

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

VŨ ANH KIỆT

KHÓA LUẬN TỐT NGHIỆP
PHÁT HIỆN LỖ HỔNG TRONG HỢP ĐỒNG THÔNG
MINH TRÊN MẠNG LIÊN CHUỖI KHỐI BẰNG
PHƯƠNG PHÁP HỌC MÁY
**SMART CONTRACT VULNERABILITIES AUTOMATIC
DETECTION ON THE CROSS-CHAIN NETWORK USING
MACHINE LEARNING**

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh - 2023

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

VŨ ANH KIỆT - 20520605

KHÓA LUẬN TỐT NGHIỆP
PHÁT HIỆN LỖ HỔNG TRONG HỢP ĐỒNG THÔNG
MINH TRÊN MẠNG LIÊN CHUỖI KHỐI BẰNG
PHƯƠNG PHÁP HỌC MÁY
**SMART CONTRACT VULNERABILITIES AUTOMATIC
DETECTION ON THE CROSS-CHAIN NETWORK USING
MACHINE LEARNING**

CỦ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:

TS. Phạm Văn Hậu
ThS. Trần Tuấn Dũng

TP. Hồ Chí Minh - 2023

LỜI CẢM ƠN

Trước hết, em muốn bày tỏ lòng biết ơn chân thành đến Ban giám hiệu của Trường Đại học Công nghệ Thông tin - Đại học Quốc Gia Thành Phố Hồ Chí Minh vì đã tạo điều kiện thuận lợi về cơ sở vật chất để hỗ trợ quá trình học tập và nghiên cứu, cũng như việc thực hiện khoá luận. Em muốn gửi lời tri ân đặc biệt đến quý thầy cô giáo viên tại trường, những người đã chia sẻ kiến thức quý báu và kinh nghiệm thực tế trong suốt thời gian học. Em muốn bày tỏ lòng biết ơn đặc biệt đến thầy Phạm Văn Hậu và thầy Trần Tuấn Dũng vì sự hướng dẫn tận tình, những ý kiến đóng góp và động viên suốt quá trình nghiên cứu. Cả hai thầy không chỉ truyền đạt kiến thức mà còn hướng dẫn cách học, cách suy nghĩ và phương pháp nghiên cứu khoa học để hoàn thiện khoá luận. Em cũng muốn bày tỏ lòng biết ơn đến gia đình và bạn bè, những người đã động viên và đóng góp ý kiến quý báu trong quá trình nghiên cứu. Cuối cùng, do hạn chế kiến thức, khoá luận của em có thể không tránh khỏi những thiếu sót. Em hy vọng nhận được phản hồi, ý kiến đóng góp và phê bình từ quý thầy cô để làm cho khoá luận trở nên hoàn thiện hơn.

Võ Anh Kiệt

MỤC LỤC

LỜI CẢM ƠN	i
MỤC LỤC	ii
DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT	iv
DANH MỤC CÁC HÌNH VẼ	v
DANH MỤC CÁC BẢNG BIỂU	vi
TÓM TẮT KHOÁ LUẬN	1
 CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI	2
1.1 Tổng quan tình hình nghiên cứu	2
1.2 Các công trình nghiên cứu liên quan	7
1.3 Mục tiêu, đối tượng, và phạm vi nghiên cứu	12
1.3.1 Mục tiêu nghiên cứu	12
1.3.2 Đối tượng nghiên cứu	12
1.3.3 Phạm vi nghiên cứu	12
1.3.4 Cấu trúc khoá luận tốt nghiệp	13
 CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	14
2.1 Tổng quan về các hệ thống phân tán	14
2.1.1 Tổng quan về hệ thống mạng Blockchain - Chuỗi khối . .	14
2.1.2 Tổng quan về hệ thống mạng liên chuỗi khối	18
2.2 Tổng quan về hợp đồng thông minh	19
2.2.1 Hợp đồng thông minh	19
2.2.2 Ngôn ngữ solidity	21
2.2.3 Các loại lỗ hổng trong hợp đồng	28
2.3 Các phương pháp phát hiện lỗ hổng tự động	34
2.3.1 Phương pháp học máy	34

2.3.2 Phương pháp học sâu	41
CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT	47
3.1 Tổng quan mô hình	47
3.1.1 Các thành phần chính của mô hình	47
3.1.2 Mô hình cầu nối	49
3.2 Phát hiện các lỗ hổng bằng phương pháp học máy và học sâu . .	51
3.2.1 Xây dựng tập dữ liệu	51
3.2.2 Gán nhãn các mẫu và xử lý dữ liệu	53
3.2.3 Ứng dụng các mô hình học máy tổng việc phát hiện các lỗ hổng	55
3.2.4 Ứng dụng các mô hình học sâu trong việc phát hiện các lỗ hổng	55
CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ	57
4.1 Thiết lập thực nghiệm	57
4.1.1 Môi trường thực nghiệm	57
4.1.2 Chỉ số đánh giá	58
4.1.3 Kịch bản thực nghiệm	60
4.2 Kết quả thực nghiệm	62
4.2.1 Kết quả về mặt hiệu suất	62
4.2.2 Kết quả về mặt thời gian	63
4.3 Thảo luận	64
CHƯƠNG 5. KẾT LUẬN	66
5.1 Kết luận	66
5.2 Hướng phát triển	67
TÀI LIỆU THAM KHẢO	68

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Pre-Training Approach
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machine
CNN	Convolutional Neural Network
LSTM	Long short term memory
FNN	Feedforward Neural Network
ML	Machine learning
DL	Deep learning
SC	Smart Contract
Smart Contract	Hợp đồng thông minh
ETH	Đồng tiền kỹ thuật số Ethereum

DANH MỤC CÁC HÌNH VẼ

Hình 1.1	Công trình nghiên cứu của Xu và nhóm nghiên cứu	8
Hình 1.2	Phương pháp đề xuất của Deng và nhóm nghiên cứu	8
Hình 1.3	Hệ thống của của Huang và nhóm nghiên cứu	9
Hình 1.4	Tổng quan phương pháp của Jiang và nhóm nghiên cứu . . .	10
Hình 1.5	Công cụ phân tích của Parizi và nhóm nghiên cứu	11
Hình 2.1	Tỷ trọng các lĩnh vực ứng dụng công nghệ Blockchain	15
Hình 2.2	Các ứng dụng khi sử dụng nền tảng blockchain	17
Hình 2.3	Giao tiếp liên chuỗi giữa hai chuỗi khối thông qua sidechain	18
Hình 2.4	Sự khác nhau giữa hợp đồng truyền thông và hợp đồng thông minh	21
Hình 2.5	Quá trình biên dịch và thực thi hợp đồng thông minh	23
Hình 2.6	Sự tương quan của source code, bytecode và opcode của hợp đồng thông minh	24
Hình 2.7	Cấu trúc của bytecode trong hợp đồng thông minh	25
Hình 2.8	Cấu trúc của opcode trong hợp đồng thông minh	26
Hình 2.9	Mô phỏng lỗ hổng Reentrancy	29
Hình 2.10	Mô phỏng lỗ hổng Interger Overflow	30
Hình 2.11	Mô phỏng lỗ hổng Interger Underflow	30
Hình 2.12	Mô phỏng lỗ hổng Unprotected Ether Withdrawal	31
Hình 2.13	Mô phỏng lỗ hổng Timestamp Dependence	32
Hình 2.14	Mô phỏng lỗ hổng Front Running	33
Hình 2.15	Mô hình học máy Decision Tree	34
Hình 2.16	Mô hình học máy Random Forest	36
Hình 2.17	Mô hình học máy XGBoost	37

Hình 2.18 Mô hình học máy Support Vector Machine	39
Hình 2.19 Mô hình học máy Logistic Regression	40
Hình 2.20 Mô hình học sâu Convolutional Neural Network	41
Hình 2.21 Mô hình học sâu Long Short-Term Memory	43
Hình 2.22 Mô hình học sâu Feedforward neuron network	44
Hình 2.23 Mô hình học sâu RoBERTa	46
Hình 3.1 Mô hình tổng quan	48
Hình 3.2 Các bước chuyển đổi dữ liệu qua cầu nối Sidechain	51
Hình 3.3 Phân bố mẫu lành tính và độc hại	52
Hình 3.4 Phân bố mẫu lành tính và độc hại tương ứng với các lỗ hổng	52
Hình 3.5 Quá trình xử lý dữ liệu	54
Hình 4.1 Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi .	64

DANH MỤC CÁC BẢNG BIỂU

Bảng 4.1	Môi trường thực nghiệm mạng liên chuỗi	57
Bảng 4.2	Môi trường thực nghiệm các phương pháp học máy	58
Bảng 4.3	Hiệu suất của các mô hình học máy trong việc phát hiện lỗ hởng bảo mật trong các hợp đồng thông minh liên chuỗi	62
Bảng 4.4	Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hởng bảo mật trong các hợp đồng thông minh liên chuỗi . .	63

TÓM TẮT KHOÁ LUẬN

Trong những năm gần đây, có nhiều tiến triển trong quá trình phát triển và triển khai hợp đồng thông minh trên mạng chuỗi chéo, việc này giúp tạo điều kiện cho giao tiếp và trao đổi dữ liệu giữa các blockchain khác nhau được diễn ra một cách hiệu quả. Tuy nhiên, việc triển khai các ứng dụng này mang theo rủi ro về an toàn thông tin, đặc biệt là trong việc phát hiện lỗ hổng trong các hợp đồng thông minh, có thể gây nguy hiểm đến tính bảo mật. Trong các nghiên cứu trước đây đã tập trung vào việc xác định và phát hiện lỗ hổng trong hợp đồng thông minh bằng các phương pháp kiểm tra ký tự và thực thi, tuy nhiên, các phương pháp hiện nay vẫn chưa đạt được khả năng phân tích toàn diện. Do đó, trong nghiên cứu này, nhóm đề xuất sử dụng các phương pháp học máy và học sâu để phân tích các lỗ hổng này một cách hiệu quả hơn.

Ở công trình nghiên cứu này, em giới thiệu các phương pháp học máy và học sâu dựa trên ChainSniper - một khung phân tích tích hợp học máy dựa trên sidechain để tự động đánh giá lỗ hổng hợp đồng thông minh chéo chuỗi. Phương pháp mà nhóm đề xuất là xây dựng một tập dữ liệu quy mô lớn gồm 300 đoạn mã có gán nhãn thủ công, được gọi là CrossChainSentinel, đã được biên soạn để huấn luyện các mô hình phân biệt mã dễ bị tấn công và mã an toàn. Các đoạn mã này bao gồm các lỗ hổng: Reentrancy, Interger Overflow/Underflow và Unprotected Ether Withdrawl. Kết quả thực nghiệm đã chứng minh được tính hiệu quả của việc ứng dụng học máy và học sâu giúp tăng sự hiệu quả của việc kiểm tra hợp đồng thông minh cho các ứng dụng phi tập trung phân tán trên nhiều blockchain. Độ chính xác phát hiện đạt mức đáng kể, khẳng định tiềm năng của ChainSniper trong việc tăng cường an ninh thông qua đánh giá tự động và toàn diện mã hợp đồng.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Trong phần này, em sẽ trình bày tổng quan về bối cảnh của vấn đề và đề cập đến các công trình nghiên cứu liên quan để làm sáng tỏ vấn đề đang xét. Đồng thời, em cũng sẽ mô tả phạm vi và cấu trúc của khoá luận, nhằm cung cấp một cái nhìn rõ ràng và toàn diện về nội dung và cách tiếp cận của công trình nghiên cứu.

1.1. Tổng quan tình hình nghiên cứu

Sự phát triển của công nghệ blockchain đã tạo ra bước tiến lớn trong việc hình thành các mạng không tập trung, mang lại khả năng lưu trữ và trao đổi thông tin một cách an toàn và chính xác. Điều này được thực hiện thông qua cơ chế đồng thuận phân tán giữa các nút mạng [1, 2]. Tuy nhiên, một hạn chế của hệ thống blockchain là việc chúng thường được xây dựng một cách độc lập với quy tắc và giao thức riêng, dẫn đến khó khăn trong việc tương tác và trao đổi dữ liệu giữa các blockchain khác nhau.

Đáp ứng vấn đề này, công nghệ liên chuỗi (cross-chain) đã được phát triển như một giải pháp cho phép các mạng blockchain tương tác với nhau một cách an toàn và hiệu quả [3]. Công nghệ này hỗ trợ việc chuyển giao tài sản số giữa các chuỗi khác nhau, từ đó thúc đẩy khả năng tích hợp và phát triển của hệ sinh thái blockchain [4]. Các giải pháp như Polkadot, chẵng hạn, đã cung cấp một khung làm việc cho phép các blockchain độc lập, với cấu trúc và chức năng khác nhau, giao tiếp và tương tác với nhau, mở ra một bước tiến mới trong lĩnh vực công nghệ blockchain và các ứng dụng phi tập trung [5].

Công nghệ liên chuỗi nhằm kết nối các hệ sinh thái blockchain bị cô lập, cho phép tài sản và dữ liệu được chuyển giao và chia sẻ một cách thuận tiện giữa các blockchain khác nhau [6]. Đóng góp này tiến hành giải quyết một vấn đề rất

quan trọng trong việc xử lý các điểm hạn chế về khả năng xử lý cũng như khả năng về mở rộng và đồng thời là chức năng mà các blockchain riêng lẻ thường gặp phải. Hiện tại tồn tại ba cách tiếp cận chính để kết nối và cho phép chuyển tài sản giữa các chuỗi khối khác nhau: các giải pháp công chứng, khóa băm, và các relays/sidechains [7–9]. Cơ chế công chứng là phương pháp đồng thuận trong đó các bên thứ ba tin cậy (công chứng viên) xác minh giao dịch bằng chữ ký số trước khi chúng được thêm vào blockchain, nhằm ngăn giao dịch kép. Giải pháp khóa băm có thể triển khai thông qua cổng để truy cập các hợp đồng khóa thời gian khóa băm (Hash Time Lock Contract - HTLC) trên blockchain từ xa, đảm bảo việc nhận thanh toán trước khi phục dựng tài sản trên blockchain đích. Sidechains là các blockchain phụ được gắn nối với blockchain chính thông qua thanh gài 2 chiều, cho phép chuyển tài sản giữa các chuỗi, thêm tính năng mới cho blockchain chính mà không cần sửa đổi giao thức, giải quyết những vấn đề thách thức như việc mở rộng hệ thống và đảm bảo vấn đề về sự riêng tư của các thông tin. Tuy nhiên, giao thức tương tác chuỗi khối này vẫn tồn tại các vấn đề về tính bí mật, tính riêng tư và sự sẵn sàng của hệ thống tư cần giải quyết.

Hợp đồng thông minh, được biết đến như là các chương trình máy tính dưới dạng mã nguồn, có năng lực tự động hóa việc thực hiện các điều khoản và điều kiện của hợp đồng mà không cần đến sự can thiệp bất kỳ nào đến từ bên ngoài [10]. Chúng được lập trình để tự động kiểm tra và đảm bảo tuân thủ các giao dịch theo logic đã được thiết lập trước. Tuy nhiên, hợp đồng thông minh cũng tiềm ẩn những rủi ro, đặc biệt là trong việc phòng chống những cuộc tấn công mà tận dụng lỗ hổng bảo mật trong mã nguồn của chúng.

Đặc biệt, các cuộc tấn công liên chuỗi nhằm vào việc lợi dụng các điểm yếu trong hợp đồng thông minh tiến hành thực thi trên nhiều mạng blockchain riêng biệt, có thể gây ra tổn thất tài chính nghiêm trọng. Một ví dụ điển hình là vụ việc tấn công gần đây, nơi mà lỗ hổng trong một hợp đồng thông minh liên chuỗi đã bị khai thác, dẫn đến việc mất mát tài sản trị giá hơn 600 triệu đô la [11]. Một trường hợp khác, vào năm 2016, dự án gây quỹ The DAO trên Ethereum

cũng đã trở thành nạn nhân của một cuộc tấn công, mất đi hơn 50 triệu đô la do lỗ hổng trong hợp đồng thông minh của họ. Các sự cố tấn công hợp đồng thông minh tiếp tục xảy ra, ví dụ như vụ tấn công vào ví Parity tháng 7/2017 khiến mất mát 30 triệu đô la tiền điện tử Ether. Hay vụ đánh cắp gần 300.000 đô la từ nền tảng KingDice tháng 8/2017 cũng do lợi dụng lỗ hổng trong mã hợp đồng. Hơn thế nữa, trong gian đoạn gần đây, thị trường blockchain đã tiếp nhận 1 cuộc tấn công là loạt vụ tấn công vào các hợp đồng thông minh trên Binance Smart Chain năm 2021, trong đó có vụ đánh cắp hơn 200 triệu đô la thông qua hợp đồng của Venus Protocol [12, 13]. Như vậy, việc phân tích và bảo mật hợp đồng thông minh trước khi tiến hành thực thi và sử dụng mang tính cấp thiết lớn trong việc hạn chế rủi ro mất mát tài sản.

Hợp đồng thông minh, là các giao thức số được thiết kế để làm đơn giản hóa, kiểm chứng hoặc thực thi các quy trình đàm phán và thực hiện hợp đồng. Chúng được ứng dụng rộng rãi, từ dịch vụ tài chính đến thị trường dự đoán và trong lĩnh vực Internet vạn vật [14]. Những hợp đồng này hoạt động hiệu quả trên các nền tảng blockchain, tự động hóa các hành động theo các điều kiện đã đặt ra trước, giảm bớt nhu cầu cho các bên trung gian. Hợp đồng thông minh, do đó, tạo điều kiện cho việc giao dịch không dựa vào sự tin tưởng và tự động hóa thực hiện các quy trình trong hệ thống blockchain. Với sự phát triển nhanh chóng của công nghệ blockchain và việc ứng dụng nó trong nhiều ngành nghề khác nhau, việc phân tích và kiểm tra bảo mật cho hợp đồng thông minh trở nên cực kỳ quan trọng trước khi chúng được triển khai [15]. Do chúng hoạt động dựa trên mã tự thực thi, bất kỳ lỗ hổng nào cũng có thể gây ra hậu quả lớn. Mặc dù việc kiểm tra và phân tích mã bằng phương pháp thủ công là thiết yếu, nhưng quá trình này lại mất nhiều thời gian và công sức, và cũng dễ phát sinh lỗi do con người. Trong các hệ thống blockchain có liên kết chéo như Polkadot, việc bảo mật và kiểm thử càng trở nên phức tạp và thách thức [16]. Hiện nay, hệ sinh thái Ethereum đang phải đổi mới với nhiều lỗ hổng bảo mật nghiêm trọng như các cuộc tấn công lặp lại (Reentrancy) [16], vấn đề tràn số

(Overflow/Underflow) [17] và các vấn đề liên quan đến việc rút token không an toàn trong các hợp đồng thông minh [18]. Những vấn đề này đều là những rủi ro lớn đối với các ứng dụng phi tập trung (dApps) trên Ethereum. Do đó, cần có các giải pháp và công cụ kiểm thử bảo mật hiệu quả hơn để đáp ứng với tốc độ phát triển của các ứng dụng blockchain hiện đại.

Mạng chuỗi liên kết chéo (cross-chain) mở ra cơ hội phát triển các ứng dụng phi tập trung phức tạp hơn bằng cách kết nối nhiều blockchain với nhau. Tuy nhiên, điều này cũng tạo ra nhiều lỗ hổng bảo mật hơn, mà kẻ tấn công có thể khai thác để gây ra các vụ tấn công [19]. Một trong những cuộc tấn công đáng được chú ý đó là cuộc tấn công lặp lại (reentrancy), cho phép đối tượng tấn công lặp đi lặp lại lời gọi hàm của hợp đồng thông minh trước khi hoàn thành yêu cầu trước đó, dẫn đến các hậu quả khó lường [20]. Bên cạnh đó, các lỗ hổng tràn số và underflow cũng rất nguy hiểm, xảy ra khi giá trị vượt ngưỡng trên hoặc ngưỡng dưới cho phép, sẽ gây ra hậu quả khôn lường [21]. Đặc biệt, lỗ hổng rút tiền điện tử mà không được xác thực (unprotected ether withdrawal) cũng đang nhận được sự quan tâm, khi một hợp đồng thông minh không xác minh chính xác yêu cầu rút tiền, cho phép hacker rút Ether một cách bất hợp pháp [22]. Những hậu quả từ các lỗ hổng bảo mật nói trên đã và đang lan rộng trong hệ sinh thái Ethereum, gây thiệt hại tài chính lớn cho nhiều bên liên quan [12]. Tình hình nghiêm trọng hiện nay nhấn mạnh sự cần thiết của việc phát triển và tuân thủ nghiêm ngặt các chính sách và quy trình bảo mật trong quá trình tạo ra các ứng dụng blockchain. Điều này đòi hỏi các doanh nghiệp và tổ chức phải tập trung đầu tư vào các hoạt động kiểm tra, phát hiện lỗ hổng và cải tiến mã nguồn của sản phẩm trước khi chúng được triển khai. Bằng cách này, có thể giảm thiểu rủi ro và thiệt hại do các lỗi bảo mật trong ứng dụng blockchain gây ra.

Học máy, một lĩnh vực của trí tuệ nhân tạo, cho phép máy tính học hỏi và cải thiện từ kinh nghiệm mà không cần sự lập trình chi tiết. Sự tiến bộ trong học máy, đặc biệt là phát triển của học sâu, đã mở ra khả năng xử lý dữ liệu lớn, dự

đoán và ra quyết định trong nhiều lĩnh vực vượt trội so với con người [23, 24]. Trong lĩnh vực hợp đồng thông minh, các mô hình học máy có thể được sử dụng để phân tích mã và nhận diện các vấn đề như lỗi lặp, tấn công tràn số, và tấn công từ chối dịch vụ (DoS) [25]. Các mô hình này, được huấn luyện từ cả các ví dụ về hợp đồng thông minh dễ tổn thương và an toàn, có thể thực hiện xác minh hợp đồng một cách hiệu quả và tự động ở quy mô lớn.

Các nghiên cứu đã tập trung phát triển mô hình học máy có khả năng phát hiện tự động các đoạn mã chứa lỗi hoặc lỗ hổng bảo mật. Tôi đã phát triển một bộ dữ liệu mã nguồn Solidity, tập trung vào việc phát hiện các lỗ hổng tiềm ẩn. Sử dụng bộ dữ liệu này, tôi đã huấn luyện nhiều mô hình phân loại khác nhau như cây quyết định, rừng ngẫu nhiên, SVM, LSTM, CNN, để cải thiện khả năng phát hiện các đoạn mã dễ bị tấn công. Học sâu, một nhánh của học máy, sử dụng mạng nơ-ron nhân tạo sâu, có khả năng tự động trích xuất đặc trưng và mẫu từ dữ liệu phức tạp như hình ảnh, âm thanh, và ngôn ngữ tự nhiên. Việc áp dụng học sâu trong phân tích mã nguồn cũng mang lại kết quả tích cực, ví dụ như mô hình Roberta đã được chứng minh là hiệu quả trong việc phân loại và dự đoán lỗ hổng. Đây là một hướng nghiên cứu tiềm năng để tự động hóa quá trình kiểm tra và bảo mật mã nguồn.

