự đặc biệt
    text = re.sub(r"\s+", " ", text)  # Chuẩn hóa khoảng trắng
    return text.strip()

df['clean_text'] = df['Email Text'].apply(clean_text)
```

Hình 12. Làm sạch nội dung văn bản Data Phishing Email

Bước 2.2: Làm sạch nội dung văn bản Data Phishing Tank

```
df = pd.read_excel('/content/drive/MyDrive/DataFromPhishingTank.xlsx')
df = df.dropna(subset=['URLs', 'Labels'])
df = df[df['URLs'].str.strip() != '']
df = df[df['Labels'].isin([0, 1])]
urls = df['URLs'].values
labels = df['Labels'].astype(np.float32).values
print(f"Loaded {len(urls)} URLs.")
```

Hình 13. Làm sạch nội dung văn bản Data Phishing Tank

Đoạn code trên xử lý văn bản trong cột Email Text của DataFrame df bằng hàm clean_text:

- text.lower(): Chuyển văn bản về chữ thường.
- re.sub(r"http\S+", "", text): Xóa các URL (bắt đầu bằng http).
- re.sub(r"[^a-z\s]", "", text): Xóa tất cả ký tự không phải chữ cái hoặc khoảng trắng.
- re.sub(r"\s+", " ", text): Thay nhiều khoảng trắng liên tiếp bằng một khoảng trắng.
- text.strip(): Xóa khoảng trắng thừa ở đầu và cuối.
- df['clean_text'] = df['Email Text'].apply(clean_text): Áp dụng hàm clean_text cho cột Email Text và lưu kết quả vào cột mới clean_text.

Kết quả: Văn bản được làm sạch, chỉ chứa chữ cái và khoảng trắng, chuẩn hóa định dạng.

Bước 3: Mã hóa văn bản bằng Tokenizer (ví dụ với BERT)

Bước 3.1: Mã hóa văn bản bằng Tokenizer (ví dụ với BERT) Data Phishing Email

```
# Tokenization
MAX_WORDS = 20000
MAX_LEN = 200
tokenizer = Tokenizer(num_words=MAX_WORDS, oov_token="<OOV>")
tokenizer.fit_on_texts(df['Email Text'])
X_seq = tokenizer.texts_to_sequences(df['Email Text'])
X_pad = pad_sequences(X_seq, maxlen=MAX_LEN, padding='post', truncating='post')
y = np.array(df['label'])
```

Hình 14. Mã hóa văn bản bằng Tokenizer (ví dụ với BERT) Data Phishing Email

Bước 3.2: Mã hóa văn bản bằng Tokenizer (ví dụ với BERT) Data Phishing Tank

```
tokenizer = Tokenizer(num_words=MAX_WORDS, char_level=True, oov_token='<OOV>')
tokenizer.fit_on_texts(urls)
sequences = tokenizer.texts_to_sequences(urls)
X = pad_sequences(sequences, maxlen=MAX_LEN, padding='post', truncating='post')
print(f"Tokenized {len(X)} URLs. Vocabulary size: {len(tokenizer.word_index) + 1}")
```

Hình 15. Mã hóa văn bản bằng Tokenizer (ví dụ với BERT) Data Phishing Tank

Đoạn code trên xử lý văn bản trong cột Email Text và danh sách URL bằng Tokenizer như sau:

- train_test_split(...): Chia tập dữ liệu ban đầu X, y thành:
  - Tập huấn luyện: 80%
  - Tập kiểm tra: 20%
  - random_state=42: Đảm bảo kết quả chia tách giống nhau mỗi lần chạy.
- Counter(y_train): In ra số lượng mẫu thuộc từng nhãn trong tập huấn luyện trước khi cân bằng. Thường sẽ thấy lớp thiểu số có rất ít mẫu so với lớp chiếm đa số.
- ADASYN(...): Khởi tạo bộ tạo mẫu tổng hợp từ lớp thiểu số (kỹ thuật học không giám sát), mục tiêu là:
  - Tự động xác định điểm dữ liệu khó học trong lớp thiểu số.
  - Tạo thêm mẫu tổng hợp ở các vùng này.
  - Giúp cải thiện hiệu suất mô hình trên lớp thiểu số.
- fit_resample(X_train, y_train): Áp dụng ADASYN để tạo dữ liệu huấn luyện mới cân bằng hơn.
  - Kết quả được lưu vào X_train_balanced, y_train_balanced.
- Counter(y_train_balanced): In ra số lượng mẫu thuộc từng nhãn sau khi cân bằng, cho thấy số lượng mẫu của lớp thiểu số đã được tăng lên gần bằng lớp đa số.

Kết quả:

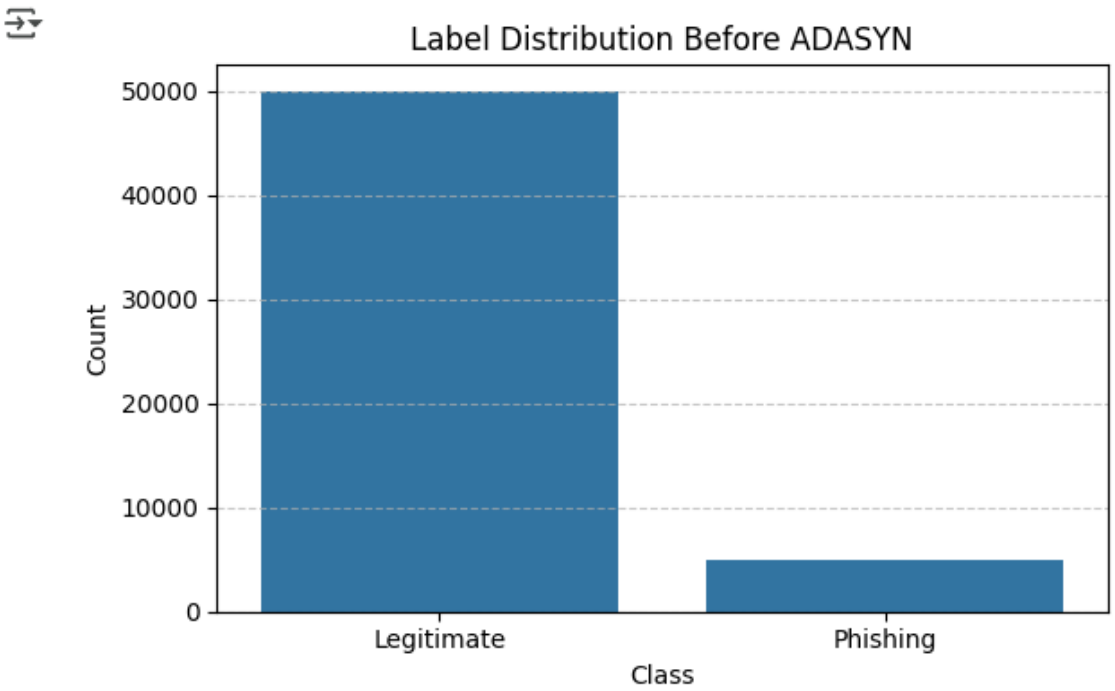
- Trước khi áp dụng ADASYN: Dữ liệu mất cân đối, lớp thiểu số có rất ít mẫu → dễ gây sai lệch cho mô hình.
- Sau khi áp dụng ADASYN: Dữ liệu huấn luyện đã cân bằng hơn → mô hình học tốt cả hai lớp, đặc biệt cải thiện khả năng phát hiện lớp hiếm/phishing.

Bước 4: Xử lý mất cân bằng dữ liệu bằng ADASYN

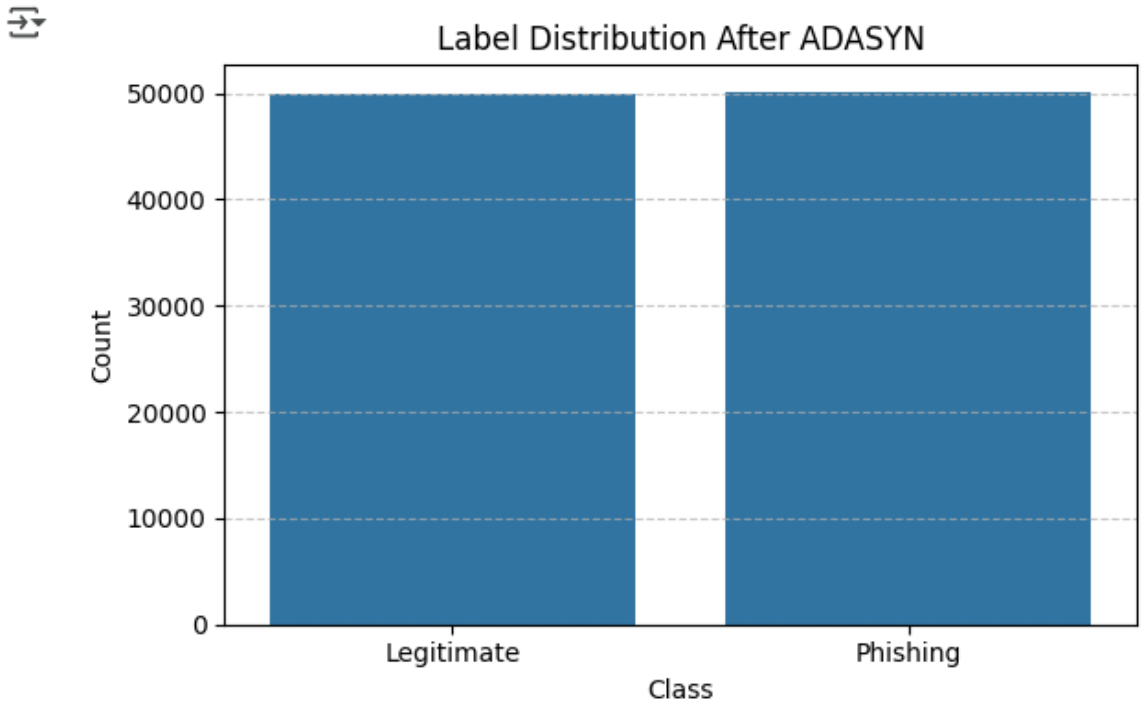
Bước 4.1: Xử lý mất cân bằng dữ liệu bằng ADASYN Data Phishing Tank

```
# Xử lý mất cân bằng dữ liệu bằng ADASYN - Dành cho DataPhishingtank.ipynb

from imblearn.over_sampling import ADASYN
from collections import Counter

# Giả định bạn đã có X_train, y_train từ train_test_split
print("Phân phối dữ liệu ban đầu:", Counter(y_train))

# Tạo và áp dụng ADASYN
adasyn = ADASYN(random_state=42)
X_train_adasyn, y_train_adasyn = adasyn.fit_resample(X_train, y_train)

print("Phân phối sau ADASYN:", Counter(y_train_adasyn))

# Sau đó bạn dùng X_train_adasyn và y_train_adasyn để huấn luyện mô hình
```

Hình 16. Xử lý mất cân bằng dữ liệu bằng ADASYN Data Phishing Tank



Hình 17. Trước ADASYN

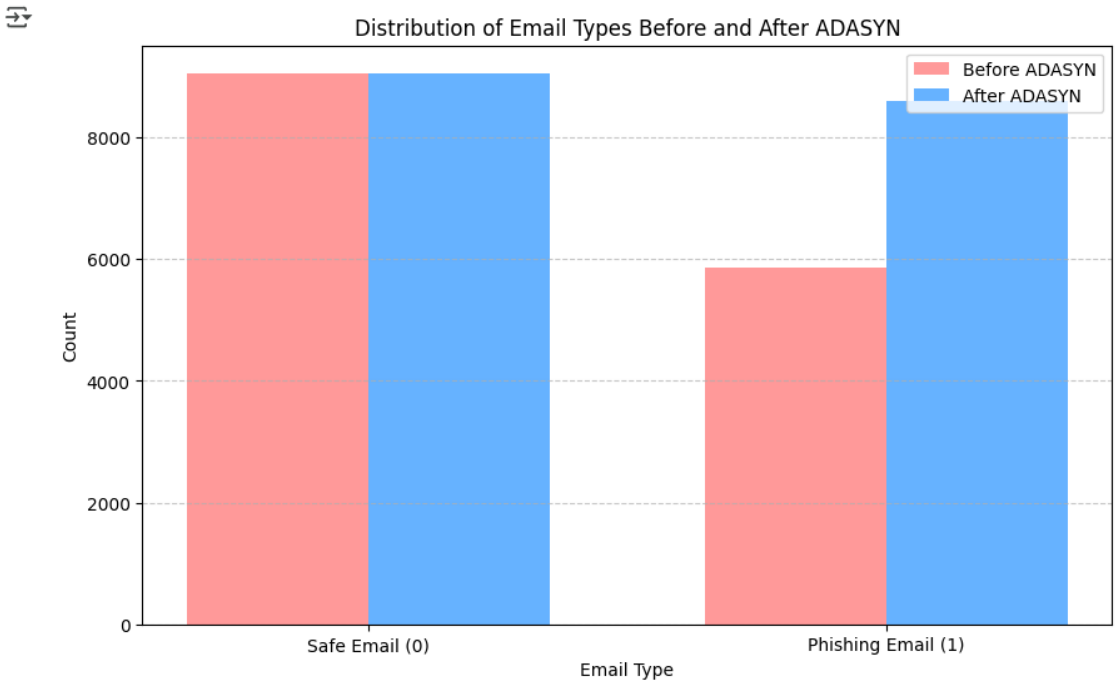


Hình 18. Sau ADASYN

Bước 4.2: Xử lý mất cân bằng dữ liệu bằng ADASYN Data Phishing Email

```
# Xử lý mất cân bằng dữ liệu bằng ADASYN - Dành cho BTL_TTNT.ipynb

from imblearn.over_sampling import ADASYN
from collections import Counter

# Giả sử bạn đã tách dữ liệu thành X_train, y_train
print("Số lượng trước khi xử lý ADASYN:", Counter(y_train))

# Áp dụng ADASYN
adasyn = ADASYN(random_state=42)
X_train_balanced, y_train_balanced = adasyn.fit_resample(X_train, y_train)

print("Số lượng sau ADASYN:", Counter(y_train_balanced))

# Dùng X_train_balanced và y_train_balanced cho bước huấn luyện mô hình
```

Hình 19. Xử lý mất cân bằng dữ liệu bằng ADASYN Data Phishing Email



Hình 20. Distribution of Email Types Trước và sau ADASYN

Đoạn code sử dụng ADASYN để cân bằng dữ liệu không cân đối trong tập huấn luyện:

- Chia dữ liệu: Dữ liệu X, y được chia thành tập huấn luyện (X_train, y_train) và tập kiểm tra (X_test, y_test) với tỉ lệ 80:20, sử dụng train_test_split.
- Cân bằng tập huấn luyện: ADASYN (Adaptive Synthetic Sampling) được dùng để tạo mẫu tổng hợp cho lớp thiểu số trong X_train, y_train, giúp cân bằng phân bố nhãn, kết quả lưu vào X_train_balanced, y_train_balanced.
- Kiểm tra phân bố nhãn: Sử dụng Counter để in số lượng mẫu của từng nhãn trước (y_train) và sau (y_train_balanced) khi áp dụng ADASYN, nhằm đánh giá hiệu quả cân bằng.

Kết quả cụ thể sẽ hiển thị số lượng mẫu chính xác trước và sau khi cân bằng, cho thấy ADASYN đã giảm sự mất cân bằng bằng cách tăng số lượng mẫu của lớp thiểu số.

Bước 5: Tách tập dữ liệu

Bước 5.1: Tách tập dữ liệu Data Phishing Email

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    tokens['input_ids'], df['label'], test_size=0.2, random_state=42
)
```

Hình 21. Tách tập dữ liệu Data Phishing Email

Bước 5.2: Tách tập dữ liệu Data Phishing Tank

```
from sklearn.model_selection import train_test_split

# Tách tập dữ liệu
X_train, X_temp, y_train, y_temp = train_test_split(X, labels, test_size=0.3, random_sta
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_st

print(f"Training set: {len(X_train)} samples")
print(f"Validation set: {len(X_val)} samples")
print(f"Test set: {len(X_test)} samples")
```

Hình 22. Tách tập dữ liệu Data Phishing Tank

Code trên sử dụng train_test_split từ sklearn.model_selection để chia dữ liệu thành tập huấn luyện và tập kiểm tra:

- train_test_split(...): Chia dữ liệu với các tham số:
  - tokens['input_ids']: Dữ liệu đầu vào (tensor mã token từ BERT tokenizer).
  - df['label']: Nhãn tương ứng (cột label trong DataFrame df).
  - test_size=0.2: 20% dữ liệu được dùng cho tập kiểm tra, 80% cho tập huấn luyện.
  - random_state=42: Đặt seed để đảm bảo kết quả chia dữ liệu có thể tái lập.