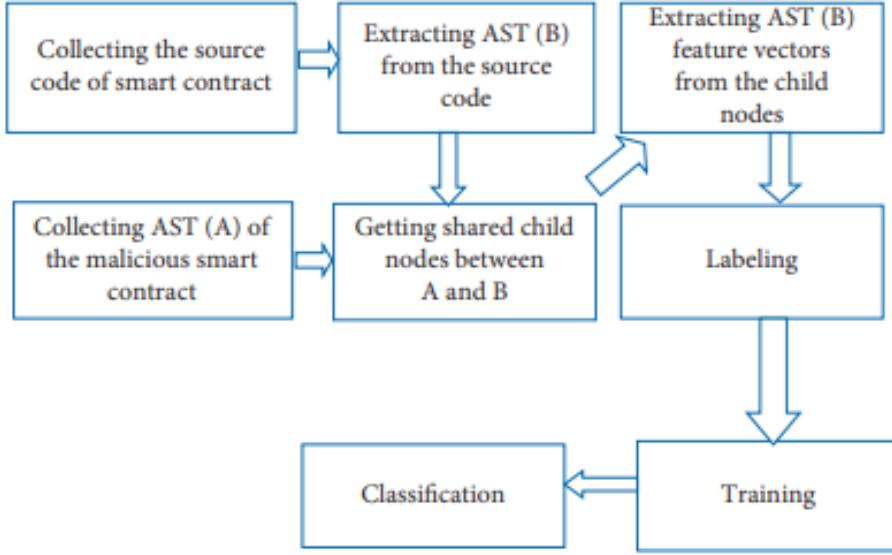
Để cải thiện bảo mật và độ tin cậy trong giao dịch chuyển tiền giữa các blockchain qua cầu nối sidechain, em đã tiến hành nghiên cứu và phát triển giải pháp ChainSniper. Phương pháp này kết hợp các chức năng ghi nhật ký sự kiện một cách cẩn thận, quét hợp đồng thông minh để phát hiện lỗ hổng tiềm ẩn và giám sát liên tục trong quá trình triển khai chúng. Để kiểm tra độ chính xác và hiệu quả của ChainSniper cũng như các mô hình học máy trong việc phát hiện lỗ hổng của hợp đồng, em đã tạo ra một bộ dữ liệu mới có tên là CrossChainSentinel. Bộ dữ liệu này chứa 300 hợp đồng thông minh, được chú thích và phân loại thủ công, trong đó 158 hợp đồng là lành tính và 142 hợp đồng còn lại chứa các lỗ hổng phổ biến như lỗi reentrancy, tràn số, underflow và các vấn đề liên quan đến rút tiền điện tử không an toàn. Tiếp theo, em đã huấn

luyện và so sánh nhiều mô hình học máy trên bộ dữ liệu CrossChainSentinel để đánh giá khả năng dự đoán của chúng, nhằm phát hiện hợp đồng độc hại trong môi trường chéo chuỗi. Kết quả thử nghiệm ban đầu đã cho thấy rằng các kỹ thuật học máy có thể được áp dụng hiệu quả trong việc kiểm toán và phân tích lỗ hổng của hợp đồng thông minh trong môi trường liên kết chéo của nhiều blockchain.

1.2. Các công trình nghiên cứu liên quan

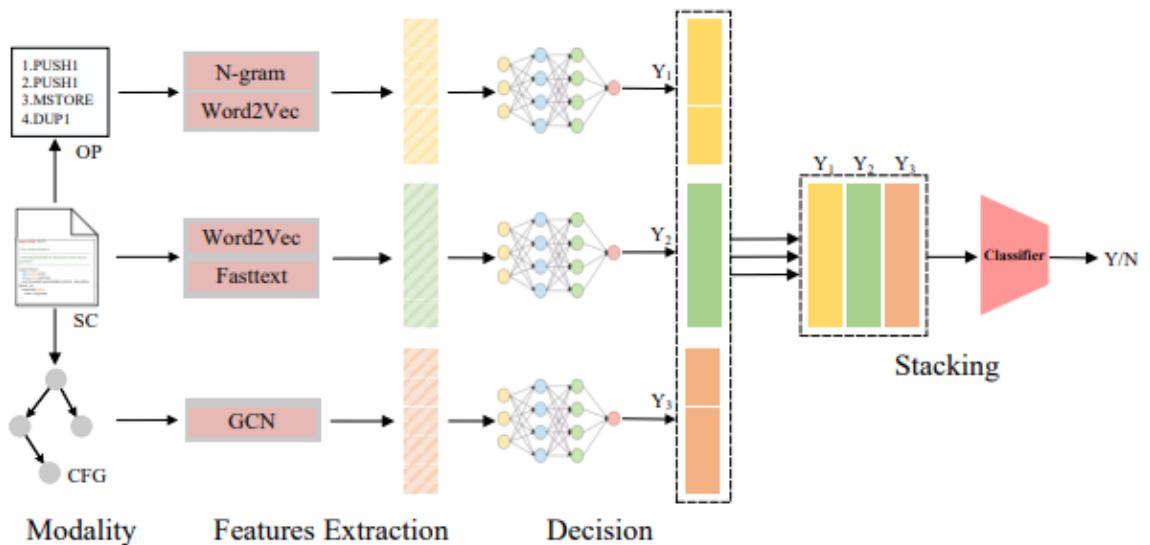
Ở thời điểm hiện tại, đã có một số đóng góp đáng kể đề cập đến vấn đề em đang nghiên cứu như Xu và nhóm nghiên cứu [26] tiến hành thực hiện một các tổng quan về việc triển khai ứng dụng các kỹ thuật học máy để phát hiện lỗ hổng trong hợp đồng thông minh (**Hình 1.1**). Họ sử dụng chung các nút con nhằm thực hiện quá trình phân tích và kết hợp mô hình k láng giềng gần nhất nhằm đẩy mạnh quá trình phân tích và cải thiện sự hiệu quả trong việc phát hiện lỗ hổng. Qua các quá trình tiến hành các thực nghiệm, công trình của Xu vượt trội hơn công cụ Oyente và SmartCheck về độ chính xác. Tuy nhiên, mô hình chủ yếu tập trung vào ngôn ngữ Solidity nên cần tinh chỉnh thêm để xác định chính xác các dòng code dễ bị lỗi.

Các nỗ lực ban đầu đã được ghi nhận trong bài nghiên cứu [27] đã tiến hành quá trình thực hiện công việc đánh giá bao quát về việc lấy ý kiến đa chiều (multivocal literature review - MLR) về những thách thức bảo mật và riêng tư trong tương tác giữa các blockchain. Các tác giả đã xác định một số lỗ hổng then chốt bao gồm: các cuộc tấn công loại wormhole tiến hành khai thác trên các lỗ hổng về an toàn thông tin để đánh cắp tài sản, thực hiện quá trình tấn công từ chối dịch vụ ngắt quãng làm hệ thống bị tê liệt, ngưng hoạt động, tấn công dựa trên thời gian thực hiện các giao dịch, sử dụng mật mã không tương thích giữa các blockchain dẫn đến mất tài sản trong quá trình chuyển đổi, và rò rỉ thông tin định danh trong các hợp đồng khóa thời gian băm mà các bên sử dụng để bảo mật giao dịch.



Hình 1.1: Công trình nghiên cứu của Xu và nhóm nghiên cứu

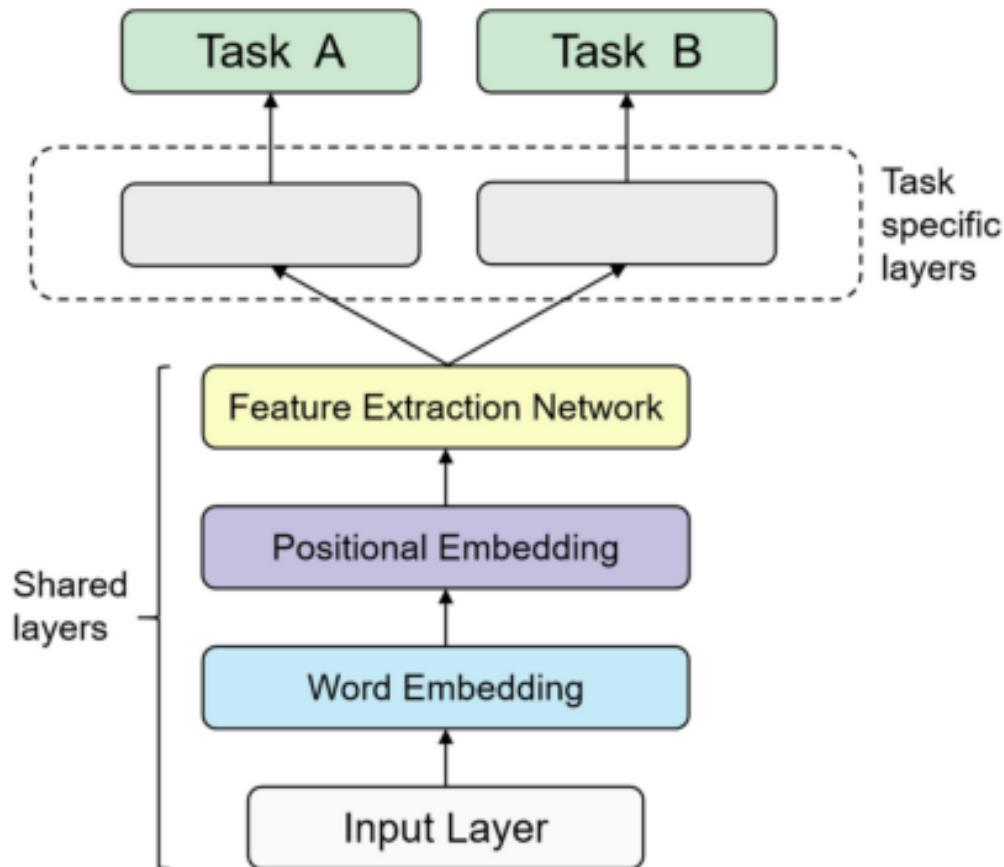
Mặt khác, Deng và nhóm nghiên cứu [28] đề xuất phương pháp mới phát hiện hợp đồng thông minh có lỗ hổng dựa trên học sâu và kết hợp nhiều chế độ quyết định (**Hình 1.2**). Họ trích xuất 5 đặc trưng khác nhau đại diện cho các hợp đồng và đạt độ chính xác cao nhờ quy trình tinh vi. Tuy nhiên, nghiên cứu bô qua học không giám sát dù nó có tiềm năng cho vấn đề phát hiện lỗ hổng.



Hình 1.2: Phương pháp đề xuất của Deng và nhóm nghiên cứu

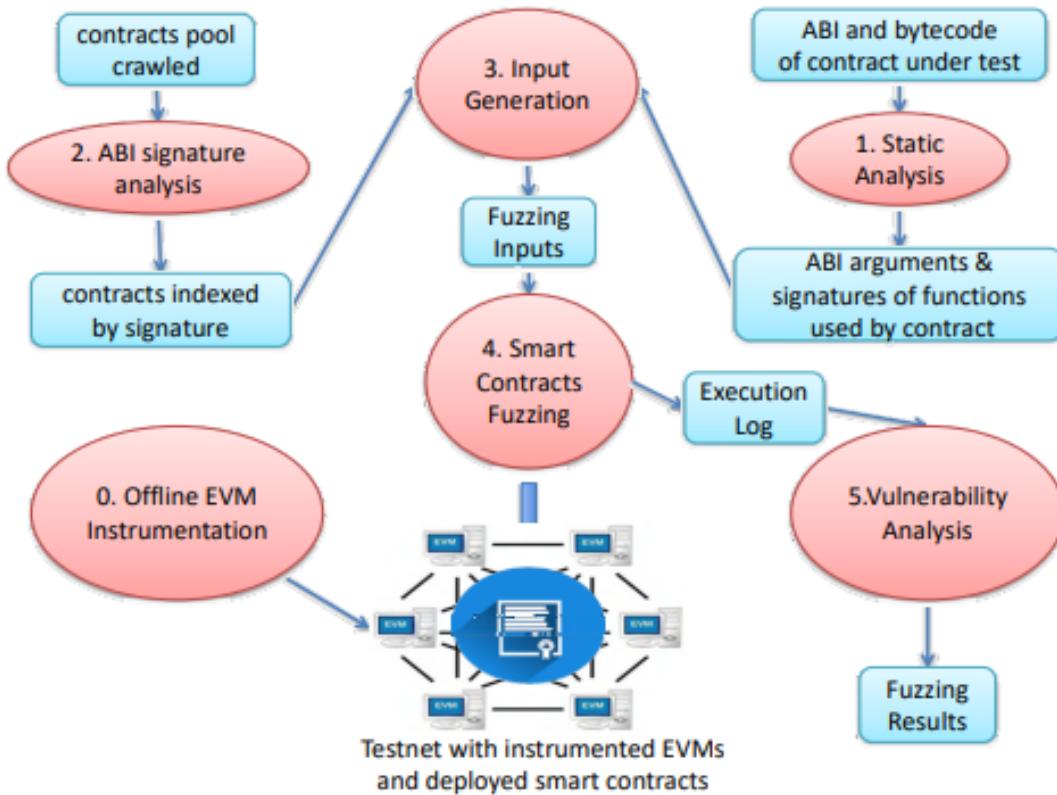
Trong một nghiên cứu tiên phong [29], He và cộng sự đã làm sáng tỏ phỗ hợp đồng thông minh có lỗ hổng và tiến hành công việc đưa ra các biện pháp khắc phục nhằm giảm thiểu những rủi ro này. Họ đã tóm tắt ngắn gọn và so sánh giữa các công cụ hiện có được tinh chỉnh để giải quyết các lỗ hổng đó.

Huang và nhóm nghiên cứu [30] đề xuất mô hình phát hiện lỗ hổng hợp đồng thông minh dựa trên học máy đa nhiệm (**Hình 1.3**). Cụ thể, mô hình bao gồm lớp dưới chia sẻ để trích xuất đặc trưng từ các lệnh, và các nhánh nhiệm vụ phụ trợ để phát hiện và nhận dạng lỗ hổng. Thông qua thiết lập nhiệm vụ nhận dạng bổ trợ, mô hình học các đặc trưng có hướng để nâng cao khả năng dự đoán. Kết quả thử nghiệm cho thấy phương pháp xác định hiệu quả các lỗi như toán học, lặp lại và các cuộc gọi không xác định. Nhìn chung, học máy đa nhiệm rất có tiềm năng trong việc tự động hóa phân tích bảo mật của các hợp đồng.



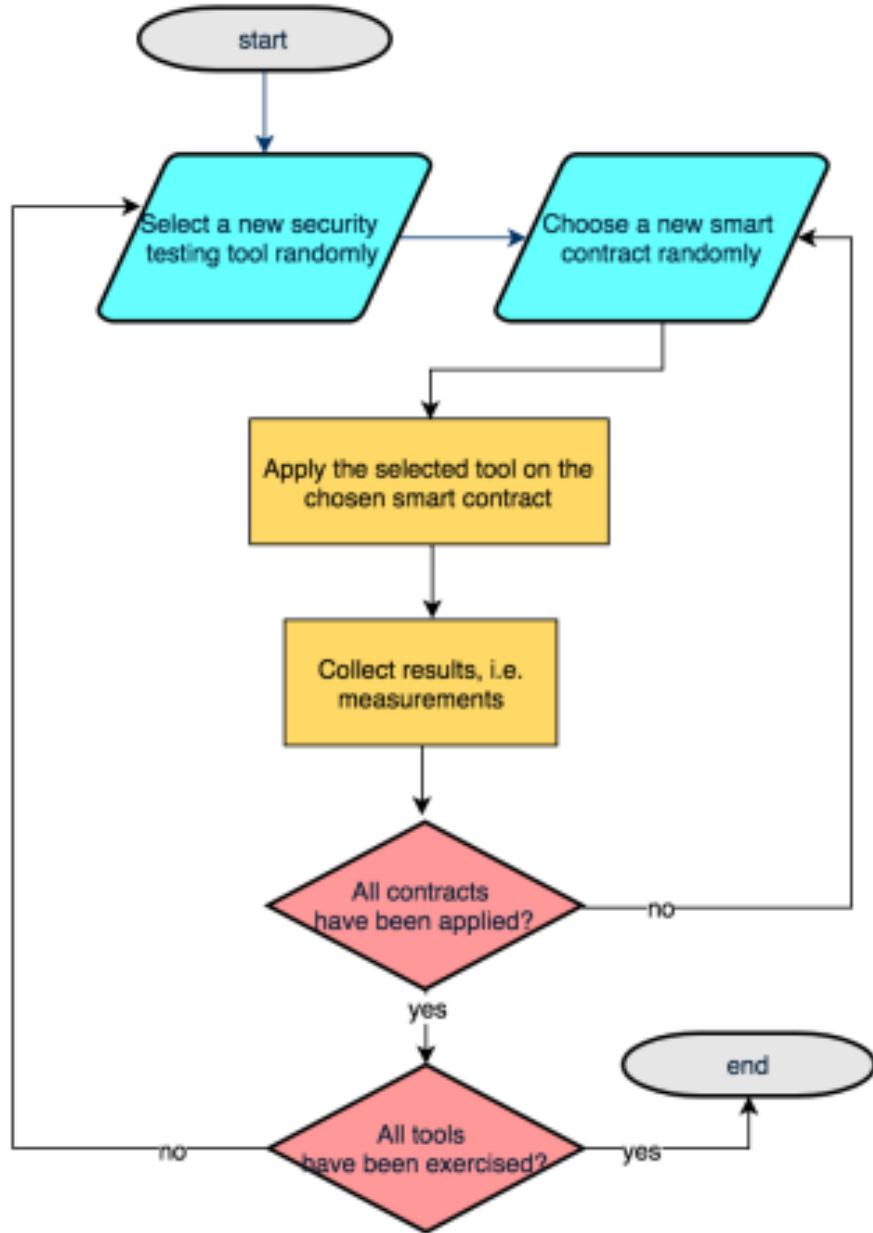
Hình 1.3: Hệ thống của Huang và nhóm nghiên cứu

Bên cạnh đó, Jiang và cộng sự [31] chứng minh độ hiệu quả của kỹ thuật fuzzing và giám sát thời gian chạy đối với việc phát hiện lỗ hổng hợp đồng (**Hình 1.4**). Công cụ ContractFuzzer của họ phân tích đặc tả ABI để sinh test case, xác định các biến kiểm soát để kiểm tra lỗ hổng, ghi nhật ký chi tiết EVM và giám sát thực thi để đánh dấu các lỗi tiềm ẩn. Thử nghiệm cho thấy công cụ phát hiện chính xác hơn 450 lỗ hổng, bao gồm cả lỗi DAO.



Hình 1.4: Tổng quan phương pháp của Jiang và nhóm nghiên cứu

Ngoài ra, Parizi và cộng sự [32] đã đóng góp đáng kể vào lĩnh vực này bằng cách giới thiệu một công cụ kiểm thử toàn diện được thiết kế để phát hiện lỗ hổng, minh họa kết quả của công cụ của họ, và cung cấp các so sánh sâu sắc với các đối tác hiện có (**Hình 1.5**). Tuy nhiên, các nghiên cứu này không áp dụng cho khái niệm chuỗi khôi liên kết ngang (cross-chain) với các lỗ hổng trong hợp đồng thông minh.



Hình 1.5: Công cụ phân tích của Parizi và nhóm nghiên cứu

1.3. Mục tiêu, đối tượng, và phạm vi nghiên cứu

1.3.1. Mục tiêu nghiên cứu

Đề tài này tập trung vào việc nghiên cứu và phát triển một cách thức dựa trên học máy để phát hiện lỗ hổng trong hợp đồng thông minh, đặc biệt trong môi trường của mạng liên chuỗi khối. Mục đích là nâng cao độ bảo mật và tin cậy của các hợp đồng thông minh, góp phần ngăn chặn và giảm bớt các rủi ro tiềm ẩn, qua đó bảo vệ lợi ích của những người tham gia.

1.3.2. Đối tượng nghiên cứu

Các đối tượng của nghiên cứu bao gồm:

- Công nghệ chuỗi khối
- Công nghệ liên chuỗi
- Bảo mật smart contract
- Công nghệ học máy

1.3.3. Phạm vi nghiên cứu

Phạm vi của đề tài bao gồm:

- Tìm hiểu về công nghệ mạng liên chuỗi khối và triển khai tấn công trên mạng liên chuỗi: Nắm vững cấu trúc, cơ chế hoạt động và khả năng ứng dụng của mạng liên chuỗi khối, đặc biệt là ở hợp đồng thông minh trên nền tảng mạng liên chuỗi và thực hiện tấn công trên mạng liên chuỗi khối.
- Xác định lỗ hổng và sự cần thiết phát hiện lỗ hổng: Định rõ các lỗ hổng phổ biến trên hợp đồng thông minh trên mạng liên chuỗi và lý do cần phải phát hiện chúng.

- Thiết kế một mô hình học máy cho việc phát hiện lỗ hổng: Tạo ra một mô hình học máy đặc biệt nhằm phát hiện ra các lỗ hổng trong hợp đồng thông minh trên các mạng chuỗi liên kết. Mô hình này sẽ áp dụng những kỹ thuật học máy tiên tiến và được tối ưu hóa để tăng cường hiệu suất và cải thiện độ chính xác.
- Đánh giá mô hình: Triển khai mô hình đã được phát triển để phát hiện lỗ hổng trên các hợp đồng thông minh thực tế trong mạng chuỗi liên kết, và tiến hành đánh giá hiệu quả làm việc của mô hình này.

1.3.4. Cấu trúc khoá luận tốt nghiệp

Cấu trúc Khoa học luận tốt nghiệp bao gồm:

- Chương 1: Tổng Quan Đề Tài: Mở đầu với cái nhìn toàn diện về đề tài cùng với việc xem xét các nghiên cứu liên quan đã được thực hiện.
- Chương 2: Cơ Sở Lý Thuyết: Trình bày cơ sở lý thuyết và các nền tảng quan trọng liên quan đến đề tài.
- Chương 3: Mô Hình Đề Xuất: Mô tả chi tiết về mô hình ChainSniper, sử dụng trong việc phát hiện lỗ hổng của hợp đồng thông minh trong mạng liên chuỗi.
- Chương 4: Thí Nghiệm và Đánh Giá: Trình bày và phân tích các kết quả thực nghiệm thu được từ mô hình.
- Chương 5: Kết Luận: Đưa ra kết luận từ nghiên cứu và bàn luận về các định hướng phát triển trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Trong phần này, em sẽ tập trung phân tích sâu hơn vào lĩnh vực nghiên cứu đã được thực hiện trước đây, bao gồm các hệ thống phân tán, hợp đồng thông minh và các lỗ hổng liên quan đến chúng. Em nghiên cứu và thảo luận về các phương pháp và kỹ thuật hiện có được áp dụng để phát hiện và giải quyết những lỗ hổng này.

2.1. Tổng quan về các hệ thống phân tán

2.1.1. Tổng quan về hệ thống mạng Blockchain - Chuỗi khối

Blockchain là một bước lớn trong cách mạng công nghệ, quá trình này đang thay đổi cách thức tương tác và trao đổi thông tin trong thế giới số. Xuất phát từ việc hỗ trợ các loại tiền điện tử như Ethereum, Bitcoin, Link, blockchain hiện nay đã vượt qua những giới hạn ban đầu chỉ là tiền mã hóa thành nền tảng then chốt cho nhiều ứng dụng khác. Về bản chất, blockchain được cấu thành từ một loạt các khối dữ liệu nối tiếp nhau một cách an toàn và phi tập trung. Nó tạo nên một hệ thống phân tán, minh bạch, bất biến và đồng thuận giữa các thành viên trong mạng.

So với các cơ sở dữ liệu truyền thống, blockchain có một số đặc điểm nổi bật:

- **An toàn, bảo mật:** Dữ liệu được tiến hành các thao tác mã hóa và xác thực bằng công nghệ mật mã, ngăn chặn nguy cơ giả mạo.
- **Minh bạch:** Mọi người tham gia đều có khả năng truy cập và kiểm chứng các giao dịch trên blockchain.
- **Phi tập trung:** Thông tin được bảo quản và điều hành trên một hệ thống máy tính phân phối, không tập trung vào một điểm chủ chốt. Điều này đảm

bảo sự vận hành không ngừng nghỉ và tránh lỗi do sự cố tại một điểm duy nhất.

- **Bất biến:** Dữ liệu sau khi ghi vào blockchain sẽ không thể thay đổi được nữa.

Nhờ những lợi ích nổi bật, blockchain đang được áp dụng rộng rãi trong nhiều ngành nghề, góp phần quan trọng vào sự phát triển của các công nghệ mới (**Hình 2.1**). Các chuyên gia trong lĩnh vực công nghệ dự báo rằng, trong tương lai, blockchain sẽ có sự biến đổi sâu rộng đối với nhiều phương diện trong đời sống của con người.



Hình 2.1: Tỷ trọng các lĩnh vực ứng dụng công nghệ Blockchain

Cơ sở lý thuyết của Blockchain dựa trên một vài nguyên lý cốt lõi:

- Đầu tiên, các khối dữ liệu (block) trên blockchain được nối liền với nhau thông qua việc sử dụng mã hash. Mã hash này giữ vai trò quan trọng trong việc đảm bảo sự nguyên vẹn của dữ liệu và liên kết logic giữa các khối. Các khối mới được thêm vào phía cuối của chuỗi theo thứ tự thời gian chính

xác, từ đó tạo ra một chuỗi lịch sử giao dịch không thể bị thay đổi hay giả mạo.

- Thứ hai, cơ chế đồng thuận giữa các nút trên mạng blockchain, với các phương pháp phổ biến như Proof of Work và Proof of Stake. Điều này bảo đảm sự đồng thuận và nhất quán về trạng thái và dữ liệu của chuỗi khối giữa tất cả các nút. Nó đóng vai trò quan trọng trong việc duy trì sự an toàn, minh bạch và phi tập trung của blockchain.
- Thứ ba, khả năng lập trình và thực thi hợp đồng thông minh (smart contracts) trên blockchain. Tính năng này giúp tự động hóa các quy trình giao dịch và thực hiện các điều khoản của hợp đồng mà không cần đến sự can thiệp của bên thứ ba.

Dựa trên những nguyên tắc và công nghệ chủ chốt này, blockchain có khả năng vận hành một cách độc lập như một hệ thống phân tán, đảm bảo an toàn, minh bạch và hiệu quả mà không cần sự quản lý của bất kỳ tổ chức trung ương nào. Đây là những yếu tố cốt lõi làm nên sự đột phá của công nghệ blockchain hiện đại.

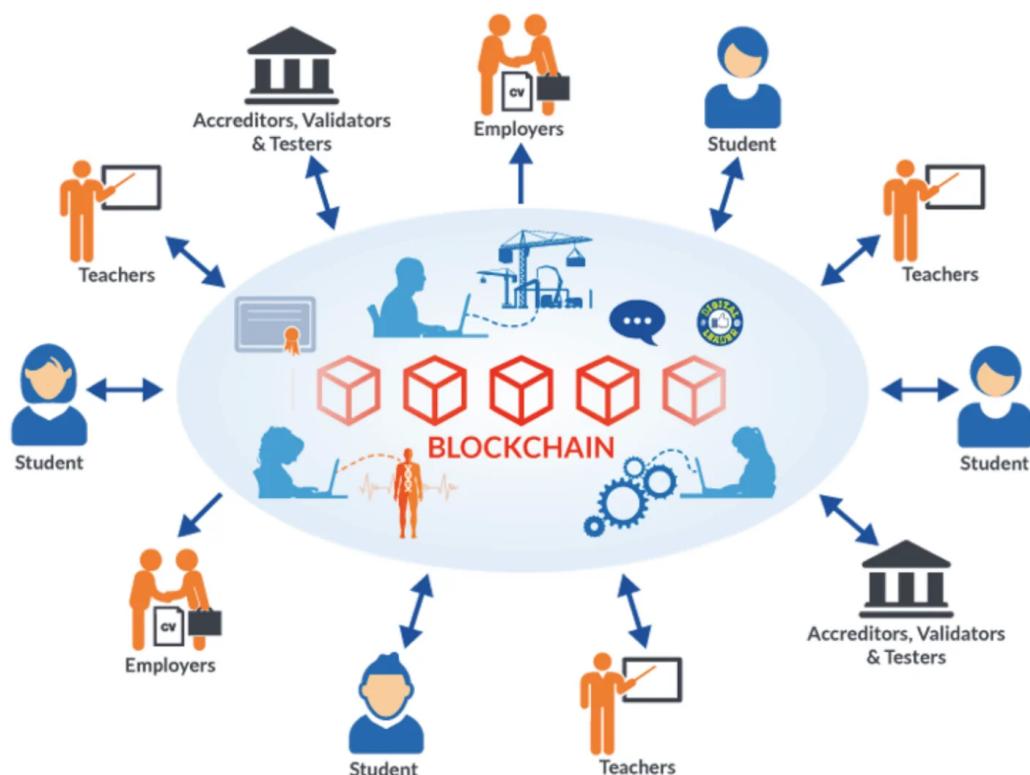
Blockchain không chỉ giới hạn ở việc hỗ trợ các loại tiền điện tử mà còn được áp dụng trong nhiều lĩnh vực khác của đời sống kinh tế và xã hội.

Trong ngành tài chính, blockchain giúp tăng tốc độ và giảm chi phí giao dịch, đồng thời cung cấp mức độ an toàn và minh bạch cao cho các hoạt động tài chính. Nó có khả năng được sử dụng trong việc xác minh danh tính, chuyển nhượng tài sản, và thực hiện các hợp đồng tài chính mà không cần bên ngoài can thiệp vào, giúp tiết kiệm thời gian và chi phí cũng như đảm bảo sự đồng bộ và chính xác.

Trong lĩnh vực chuỗi cung ứng và logistics, blockchain giúp doanh nghiệp theo dõi chính xác nguồn gốc và xuất xứ của sản phẩm từ khâu sản xuất đến khi giao đến tay người tiêu dùng. Điều này tăng cường minh bạch, giúp nhanh chóng phát hiện gian lận thương mại và đảm bảo an toàn thực phẩm. Tính không thể

thay đổi của blockchain cũng giúp ngăn chặn làm giả hàng hóa và đánh cắp dữ liệu trong quá trình thực hiện công việc vận chuyển giữa các điểm.

Trong ngành y tế, blockchain cho phép an toàn lưu trữ và chia sẻ hồ sơ, dữ liệu bệnh án giữa các bệnh viện, phòng mạch và bác sĩ, ngăn chặn việc bệnh nhân cần phải nhập lại thông tin nhiều lần. Các nghiên cứu y sinh học cũng có thể được chia sẻ an toàn, nhanh chóng và tiện lợi hơn qua blockchain.



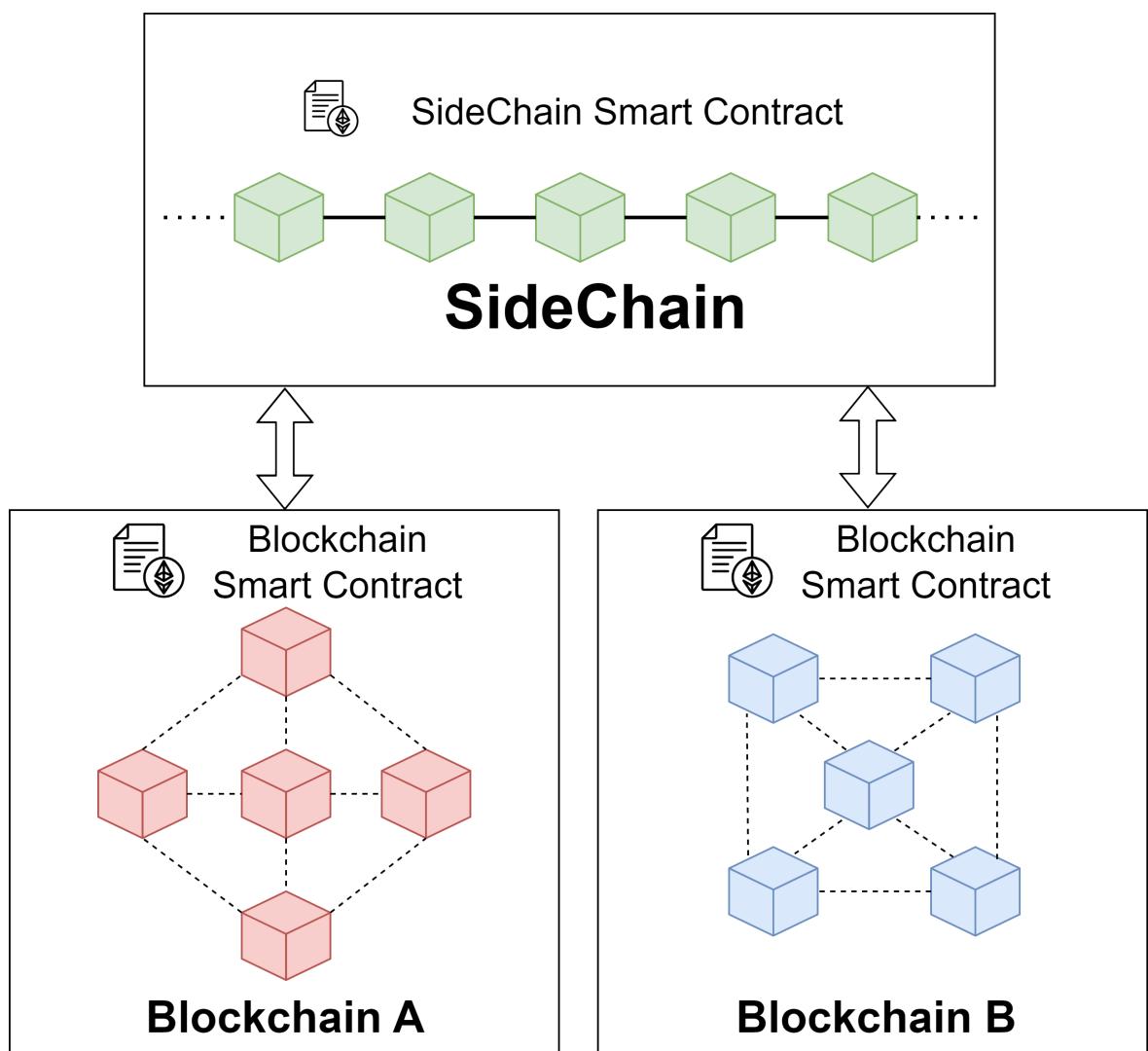
Hình 2.2: Các ứng dụng khi sử dụng nền tảng blockchain

Do đó, nhờ vào khả năng phân quyền và xác thực thông tin mà không cần tập trung, blockchain đang mở ra những cơ hội rộng lớn cho tương lai, nơi thông tin có thể được bảo quản, chia sẻ và truy cập một cách an toàn, minh bạch và độc lập với bên trung gian. Điều này cho thấy vai trò và ảnh hưởng của blockchain vượt xa khỏi lĩnh vực tiền điện tử, mở ra nhiều khả năng mới và tiềm năng phong phú.

2.1.2. Tổng quan về hệ thống mạng liên chuỗi khối

Mạng liên chuỗi (cross-chain) là một công nghệ blockchain nâng cao, cho phép xây dựng các giao thức để kết nối và tương tác giữa nhiều chuỗi khối (blockchain) khác nhau (**Hình 2.3**). Các chuỗi khối này có thể dựa trên những công nghệ khác nhau, do nhiều tổ chức phát triển độc lập.

Mạng liên chuỗi ra đời để cải thiện hiện trạng cô lập giữa các blockchain/chuỗi khối, và xây dựng một hệ sinh thái mở, tự do trao đổi dữ liệu, tài sản và ứng dụng. Điều này sẽ thúc đẩy đổi mới và mở rộng tiềm năng của blockchain.



Hình 2.3: Giao tiếp liên chuỗi giữa hai chuỗi khối thông qua sidechain

Các đặc điểm chính của mạng liên chuỗi:

- **Tính tương thích:** Các blockchain có thể giao tiếp, trao đổi dữ liệu hiệu quả với nhau qua mạng liên chuỗi. Yêu cầu cách biểu diễn tính nhất quán về mặt dữ liệu giữa các blockchain.
- **Tính an toàn và bảo mật:** Áp dụng các biện pháp bảo mật chặt chẽ để bảo vệ an toàn trong quá trình giao tiếp dữ liệu giữa các chuỗi khôi. Các phương pháp này thường bao gồm việc sử dụng các thuật toán mã hóa nâng cao và ký số điện tử tiên tiến.
- **Tính sở hữu đa chuỗi:** Người dùng có khả năng chuyển và quản lý tài sản của mình một cách dễ dàng giữa các chuỗi khôi khác nhau, điều này tăng cường sự linh hoạt và hiệu quả trong việc sử dụng tài sản trên nhiều nền tảng blockchain.

2.2. Tổng quan về hợp đồng thông minh

2.2.1. Hợp đồng thông minh

Hợp đồng thông minh, còn được biết đến như hợp đồng điện tử thông minh, là loại thỏa thuận điện tử dựa trên công nghệ blockchain. Điểm đặc biệt của hợp đồng này là khả năng tự động thực hiện các điều khoản đã được lập trình trước mà không yêu cầu can thiệp từ bất kỳ bên thứ ba nào.

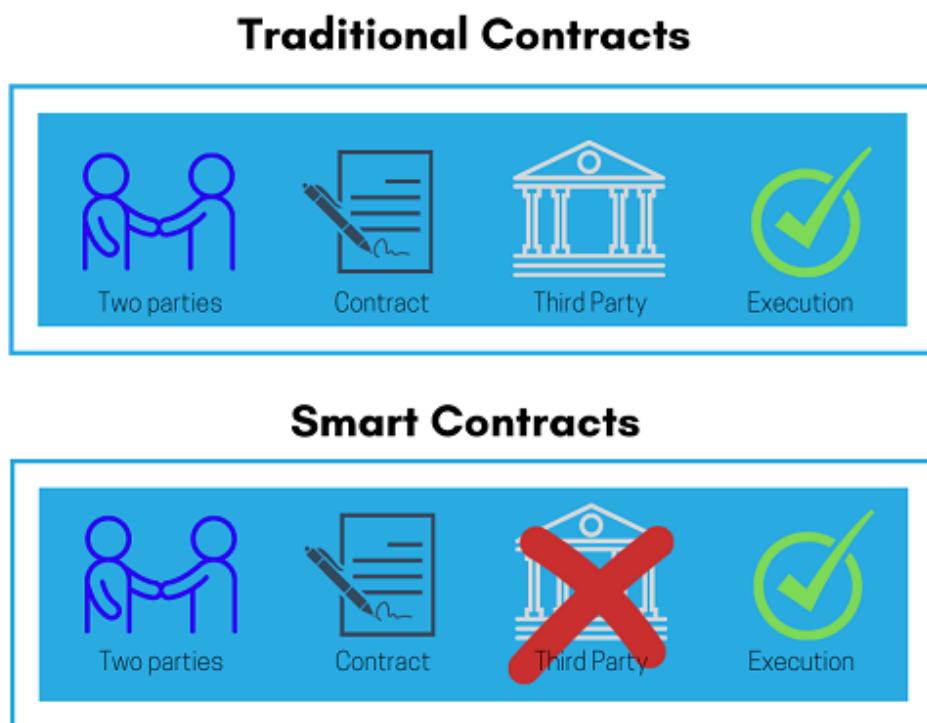
Lợi ích chính của hợp đồng thông minh nằm ở tính minh bạch và khả năng quản lý chi phí và thời gian thực hiện hợp đồng một cách hiệu quả, vượt trội hơn so với các hợp đồng truyền thống (**Hình 2.4**). Các điều khoản trong hợp đồng được chuyển đổi thành mã lập trình không thể sửa đổi và được lưu trữ một cách công khai trên mạng lưới phân tán. Điều này giảm thiểu rủi ro về tranh chấp và giả mạo thông tin khi thực hiện hợp đồng.

Các tính chất đặc trưng của hợp đồng thông minh gồm:

- Hợp đồng thông minh có khả năng tự động thực thi các điều khoản đã lập trình sẵn mà không cần sự can thiệp của bất kỳ bên thứ ba nào. Các quy định và điều kiện trong hợp đồng được biểu diễn rõ ràng qua ngôn ngữ lập trình. Khi các điều kiện này được đáp ứng, hợp đồng sẽ tự kích hoạt mà không cần sự tác động từ bên ngoài.
- Hợp đồng thông minh còn được đặc trưng bởi việc chúng được sao lưu và phân bổ trên nhiều máy tính trong mạng blockchain. Điều này giúp tăng cường bảo mật và đảm bảo sự toàn vẹn của hợp đồng, ngăn chặn việc sửa đổi hoặc xoá bỏ không phép.
- Một tính năng khác là tính bất biến: một khi hợp đồng đã được tải lên blockchain, nội dung của nó không thể bị thay đổi. Tính năng này rất quan trọng trong việc bảo vệ sự an toàn và tin cậy của các giao dịch.
- Hợp đồng thông minh cũng được đánh giá cao về sự đáng tin cậy. Chúng hoạt động theo đúng cách thiết kế mà không bị tác động từ bên ngoài, đảm bảo mọi thứ diễn ra chính xác và hạn chế rủi ro sai sót. Hơn nữa, cơ chế đồng thuận của blockchain cũng góp phần vào việc duy trì độ tin cậy này.
- Tính minh bạch cũng là một ưu điểm quan trọng: mã nguồn của hợp đồng thông minh có thể được truy cập và xem xét bởi bất kỳ ai, kể cả những người không tham gia trực tiếp vào hợp đồng.
- Cuối cùng, hợp đồng thông minh giúp tiết kiệm thời gian và chi phí cho các bên liên quan. Chúng tự hoạt động mà không cần sự hỗ trợ từ đối tác hoặc người công chứng.

Tuy nhiên, hợp đồng thông minh vẫn chưa thể hoàn toàn thay thế hợp đồng truyền thống trong mọi tình huống (**Hình 2.4**). Không phải tất cả các loại hợp đồng hiện có đều thích hợp để chuyển đổi sang dạng hợp đồng thông minh. Việc sử dụng chúng phụ thuộc vào tính chất và phạm vi của từng thỏa thuận cụ thể. Hơn nữa, các quy định pháp lý hiện tại và bối cảnh pháp lý ở mỗi quốc gia và

khu vực cũng ảnh hưởng đến việc triển khai công nghệ hợp đồng thông minh. Do đó, hợp đồng thông minh còn tồn tại một số hạn chế và không thể áp dụng một cách toàn diện trong mọi hoàn cảnh. Sự phù hợp của hợp đồng thông minh cần được xem xét kỹ lưỡng dựa trên đặc thù và phạm vi của từng loại hợp đồng, cũng như dựa trên các quy định pháp lý ở mỗi quốc gia.



Hình 2.4: Sự khác nhau giữa hợp đồng truyền thống và hợp đồng thông minh

2.2.2. Ngôn ngữ solidity

2.2.2.1. Source code

Solidity là một ngôn ngữ lập trình cấp cao, được tạo ra đặc biệt cho việc xây dựng hợp đồng thông minh trên nền tảng Ethereum. Đây là ngôn ngữ có kiểu dữ liệu tinh, với cú pháp tương đồng với các ngôn ngữ như JavaScript, C++ và Python. Nhờ vào điều này, các lập trình viên có thể dễ dàng học và làm việc với Solidity.

Trong Solidity, mọi thứ đều được mã hóa dưới dạng hợp đồng và các hàm (function). Các hàm này chứa các điều kiện, logic và các bước thực thi mà hợp đồng cần thực hiện. Lập trình viên sẽ sử dụng solidity để khai báo các biến, hàm và xây dựng tương tác giữa chúng để mô phỏng bất kỳ quy tắc nghiệp vụ nào. Từ đó, hợp đồng thông minh có thể tự động được thực thi.

Một trong những lợi thế đáng chú ý của Solidity là nó cho phép phát triển hợp đồng thông minh theo mô hình OOP (lập trình hướng đối tượng). Điều này giúp chương trình dễ cấu hình, bảo trì và nâng cấp. Hơn nữa, Solidity còn hỗ trợ thư viện và giao thức có khả năng tích hợp thêm các tiện ích đa dạng từ nhiều nguồn khác nhau giúp cho phát triển blockchain như quản lý danh tính, gửi/nhận token, giao dịch...