Kết quả:

- X_train, X_test: Tensor input_ids cho tập huấn luyện và kiểm tra.
- y_train, y_test: Nhãn tương ứng cho tập huấn luyện và kiểm tra.

CHƯƠNG IV. XÂY DỰNG MÔ HÌNH PHÁT HIỆN PHISHING EMAIL BẰNG TRANSFORMER

4.1. Kiến trúc tổng quát của mô hình

Trong nghiên cứu này, chúng em đề xuất một mô hình phát hiện hành vi phishing email dựa trên kiến trúc Transformer Encoder, một biến thể đơn giản hóa của mô hình Transformer gốc. Mô hình được thiết kế để học trực tiếp từ nội dung văn bản email và khai thác hiệu quả các mối quan hệ ngữ nghĩa giữa các từ trong chuỗi.

Kiến trúc tổng quát của mô hình bao gồm các thành phần chính như sau:

- *Embedding Layer:* Chuyển đổi các chỉ mục token (sau khi tokenization) thành các vector đặc trưng có kích thước cố định, từ đó cung cấp thông tin ngữ nghĩa cơ bản cho mô hình.
- *Positional Encoding:* Vì Transformer không có cơ chế xử lý tuần tự như RNN, nên thông tin về thứ tự từ trong chuỗi được thêm vào thông qua hàm mã hóa vị trí. Thành phần này sử dụng các hàm điều hòa sin và cos ở các tần số khác nhau để mã hóa vị trí tuyệt đối của các token trong chuỗi.
- *Multi-Head Self-Attention:* Cho phép mô hình học được các mối liên hệ giữa các từ bất kỳ trong chuỗi, bất kể khoảng cách vị trí, nhờ vào khả năng tự chú ý tại nhiều "góc nhìn" khác nhau (attention heads).
- *Feed-Forward Network (FFN):* Áp dụng một mạng nơ-ron gồm hai lớp tuyến tính kết hợp với hàm kích hoạt ReLU, hoạt động độc lập trên mỗi vị trí trong chuỗi, nhằm tăng tính phi tuyến và khả năng trích xuất đặc trưng sâu.
- *Global Average Pooling + Dense Layers:* Sau khi qua các khối encoder, đầu ra được gộp bằng phép trung bình toàn cục để tạo ra một vector đặc trưng đại diện cho toàn bộ văn bản, sau đó đưa qua các lớp Dense nhằm thực hiện phân loại nhị phân (phishing hoặc hợp lệ).
- *Loss Function:* Hàm mất mát binary_crossentropy được sử dụng để đo lường sai số trong bài toán phân loại nhị phân.
- *Optimizer:* Bộ tối ưu hóa Adam được sử dụng để cập nhật trọng số của mô hình.

Toàn bộ mô hình được triển khai bằng thư viện TensorFlow/Keras với cấu trúc rõ ràng và có khả năng mở rộng cho các bài toán NLP khác.

4.2. Hàm Positional Encoding

Mô hình Transformer không có cơ chế xử lý dữ liệu tuần tự theo thời gian như RNN nên không thể tự nhận biết vị trí tương đối hoặc tuyệt đối của các từ trong chuỗi. Để khắc phục điều này, hàm mã hóa vị trí (Positional Encoding) được đưa vào nhằm truyền thông tin về vị trí của từng token trong chuỗi đầu vào.

```
[ ]  def positional_encoding(position, d_model):
        angle_rads = np.arange(position)[:, np.newaxis] / np.power(10000, (2 * (np.arange(d_model)[np.newaxis, :] // 2)) / np.float32(d_model))
        angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
        angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
        return tf.cast(angle_rads[np.newaxis, ...], dtype=tf.float32)
```

Hình 23. Hàm mã hóa vị trí positional encoding

4.3. Transformer Block

Mỗi khối Transformer (Transformer Encoder Block) là một đơn vị xử lý cơ bản trong kiến trúc mô hình, gồm các thành phần sau:

- *Multi-Head Self-Attention Layer:* Học sự liên kết giữa các token trong cùng một chuỗi, giúp mô hình hiểu ngữ cảnh sâu hơn.
- *Residual Connection và Layer Normalization:* Đầu ra của attention layer được cộng với đầu vào ban đầu (residual connection), sau đó chuẩn hóa bằng layer normalization để tăng tính ổn định khi huấn luyện.
- *Feed-Forward Network (FFN):* Gồm hai lớp Dense tuyến tính với hàm ReLU ở giữa, hoạt động riêng biệt trên từng vị trí trong chuỗi.
- *Một lần nữa Residual + LayerNorm:* Sau FFN, tiếp tục áp dụng kết nối tắt và chuẩn hóa lớp để đảm bảo gradient lan truyền ổn định

```
def transformer_block(inputs, embed_dim, num_heads, ff_dim, rate=0.3):
    attn_output = MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim,
                                     kernel_regularizer=regularizers.l2(1e-2))(inputs, inputs)
    attn_output = Dropout(rate)(attn_output)
    out1 = LayerNormalization(epsilon=1e-6)(inputs + attn_output)

    ffn = Dense(ff_dim, activation='relu', kernel_regularizer=regularizers.l2(1e-2))(out1)
    ffn = Dense(embed_dim, kernel_regularizer=regularizers.l2(1e-2))(ffn)
    ffn = Dropout(rate)(ffn)
    return LayerNormalization(epsilon=1e-6)(out1 + ffn)
```

Hình 24. Khối transformer block

4.4. Xây dựng mô hình hoàn chỉnh

Chúng em xây dựng mô hình Transformer Encoder bằng cách xếp chồng nhiều Transformer Block, mỗi block giúp mô hình học được các đặc trưng ngày càng phức tạp hơn. Cụ thể:

- Nhập chuỗi văn bản đã được tiền xử lý và mã hóa bằng Tokenizer.
- Ánh xạ thành vector embedding.
- Cộng với positional encoding.
- Đưa qua nhiều block encoder (số lớp có thể điều chỉnh, thường là 2–4 cho tác vụ nhỏ).
- Sau đó gộp lại bằng GlobalAveragePooling1D.
- Đưa qua các lớp Dense, cuối cùng là một lớp sigmoid cho phân loại nhị phân.

```
[ ]  def build_model(maxlen, vocab_size, embed_dim, num_heads, ff_dim, num_blocks=2, dropout_rate=0.3):
         inputs = Input(shape=(maxlen,))
         x = Embedding(input_dim=vocab_size, output_dim=embed_dim)(inputs)
         x = LayerNormalization(epsilon=1e-6)(x)
         x += positional_encoding(maxlen, embed_dim)[:, :maxlen, :]

         for _ in range(num_blocks):
             x = transformer_block(x, embed_dim, num_heads, ff_dim, rate=dropout_rate)

         x = GlobalAveragePooling1D()(x)
         x = Dropout(dropout_rate)(x)
         x = Dense(64, activation='relu', kernel_regularizer=regularizers.l2(1e-4))(x)
         x = Dropout(dropout_rate)(x)
         outputs = Dense(1, activation='sigmoid')(x)

         model = Model(inputs=inputs, outputs=outputs)
         model.compile(optimizer=Adam(1e-4), loss='binary_crossentropy',
                       metrics=['accuracy', Precision(name='precision'), Recall(name='recall')])
         return model
```

Hình 25. Mô hình transformer hoàn chỉnh

4.5. Huấn luyện mô hình

Sau khi mô hình được xây dựng, chúng tôi tiến hành huấn luyện với các tham số:

- Epochs: 50
- Batch size: 32
- Validation split: 20%
- Callbacks: EarlyStopping và ReduceLROnPlateau để tránh overfitting và cải thiện tốc độ hội tụ.

```
early_stopping = EarlyStopping(monitor='val_loss', patience=5, min_delta=0.005, mode='min', restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=5,
    min_lr=1e-6
)
# Train
history = model.fit(
    X_train, y_train,
    epochs=50,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping, reduce_lr]
)
```

Hình 26. Tiến hành huấn luyện với các tham số

4.6. Nhận xét ban đầu

Mô hình Transformer với self-attention cho thấy khả năng học đặc trưng tốt từ nội dung email, đặc biệt khi xử lý các câu dài hoặc các biểu hiện tinh vi của hành vi phishing. Việc áp dụng các kỹ thuật như early stopping và regularization (L2) giúp mô hình tránh overfitting, từ đó tăng độ chính xác khi triển khai trên dữ liệu thực tế.

CHƯƠNG V.
KẾT QUẢ ĐẠT ĐƯỢC VÀ ĐÁNH GIÁ MÔ HÌNH

5.1. Kết quả đạt được với dữ liệu phishing email từ kaggle

5.1.1. Kết quả huấn luyện mô hình

Mô hình Transformer được huấn luyện trong 50 epoch với bộ dữ liệu email đã qua xử lý và được gán nhãn (Phishing và Safe). Từ biểu đồ loss và log training, mô hình hội tụ nhanh chóng chỉ sau khoảng 10 epoch đầu tiên. Loss huấn luyện giảm mạnh từ ~9 xuống dưới 0.1, trong khi validation loss cũng giảm đều, cho thấy mô hình học hiệu quả mà không bị overfitting.

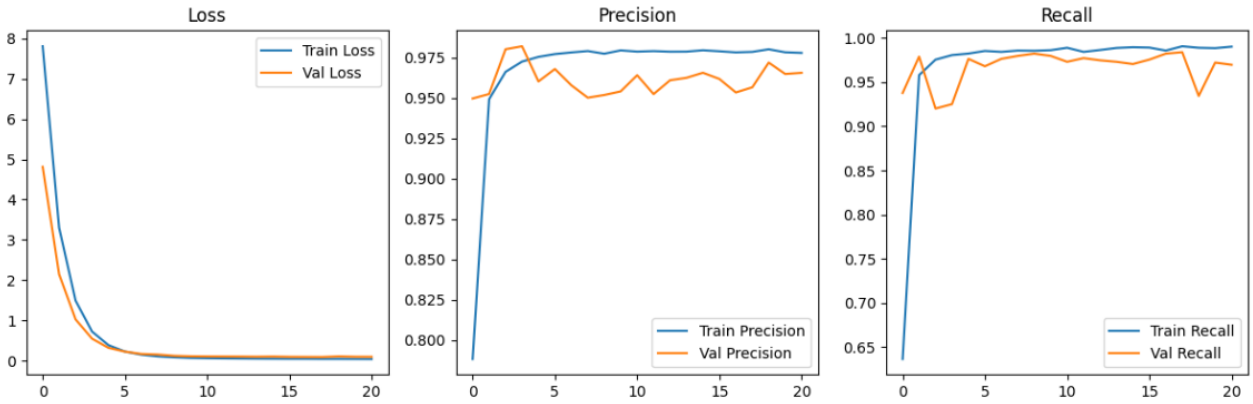
```
Epoch 1/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 34s 47ms/step - loss: 9.7093 - precision: 0.6298 - recall: 0.4144 - val_loss: 4.8131 - val_precision: 0.9498 - val_recall: 0.9378 - learning_rate: 1.0000e-04
Epoch 2/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 27s 34ms/step - loss: 3.9890 - precision: 0.9441 - recall: 0.9548 - val_loss: 2.1445 - val_precision: 0.9526 - val_recall: 0.9790 - learning_rate: 1.0000e-04
Epoch 3/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 34ms/step - loss: 1.7807 - precision: 0.9663 - recall: 0.9783 - val_loss: 1.0252 - val_precision: 0.9803 - val_recall: 0.9202 - learning_rate: 1.0000e-04
Epoch 4/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 35ms/step - loss: 0.8440 - precision: 0.9749 - recall: 0.9826 - val_loss: 0.5465 - val_precision: 0.9822 - val_recall: 0.9252 - learning_rate: 1.0000e-04
Epoch 5/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 34ms/step - loss: 0.4372 - precision: 0.9761 - recall: 0.9822 - val_loss: 0.3186 - val_precision: 0.9603 - val_recall: 0.9765 - learning_rate: 1.0000e-04
Epoch 6/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 35ms/step - loss: 0.2533 - precision: 0.9765 - recall: 0.9842 - val_loss: 0.2243 - val_precision: 0.9681 - val_recall: 0.9681 - learning_rate: 1.0000e-04
Epoch 7/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 21s 35ms/step - loss: 0.1604 - precision: 0.9795 - recall: 0.9827 - val_loss: 0.1660 - val_precision: 0.9580 - val_recall: 0.9765 - learning_rate: 1.0000e-04
Epoch 8/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 21s 36ms/step - loss: 0.1149 - precision: 0.9770 - recall: 0.9828 - val_loss: 0.1513 - val_precision: 0.9503 - val_recall: 0.9798 - learning_rate: 1.0000e-04
Epoch 9/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 36ms/step - loss: 0.0850 - precision: 0.9788 - recall: 0.9880 - val_loss: 0.1191 - val_precision: 0.9520 - val_recall: 0.9824 - learning_rate: 1.0000e-04
Epoch 10/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 36ms/step - loss: 0.0735 - precision: 0.9767 - recall: 0.9877 - val_loss: 0.1079 - val_precision: 0.9542 - val_recall: 0.9798 - learning_rate: 1.0000e-04
Epoch 11/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 36ms/step - loss: 0.0633 - precision: 0.9772 - recall: 0.9879 - val_loss: 0.1039 - val_precision: 0.9642 - val_recall: 0.9731 - learning_rate: 1.0000e-04
Epoch 12/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 36ms/step - loss: 0.0574 - precision: 0.9761 - recall: 0.9830 - val_loss: 0.1018 - val_precision: 0.9525 - val_recall: 0.9773 - learning_rate: 1.0000e-04
Epoch 13/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 35ms/step - loss: 0.0526 - precision: 0.9756 - recall: 0.9880 - val_loss: 0.1004 - val_precision: 0.9611 - val_recall: 0.9748 - learning_rate: 1.0000e-04
Epoch 14/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 35ms/step - loss: 0.0502 - precision: 0.9775 - recall: 0.9896 - val_loss: 0.0969 - val_precision: 0.9626 - val_recall: 0.9731 - learning_rate: 1.0000e-04
Epoch 15/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 36ms/step - loss: 0.0494 - precision: 0.9790 - recall: 0.9899 - val_loss: 0.1010 - val_precision: 0.9657 - val_recall: 0.9706 - learning_rate: 1.0000e-04
Epoch 16/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 21s 37ms/step - loss: 0.0485 - precision: 0.9777 - recall: 0.9853 - val_loss: 0.0940 - val_precision: 0.9619 - val_recall: 0.9756 - learning_rate: 1.0000e-04
Epoch 17/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 37ms/step - loss: 0.0463 - precision: 0.9768 - recall: 0.9848 - val_loss: 0.0917 - val_precision: 0.9535 - val_recall: 0.9824 - learning_rate: 1.0000e-04
Epoch 18/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 37ms/step - loss: 0.0435 - precision: 0.9786 - recall: 0.9903 - val_loss: 0.0899 - val_precision: 0.9567 - val_recall: 0.9840 - learning_rate: 1.0000e-04
Epoch 19/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 14s 36ms/step - loss: 0.0417 - precision: 0.9804 - recall: 0.9906 - val_loss: 0.1049 - val_precision: 0.9720 - val_recall: 0.9345 - learning_rate: 1.0000e-04
Epoch 20/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 14s 37ms/step - loss: 0.0445 - precision: 0.9779 - recall: 0.9887 - val_loss: 0.0960 - val_precision: 0.9650 - val_recall: 0.9723 - learning_rate: 1.0000e-04
Epoch 21/50
373/373 ━━━━━━━━━━━━━━━━━━━━ 13s 36ms/step - loss: 0.0399 - precision: 0.9805 - recall: 0.9874 - val_loss: 0.0942 - val_precision: 0.9657 - val_recall: 0.9697 - learning_rate: 1.0000e-04
```

Hình 27. Quá trình train mô hình với dữ liệu phishing email từ kaggle

Một số chỉ số tiêu biểu:

- Precision (Huấn luyện): ~0.97
- Recall (Huấn luyện): ~0.98 đến 0.99
- Precision (Validation): dao động quanh 0.95–0.96
- Recall (Validation): đạt đến 0.98 trong nhiều epoch

Biểu đồ loss, precision và recall phản ánh rõ sự ổn định và độ chính xác cao của mô hình trong quá trình huấn luyện và đánh giá.