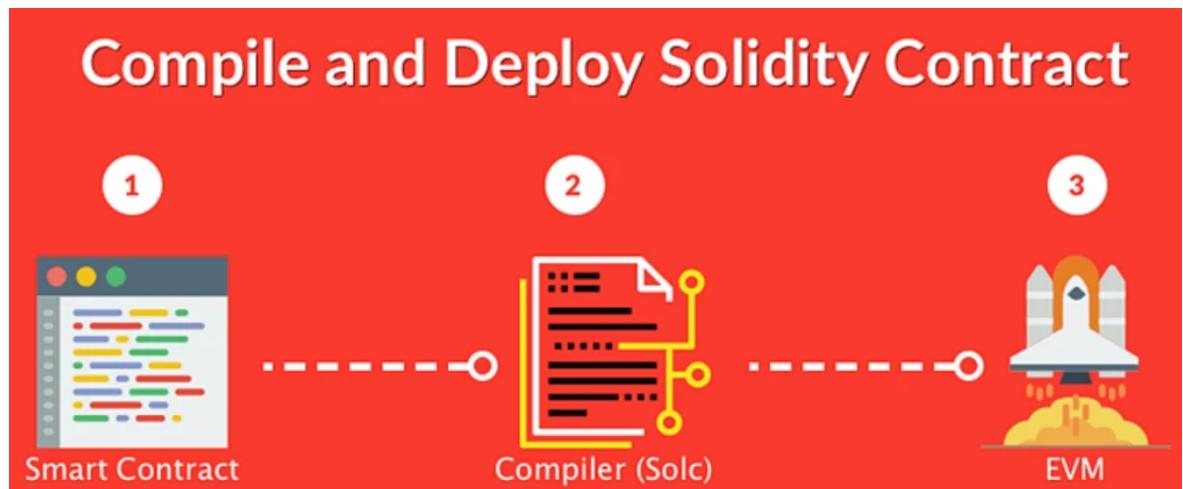
Do đó, Solidity hiện được coi là công cụ lý tưởng để tạo ra các ứng dụng blockchain và hợp đồng thông minh trên Ethereum. Ngôn ngữ này cung cấp một sự cân bằng giữa sự đơn giản và dễ dàng sử dụng, đồng thời vẫn duy trì được khả năng linh hoạt và hiệu suất cao trong quá trình phát triển.

Cấu trúc của một hợp đồng thông minh:

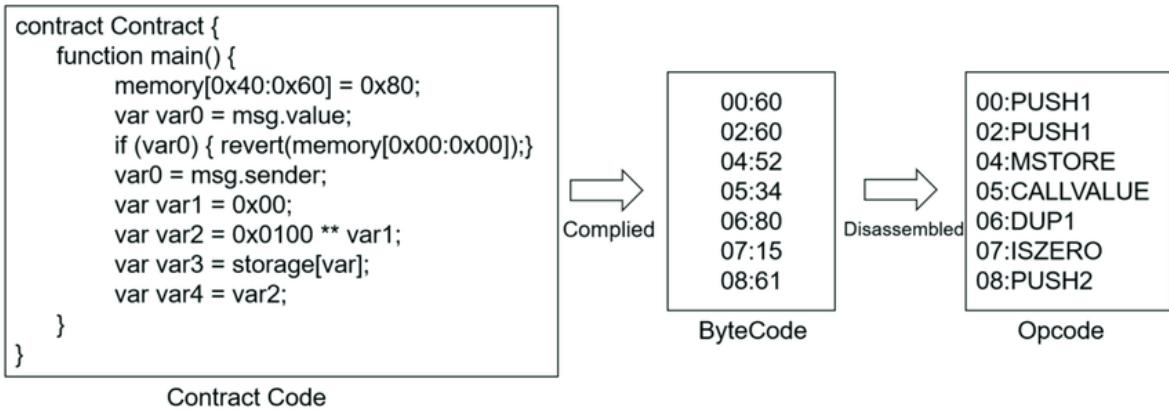
- **Pragma directive:** Là dòng khai báo phiên bản trình biên dịch Solidity mà source code sẽ được viết trên đó. Việc khai báo này giúp đồng bộ phiên bản giữa code và compiler, tránh rủi ro mất tương thích.
- **Contract:** Đây là khái niệm cơ bản nhất trong Solidity, tương đương với lớp (class) trong OOP. Contract sẽ chứa toàn bộ dữ liệu và hàm thành viên cần thiết để cấu thành nên hợp đồng thông minh.
- **State Variables:** Đây là các biến được xác định bên trong contract, lưu trữ trạng thái và dữ liệu của hợp đồng. Chúng được lưu trữ vĩnh viễn trên blockchain.
- **Functions:** Các hàm là nơi định nghĩa chức năng và nghiệp vụ của hợp đồng. Chúng gồm các câu lệnh, điều kiện và logic cần thiết để thực thi.

- **Events:** Sự kiện trong hợp đồng thông minh được kích hoạt khi một hành động cụ thể nào đó xảy ra, thực hiện ghi log và xác nhận các hoạt động.

Thêm vào đó, sự xuất hiện của các công cụ biên dịch hợp đồng thông minh nhằm làm cho quá trình phát triển hợp đồng thông minh trở nên đơn giản hơn. Chúng cho phép người dùng viết hợp đồng bằng các ngôn ngữ phổ biến hơn như Python, sau đó tự động biên dịch ra các ngôn ngữ chuyên dụng như Solidity. Do đó, quá trình tạo lập và phát triển hợp đồng thông minh trở nên thuận tiện và dễ dàng hơn, mở rộng khả năng tiếp cận cho nhiều lập trình viên. Điều này giúp mở rộng khả năng ứng dụng của công nghệ blockchain và hợp đồng thông minh. Quá trình biên dịch smart contract từ ngôn ngữ lập trình sang bytecode được thực hiện bởi trình biên dịch như solc cho Solidity. Bytecode sau đó được triển khai lên blockchain thông qua các giao dịch. Khi smart contract được thực thi, bytecode sẽ được biên dịch tiếp thành opcode - đây là ngôn ngữ máy ở mức thấp hơn mà EVM đọc được và tiến hành quá trình thực thi trực tiếp. Việc biên dịch này giúp tối ưu hóa hiệu suất thực thi smart contract, vì opcode có kích thước nhỏ hơn và EVM chỉ cần đọc và thực thi các lệnh đơn giản (**Hình 2.5** và **Hình 2.6**).



Hình 2.5: Quá trình biên dịch và thực thi hợp đồng thông minh



Hình 2.6: Sự tương quan của source code, bytecode và opcode của hợp đồng thông minh

2.2.2.2. Bytecode

Bytecode là mã của Máy ảo Ethereum (EVM) được tạo ra từ việc biên dịch hợp đồng thông minh, khi chúng được viết bằng ngôn ngữ lập trình Solidity (**Hình 2.7**). Bytecode cho phép chuyển đổi mã nguồn Solidity thành các câu lệnh máy có thể đọc và thực thi được bởi Ethereum Virtual Machine. Đây là bước trung gian cần thiết để triển khai hợp đồng lên blockchain Ethereum. Các biến, cấu trúc dữ liệu, câu lệnh điều khiển và các hàm được khai báo bằng ngôn ngữ Solidity cấp cao sẽ được biên dịch xuống các kiểu dữ liệu và câu lệnh cơ bản hơn trong bytecode. Ví dụ biến kiểu string trong Solidity sẽ trở thành một mảng các ký tự byte. Biến bool được biểu diễn bằng giá trị 0 hoặc 1.

Trong quá trình biên dịch mã nguồn Solidity, trình biên dịch sẽ chuyển đổi các hợp đồng thông minh thành một chuỗi lệnh máy, được biết đến là opcode. Mỗi opcode tương ứng với một hành động, thao tác hoặc lệnh cụ thể mà EVM có thể hiểu và thực thi được.

Hình 2.7: Cấu trúc của bytecode trong hợp đồng thông minh

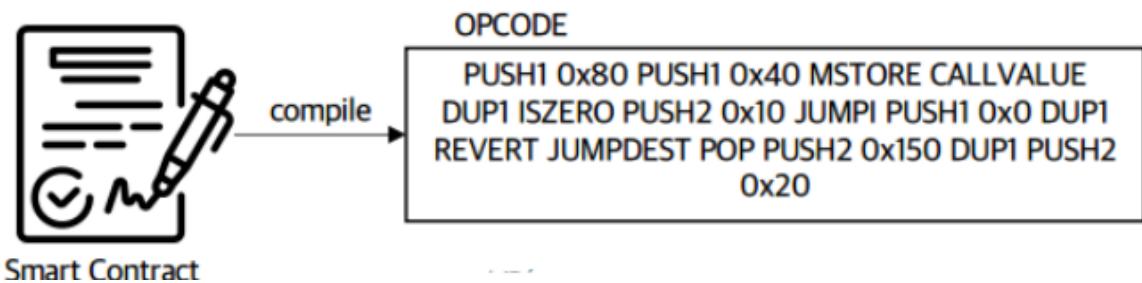
Một số opcode phổ biến bao gồm các lệnh toán học ADD, MUL, dùng cho phép cộng, nhân; lệnh SSTORE ghi dữ liệu vào bộ nhớ, SLOAD đọc dữ liệu; JUMP dùng để nhảy tới vị trí khác trong chương trình. Tập hợp các opcode này tạo thành bytecode cho hợp đồng.

Sau khi hoàn thành quá trình biên dịch, hợp đồng thông minh Solidity sẽ được triển khai lên blockchain Ethereum dưới hình thức bytecode. Đây chính là các câu lệnh mà EVM có thể đọc và thực thi khi có giao dịch gọi tới hợp đồng. Bytecode có chức năng chuyển đổi mã Solidity cấp cao thành ngôn ngữ máy, cho phép nó có thể được thực thi trên nền tảng Ethereum. Nó là cầu nối để code do người viết có thể vận hành trên môi trường blockchain. Khi triển khai, bytecode sẽ được lưu lại bên trong các khối của Ethereum.

2.2.2.3. *Opcode*

Opcode là các lệnh máy ảo được sử dụng trong smart contract để thực thi logic của contract. Các opcode này tương tự như các lệnh assembly trong lập trình

máy tính. Mỗi opcode đại diện cho một hành động cụ thể mà smart contract có thể thực hiện, chẳng hạn như ghi dữ liệu, đọc dữ liệu, thực hiện phép toán, nhảy đến địa chỉ opcode tiếp theo. Các opcode cho phép smart contract thao tác với bộ nhớ, thực thi logic và giao tiếp với blockchain. Chúng được thiết kế để thực thi an toàn, tiện lợi và đáng tin cậy. Các lập trình viên smart contract sẽ sử dụng các opcode thông qua ngôn ngữ lập trình smart contract để viết logic cho contract. Sau đó mã nguồn sẽ được biên dịch thành các opcode tương ứng để máy ảo Ethereum thực thi.



Hình 2.8: Cấu trúc của opcode trong hợp đồng thông minh

Các loại opcode thông dụng:

- Các opcode thao tác dữ liệu: SLOAD, SSTORE, MLOAD, MSTORE
- Các opcode tính toán: ADD, MUL, SUB, DIV, EXP, MOD
- Các opcode logic: NOT, AND, OR, XOR, LT, GT
- Các opcode nhảy: JUMP, JUMPI

Chức năng của opcode thực hiện các thao tác dữ liệu:

- SLOAD: Đọc giá trị từ một vị trí bộ nhớ lưu trữ (storage) của hợp đồng thông minh. SLOAD sẽ truy xuất giá trị tại một địa chỉ bộ nhớ lưu trữ cụ thể và đẩy giá trị đó lên stack.
- SSTORE: Lưu trữ một giá trị vào một vị trí bộ nhớ lưu trữ của hợp đồng thông minh. Lệnh SSTORE trong Ethereum nhận hai tham số: tham số

đầu tiên xác định địa chỉ trong bộ nhớ lưu trữ, và tham số thứ hai là giá trị cần được lưu vào địa chỉ đó.

- MLOAD: Tải một word (32 byte) từ bộ nhớ (memory) của EVM (Ethereum Virtual Machine) và đẩy lên stack.
- MSTORE: Lệnh này dùng để lưu trữ một word từ stack vào bộ nhớ. Nó yêu cầu hai tham số: tham số thứ nhất xác định vị trí trong bộ nhớ cần lưu trữ, và tham số thứ hai là giá trị cần được lưu vào vị trí đó.

Như vậy, SLOAD và SSTORE làm việc với bộ nhớ lưu trữ lâu dài, trong khi MLOAD và MSTORE làm việc với bộ nhớ tạm thời của EVM. Chúng được sử dụng để thực hiện việc đọc và ghi dữ liệu giữa các vùng nhớ khác nhau.

Chức năng của opcode thực hiện các thao tác về các phép tính toán số học:

- ADD: Phép cộng hai số. Lấy hai giá trị trên đầu stack, cộng lại và đẩy kết quả lên stack.
- MUL: Phép nhân hai số. Tương tự ADD, lấy hai giá trị đầu stack nhân với nhau và đẩy kết quả lên.
- SUB: Phép trừ hai số. Lấy giá trị thứ 2 trên stack trừ đi giá trị đầu stack.
- DIV: Chia hai số nguyên. Tương tự SUB, chia số thứ 2 cho số đầu tiên.
- EXP: Lũy thừa - Giá trị thứ 2 mũ giá trị đầu tiên.
- MOD: Chia lấy dư - Tính số dư của phép chia giá trị đầu cho giá trị thứ 2.

Các opcode này hỗ trợ việc thực hiện các phép toán cơ bản, đồng thời hỗ trợ xử lý số liệu và tính toán trong hợp đồng thông minh. Chúng lấy các giá trị input từ stack, thực hiện phép toán và đẩy kết quả lên stack.

JUMP và JUMPI cho phép điều khiển luồng code một cách linh hoạt hơn, tạo ra các cấu trúc rẽ nhánh và lặp trong smart contract. Chức năng của opcode nhảy:

- JUMP: Nhảy đến một địa chỉ tuyệt đối trong code. Nó nhận vào một tham số duy nhất là địa chỉ cần nhảy tới. Sau khi thực thi JUMP, con trỏ lệnh sẽ chuyển đến địa chỉ được chỉ định và tiếp tục thực thi.
- JUMPI: Nhảy có điều kiện dựa trên kết quả kiểm tra logic. JUMPI nhận vào 2 tham số: Tham số 1: Địa chỉ cần nhảy tới nếu điều kiện khớp (tương tự như trong JUMP). Tham số 2: Giá trị điều kiện (khác 0 là True). Nếu giá trị điều kiện là True (khác 0), JUMPI sẽ chuyển con trỏ đến địa chỉ nhảy. Ngược lại con trỏ sẽ tiếp tục xuống dòng tiếp theo.

2.2.3. Các loại lỗ hổng trong hợp đồng

2.2.3.1. Reentrancy

Lỗ hổng reentrancy xảy ra do cách thức hoạt động của Ethereum Virtual Machine (EVM). Cụ thể, mỗi khi một hàm smart contract được gọi, EVM sẽ tạo ra một stack frame mới. Tại đây, các biến local sẽ được sao chép giá trị từ state của blockchain. Sau khi hàm kết thúc, giá trị cập nhật của các biến local mới được commit vào blockchain state.

Chính vì thế, nếu trong quá trình thực thi, hàm smart contract gọi tới một hàm khác trước khi kết thúc (ví dụ gọi đến một hàm bên ngoài hay hàm fallback()), điều này cho phép mã độc hại được thực thi lại trên cùng một stack frame. Hacker có thể lợi dụng điều này để gọi đệ quy hàm bị lỗi, rút lui tiền nhiều lần trước khi trạng thái thực sự được cập nhật. **Hình 2.9** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract ReentrancyExample {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "I_balance");

        // Malicious reentrancy attack
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Reentrancy attack failed");

        balances[msg.sender] -= amount;
    }
}

```

Hình 2.9: Mô phỏng lỗ hổng Reentrancy

Điển hình như trường hợp hàm rút tiền ETH trong ví smart contract bị lỗi. Hacker có thể liên tục gọi lại hàm này để "đánh lừa" contract rằng số dư vẫn chưa thay đổi, cho phép rút hết số ETH trong ví. Điều này tạo ra lỗ hổng reentrancy, được xem là lỗ hổng bảo mật nghiêm trọng nhất, có thể gây ra những tổn thất lớn.

Các contract thông minh cần cẩn trọng trong việc thiết kế các hàm dễ bị lạm dụng, đặc biệt là những hàm liên quan tới tiền hay quyền sở hữu tài sản.

2.2.3.2. Integer Overflow/Underflow

Integer Overflow/Underflow xảy ra khi một biến số nguyên vượt quá giới hạn lưu trữ của kiểu dữ liệu. Cụ thể, đối với kiểu uint256 trong Solidity chỉ có thể lưu trữ các số nguyên từ 0 đến $2^{256}-1$.

Trong các phép tính số học như cộng, trừ, nhân, chia, nếu kết quả vượt qua giới hạn số lớn nhất có thể lưu trữ, sự cố overflow sẽ xảy ra. Ngược lại, nếu kết

quả thấp hơn giới hạn số nhỏ nhất, tức là dưới 0, thì sự cố underflow sẽ xuất hiện. Trong smart contract, hacker có thể cố tình đưa vào các giá trị input khiến overflow/underflow xảy ra, dẫn đến kết quả sai và làm lợi cho chúng.

Chẳng hạn trong hàm rút tiền, hacker có thể gửi vào một số âm rất lớn, khiến biến cộng với số âm này bị underflow và đảo dấu thành số dương rất lớn. Điều này cho phép chúng rút nhiều tiền hơn số dư thực tế. **Hình 2.10** và **Hình 2.11** đã mô phỏng lại cuộc tấn công trong smart contract.

```
pragma solidity ^0.8.0;

contract IntegerOverflowUnderflowExample {
    mapping(address => uint256) public balances;

    function deposit(uint256 amount) public {
        balances[msg.sender] += amount;
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance");

        // Integer underflow vulnerability
        balances[msg.sender] -= amount;

        // Malicious integer overflow attack
        uint256 maliciousAmount = type(uint256).max - balances[msg.sender] + 1;
        balances[msg.sender] += maliciousAmount;
    }
}
```

Hình 2.10: Mô phỏng lỗ hổng Integer Overflow

```
pragma solidity ^0.8.0;

contract IntegerUnderflowExample {
    mapping(address => uint256) public balances;

    function deposit(uint256 amount) public {
        balances[msg.sender] += amount;
    }

    function withdraw(uint256 amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance");

        // Integer underflow vulnerability
        balances[msg.sender] -= amount;

        // Transfer the amount to the user
        payable(msg.sender).transfer(amount);
    }
}
```

Hình 2.11: Mô phỏng lỗ hổng Integer Underflow

2.2.3.3. Unprotected Ether Withdrawal

Unprotected Ether Withdrawal xảy ra khi smart contract không giới hạn quyền rút Ether của người dùng. Điều này có nghĩa là bất kỳ người dùng nào cũng có thể gọi hàm để rút Ether mà không cần trải qua quá trình xác thực.

Trong trường hợp contract lưu trữ số lượng lớn Ether, hacker có thể dễ dàng gọi hàm rút tiền và chuyển tất cả về ví của mình. Đây là một lỗ hổng bảo mật nghiêm trọng, vì nó tạo ra mối đe dọa trực tiếp đến tài sản cá của chủ sở hữu lẫn người dùng.

Nguyên nhân của lỗ hổng thường là do các nhà phát triển bỏ qua việc kiểm soát quyền truy cập vào hàm rút tiền. Hoặc họ có thể cho rằng quá trình xác thực đã diễn ra trong một giai đoạn trước đó. Tuy nhiên không có cơ chế nào ngăn chặn hacker gọi trực tiếp đến hàm sensitive. **Hình 2.12** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract UnprotectedWithdrawalExample {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        // Unprotected Ether Withdrawal vulnerability
        uint256 amount = balances[msg.sender];
        msg.sender.transfer(amount);

        // Set balance to 0 after withdrawal
        balances[msg.sender] = 0;
    }
}

```

Hình 2.12: Mô phỏng lỗ hổng Unprotected Ether Withdrawal

2.2.3.4. Timestamp Dependence

Lỗ hổng Timestamp Dependence xảy ra khi smart contract phụ thuộc quá nhiều vào dữ liệu thời gian. Trong môi trường blockchain, thời gian có thể bị giả mạo. Nếu không xử lý cẩn thận, điều này có thể dẫn đến những hậu quả nghiêm trọng, bao gồm cả việc lừa đảo tài chính. Kẻ gian có thể tấn công bằng cách làm sai lệch giá trị thời gian. Nếu hợp đồng dựa vào đó để xử lý các hoạt động then chốt, hậu quả có thể rất lớn. Cần có thêm các biện pháp kiểm soát khác để ngăn chặn. **Hình 2.13** đã mô phỏng lại cuộc tấn công trong smart contract.

```
pragma solidity ^0.8.0;

contract TimestampDependenceExample {
    mapping(address => uint256) public lastAccessTime;

    function accessRestrictedResource() public {
        require(block.timestamp - lastAccessTime[msg.sender] > 1 days, "Access allowed once per day");

        // Do something important here

        lastAccessTime[msg.sender] = block.timestamp;
    }
}
```

Hình 2.13: Mô phỏng lỗ hổng Timestamp Dependence

Thử thách lớn nhất là khả năng hacker làm giả dữ liệu thời gian. Chúng có thể thay đổi thời gian của các nút mạng để đánh lừa smart contract. Điều đó ảnh hưởng tới quyết định của hợp đồng và mở ra cơ hội gian lận.

Như vậy, lỗ hổng này liên quan tới việc smart contract quá phụ thuộc vào dữ liệu thời gian dễ bị làm giả, dẫn tới những hậu quả tiêu cực.

2.2.3.5. Front Running

Front Running đang nổi lên như một mối đe dọa lớn đối với an toàn của các ứng dụng tài chính phi tập trung hoạt động trên nền tảng blockchain. Bản chất của vấn đề này là hacker theo dõi, nhận diện các giao dịch lớn của người dùng, sau đó cố tình đẩy lệnh giao dịch của mình lên trước. Kết quả là giá và điều kiện thực thi giao dịch ban đầu bị ảnh hưởng, từ đó hacker hưởng lợi.