Hình 28. Biểu đồ loss, precision và recall

5.1.2. Ma trận nhầm lẫn

Ma trận nhầm lẫn ở hình 29 thể hiện hiệu suất phân loại mô hình trên tập kiểm thử:

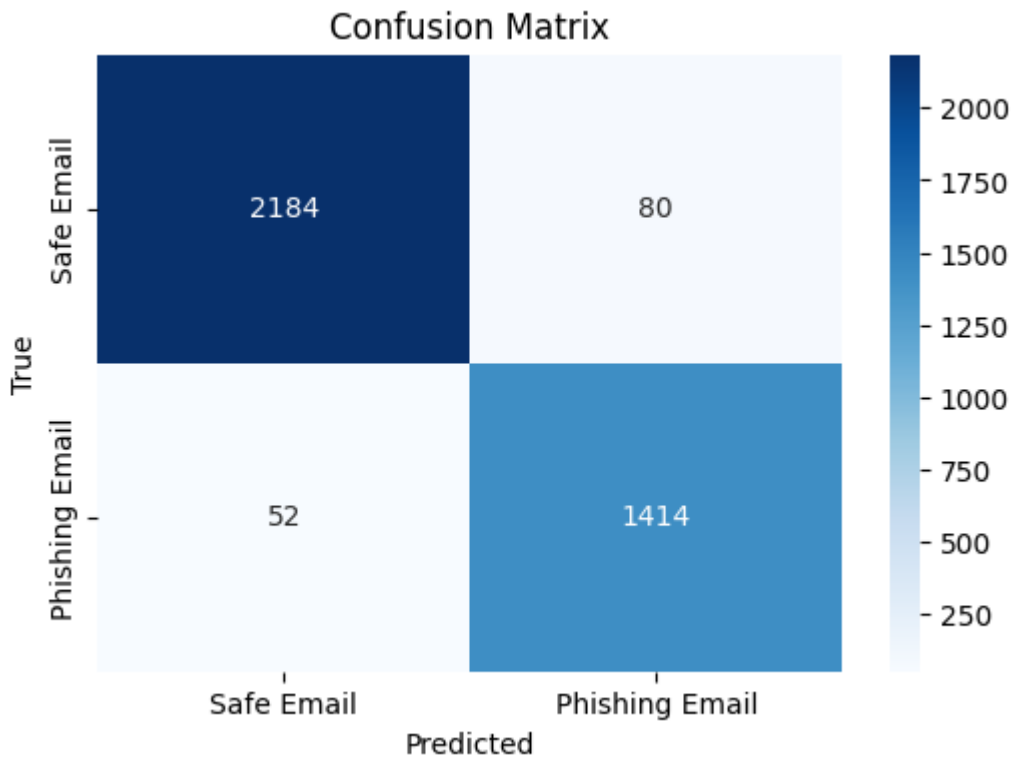
	Dự đoán: Safe	**Dự đoán: Phishing**
Thực tế: Safe	2184	80
Thực tế: Phishing	52	1414

Từ ma trận trên:

- True Positives (TP) = 1414 (phishing → đúng)
- True Negatives (TN) = 2184 (safe → đúng)
- False Positives (FP) = 80 (safe bị đoán nhầm là phishing)
- False Negatives (FN) = 52 (phishing bị bỏ sót)

Ta tính được:

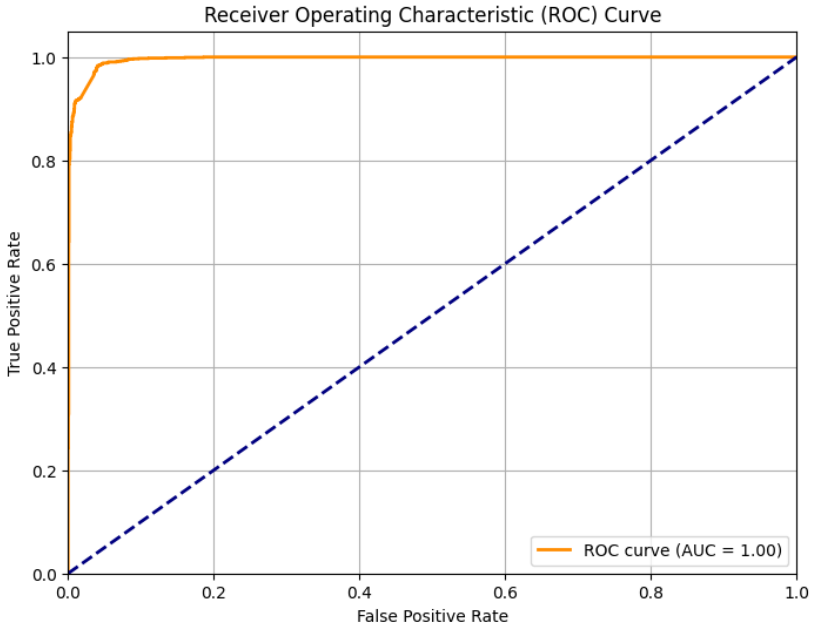
- Độ chính xác (Accuracy) ≈ (2184 + 1414) / (2184 + 1414 + 80 + 52) ≈ 96.6%
- Precision ≈ 1414 / (1414 + 80) ≈ 94.6%
- Recall ≈ 1414 / (1414 + 52) ≈ 96.4%



Hình 29.Ma trận nhầm lẫn

5.1.3. Đường cong ROC và AUC

Hình biểu diễn đường cong ROC. Với AUC = 1.00, mô hình đạt hiệu năng phân loại gần như hoàn hảo, cho thấy khả năng tách biệt rõ ràng giữa hai lớp: "phishing email" và "safe email".



Hình 30. Đường cong ROC

5.1.4. Đánh giá tổng quan mô hình

a, Ưu điểm:

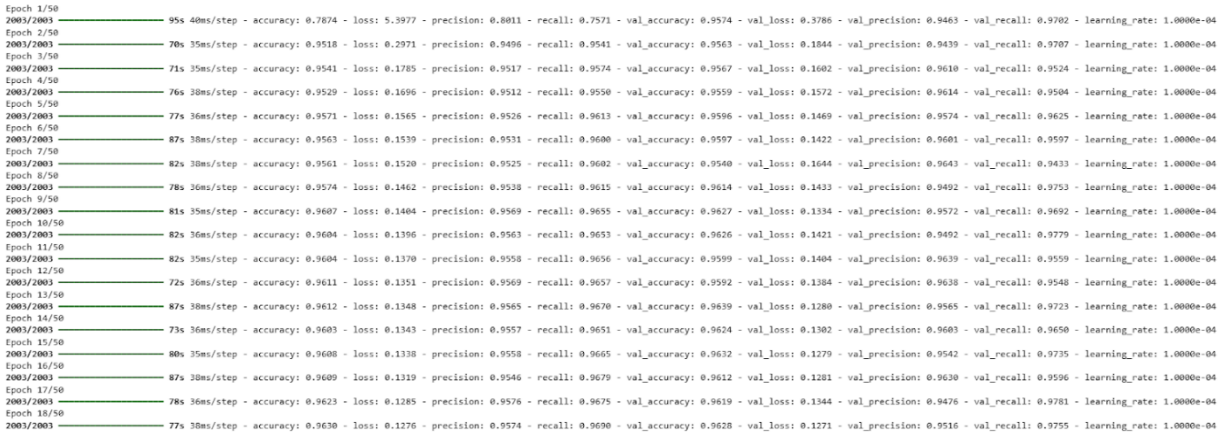
- Mô hình học rất nhanh và ổn định, không bị overfitting.
- Precision và Recall đều cao, đặc biệt phù hợp với bài toán phishing vốn cần giảm thiểu False Negative.
- AUC đạt 1.00 cho thấy khả năng phân loại tuyệt đối về mặt lý thuyết.

b, Nhược điểm:

- Có thể có một số email phishing tinh vi bị bỏ sót (FN = 52).
- Số lượng False Positive (80) tuy thấp nhưng cần kiểm soát nếu triển khai trong môi trường thực tế (tránh cảnh báo giả).