Cụ thể, hacker sẽ liên tục giám sát những giao dịch tiềm năng có thể tác động mạnh đến thị trường. Khi phát hiện một giao dịch lớn, chúng nhanh tay đặt lệnh trước với giá khác biệt để thu lợi từ sự chênh lệch. Điều này không chỉ khiến hacker hưởng lợi bất chính mà còn gây rủi ro, thiệt hại cho người dùng thực sự do phải mua bán với giá không công bằng. **Hình 2.14** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract FrontRunningVulnerableContract {
    address public owner;
    uint256 public price;

    constructor() {
        owner = msg.sender;
        price = 1 ether;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Not the contract owner");
       _;
    }

    function setPrice(uint256 newPrice) public onlyOwner {
        price = newPrice;
    }

    function buy() public payable {
        require(msg.value >= price, "Insufficient funds");

        // Simulate front running vulnerability
        // In a real scenario, this vulnerable part could be more complex
        if (msg.value > price) {
            // Front running vulnerability: Refund the excess value
            payable(msg.sender).transfer(msg.value - price);
        }

        // Complete the purchase
        owner = msg.sender;
        price = msg.value;
    }

    function withdraw() public onlyOwner {
        // Owner can withdraw the balance
        payable(owner).transfer(address(this).balance);
    }
}

```

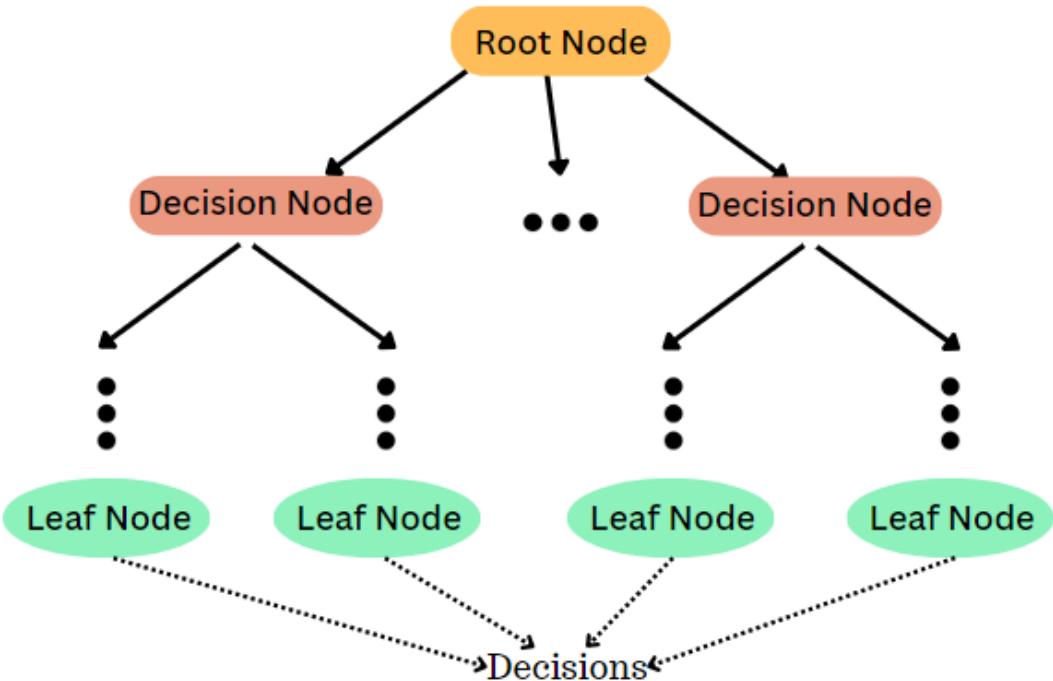
Hình 2.14: Mô phỏng lỗ hổng Front Running

2.3. Các phương pháp phát hiện lỗ hổng tự động

2.3.1. Phương pháp học máy

2.3.1.1. Decision Tree

Decision Tree là một phương pháp phân loại và dự báo được ưa chuộng trong lĩnh vực học máy, nổi bật với khả năng dễ giải thích và hiểu. Nó hoạt động dựa trên việc tạo ra một cấu trúc dạng cây, trong đó mỗi nút thể hiện một thuộc tính cụ thể và mỗi nhánh biểu diễn một giá trị của thuộc tính đó. Quá trình huấn luyện cây sử dụng các thước đo độ thuần khiết như Entropy hay Gini Index để liên tục phân chia dữ liệu cho đến khi đạt được điểm dừng. **Hình 2.15** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.15: Mô hình học máy Decision Tree

Các ưu điểm chính của Decision Tree bao gồm: khả năng xử lý hiệu quả dữ liệu phi tuyến và tương tác giữa các thuộc tính; không cần phải tiền xử lý dữ

liệu trước khi huấn luyện; khả năng xử lý được cả dữ liệu định danh và số liệu; và khả năng trực quan hóa, giúp dễ dàng giải thích quá trình ra quyết định. Tuy nhiên, mô hình này cũng có một số hạn chế, như dễ bị overfitting khi cấu trúc cây quá phức tạp, và sự nhạy cảm với nhiễu trong dữ liệu.

Decision Tree được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm: y tế, cho việc chẩn đoán bệnh và dự đoán biến chứng; tài chính, trong việc đánh giá rủi ro tín dụng và phát hiện gian lận thẻ tín dụng; bán lẻ, để dự đoán hành vi khách hàng; viễn thông, trong việc nhận diện cuộc gọi rác, và nhiều ứng dụng khác. Các ứng dụng điển hình bao gồm hệ thống hỗ trợ ra quyết định, phát hiện bất thường, dự đoán xu hướng, phân khúc khách hàng, lọc các trường hợp quan tâm, v.v.

Tóm lại, với tính chất đơn giản, dễ hiểu và hiệu quả, Decision Tree đã trở thành một trong những mô hình quan trọng và không thể thiếu trong lĩnh vực học máy và thống kê. Nó đã và đang được ứng dụng một cách rộng rãi trong nhiều lĩnh vực đa dạng. Dự kiến, kỹ thuật này sẽ tiếp tục phát triển và mở rộng ứng dụng của mình trong tương lai.

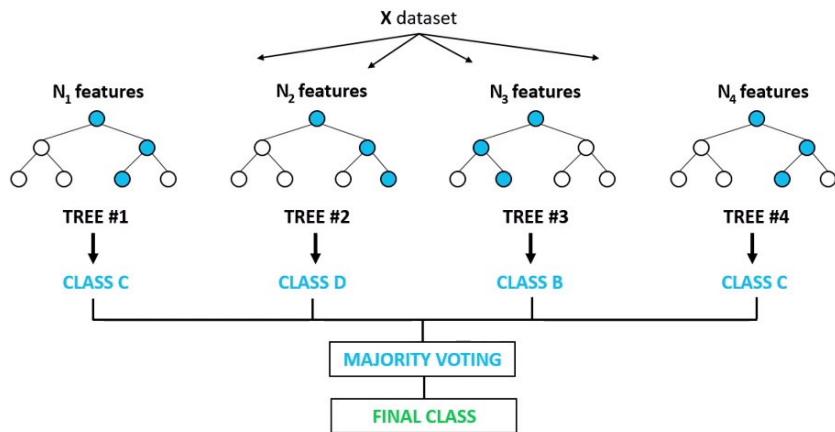
2.3.1.2. Random Forest

Random Forest là một kỹ thuật học máy mạnh mẽ dựa trên nền tảng của Decision Tree. Ý tưởng chính là xây dựng một tập hợp gồm nhiều cây quyết định thay vì chỉ một cây đơn lẻ. Mỗi cây trong mô hình được huấn luyện dựa trên một tập con được chọn một cách ngẫu nhiên từ tập dữ liệu gốc. Cách tiếp cận này tạo ra sự đa dạng trong mô hình và giúp cải thiện độ chính xác so với việc sử dụng chỉ một decision tree đơn lẻ.

Trong quá trình huấn luyện, bước đầu tiên là tạo các tập dữ liệu con bằng cách chọn ngẫu nhiên từ tập dữ liệu ban đầu. Sau đó, mỗi tập dữ liệu con này sẽ được dùng để huấn luyện một cây quyết định riêng lẻ, sử dụng thuật toán decision tree chuẩn. Quá trình này được tiếp tục cho đến khi số lượng cây quyết định mong muốn được xây dựng. Khi thực hiện dự đoán, mỗi cây quyết định

đưa ra phán đoán của mình và kết quả cuối cùng là sự đồng thuận (đa số bình chọn) từ tất cả các cây. **Hình 2.16** đã mô phỏng lại kiến trúc của mô hình.

Random Forest Classifier



Hình 2.16: Mô hình học máy Random Forest

Ưu điểm nổi trội của Random Forest là khả năng giảm overfitting và tăng khả năng tổng quát hóa so với các thuật toán cơ bản như decision tree. Sự đa dạng của các cây quyết định giúp tránh hiện tượng quá khớp với tập huấn luyện. Bên cạnh đó, nhược điểm là Random Forest tốn nhiều tài nguyên tính toán và khó giải thích kết quả hơn.

Random Forest được áp dụng trong nhiều lĩnh vực khác nhau, từ việc giải quyết các bài toán phân loại và dự đoán, đến phân tích hình ảnh và xử lý ngôn ngữ tự nhiên. Nó được sử dụng rộng rãi trong các ngành như tài chính, y tế, gen và hình ảnh. Nhờ vào độ chính xác vượt trội và khả năng chịu đựng nhiễu, Random Forest đã trở thành một trong những mô hình học máy phổ biến và hiệu quả.

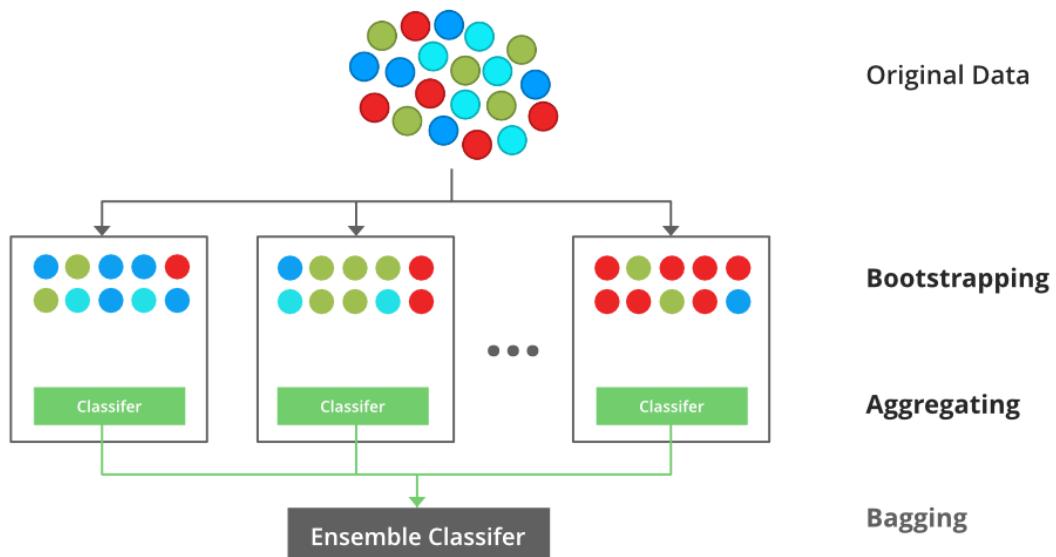
Tóm lại, với những ưu điểm như độ chính xác cao, khả năng chống nhiễu và tính linh hoạt, Random Forest là một công cụ mạnh mẽ trong nhiều ứng dụng

học máy và đã được sử dụng rộng rãi trong thực tiễn. Nó chắc chắn là một mô hình không thể thiếu trong học máy hiện đại.

2.3.1.3. XGBoost

XGBoost đứng đầu trong số các thuật toán học máy tiên tiến và hiệu quả, với hiệu suất vượt trội so với nhiều thuật toán cơ bản khác. Là phiên bản cải tiến của Gradient Boosting, XGBoost tích hợp nhiều kỹ thuật tối ưu hóa để cải thiện đáng kể cả về độ chính xác lẫn hiệu suất.

Cơ chế hoạt động của XGBoost là xây dựng mô hình dựa trên việc kết hợp nhiều cây quyết định "yếu". Mỗi cây mới được thêm vào nhằm giảm thiểu những khuyết điểm của các cây đã tồn tại, từ đó liên tục cải thiện độ chính xác của mô hình. Quá trình huấn luyện tập trung vào việc tối thiểu hóa lỗi dự đoán thông qua việc tối ưu hóa hàm loss. Để tránh overfitting, XGBoost sử dụng kỹ thuật regularization và early stopping. **Hình 2.17** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.17: Mô hình học máy XGBoost

XGBoost nổi bật với độ chính xác rất cao, thường cao hơn nhiều so với các thuật toán khác. Ngoài ra, thuật toán này còn có tốc độ huấn luyện nhanh, khả năng xử lý hiệu quả đối với dữ liệu lớn và phức tạp. Tuy nhiên, nhược điểm lớn nhất của XGBoost là yêu cầu tài nguyên tính toán cao và thách thức trong việc giải thích mô hình.

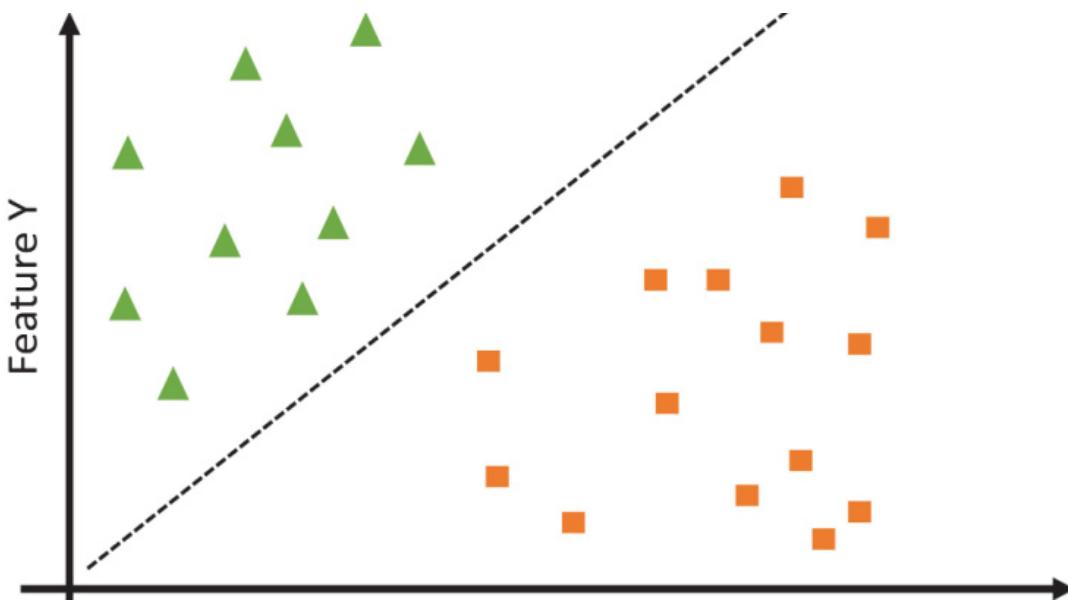
XGBoost có ứng dụng rất rộng rãi, từ việc dự đoán trong lĩnh vực tài chính, phân tích dữ liệu khách hàng, nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, cho đến các vấn đề dự báo trong nông nghiệp và giao thông. Trong nhiều cuộc thi AI và dự án nghiên cứu, XGBoost thường được chọn lựa do khả năng mô hình hóa xuất sắc của nó.

Tóm lại, với sự kết hợp của nhiều kỹ thuật tối ưu hiện đại và khả năng nâng cao hiệu suất, XGBoost đã trở thành một trong những thuật toán học máy hàng đầu và phổ biến nhất trong những năm gần đây.

2.3.1.4. Support Vector Machine

Support Vector Machine (SVM) là một thuật toán học máy được ưa chuộng, áp dụng cho cả phân loại và hồi quy. Điểm nổi bật của SVM là khả năng xử lý hiệu quả dữ liệu trong không gian chiều cao, đặc biệt trong các trường hợp dữ liệu lớn và phức tạp.

SVM hoạt động bằng cách xác định một siêu phẳng (hyperplane) tối ưu phân chia không gian thành hai phần để phân loại các nhóm dữ liệu. Trong trường hợp dữ liệu có thể phân loại theo dạng tuyến tính, siêu phẳng này sẽ được xác định sao cho khoảng cách giữa các điểm dữ liệu của các lớp khác nhau là lớn nhất. Đối với dữ liệu không tuyến tính, SVM sử dụng hàm kernel để chuyển dữ liệu vào không gian chiều cao hơn, giúp việc xác định siêu phẳng phân loại trở nên khả thi hơn. **Hình 2.18** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.18: Mô hình học máy Support Vector Machine

Một ưu điểm chính của SVM là hiệu suất cao khi làm việc trong không gian chiều cao, cùng với khả năng xử lý hiệu quả các tập dữ liệu lớn. SVM cũng rất linh hoạt với việc thay đổi hàm kernel, cho phép nó xử lý nhiều loại dữ liệu khác nhau một cách hiệu quả.

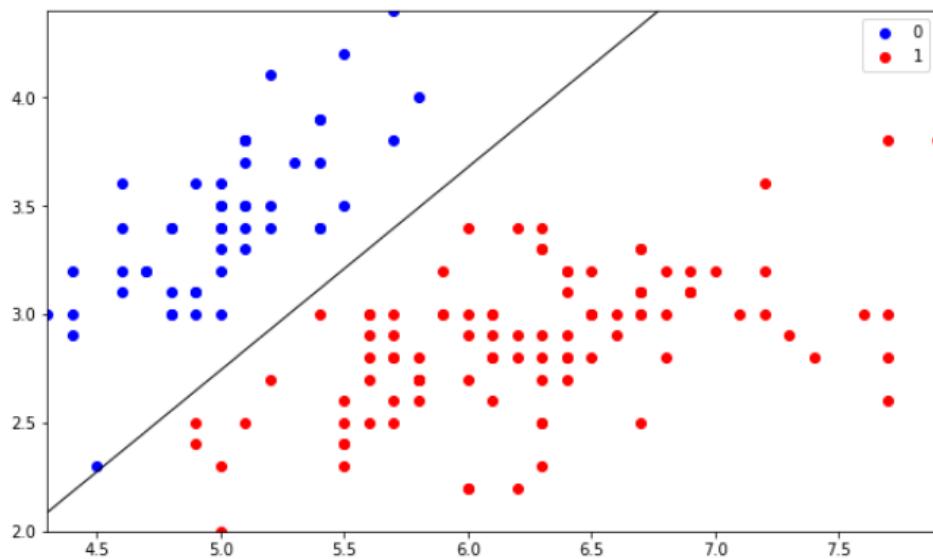
Tuy vậy, khi xử lý các tập dữ liệu cực lớn, SVM có thể trở nên khá phức tạp và yêu cầu lượng lớn tài nguyên. Việc chọn lựa hàm kernel và tham số phù hợp cũng cần có kiến thức chuyên sâu và sự hiểu biết sâu rộng về bản chất của dữ liệu đang được xử lý.

Ứng dụng của SVM rất đa dạng, từ phân loại văn bản, nhận diện khuôn mặt, đến dự báo giá chứng khoán. Đối với các tập dữ liệu có cấu trúc rõ ràng và không gian lớn, SVM thường là một lựa chọn ưu việt với tính chất phân loại đạt độ chính xác cao và hiệu suất ổn định.

2.3.1.5. Logistic Regression

Logistic Regression là một kỹ thuật phân loại cơ bản và thiết yếu trong học máy, được sử dụng để dự đoán xác suất thuộc tính của một đối tượng vào một nhóm cụ thể, dựa trên các đặc trưng của đối tượng đó.

Trong Logistic Regression, mối quan hệ giữa các biến độc lập - tức là các thuộc tính của đối tượng - và xác suất dự đoán kết quả (đầu ra) được mô hình hóa thông qua hàm logistic. Quá trình huấn luyện mô hình bao gồm việc cập nhật các tham số để chính xác phản ánh mối liên hệ này. Mô hình cuối cùng sau đó có thể được sử dụng để dự đoán xác suất cho các tình huống mới. **Hình 2.19** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.19: Mô hình học máy Logistic Regression

Ưu điểm lớn nhất của Logistic Regression là đơn giản, dễ hiểu và dễ giải thích cách thức hoạt động. Nó cũng khá linh hoạt, có thể mở rộng ứng dụng trong việc giải quyết các bài toán phân loại đa dạng, từ nhị phân tới đa lớp. Tuy vậy, giả định tuyến tính của mô hình là một điểm yếu lớn khi dữ liệu phức tạp.

Logistic Regression được ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm y học để dự đoán các bệnh và phân loại giai đoạn của bệnh ung thư; trong lĩnh vực ngân hàng để đánh giá rủi ro tín dụng; và trong marketing để dự đoán xác suất khách hàng mua sắm trực tuyến. Đây là công cụ hữu ích cho hầu hết các nhiệm vụ phân loại cơ bản, đặc biệt khi yêu cầu mô hình cần có khả năng giải thích rõ ràng.

Tóm lại, Logistic Regression được đánh giá cao do tốc độ huấn luyện nhanh,

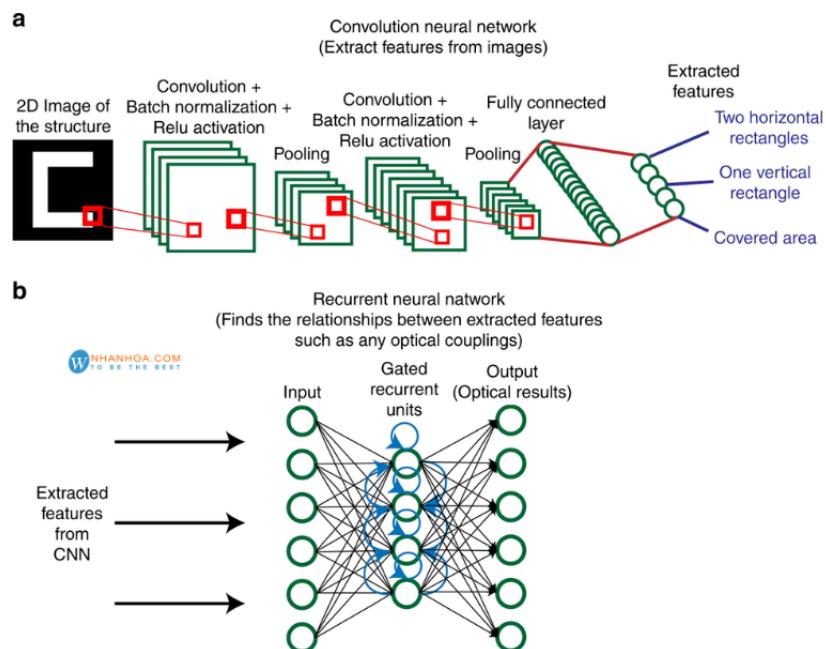
dễ dàng triển khai và khả năng giải thích mô hình một cách rõ ràng, làm cho nó trở thành một trong những kỹ thuật học máy cơ bản và quan trọng cho hầu hết các tác vụ phân loại.

2.3.2. Phương pháp học sâu

2.3.2.1. Convolutional Neural Network

Convolutional Neural Network (CNN) đã trở thành một mô hình chủ chốt trong công nghệ nhận dạng hình ảnh hiện đại, được biết đến với khả năng hiệu quả trong việc học các đặc trưng phức tạp từ dữ liệu hình ảnh và video.

CNN là kiến trúc mạng nơ-ron sâu được thiết kế đặc biệt cho việc xử lý hình ảnh. Nó bao gồm các lớp tích chập và lớp gộp là hai yếu tố quan trọng nhất. Các lớp tích chập dùng bộ lọc để phát hiện các đặc trưng cục bộ như cạnh, góc và kết cấu của hình ảnh. Sau đó, các lớp gộp giúp giảm kích thước của hình ảnh, loại bỏ thông tin không cần thiết, và tăng tốc độ xử lý hình ảnh. **Hình 2.20** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.20: Mô hình học sâu Convolutional Neural Network

Ưu điểm vượt trội của CNN là khả năng học các đặc trưng trừu tượng cấp cao, biểu diễn đa chiều của hình ảnh mà không cần kỹ thuật trích chọn đặc trưng thủ công phức tạp. Điều này giúp CNN vượt trội trong các tác vụ phân loại, phát hiện đối tượng trên ảnh. Song, CNN yêu cầu về mặt số lượng lớn ở dữ liệu và tài nguyên cho việc thực hiện các công việc tính toán lớn.

Ứng dụng của CNN bao trùm các bài toán liên qua đến việc xử lý ảnh: nhận dạng khuôn mặt, phân loại đối tượng trong ảnh vệ tinh, dịch chữ viết tay, dự đoán bệnh trên ảnh y khoa,... CNN là “con ngựa chiến” cho mọi hệ thống thị giác.

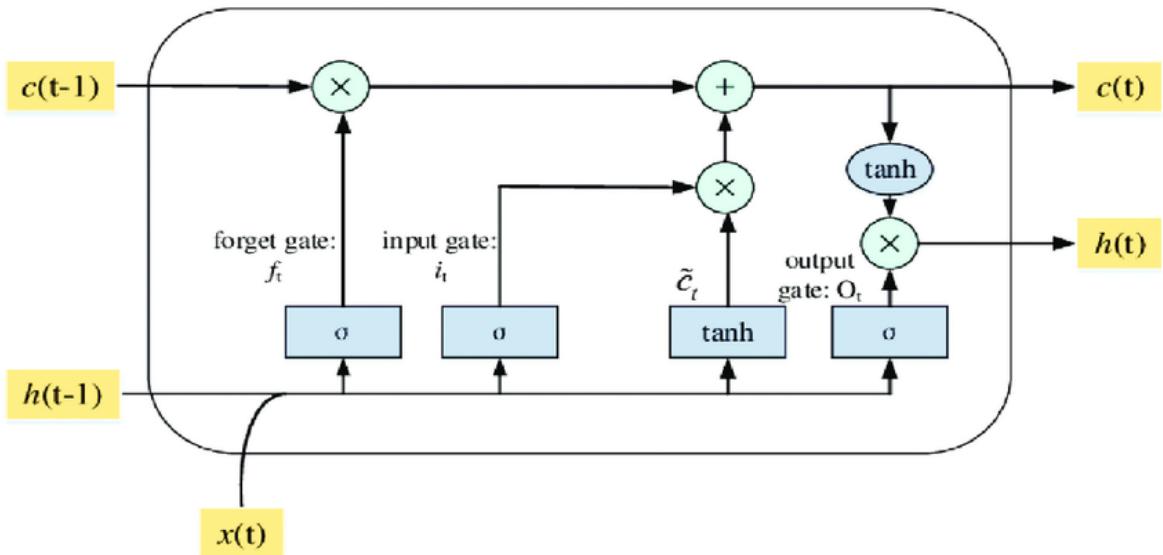