5.2. Kết quả đạt được với dữ liệu phishing email từ kaggle

5.2.1. Kết quả huấn luyện mô hình



Hình 31. Kết quả huấn luyện mô hình

Nhận xét:

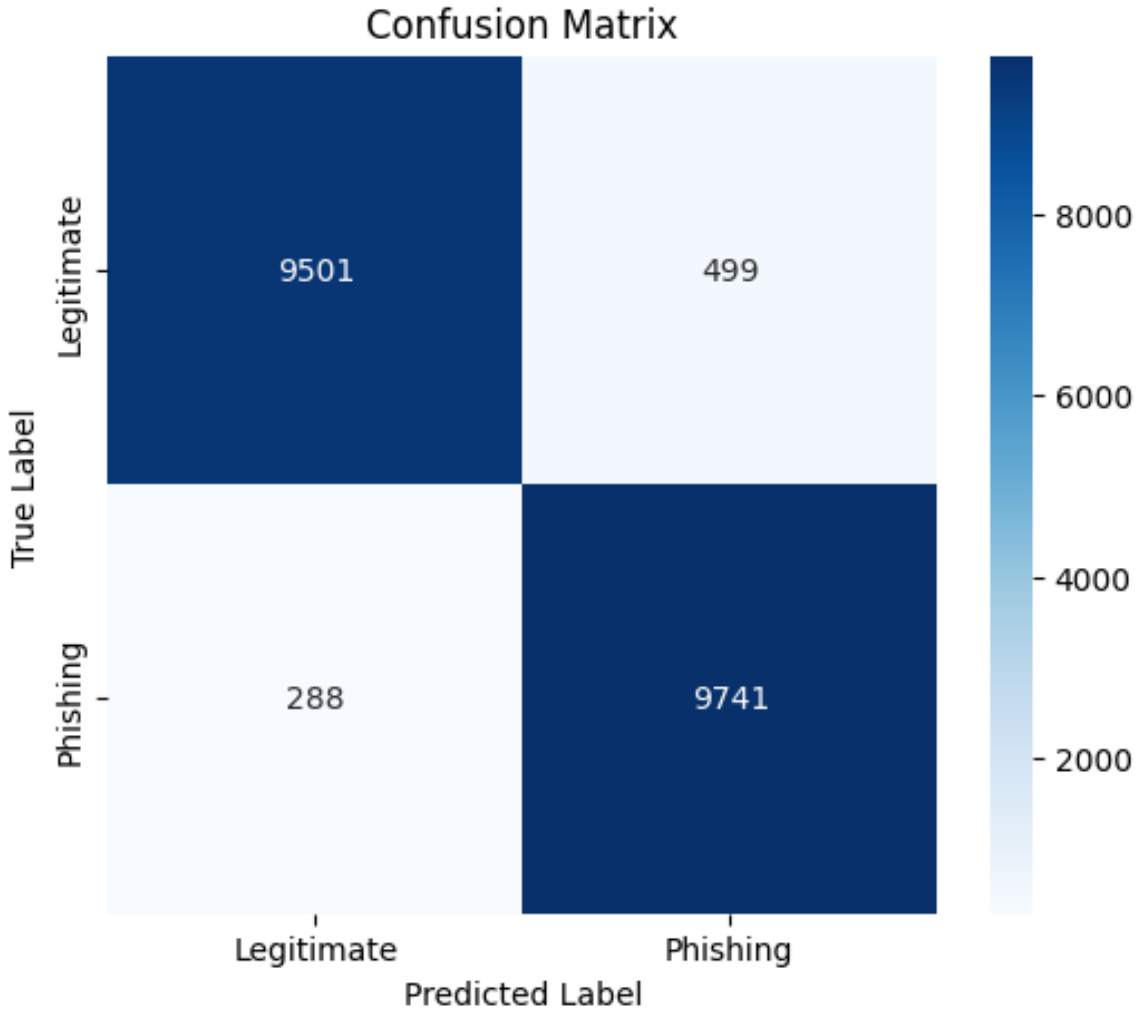
- Epochs: 50
- Độ chính xác huấn luyện (training accuracy) cuối cùng: ≈ 96.30%
- Độ chính xác kiểm tra (validation accuracy) cuối cùng: ≈ 96.28%
- Precision (val): ~95.16%
- Recall (val): ~97.55%
- Loss giảm đều, không có dấu hiệu overfitting rõ rệt → mô hình ổn định.

5.2.2. Confusion Matrix (Ma trận nhầm lẫn)

	Dự đoán: Legitimate	**Dự đoán: Phishing**
Thực tế: Legitimate	9501 (TN)	499 (FP)
Thực tế: Phishing	288 (FN)	9741 (TP)

Nhận xét:

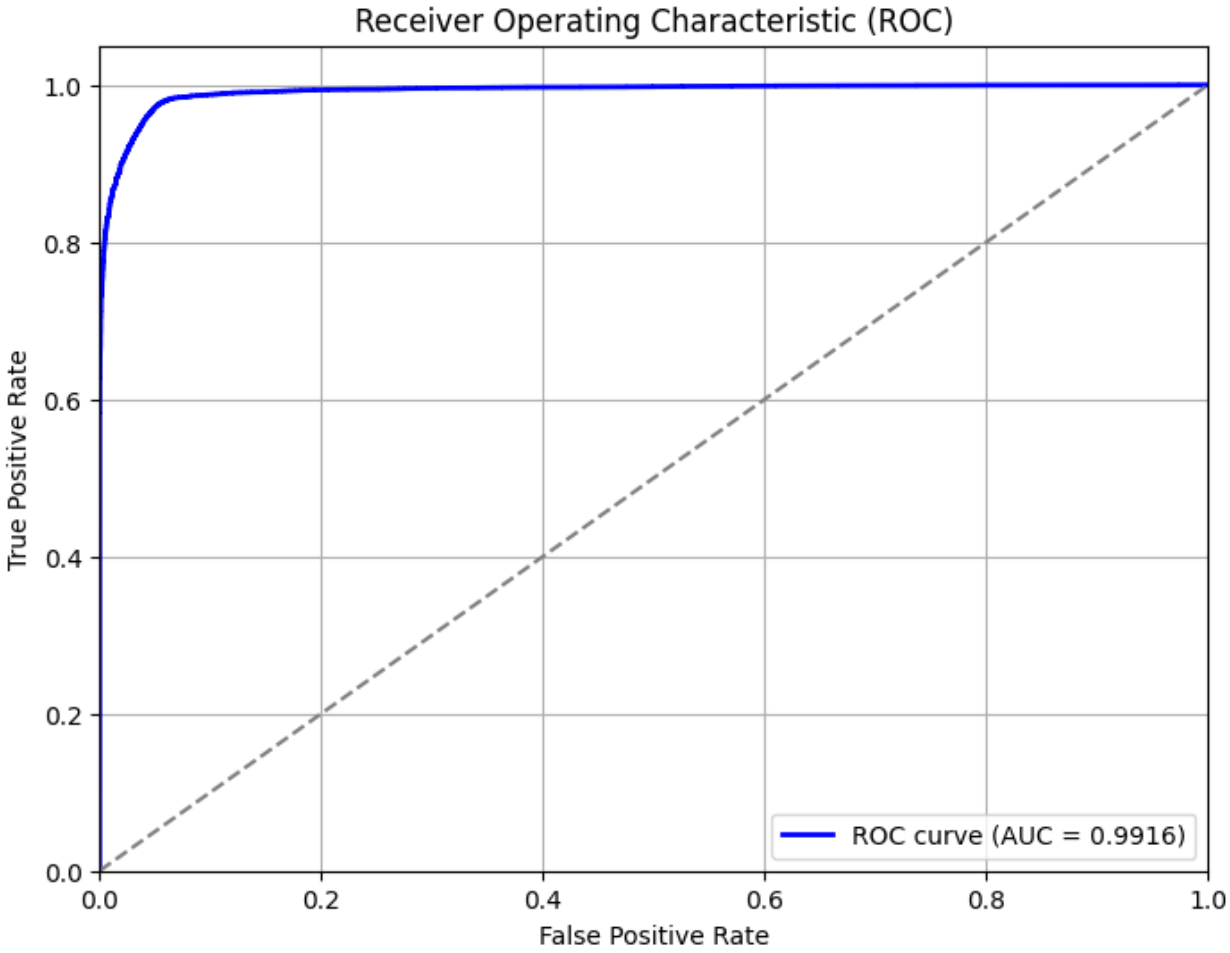
- True Positive (TP): 9741 → Phishing được dự đoán đúng
- True Negative (TN): 9501 → Legitimate được dự đoán đúng
- False Positive (FP): 499 → Legitimate bị nhầm là phishing
- False Negative (FN): 288 → Phishing bị nhầm là legitimate



Hình 32. Ma trận nhầm lẫn

5.2.3. Đường cong ROC-AUC

- AUC (Area Under Curve): 0.9916
- ROC đường cong gần sát góc trên trái (0,1) → mô hình phân biệt rất tốt giữa phishing và legitimate.



Hình 33. Đường cong ROC

5.2.4. Đánh giá tổng quan mô hình

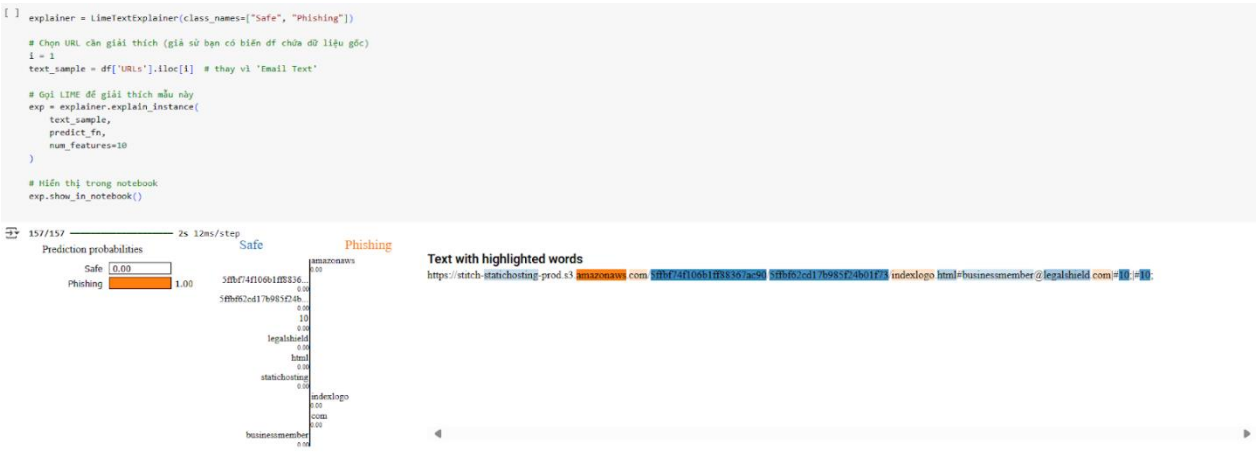
Mô hình Transformer hoạt động rất hiệu quả trên dữ liệu PhishingTank, với:

- Độ chính xác cao (>96%)
- AUC gần tiệm cận 1.0
- Recall cao → khả năng phát hiện phishing tốt
- Sai số (FN, FP) tương đối thấp.

5.3. Diễn giải mô hình bằng LIME

Để tăng tính minh bạch và khả năng kiểm chứng trong các quyết định phân loại, nhóm đã áp dụng kỹ thuật LIME (Local Interpretable Model-agnostic Explanations) nhằm giải thích lý do mô hình Transformer đưa ra dự đoán "phishing" đối với từng email cụ thể. LIME hoạt động bằng cách làm nhiễu (perturbation) văn bản đầu vào và đánh giá lại xác suất dự đoán tương ứng, từ đó xác định các thành phần có ảnh hưởng lớn nhất tới quyết định của mô hình.

Ví dụ, với một URL được trích xuất từ dữ liệu thực tế của PhishingTank, mô hình đã dự đoán với xác suất 1.00 (tức 100%) rằng đây là một email phishing. LIME xác định rằng các từ khóa như amazonaws, s3, static, các chuỗi ký tự ngẫu nhiên và tên miền .com là những yếu tố đóng góp quan trọng vào dự đoán phishing. Những từ khóa này thường xuất hiện trong các URL giả mạo và đóng vai trò mấu chốt trong việc đánh lừa người dùng truy cập vào trang web lừa đảo.



Hình 34. Diễn giải mô hình Transformer bằng LIME trên URL phishing từ PhishingTank

Hình 35. Diễn giải mô hình Transformer bằng LIME trên tập dữ liệu phishing email từ kaggle

Minh họa kết quả diễn giải từ LIME, trong đó các từ được tô màu theo mức độ đóng góp của chúng vào việc mô hình phân loại email là phishing. Màu cam biểu thị tác động mạnh tới lớp "Phishing", trong khi các từ ít liên quan hơn sẽ không được tô màu hoặc có trọng số thấp hơn.

Kết quả cho thấy mô hình không chỉ đơn thuần dựa vào một từ duy nhất, mà còn học được các mẫu cấu trúc URL đặc trưng cho phishing như tên miền lưu trữ tạm thời, thư mục chứa chuỗi ngẫu nhiên và các địa chỉ email mạo danh. Đây là tín hiệu tích cực, khẳng định mô hình thực sự học được đặc trưng sâu trong nội dung, chứ không chỉ ghi nhớ từ khóa đơn lẻ.

CHƯƠNG VI: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Trong khuôn khổ đề tài, nhóm đã tiến hành xây dựng và triển khai mô hình Transformer nhằm phát hiện hành vi phishing trong email dựa trên nội dung văn bản. Mô hình được thiết kế dựa trên kiến trúc Transformer Encoder với các thành phần chính như lớp embedding, positional encoding, multi-head self-attention, mạng feed-forward và các lớp dense kết nối đầu ra. Toàn bộ mô hình được xây dựng trên nền tảng TensorFlow/Keras và được huấn luyện với hàm mất mát binary_crossentropy.

Mô hình được huấn luyện và đánh giá trên hai tập dữ liệu khác nhau: (1) tập dữ liệu từ Kaggle, bao gồm các email đã được gán nhãn rõ ràng; (2) tập dữ liệu thực tế thu thập từ PhishingTank. Kết quả thực nghiệm cho thấy mô hình đạt được độ chính xác cao, với AUC lên tới 1.00 trên tập Kaggle và 0.9916 trên tập PhishingTank. Các chỉ số precision và recall đều đạt mức >95%, phản ánh hiệu năng tốt trong việc phân loại email phishing và email hợp lệ.

Những kết quả thu được cho thấy mô hình Transformer là một lựa chọn hiệu quả cho bài toán phát hiện phishing trong email, đặc biệt trong bối cảnh dữ liệu văn bản ngày càng phức tạp và tinh vi.

6.2. Ưu điểm của mô hình

Qua quá trình triển khai và đánh giá, mô hình Transformer thể hiện một số ưu điểm nổi bật như sau:

- *Hiệu quả phân loại cao:* Độ chính xác và AUC cao, cho thấy mô hình có khả năng phân biệt tốt giữa email phishing và email hợp lệ.
- *Khả năng tổng quát tốt:* Mô hình không chỉ hoạt động hiệu quả trên tập huấn luyện mà còn duy trì hiệu suất cao trên tập dữ liệu thực tế từ PhishingTank.
- *Phù hợp với dữ liệu dài và ngữ cảnh rộng:* Cơ chế self-attention giúp mô hình học được mối quan hệ giữa các từ cách xa nhau, điều mà các mô hình truyền thống khó thực hiện.
- *Tốc độ huấn luyện nhanh:* Thời gian huấn luyện ngắn, không đòi hỏi cấu hình phần cứng quá cao, phù hợp với môi trường học thuật và ứng dụng thực tiễn.

6.3. Hạn chế của mô hình

Bên cạnh những ưu điểm nêu trên, mô hình vẫn tồn tại một số hạn chế nhất định:

- *Tồn tại sai số phân loại:* Mô hình vẫn cho ra một lượng nhỏ false positive và false negative. Trong bối cảnh thực tế, điều này có thể dẫn đến việc cảnh báo sai hoặc bỏ sót email lừa đảo.
- *Chưa khai thác đặc trưng ngoài văn bản:* Việc chỉ sử dụng nội dung văn bản email có thể chưa đủ để phân loại chính xác trong các trường hợp phishing tinh vi. Các đặc trưng như header, địa chỉ IP, domain, liên kết URL... chưa được tích hợp.
- *Thiếu khả năng giải thích kết quả:* Transformer là mô hình dạng "hộp đen", khó lý giải tại sao một email bị gán nhãn phishing, điều này gây khó khăn trong công tác phân tích và kiểm định hệ thống.