Với sự phát triển mạnh mẽ của dữ liệu hình ảnh và video trong thập kỷ qua, cùng nhu cầu nhận dạng ngày một cao, có thể nói CNN chính là công cụ đầy quyền năng đã giúp cách mạng hóa khả năng hiểu hình ảnh của máy móc. Sự phát triển nhanh chóng và ứng dụng rộng rãi của CNN chứng tỏ rằng nó sẽ tiếp tục đóng một vai trò không thể thay thế trong tương lai của công nghệ.

2.3.2.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) là một kiến trúc mạng nơ-ron học sâu mang tính đột phá, đặc biệt trong lĩnh vực xử lý ngôn ngữ tự nhiên và dữ liệu dạng chuỗi. Nhờ vào cơ chế cỗng kiểm soát, LSTM có khả năng xử lý vấn đề biến mất của gradient, một trở ngại thường gặp khi huấn luyện mô hình trên các chuỗi dữ liệu dài.

Cụ thể, LSTM sử dụng 3 loại cỗng để quyết định xem nên lưu giữ, cập nhật hay loại bỏ thông tin từ các bước trước đó của chuỗi. Cỗng quên quyết định những gì nên loại bỏ; cỗng đầu vào quyết định những gì nên cập nhật; cỗng đầu ra xác định thông tin nào truyền tới các tế bào LSTM tiếp theo. Nhờ đó, LSTM có thể duy trì và khai thác thông tin liên quan trong những chuỗi dữ liệu dài.

Hình 2.21 đã mô phỏng lại kiến trúc của mô hình.



Hình 2.21: Mô hình học sâu Long Short-Term Memory

LSTM nổi bật với khả năng xử lý các phụ thuộc xa, một thách thức lớn đối với các mạng nơ-ron truyền thống. Nó còn có khả năng học được các mối quan hệ phức tạp trong dữ liệu chuỗi. Tuy nhiên, LSTM yêu cầu một lượng lớn dữ liệu và tốn nhiều tài nguyên tính toán hơn so với các mô hình đơn giản hơn.

LSTM được ứng dụng chủ yếu trong các hệ thống xử lý ngôn ngữ tự nhiên như dịch máy, tạo văn bản tự động, và nhận dạng giọng nói. Nó cũng được sử dụng trong dự báo chuỗi thời gian ở các lĩnh vực như tài chính, khí tượng, và sản xuất.

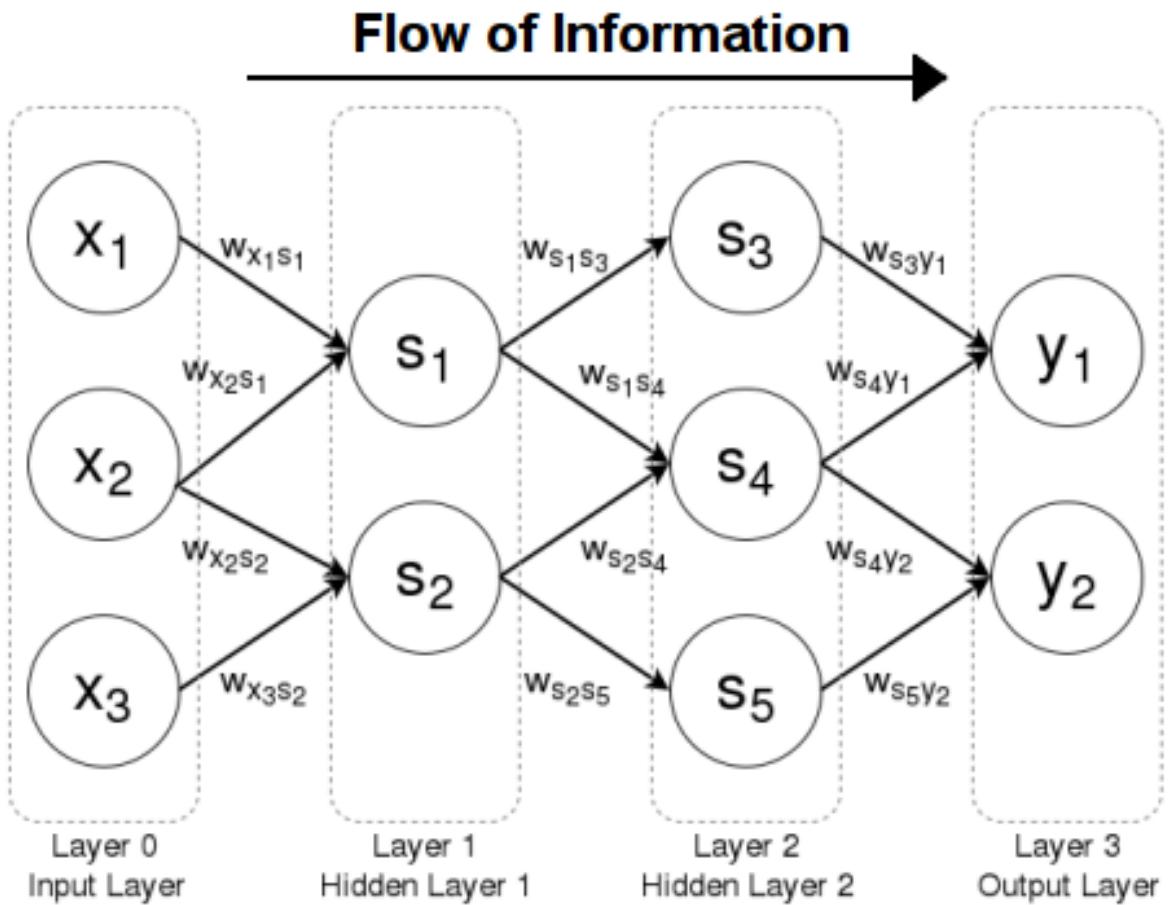
Tóm lại, với khả năng xử lý hiệu quả các phụ thuộc xa trong dữ liệu chuỗi, LSTM đã trở thành một bước đột phá quan trọng, mở ra một kỷ nguyên mới cho AI trong lĩnh vực ngôn ngữ và dự báo chuỗi. Sự phát triển vượt bậc của xử lý ngôn ngữ tự nhiên gần đây có sự đóng góp không nhỏ từ kiến trúc LSTM.

2.3.2.3. Feed forward network

Feedforward Neural Network (FNN) là kiến trúc mạng nơ-ron nhân tạo cơ bản, nổi bật với đặc trưng không có chu kỳ hay vòng lặp, và dữ liệu chuyển tiếp một chiều từ lớp đầu vào, qua các lớp ẩn, đến lớp đầu ra. Mô hình này vẫn giữ

một vai trò quan trọng và được áp dụng rộng rãi.

FNN cấu tạo từ các lớp nơ-ron, với mỗi lớp được kết nối liên tục và thẳng hàng từ đầu vào đến đầu ra. Mỗi nơ-ron trong một lớp kết nối với tất cả nơ-ron ở lớp kế tiếp. Trong quá trình lan truyền thông tin qua mạng, trọng số giữa các nơ-ron được điều chỉnh để tối ưu hóa kết quả cuối cùng. **Hình 2.22** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.22: Mô hình học sâu Feedforward neuron network

Ưu điểm chính của FNN chính là sự đơn giản, dễ hiểu và khả năng triển khai một cách thuận tiện. FNN có thể xử lý nhiều loại dữ liệu khác nhau và thích hợp với đa dạng các loại bài toán. Tuy nhiên, FNN không hiệu quả lắm trong việc nắm bắt các mối quan hệ phức tạp hoặc phi tuyến tính trong dữ liệu.

Ứng dụng của FNN rất đa dạng, bao gồm nhận dạng chữ số viết tay, phân

loại văn bản/email, phân loại hình ảnh, dự đoán chuỗi thời gian tài chính. Đối với các bài toán đơn giản hóa hoặc dữ liệu có cấu trúc phân lớp rõ ràng, FNN thường đạt hiệu quả cao.

Tóm lại, với tốc độ huấn luyện nhanh, khả năng triển khai dễ dàng và hiệu suất tốt trên nhiều loại bài toán, FNN vẫn giữ một vai trò cốt lõi trong lĩnh vực mạng nơ-ron nhân tạo hiện đại. Sự phát triển mạnh mẽ của học sâu trong những năm gần đây cũng bắt nguồn từ những ý tưởng sơ khai về FNN.

2.3.2.4. Robustly Optimized BERT Pretraining Approach

RoBERTa (Robustly Optimized BERT Pretraining Approach) là một cải tiến đáng kể trong lĩnh vực xử lý ngôn ngữ tự nhiên, mang lại khả năng hiểu và mô hình hóa ngôn ngữ với độ chính xác và sâu sắc chưa từng thấy. Đây là phiên bản được tối ưu hóa từ kiến trúc BERT ban đầu.

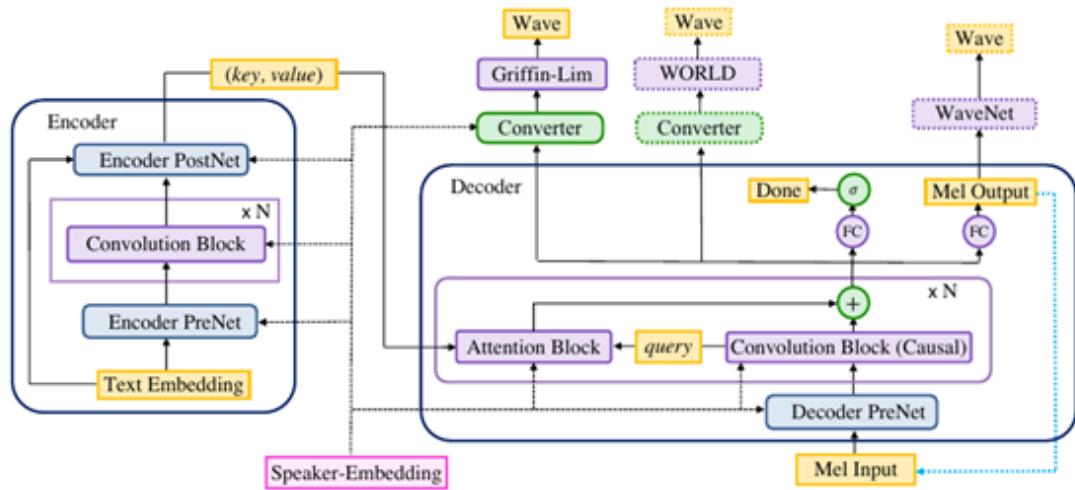
RoBERTa sử dụng cơ chế encoder-decoder tương tự như BERT để thu thập các đặc trưng ngữ nghĩa phong phú từ văn bản. Tuy nhiên, RoBERTa đã loại bỏ một số thành phần không cần thiết và áp dụng các kỹ thuật tối ưu hóa trong quá trình huấn luyện, giúp cải thiện hiệu suất đáng kể.

Điểm mạnh nhất của RoBERTa so với các mô hình trước đây là khả năng xử lý ngôn ngữ tự nhiên một cách chính xác và sâu sắc hơn. RoBERTa đã đạt được kết quả xuất sắc trên nhiều tác vụ xử lý ngôn ngữ tự nhiên phức tạp. Tuy nhiên, mô hình này yêu cầu lượng lớn dữ liệu và tài nguyên tính toán.

Ứng dụng của RoBERTa rất đa dạng, từ hỗ trợ trả lời câu hỏi, chatbot, dịch máy, đến phân loại và tóm tắt văn bản tự động. Với độ chính xác cao và khả năng hiểu ngôn ngữ sâu sắc, RoBERTa đã trở thành một bước đột phá trong nhiều ứng dụng AI liên quan đến ngôn ngữ.

Tóm lại, với sự phát triển về kiến trúc và các kỹ thuật tối ưu hóa, RoBERTa đánh dấu một bước tiến mới trong mô hình xử lý ngôn ngữ tự nhiên, mở ra một kỷ nguyên mới trong việc hiểu và xử lý ngôn ngữ bởi máy móc. **Hình 2.23** đã

mô phỏng lại kiến trúc của mô hình.



Hình 2.23: Mô hình học sâu RoBERTa

CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT

Trong phần này, em sẽ mô tả cách thức xây dựng và phát triển mô hình ChainSniper, với mục tiêu chính là tự động phát hiện các lỗ hổng trong hợp đồng thông minh trên mạng liên chuỗi. Em sẽ trình bày chi tiết về phương pháp và quy trình xây dựng mô hình, cũng như việc tạo ra tập dữ liệu dùng để đánh giá hiệu quả của các mô hình học máy trong hệ thống ChainSniper.

3.1. Tổng quan mô hình

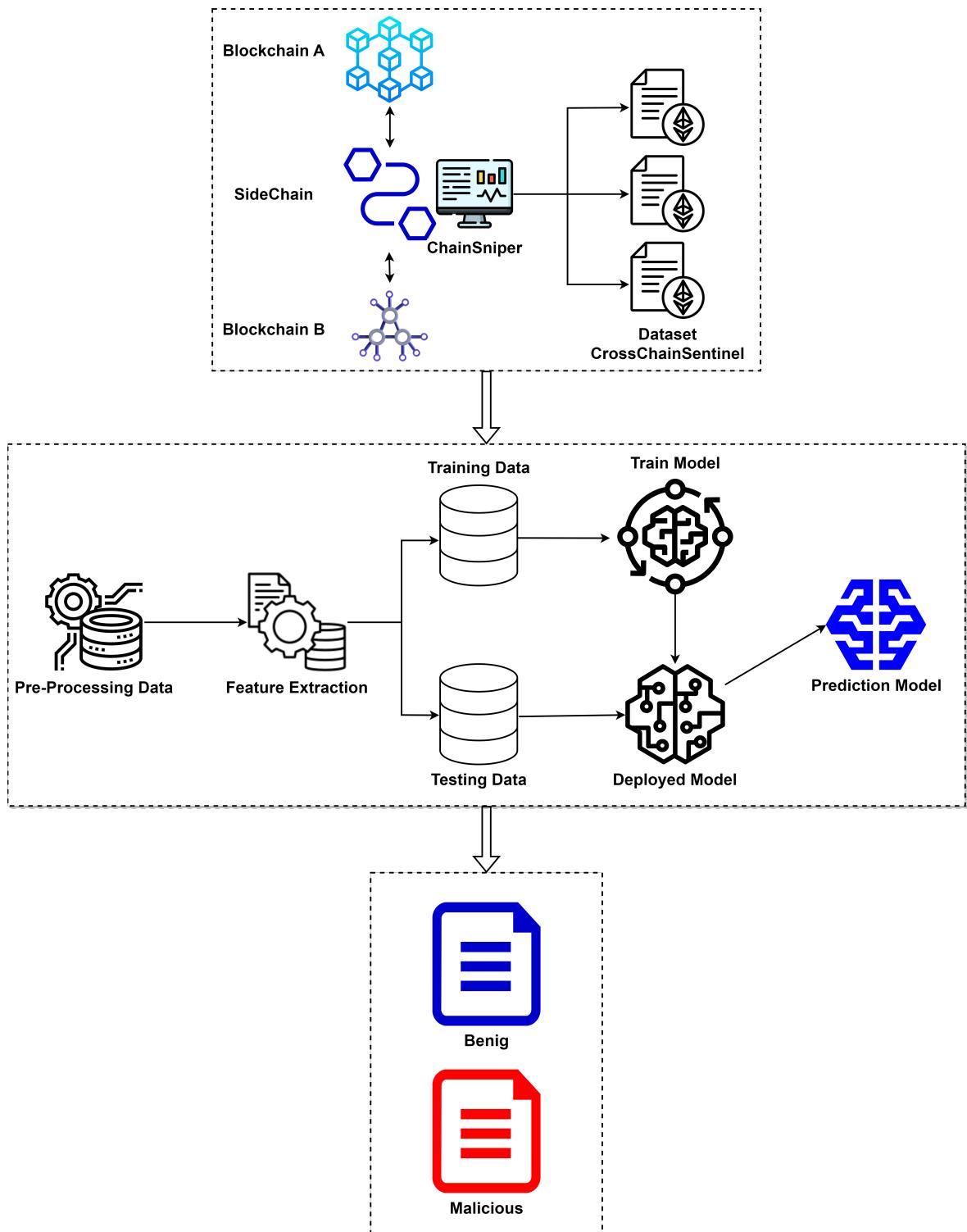
3.1.1. Các thành phần chính của mô hình

Mô hình chính của ChainSniper bao gồm 5 thành phần:

1. Ethereum Sepolia đóng vai trò là Blockchain A
2. Quorum đóng vai trò là Blockchain B
3. Sidechain đóng vai trò quan trọng trong việc chuyển dữ liệu và ghi lại các hợp đồng thông minh
4. Mô hình Học máy phát hiện lỗ hổng
5. Module phân loại trong hệ thống sử dụng sidechain có chức năng xác định và phân loại các hợp đồng thông minh

Hệ thống này gồm hai mạng blockchain liên kết thông qua cầu nối sidechain, giúp chuyển dữ liệu và lưu trữ thông tin của hợp đồng thông minh. Dữ liệu này sẽ được xử lý và chuẩn bị trước khi đưa vào các mô hình học máy, đã được huấn luyện trên tập dữ liệu có nhãn, để nhận diện các hợp đồng thông minh độc hại. Sau đó, hiệu suất của các mô hình này được đánh giá một cách kỹ lưỡng. Cuối cùng, module sử dụng kết quả dự đoán từ các mô hình để phân loại hợp đồng

thông minh trên sidechain thành lành tính hoặc độc hại. **Hình 3.1** minh họa mô hình chính:



Hình 3.1: Mô hình tổng quan

3.1.2. Mô hình cầu nối

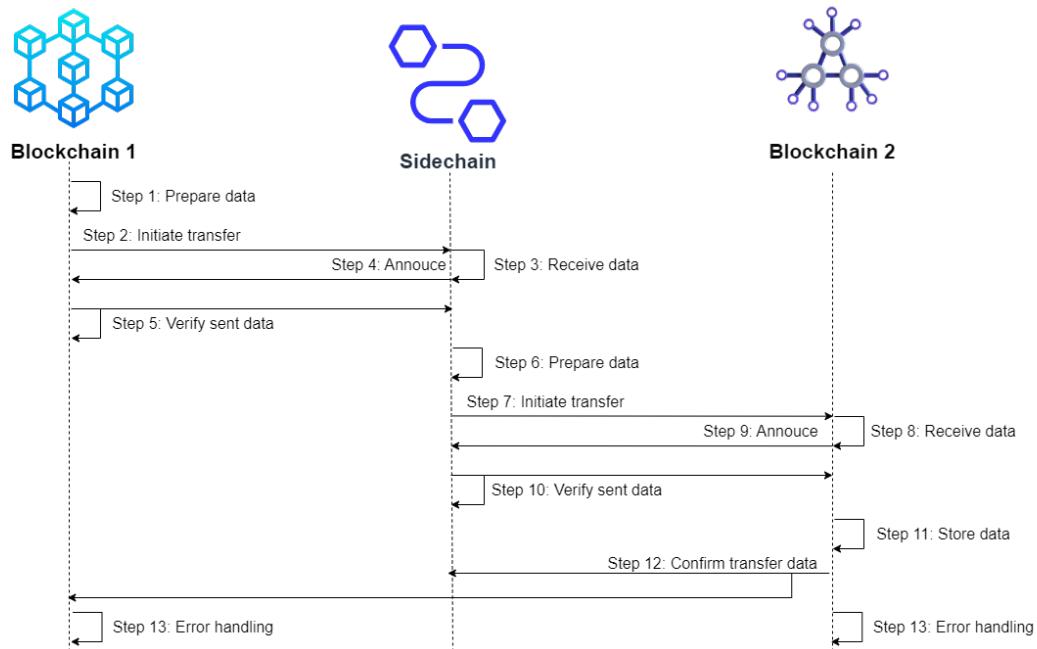
Hệ thống ChainSniper cho phép tương tác giữa các blockchain không đồng nhất qua cầu nối sidechain. Sidechain này đóng vai trò như một lớp trung gian, xử lý và chuyển dữ liệu giữa các mạng blockchain kết nối với nhau. Để có thể truyền tải dữ liệu, tài sản giữa các chuỗi được khóa lại ở một chuỗi này và giải khóa những biểu diễn tương đương ở chuỗi kia thông qua cơ chế neo hai chiều dựa trên hợp đồng đa chữ ký.

Khi có các giao dịch xuyên chuỗi diễn ra, các nút của sidechain sẽ ghi lại các thông tin, dữ liệu của giao dịch đó như địa chỉ của các hợp đồng thông minh tham gia, dấu thời gian diễn ra, các lời gọi hàm, các tham số được truyền vào, giá trị trả về, cũng như các ngoại lệ nếu có, sẽ được ghi lại trong nhật ký giao dịch. Nhật ký này sau đó được tổng hợp và xử lý để tạo nên một tập dữ liệu, cung cấp cái nhìn sâu sắc về hành vi và cách thức hoạt động của hợp đồng thông minh, cũng như các mẫu thực thi khác nhau của chúng.

Các bước thực hiện việc chuyển đổi dữ liệu qua sidechain (**Hình 3.2**):

1. Chuẩn bị dữ liệu: Xác định cấu trúc và bảo đảm tính nguyên vẹn của dữ liệu trước khi thực hiện chuyển đổi.
2. Khởi tạo quá trình chuyển đổi: Kết nối với sidechain để khởi tạo quá trình chuyển đổi.
3. Nhận dữ liệu ở sidechain: Sidechain nhận dữ liệu đến từ hệ thống blockchain A sau khi quá trình chuyển đổi được khởi tạo.
4. Sidechain thông báo nhận dữ liệu: blockchain A nhận thông báo từ sidechain về sự kiện chuyển đổi dữ liệu.
5. Kiểm tra Dữ liệu Trên Sidechain: Sidechain tiến hành xác nhận độ chính xác và tính nguyên vẹn của dữ liệu nhận được từ hệ thống blockchain chính.
6. Chuẩn bị dữ liệu (lần 2): Chuẩn bị lại dữ liệu cho lần chuyển đổi tiếp theo

7. Khởi tạo quá trình chuyển đổi (lần 2): Blockchain B tiếp tục kết nối với sidechain để khởi tạo lần chuyển đổi tiếp theo
8. Nhận dữ liệu (lần 2): Sidechain gửi dữ liệu mới đến Blockchain B.
9. Thông báo: Blockchain B nhận thông báo từ sidechain về sự kiện chuyển đổi dữ liệu lần thứ hai.
10. Xác minh dữ liệu đã gửi từ sidechain (lần 2): Sidechain tiến hành kiểm tra và đảm bảo tính chính xác cũng như nguyên vẹn của dữ liệu vừa được chuyển giao.
11. Bảo quản Dữ liệu: Dữ liệu được bảo lưu trên Blockchain B để sử dụng trong các hoạt động tiếp theo.
12. Xác nhận Chuyển đổi Dữ liệu: Blockchain B kiểm chứng rằng quá trình chuyển đổi đã hoàn tất thành công và dữ liệu đã được biến đổi một cách chính xác.
13. Quản lý Lỗi: Trong trường hợp xuất hiện lỗi, các mạng blockchain phải tiến hành xử lý, bao gồm cả việc gửi thông báo lỗi và ghi nhận chúng vào log để có thể theo dõi và giải quyết vấn đề.



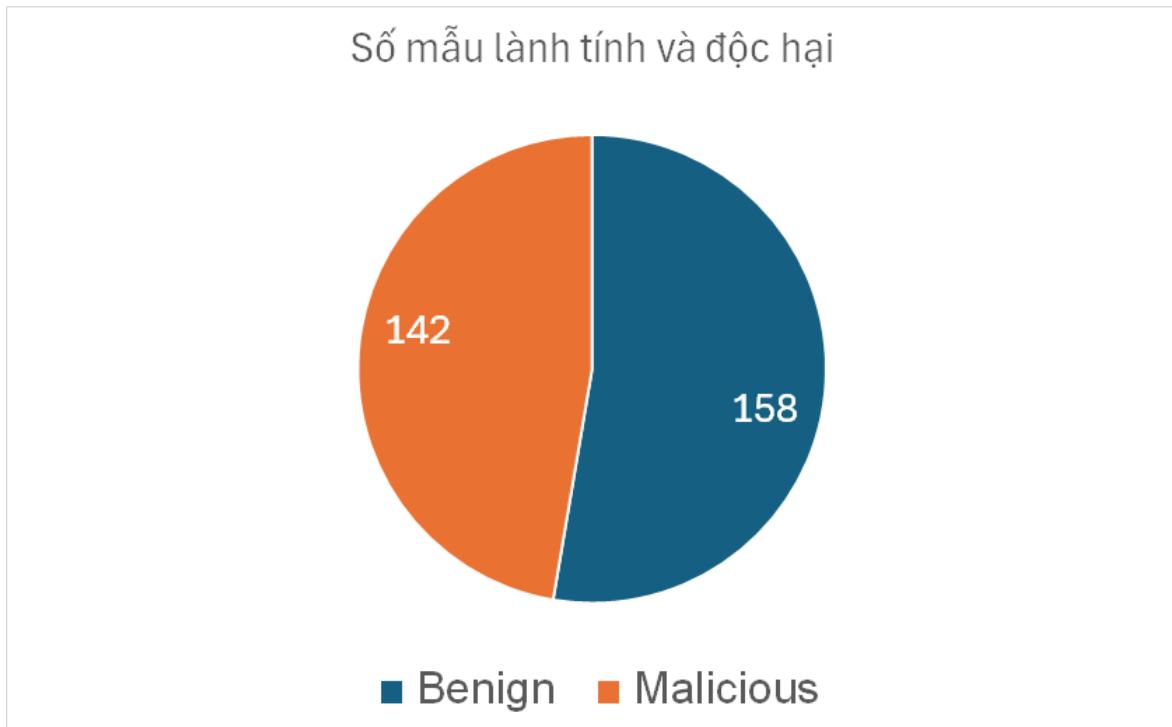
Hình 3.2: Các bước chuyển đổi dữ liệu qua cầu nối Sidechain

3.2. Phát hiện các lỗ hổng bằng phương pháp học máy và học sâu

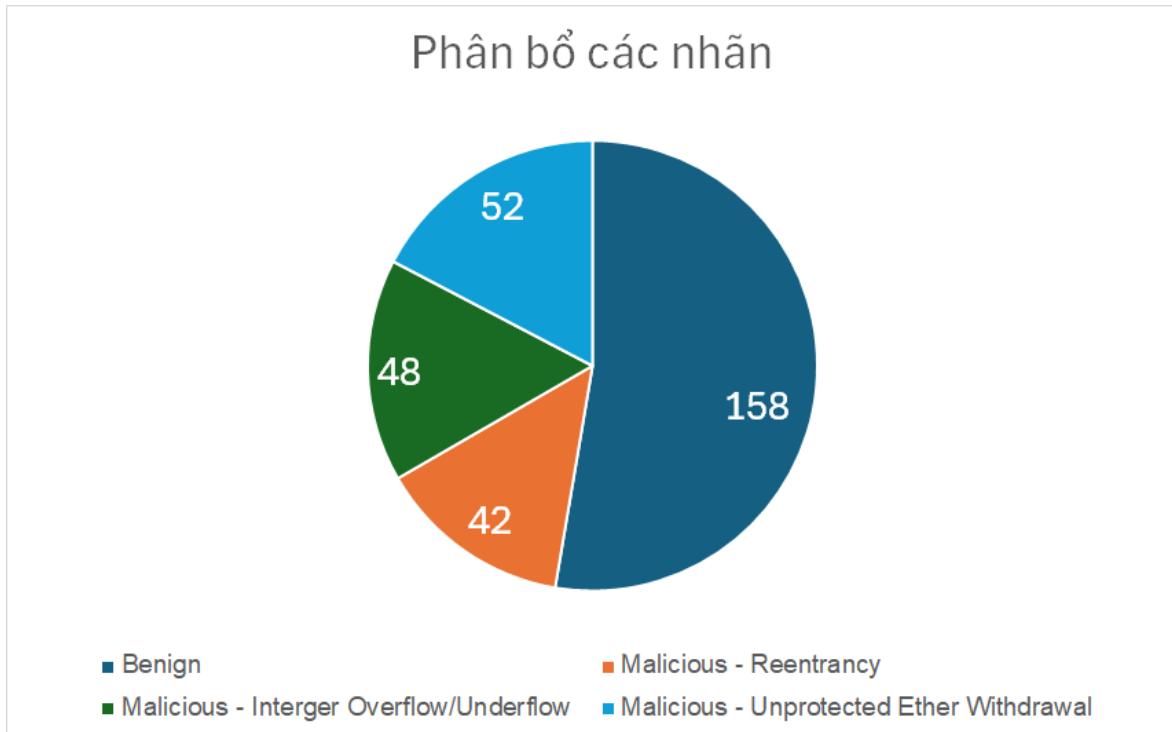
3.2.1. Xây dựng tập dữ liệu

Tập dữ liệu CrossChainSentinel¹ được tạo ra với mục đích phân tích các lỗ hổng tiềm ẩn trong các hợp đồng thông minh trên cầu nối sidechain. Tập dữ liệu này chứa 300 tệp hợp đồng thông minh, trong đó 158 mẫu được xác định là lành tính, 142 mẫu còn lại được xếp vào loại độc hại (**Hình 3.3**). Cụ thể, số mẫu độc hại bao gồm 42 hợp đồng bị lỗ hổng tái tham chiếu (Reentrancy), 48 hợp đồng chứa lỗi tràn/cạn số nguyên (Integer Overflow/Underflow) và 52 hợp đồng có vấn đề rút Ether ra mà không được bảo vệ (Unprotected Ether Withdrawal) (**Hình 3.4**).

¹<https://github.com/anhkiet1227/CrossChainSentinel>



Hình 3.3: Phân bố mẫu lành tính và độc hại



Hình 3.4: Phân bố mẫu lành tính và độc hại tương ứng với các lỗ hổng

Các lỗ hổng này được xác định thông qua sử dụng các kỹ thuật từ các nghiên cứu trước như Smartbugs-wild², SolidiFi-benchmark³ cùng Danh sách đen các địa chỉ Ethereum liên quan tới các cuộc tấn công đa chuỗi. Dữ liệu về cầu nối chuỗi chéo được mô phỏng theo 15 nhà cung cấp thực tế khác nhau gồm Commos, Avalanche, Chainlink... Tập dữ liệu CrossChainSentinel tập trung vào những lỗ hổng phổ biến có thể xảy ra khi tài sản được chuyển đổi giữa các blockchain thông qua hợp đồng sidechain.

Việc đưa ra các rủi ro tiềm ẩn đó trong một tập dữ liệu có nhãn rộng lớn giúp CrossChainSentinel hỗ trợ tăng cường khả năng kiểm toán, thử nghiệm và phát hiện những lỗ hổng nguy hiểm trong cơ chế cầu nối chuỗi chéo trước khi chúng được triển khai trên mạng chính. Sự đa dạng về nguồn chuỗi chéo và các loại lỗ hổng khiến đây là tập dữ liệu (dataset) lý tưởng để thực hiện công việc đánh giá công cụ và các mô hình nhằm tăng cường bảo mật chuỗi

3.2.2. Gán nhãn các mẫu và xử lý dữ liệu

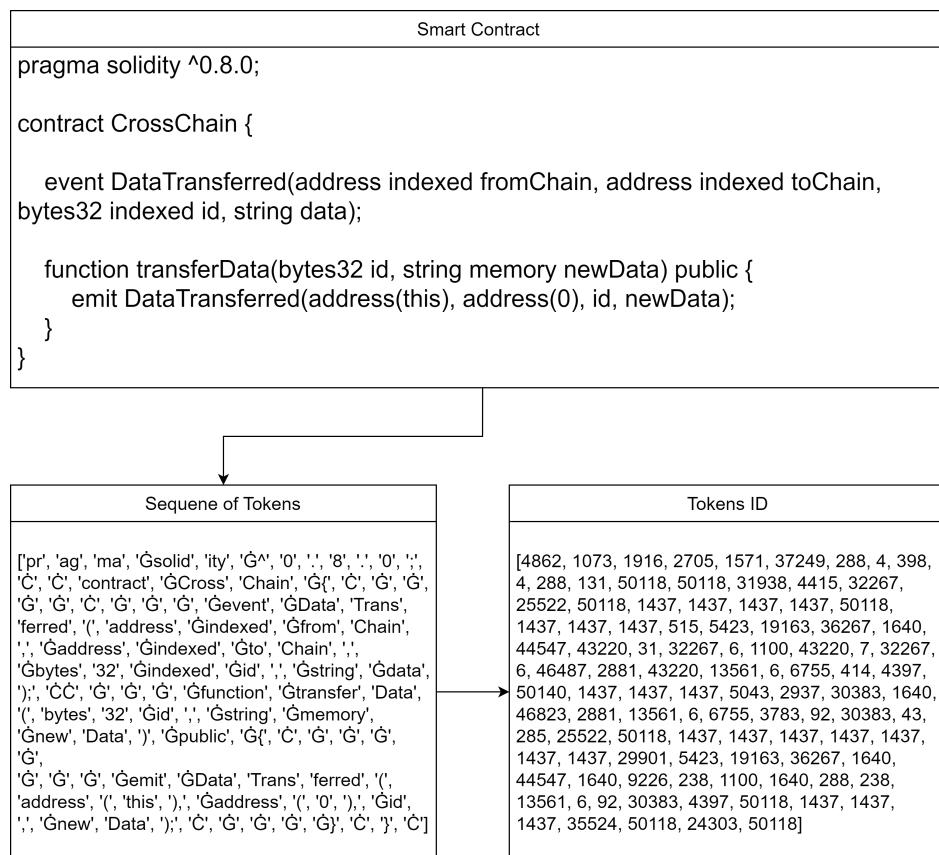
Tập dữ liệu chứa hai loại nhãn để phân loại hợp đồng thông minh. Tập nhãn đầu tiên là một tập nhãn nhị phân, xác định xem một hợp đồng có đặc điểm an toàn hay nguy hiểm. Tập nhãn thứ hai chi tiết hóa các nhãn đa lớp, phân loại các hợp đồng độc hại thành các loại lỗ hổng cụ thể như an toàn, có khả năng bị Reentrancy, Integer Overflow/Underflow và Unprotected Ether Withdrawal. Dữ liệu được xác định bằng các đặc điểm như tên file dự án, ID commit, nhãn mục tiêu, cấu trúc và nội dung của hợp đồng, cũng như số thứ tự file. Trong tập nhãn nhị phân, giá trị 0 biểu thị an toàn và 1 là nguy hiểm. Trong tập nhãn đa lớp, giá trị 0 là an toàn, 1 là lỗ hổng Reentrancy, 2 là lỗ hổng Integer Overflow/Underflow và 3 là lỗ hổng Unprotected Ether Withdrawal.

Để xử lý dữ liệu, em bắt đầu bằng việc chuyển đổi các file mã nguồn hợp đồng thông minh trở thành các token thông qua quá trình tokenization (**Hình**

²<https://github.com/smartbugs/smartbugs-wild>

³<https://github.com/DependableSystemsLab/SolidiFI-benchmark>

3.5). Mỗi file code smart contract được phân chia thành các token, đơn vị nhỏ nhất của mã nguồn, như từ vựng, ký tự đặc biệt, hoặc biểu thức. Việc này giúp em biểu diễn cấu trúc và ý nghĩa của mã nguồn một cách có cấu trúc. Tiếp theo, mỗi token được chuyển đổi thành một tokenID, là một số nguyên duy nhất đại diện cho loại cụ thể của token đó. Quá trình này giúp giảm kích thước của dữ liệu và tạo ra một biểu diễn số hóa hiệu quả cho mã nguồn hợp đồng. Quá trình này tối ưu hóa việc đưa dữ liệu vào mô hình học máy, giúp mô hình có khả năng "nắm bắt" cấu trúc và ngữ cảnh của mã nguồn hợp đồng thông minh. Từ đó, em đã phát triển một bộ các tokenID, trong đó mỗi tokenID tương ứng với một phần cụ thể của mã nguồn. Điều này cho phép em tạo ra các vector đặc trưng cho mỗi hợp đồng thông minh, và cuối cùng, sử dụng chúng để đưa vào mô hình học máy nhằm phân loại các lô hàng một cách chính xác.



Hình 3.5: Quá trình xử lý dữ liệu

3.2.3. Ứng dụng các mô hình học máy tổng việc phát hiện các lỗ hổng

Việc chọn các phương pháp học máy bởi chúng cho phép tự động phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh bằng cách phân biệt các mẫu trong mã hợp đồng và cấu trúc. So với kiểm toán thủ công, các mô hình học máy có thể phân tích hợp đồng một cách hiệu quả và nhanh hơn. Em thử nghiệm vài mô hình học máy cổ điển, bao gồm Cây quyết định, Rừng ngẫu nhiên, Máy vector hỗ trợ, XGBoost và Hồi quy Logistic. Nhờ khả năng giải thích, các mô hình này cho phép hiểu được lý do tại sao một số hợp đồng bị gắn cờ là dễ bị tấn công.

Decision tree và Random forest xây dựng các quy tắc phân cấp dựa trên các thuộc tính mã để phân loại hợp đồng. Trong khi đó, SVM tìm ranh giới tối ưu giữa hợp đồng dễ bị tấn công và an toàn trong không gian đặc trưng. XGBoost tiếp tục tăng độ chính xác thông qua một tập hợp các học viên yếu. Em trích xuất các đặc trưng số liệu có giá trị thông tin như độ phức tạp hàm, luồng điều khiển và các mẫu cú pháp.

Bằng cách huấn luyện trên dữ liệu đại diện thích hợp của cả ví dụ có chiều hướng tích cực lẫn chiều hướng tiêu cực, các mô hình học cách phát hiện bền vững các lỗ hổng an toàn thông tin. Em áp dụng các kỹ thuật này trong ChainSniper để xây dựng một máy quét tự động cho các lỗ hổng như Reentrancy, Integer Overflow/Underflow và Unprotected Ether Withdrawal. Tính linh hoạt của học máy cung cấp một phương pháp luận để phát hiện lỗ hổng trong khi tránh nỗ lực thủ công mở rộng.

3.2.4. Ứng dụng các mô hình học sâu trong việc phát hiện các lỗ hổng

Quá trình lựa chọn các phương pháp học sâu được thực hiện vì khả năng của chúng trong việc mô hình hóa các mối quan hệ phức tạp của logic mã và phụ

thuộc mà không cần trích xuất thủ công các đặc trưng. Các kỹ thuật như CNN, RNN, LSTM và các mô hình Transformer như RoBERTa cho phép học từ đầu đến cuối trực tiếp từ mã nguồn hợp đồng thông minh thô để phát hiện các mẫu cú pháp và ngữ nghĩa phức tạp mà đặc trưng cho các lỗ hổng.

Việc thử nghiệm các mạng nơ-ron sâu bao gồm các mô hình tuần tự dựa trên LSTM có khả năng diễn giải các luồng điều khiển, CNN trích xuất các mẫu cú pháp cục bộ và bộ mã hóa Transformer như RoBERTa cung cấp các vector từ ngữ cảnh phong phú. Nhờ xử lý phân cấp theo tầng, các mạng này tự động học quan hệ tiềm ẩn giữa các token và cấu trúc dẫn đến sự hiểu biết về logic cho thấy lỗ hổng bảo mật.

Hơn thế nữa, các phụ thuộc dài hạn được mô hình hóa bởi LSTM có thể truy tìm các luồng thông tin trong hợp đồng để xác định các vấn đề kiểm tra cuộc gọi không kiểm soát. Các vector ngữ cảnh của RoBERTa có khả năng phân biệt các sắc thái giữa các cấu trúc có vẻ tương tự nhưng có thể hoặc không dễ bị tấn công. Các lớp chú ý có thể giải thích được của học sâu cũng hỗ trợ xác định các thành phần gây vấn đề. Việc áp dụng các kỹ thuật này trong ChainSniper để xây dựng các máy quét tự động có thể cờ các nguy cơ Reentrancy, Interger Overflow/Underflow và Unprotected Ether Withdrawal.

CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ

Trong phần này, em sẽ mô tả quá trình thiết lập thực nghiệm, bao gồm việc cấu hình môi trường thực nghiệm, xác định các chỉ số đánh giá và phác thảo các kịch bản thực nghiệm. Em cũng sẽ trình bày về kết quả đạt được, cả về hiệu quả và thời gian thực thi của mô hình.

4.1. Thiết lập thực nghiệm

4.1.1. Môi trường thực nghiệm

Để chạy mô hình liên chuỗi, cần có một môi trường với hiệu suất CPU cao, dung lượng lưu trữ lớn và bộ nhớ đủ rộng. Chính vì thế, nhóm đã thực hiện việc thiết lập một môi trường thích hợp để đáp ứng các yêu cầu như **Bảng 4.1**:

	Blockchain A	SideChain	Blockchain B
CPU	4 cores	4 cores	4 cores
GPU	N/A	N/A	N/A
RAM	8 GB	8 GB	8 GB
Hard Drive	60 GB	60 GB	60 GB
OS	Ubuntu 22.04	Ubuntu 22.04	Ubuntu 22.04

Bảng 4.1: Môi trường thực nghiệm mạng liên chuỗi

Để huấn luyện mô hình học máy, cần một môi trường có khả năng chạy liên tục trong thời gian dài với hiệu suất cao từ CPU và GPU, cùng với dung lượng lưu trữ lớn và bộ nhớ rộng lớn. Do vậy, để đáp ứng những yêu cầu tính toán nghiêm ngặt này, nhóm đã thiết lập môi trường như **Bảng 4.2**:

Thiết bị	Machine 1	Machine 2	Máy Google Collab
CPU	4 cores	4 cores	4 cores
GPU	N/A	N/A	T4
RAM	8 GB	16 GB	12.7 GB
Hard Drive	60 GB	60 GB	166 GB
OS	Ubuntu 22.04	Window 10	Ubuntu 20.04

Bảng 4.2: Môi trường thực nghiệm các phương pháp học máy

4.1.2. Chỉ số đánh giá

Trong nghiên cứu này, em đã chọn 5 chỉ số để đánh giá hiệu suất của mô hình, bao gồm: Accuracy (Độ chính xác), Precision (Mức độ chính xác), Recall (Độ phủ), F1-score (Điểm F1) và Thời gian dự đoán. Để tính toán bốn chỉ số đầu tiên, em sử dụng ma trận confusion (confusion matrix), một công cụ hữu ích trong việc đánh giá các mô hình phân loại. Confusion matrix cho phép xác định nhanh chóng các chỉ số này qua việc phân tích số lượng dữ liệu thực tế thuộc vào một lớp nhưng lại bị dự đoán nhầm lẫn vào lớp khác, dựa trên các khái niệm về True Positive, False Positive, True Negative và False Negative. Điều này giúp hiểu rõ hơn về cách thức mô hình phân loại và đánh giá chất lượng dự đoán của nó:

- True Positive (TP) là những mẫu được dự đoán đúng thuộc lớp positive và thực tế cũng thuộc lớp positive.
- True Negative (TN) là những mẫu được dự đoán đúng thuộc lớp negative và thực tế cũng thuộc lớp negative.
- False Positive (FP) là những mẫu được dự đoán nhầm thuộc lớp positive nhưng thực tế lại thuộc lớp negative.

- False Negative (FN) là những mẫu được dự đoán nhầm thuộc lớp negative nhưng thực tế lại thuộc lớp positive.

Các độ đo đánh giá như Accuracy, Precision, Recall, và F1-Score được xác định và tính toán dựa trên số lượng các mẫu phân loại là True Positive (TP), True Negative (TN), False Positive (FP) và False Negative (FN). Các định nghĩa và công thức của chúng như sau:

- Accuracy là chỉ số đánh giá độ chính xác toàn diện của mô hình. Nó được tính bằng cách lấy tỷ lệ phần trăm của tổng số dự đoán đúng (bao gồm cả true positives và true negatives) so với tổng số quan sát. Khi giá trị Accuracy càng cao, điều đó cho thấy mô hình có khả năng dự đoán càng chính xác.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

- Precision đo lường độ chính xác của những dự đoán dương tính mà mô hình thực hiện. Nó được xác định bằng cách chia số lượng true positives (các trường hợp dự đoán đúng là dương tính) cho tổng số dự đoán dương tính (bao gồm cả true positives và false positives). Giá trị Precision cao chỉ ra rằng mô hình có độ tin cậy lớn khi dự đoán các quan sát là dương tính.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4.2)$$

- Recall (còn được gọi là Sensitivity) cho biết khả năng của mô hình trong việc xác định tất cả các trường hợp dương tính thực sự. Chỉ số này được tính bằng cách chia số lượng true positives (số trường hợp dương tính mà mô hình dự đoán chính xác) cho tổng số trường hợp dương tính thực tế. Khi giá trị Recall càng cao, nó cho thấy mô hình càng có khả năng phát hiện tất cả các trường hợp dương tính mà không bỏ lỡ.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.3)$$

- F1-Score là chỉ số đánh giá sự cân bằng giữa Precision và Recall. Nó được tính bằng công thức lấy trung bình điều hòa của Precision và Recall. F1-Score cung cấp một cái nhìn toàn diện về độ chính xác và độ phủ của mô hình. Một mô hình với F1-Score cao được xem là có hiệu suất tốt, vì nó thể hiện cả hai khía cạnh quan trọng là Precision và Recall đều được cân nhắc và tối ưu hóa.

$$\text{F1-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4.4)$$

Thời gian dự đoán (Inference Time) cũng là một chỉ số quan trọng khác trong việc đánh giá hiệu suất của mô hình máy học. Chỉ số này đo lường thời gian mà mô hình cần để đưa ra dự đoán cho một hoặc một loạt các quan sát mới. Thời gian dự đoán càng ngắn, thì mô hình càng được xem là hiệu quả, điều này rất quan trọng trong các ứng dụng cần phản hồi nhanh chóng, như trong các tình huống yêu cầu thời gian thực. Mô hình có thời gian dự đoán nhanh sẽ trả về kết quả nhanh hơn, cho phép hệ thống đưa ra quyết định hoặc hành động kịp thời. Ngược lại, thời gian dự đoán chậm có thể khiến trải nghiệm người dùng kém hơn. Vì vậy, khi thiết kế và đánh giá mô hình, thời gian dự đoán là một yếu tố then chốt cần xem xét.

4.1.3. Kịch bản thực nghiệm

Trước khi em bước vào các giai đoạn chi tiết của quy trình này, hãy cùng xem xét tổng quan về quy trình mà em đã thiết kế để kiểm tra và phân loại lỗi trong các smart contract trên blockchain. Đầu tiên, em xây dựng một mô hình cầu nối blockchain, tạo ra một môi trường mô phỏng cho việc tạo ra các smart contract. Tiếp theo, em thu thập và gắn nhãn cho các smart contract để tạo ra một tập dữ liệu đủ lớn để huấn luyện và kiểm tra mô hình. Sau đó, em sử dụng các thuật toán học máy để xây dựng các mô hình phân loại lỗi trong smart contract, và quá trình huấn luyện được thực hiện trên tập dữ liệu đã chuẩn bị trước đó. Để đánh giá hiệu suất, em sử dụng các chỉ số như Accuracy, Precision, Recall, và

F1-Score, và cuối cùng, em triển khai mô hình tốt nhất và kiểm tra nó trên giao diện ứng dụng để đảm bảo tính thực tế và hiệu quả trong môi trường thực tế. Hãy cùng đi vào từng bước chi tiết của quá trình này:

1. Xây dựng mô hình cầu nối blockchain: Mục tiêu của giai đoạn này là mô phỏng quá trình sáng tạo các smart contract trên blockchain. em thiết lập một mô hình cầu nối để tái hiện các bước quan trọng trong việc tạo ra smart contract, tạo nên một môi trường mô phỏng cho nghiên cứu tiếp theo.
2. Thu thập và gắn nhãn märk smart contract: Sau khi mô phỏng, em thu thập các smart contract được tạo ra từ mô hình cầu nối. Các smart contract này sau đó được gắn nhãn để tạo thành tập dữ liệu cho quá trình đào tạo và kiểm tra mô hình.
3. Áp dụng thuật toán học máy: Với tập dữ liệu đã có, em áp dụng các thuật toán học máy như máy học và học sâu để xây dựng các mô hình phân loại lỗi trong smart contract. Các thuật toán này được chọn để hiểu và dự đoán các lỗi có thể xuất hiện trong smart contract.
4. Huấn luyện mô hình: Quá trình huấn luyện được thực hiện trên tập dữ liệu đã được chuẩn bị. Mô hình học từ dữ liệu để hiểu các đặc trưng quan trọng và mối quan hệ giữa chúng để có khả năng dự đoán lỗi một cách chính xác.
5. Đánh giá và so sánh mô hình: Sau khi huấn luyện, em đánh giá hiệu suất của các mô hình bằng cách sử dụng các chỉ số như Accuracy, Precision, Recall và F1-Score. Sự so sánh giữa các mô hình giúp em lựa chọn mô hình có hiệu suất tốt nhất.
6. Lựa chọn mô hình tốt nhất: Dựa trên kết quả đánh giá, mô hình có hiệu suất tốt nhất được lựa chọn để tiếp tục triển khai và kiểm tra trong các tình huống thực tế.
7. Triển khai trên giao diện ứng dụng: Mô hình tốt nhất được triển khai trên

giao diện ứng dụng để kiểm tra và xác nhận tính khả thi của nó trong môi trường thực tế.

4.2. Kết quả thực nghiệm

4.2.1. Kết quả về mặt hiệu suất

Bảng kết quả cung cấp một cái nhìn tổng quan về khả năng phát hiện lỗ hổng bảo mật trong các smart contract liên chuỗi của các mô hình. Qua các độ đo này, chúng ta có thể thăm dò cận kề hiệu quả của từng mô hình. **Bảng 4.3** thể hiện kết quả hiệu suất. RoBERTa nổi bật như mô hình dẫn đầu về khả năng phân loại lỗ hổng trong smart contract, với độ chính xác cao nhất đạt 96.67% so với các mô hình khác. Điểm nổi bật của RoBERTa so với các mô hình khác là độ đo và F1 Score lần lượt đạt 100% và 93.33% - cao nhất trong tất cả các mô hình, cho thấy sự cân bằng tốt giữa độ chính xác và độ nhạy.