6.4. Hướng phát triển

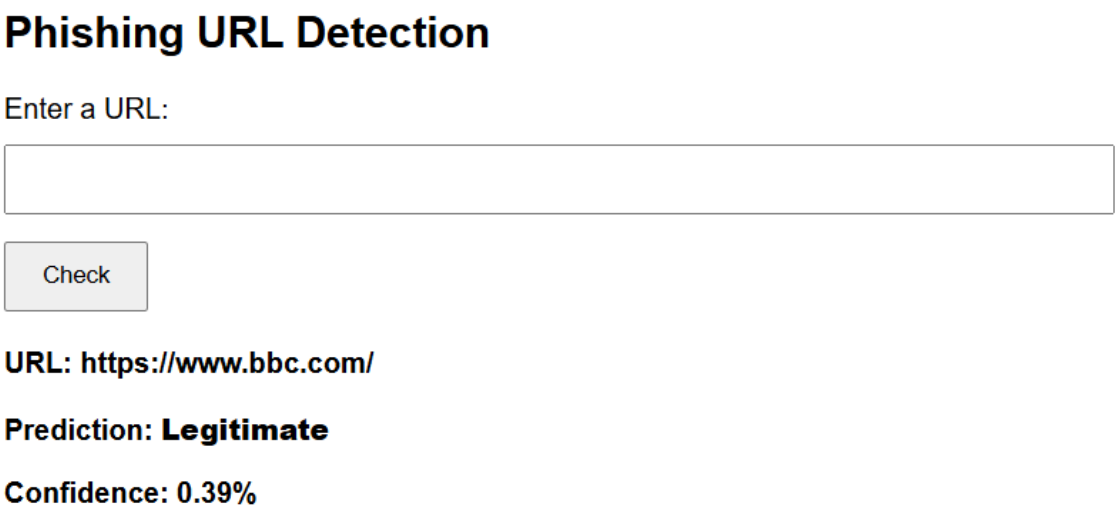
Để nâng cao hiệu quả ứng dụng mô hình trong thực tiễn, nhóm đề xuất một số hướng phát triển trong tương lai:

- *Tích hợp thêm đặc trưng phi văn bản:* Bao gồm tiêu đề email, thông tin địa chỉ gửi/nhận, domain, liên kết URL, và các tệp đính kèm... nhằm cải thiện độ chính xác.
- Sử dụng mô hình ngôn ngữ tiền huấn luyện (như BERT, RoBERTa) để tận dụng các biểu diễn ngữ nghĩa mạnh mẽ, từ đó tăng khả năng hiểu nội dung sâu sắc hơn.
- *Tối ưu triển khai thực tế:* Triển khai mô hình dưới dạng dịch vụ API hoặc tích hợp vào hệ thống lọc email trong doanh nghiệp nhằm phát hiện sớm và ngăn chặn các hành vi tấn công lừa đảo.

6.5. Triển khai mô hình

Nhằm kiểm chứng khả năng ứng dụng thực tế của mô hình Transformer trong việc phát hiện hành vi phishing qua email, nhóm đã tiến hành triển khai mô hình dưới dạng giao diện demo trên môi trường local.

- Giao diện người dùng: Giao diện được thiết kế đơn giản, trực quan với các thành phần chính:
    - *Khung nhập văn bản:* Cho phép người dùng dán hoặc gõ nội dung email cần kiểm tra.
    - *Nút "Phân loại":* Khi nhấn, mô hình sẽ xử lý văn bản và đưa ra kết quả phân loại.
    - *Kết quả dự đoán:* Hiển thị nhãn phân loại (Phishing hoặc Hợp lệ) cùng xác suất dự đoán tương ứng dưới dạng phần trăm.



Hình 36. Giao diện demo trên môi trường local

- Công nghệ sử dụng
  - *Ngôn ngữ:* Python
  - *Giao diện:* Tkinter (hoặc Gradio nếu dùng giao diện web local)
  - *Mô hình phía sau:* Được xây dựng bằng TensorFlow/Keras và nạp từ file .h5 đã huấn luyện.
  - *Pipeline xử lý đầu vào:* Bao gồm tokenizer, padding và dự đoán mô hình.
- Lợi ích
  - Giúp trực quan hóa khả năng hoạt động của mô hình với đầu vào thực tế.
  - Dễ dàng thử nghiệm với các email khác nhau để đánh giá hiệu quả mô hình.
  - Là bước đệm cho việc tích hợp vào các hệ thống lớn hơn như phần mềm lọc thư rác doanh nghiệp hoặc API phân loại email.

KẾT LUẬN

Trong bối cảnh các cuộc tấn công mạng ngày càng gia tăng cả về số lượng lẫn mức độ tinh vi, đặc biệt là các hành vi lừa đảo qua email (phishing email), việc nghiên cứu và ứng dụng các mô hình học sâu tiên tiến để phát hiện sớm và chính xác các mối đe dọa là vô cùng cần thiết. Đề tài này đã tập trung triển khai mô hình Transformer – một kiến trúc nổi bật trong lĩnh vực xử lý ngôn ngữ tự nhiên – nhằm phát hiện hành vi phishing email thông qua việc học các đặc trưng ngữ nghĩa và ngữ cảnh từ nội dung văn bản email.

Thông qua các bước tiền xử lý dữ liệu gồm tokenization, padding, cân bằng dữ liệu với ADASYN, cùng quá trình huấn luyện mô hình Transformer Encoder, kết quả thực nghiệm cho thấy mô hình đạt được độ chính xác và khả năng phân loại vượt trội so với các phương pháp truyền thống. Mô hình không chỉ học được mối quan hệ phức tạp giữa các từ trong câu, mà còn thích ứng tốt với những biến thể tinh vi trong nội dung email giả mạo.

Tuy nhiên, trong quá trình nghiên cứu, đề tài cũng gặp một số hạn chế nhất định như phụ thuộc vào chất lượng tập dữ liệu, chi phí tính toán cao của mô hình Transformer, cũng như thách thức trong việc xử lý các email có ngôn ngữ phi cấu trúc hoặc đa ngữ. Các hạn chế này mở ra hướng phát triển tiếp theo, như kết hợp thêm các mô hình ngôn ngữ tiên tiến (như BERT hoặc GPT), tích hợp thông tin metadata của email (header, IP gửi...), hoặc triển khai mô hình trong môi trường thời gian thực.

Tóm lại, đề tài đã chứng minh tiềm năng lớn của kiến trúc Transformer trong việc phát hiện hành vi phishing email, đồng thời mở ra hướng tiếp cận mới, thông minh và hiệu quả hơn cho các hệ thống phòng chống tấn công mạng hiện đại.

TÀI LIỆU THAM KHẢO

[1] M. Tran, "vnetwork," 13 June 2025. [Online]. Available: https://www.vnetwork.vn/news/phishing-email-la-gi/.

[2] S. Atawneh, "mdpi," 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/20/4261.

[3] perception-point, "perception-point," 2024. [Online]. Available: https://perception-point.io/guides/email-security/email-scams-phishing-quishing-generative-ai/.

[4] tapchinganhang, "tapchinganhang," 19 July 2024. [Online]. Available: https://tapchinganhang.gov.vn/nghien-cuu-xay-dung-mo-hinh-hoc-sau-phat-hien-tin-nhan-rac-dua-tren-bo-du-lieu-phuc-hop-duoc-cap-nhat-10485.html.

[5] M. Nhung, "pcphuyen," 11 June 2025. [Online]. Available: https://pcphuyen.cpc.vn/Tintuc_Sukien/Tin_Chitiet/articleId/95325.

[6] knowbe4, "knowbe4," March 2025. [Online]. Available: https://www.knowbe4.com/hubfs/Phishing-Threat-Trends-2025_Report.pdf.

[7] B. Santos, "anubisnetworks," 18 February 2025. [Online]. Available: https://www.anubisnetworks.com/blog/phishing-trends-for-2025.

[8] V. T. Trang, "vbee," 11 June 2025. [Online]. Available: https://vbee.vn/blog/ai/deep-learning/?srsltid=AfmBOoqRUk3-22MFO8M7Eao7HDF1J-vpkcSeJh5at8zWFGs3HTFxAZht.

[9] T. N. N. Cương, "antoanthongtin," 30 December 2024. [Online]. Available: https://antoanthongtin.vn/tin/phong-chong-su-dung-khong-gian-mang-de-lua-dao-trong-tinh-hinh-hien-nay.

[10] L. H. Khôi, "base.vn," 28 May 2025. [Online]. Available: https://base.vn/blog/transformer-model/.