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.8519	0.8209	0.8730	0.8462
Random Forest	0.7312	0.6324	1.0000	0.7748
XGBoost	0.6211	0.5485	0.9924	0.7065
SVM	0.8458	0.7551	0.9911	0.8571
Logistic Regression	0.9000	0.9071	0.8864	0.8967
CNN	0.9000	0.8235	1.0000	0.9032
LSTM	0.5500	0.5500	1.0000	0.7100
FNN	0.8125	0.8636	0.7037	0.7755
RoBERTa	0.9667	1.0000	0.8750	0.9333

Bảng 4.3: Hiệu suất của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

4.2.2. Kết quả về mặt thời gian

Quá trình thực hiện của các mô hình cho thấy sự biến thiên đáng kể về mặt thời gian. **Bảng 4.4** trình bày kết quả thời gian thực thi của chúng. Mô hình như Decision Tree, Random Forest và SVM chỉ mất khoảng 3.3 - 3.7 giây để thực hiện, phản ánh sự đơn giản trong cấu trúc của chúng, đặc biệt là Decision Tree và Random Forest.

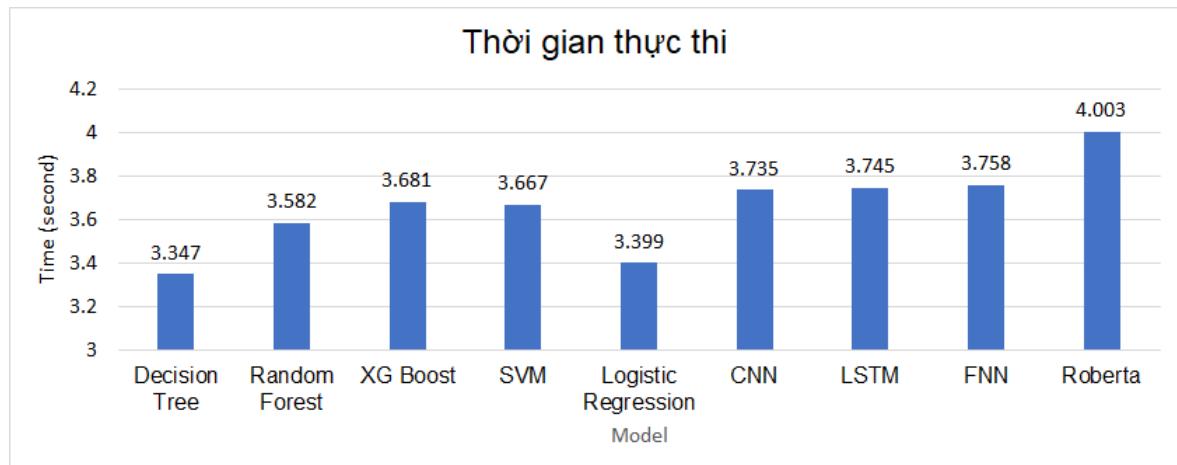
Mặt khác, mô hình RoBERTa mất nhiều thời gian nhất để hoàn thành, đạt 4.003 giây. Sự phức tạp của kiến trúc transformer có thể là nguyên nhân, nhưng thời gian này vẫn nằm trong phạm vi chấp nhận được, đặc biệt khi xét đến khả năng dự đoán xuất sắc của mô hình.

Model	Time Performance (seconds)
Decision Tree	3.347
Random Forest	3.582
XGBoost	3.681
SVM	3.667
Logistic Regression	3.399
CNN	3.735
LSTM	3.745
FNN	3.758
RoBERTa	4.003

Bảng 4.4: Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

Việc đánh giá thời gian thực hiện của mô hình là quan trọng, đặc biệt khi cân nhắc triển khai mô hình trong các ứng dụng cần phản ứng nhanh hoặc đào tạo liên tục. Trong trường hợp các ứng dụng cần thời gian phản hồi nhanh,

việc chọn mô hình như Decision Tree hoặc SVM có thể phù hợp, trong khi đó, RoBERTa vẫn là sự lựa chọn mạnh mẽ cho các nhiệm vụ đòi hỏi độ chính xác cao, mặc dù có thời gian thực hiện lớn hơn. **Hình 4.1** minh họa sự khác biệt về thời gian thực thi giữa các mô hình.



Hình 4.1: Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

4.3. Thảo luận

Hệ thống được đánh giá về khả năng phát hiện bảo mật các lỗ hổng và hành vi độc hại trong các hợp đồng thông minh xuyên chuỗi. Các mô hình học máy, đặc biệt là Roberta, có hiệu quả rất cao trong việc phân loại và xác định lỗ hổng, với độ chính xác vượt 96%. Điều này cho thấy khả năng của hệ thống trong việc diễn giải ngữ nghĩa hợp đồng để học các mô hình dự đoán chính xác phần mã độc hại.

Khả năng phát hiện bảo mật xuất phát từ cách tiếp cận độc đáo trong việc tạo tập dữ liệu xuyên chuỗi của hệ thống. Theo đó, dữ liệu hợp đồng sẽ được biến đổi thành các đặc trưng mạnh mẽ. Tổng thể, kết quả khẳng định độ bảo mật của hệ thống cho các mạng blockchain liên kết thông qua các mô hình học máy được huấn luyện trên tập dữ liệu xuyên chuỗi mới.

Việc tích hợp ChainSniper vào hệ thống cũng tăng cường thêm lớp bảo vệ

through qua khả năng phân tích động các phụ thuộc và tương tác giữa các hợp đồng xuyên blockchain. Bằng cách giám sát theo thời gian thực, ChainSniper có thể xác định các dấu hiệu của nỗ lực khai thác lõi hồng. Sự kết hợp này cho phép giám sát bảo mật chủ động trên các mạng blockchain liên kết.

Về mặt thời gian giao động, các mô hình học máy đạt mức 3.3 - 3.7 giây. Trong khi đó các mô hình học sâu cho thời gian 3.7 - 4..0 giây. Cao nhất là mô hình RoBERTa với 4.003 giây. Mặc dù vậy, xét về yếu tố như độ chính xác đều rất cao và có khả năng ứng dụng thực tế.

CHƯƠNG 5. KẾT LUẬN

Trong phần kết luận này, em sẽ tổng kết những điểm chính của công trình nghiên cứu và đề cập đến các hướng phát triển tiềm năng trong tương lai cho mô hình.

5.1. Kết luận

Nghiên cứu này mang lại một bước tiến quan trọng trong việc cải thiện bảo mật cho hệ sinh thái hợp đồng thông minh liên kết chuỗi. Cụ thể, nhóm nghiên cứu đã phát triển một mô hình mới có tên ChainSniper, dựa trên công nghệ sidechain. Hơn nữa, họ cũng tạo ra một tập dữ liệu đặc biệt, CrossChainSentinel, chứa 300 mẫu mã nguồn hợp đồng thông minh đã được gán nhãn. Sử dụng tập dữ liệu này, ChainSniper được thiết kế để kết hợp nhiều mô hình máy học tiên tiến, bao gồm cả các phân loại học sâu, nhằm phát hiện các hợp đồng thông minh độc hại trong môi trường blockchain liên kết. Qua các thử nghiệm và đánh giá, mô hình này cho thấy độ chính xác ấn tượng lên tới 96% trong việc phát hiện lỗ hổng hợp đồng thông minh trên bộ dữ liệu mới. Kết quả này là minh chứng cho tiềm năng lớn của việc kết hợp công nghệ sidechain và máy học trong việc tăng cường bảo mật cho hệ thống hợp đồng thông minh liên kết chuỗi. Nghiên cứu này không chỉ cung cấp một mô hình hiệu quả với độ chính xác cao mà còn mở ra hướng tiếp cận mới để cải thiện an toàn cho hợp đồng thông minh hoạt động trong môi trường đa blockchain. Như vậy, nghiên cứu đã định hình cơ hội và hướng phát triển mới trong việc nâng cao tính bảo mật cho hệ thống hợp đồng thông minh, góp phần thúc đẩy sự phát triển của công nghệ blockchain trong tương lai.

5.2. Hướng phát triển

Nghiên cứu này mở ra nhiều hướng phát triển tiềm năng cho mô hình Chain-Sniper. Một trong những khả năng chính là việc mở rộng bộ dữ liệu CrossChain-Sentinel, bao gồm cả việc bổ sung thêm các mẫu hợp đồng thông minh lành tính và độc hại. Việc tăng cường bộ dữ liệu sẽ cải thiện khả năng tổng quát hóa và độ chính xác của mô hình học máy. Ngoài ra, khám phá và tích hợp các kiến trúc máy học tiên tiến như mô hình Transformer và áp dụng kiến thức từ lĩnh vực xử lý ngôn ngữ tự nhiên có thể giúp cải thiện khả năng hiểu ngữ nghĩa của mã smart contract, tăng cường chất lượng trong việc phát hiện lỗ hổng. Sự kết hợp này sẽ khai thác hiệu quả ngữ liệu để phân tích mã máy, mở ra một phương thức mới trong việc đảm bảo an ninh cho hợp đồng thông minh. Mục tiêu dài hạn là liên tục cải thiện ChainSniper để nó có thể tự động phát hiện điểm yếu trong hợp đồng thông minh trước khi chúng được triển khai, giúp thực hiện kiểm định chất lượng, nâng cao độ tin cậy và an toàn cho các ứng dụng phi tập trung trên nhiều nền tảng blockchain khác nhau. Tóm lại, sự kết hợp giữa việc mở rộng bộ dữ liệu, tối ưu hóa thuật toán và nâng cao các tính năng sẽ giúp mô hình ChainSniper ngày càng hoàn thiện, trở thành một giải pháp quan trọng trong việc kiểm tra toàn diện hợp đồng thông minh, giảm thiểu rủi ro bảo mật cho các hệ thống phức tạp liên kết chuỗi.

TÀI LIỆU THAM KHẢO

Tài liệu tham khảo

- [1] P. Patel and H. Patel, “Achieving a secure cloud storage mechanism using blockchain technology,” p. 130–142, 2023. [Online]. Available: <http://dx.doi.org/10.7763/IJCTE.2023.V15.1342>
- [2] M. Kumarathunga, R. N. Calheiros, and A. Ginige, “Sustainable microfinance outreach for farmers with blockchain cryptocurrency and smart contracts,” p. 9–14, 2022. [Online]. Available: <http://dx.doi.org/10.7763/IJCTE.2022.V14.1304>
- [3] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, “zkbridge: Trustless cross-chain bridges made practical,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.
- [4] P. Han, Z. Yan, W. Ding, S. Fei, and Z. Wan, “A survey on cross-chain technologies,” *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 2, pp. 1–30, 2023.
- [5] Y. Hei, D. Li, C. Zhang, J. Liu, Y. Liu, and Q. Wu, “Practical agentchain: A compatible cross-chain exchange system,” *Future Generation Computer Systems*, vol. 130, pp. 207–218, 2022.
- [6] R. Lan, G. Upadhyaya, S. Tse, and M. Zamani, “Horizon: A gas-efficient, trustless bridge for cross-chain transactions,” *arXiv preprint arXiv:2101.06000*, 2021.

- [7] K. Qin and A. Gervais, “An overview of blockchain scalability, interoperability and sustainability,” *Hochschule Luzern Imperial College London Liquidity Network*, pp. 1–15, 2018.
- [8] T. Hardjono, “Blockchain gateways, bridges and delegated hash-locks,” *arXiv preprint arXiv:2102.03933*, 2021.
- [9] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghanianha, and K.-K. R. Choo, “Sidechain technologies in blockchain networks: An examination and state-of-the-art review,” *Journal of Network and Computer Applications*, vol. 149, p. 102471, 2020.
- [10] M. H. Miraz and D. C. Donald, “Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities,” *arXiv preprint arXiv:1902.04471*, 2019.
- [11] J. Zhang, J. Gao, Y. Li, Z. Chen, Z. Guan, and Z. Chen, “Xscope: Hunting for cross-chain bridge attacks,” in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–4.
- [12] S. Sayeed, H. Marco-Gisbert, and T. Caira, “Smart contract: Attacks and protections,” *IEEE Access*, vol. 8, pp. 24 416–24 427, 2020.
- [13] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings 6*. Springer, 2017, pp. 164–186.
- [14] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, “An overview of smart contract: architecture, applications, and future trends,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 108–113.

- [15] T. H.-D. Huang, “Hunting the ethereum smart contract: Color-inspired inspection of potential attacks,” *arXiv preprint arXiv:1807.01868*, 2018.
- [16] L. Zhang, W. Chen, W. Wang, Z. Jin, C. Zhao, Z. Cai, and H. Chen, “Cb-gru: A detection method of smart contract vulnerability based on a hybrid model,” *Sensors*, vol. 22, no. 9, p. 3577, 2022.
- [17] E. Lai and W. Luo, “Static analysis of integer overflow of smart contracts in ethereum,” in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, 2020, pp. 110–115.
- [18] M. Staderini, C. Palli, and A. Bondavalli, “Classification of ethereum vulnerabilities and their propagations,” in *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*. IEEE, 2020, pp. 44–51.
- [19] H. Mao, T. Nie, H. Sun, D. Shen, and G. Yu, “A survey on cross-chain technology: Challenges, development, and prospect,” *IEEE Access*, 2022.
- [20] M. Rodler, W. Li, G. O. Karame, and L. Davi, “Sereum: Protecting existing smart contracts against re-entrancy attacks,” *arXiv preprint arXiv:1812.05934*, 2018.
- [21] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, “Security analysis methods on ethereum smart contract vulnerabilities: a survey,” *arXiv preprint arXiv:1908.08605*, 2019.
- [22] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, “Systematic review of security vulnerabilities in ethereum blockchain smart contract,” *IEEE Access*, vol. 10, pp. 6605–6621, 2022.
- [23] J.-W. Liao, T.-T. Tsai, C.-K. He, and C.-W. Tien, “Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing,” in

2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). IEEE, 2019, pp. 458–465.

- [24] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, “An introduction to machine learning,” *Clinical pharmacology & therapeutics*, vol. 107, no. 4, pp. 871–885, 2020.
- [25] M. Krichen, “Strengthening the security of smart contracts through the power of artificial intelligence,” *Computers*, vol. 12, no. 5, p. 107, 2023.
- [26] Y. Xu, G. Hu, L. You, and C. Cao, “A novel machine learning-based analysis model for smart contract vulnerability,” *Security and Communication Networks*, vol. 2021, pp. 1–12, 2021.
- [27] T. Haugum, B. Hoff, M. Alsadi, and J. Li, “Security and privacy challenges in blockchain interoperability-a multivocal literature review,” in *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, 2022, pp. 347–356.
- [28] W. Deng, H. Wei, T. Huang, C. Cao, Y. Peng, and X. Hu, “Smart contract vulnerability detection based on deep learning and multimodal decision fusion,” *Sensors*, vol. 23, no. 16, p. 7246, 2023.
- [29] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, “Smart contract vulnerability analysis and security audit,” *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.
- [30] J. Huang, K. Zhou, A. Xiong, and D. Li, “Smart contract vulnerability detection model based on multi-task learning,” *Sensors*, vol. 22, no. 5, p. 1829, 2022.

- [31] B. Jiang, Y. Liu, and W. K. Chan, “Contractfuzzer: Fuzzing smart contracts for vulnerability detection,” in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 259–269.
- [32] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, “Empirical vulnerability analysis of automated smart contracts security testing on blockchains,” *arXiv preprint arXiv:1809.02702*, 2018.