Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM
Khoa Mạng máy tính và truyền thông

BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

Môn học: CƠ CHẾ HOẠT ĐỘNG CỦA MÃ ĐỘC

Tên chủ đề: HYDRA: A multimodal deep learning framework for malware classification

Mã nhóm: G01 Mã đề tài: S20

Lớp: NT230.N21.ANTT

1. THÔNG TIN THÀNH VIÊN NHÓM:

(Sinh viên liệt kê tất cả các thành viên trong nhóm)

STT	Họ và tên	MSSV	Email
1	Đỗ Minh Thọ	20521972	20521972@gm.uit.edu.vn
2	Trần Xuân Nam	20521637	20521637@gm.uit.edu.vn
3	Nguyễn Phúc Hải	20521281	20521281@gm.uit.edu.vn

2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

A. Chủ đề nghiên cứu trong lĩnh vực Mã độc: *(chọn nội dung tương ứng bên dưới)*

☐ Phát hiện mã độc

☐ Đột biến mã độc

☑Khác: Phát hiện mã độc và phân loại chúng

B. Liên kết lưu trữ mã nguồn của nhóm:

Mã nguồn của đề tài đồ án được lưu tại:

- Link dirve chứa code demo, bản báo cáo nhóm đã viết trước khi viết Final báo cáo theo mẫu và slide thuyết trình: https://drive.google.com/drive/u/1/folders/1OsUYI9GGGDCagb-OYxA0cusvfdUbFd1w

(Lưu ý: GV phụ trách phải có quyền truy cập nội dung trong Link)

C. Tên bài báo tham khảo chính:

HYDRA: A multimodal deep learning framework for malware classification

D. Dịch tên Tiếng Việt cho bài báo:

[1] Ghi nội dung tương ứng theo mô tả

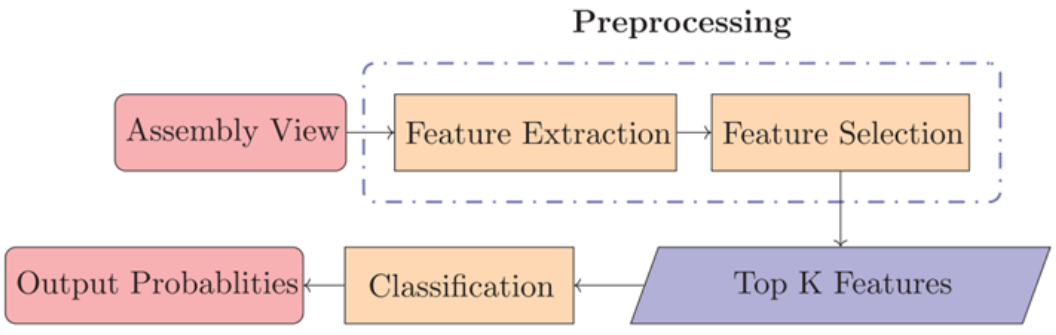
HYDRA: Một khung công cụ học sâu đa phương tiện cho phân loại mã độc.

E. Tóm tắt nội dung chính:

Bài báo giới thiệu một hệ thống phân loại phần mềm độc hại đa dạng sử dụng kỹ thuật học sâu đa phương tiện. Hệ thống này sử dụng các đặc trưng đa dạng. HYDRA sử dụng một mô hình mạng học sâu đa phương tiện để học và phân loại các loại phần mềm độc hại. Bài báo đề cập tới các phương thức sử dụng là API function call, Mnemonic và opcode-byte. Một số thử nghiệm đã được thực hiện để đánh giá hiệu quả của HYDRA, kết quả cho thấy rằng hệ thống này vượt qua các phương pháp phân loại phần mềm độc hại truyền thống và đưa ra các kết quả phân loại chính xác cao hơn. Cuối bài báo thì nhóm tác giả còn cung cấp một so sánh với các phương pháp tốt nhất hiện nay trong tài liệu, bao gồm gradient boosting và các phương pháp học sâu .HYDRA có thể được sử dụng để giúp các chuyên gia bảo mật phát hiện và ngăn chặn các mối đe dọa của phần mềm độc hại.

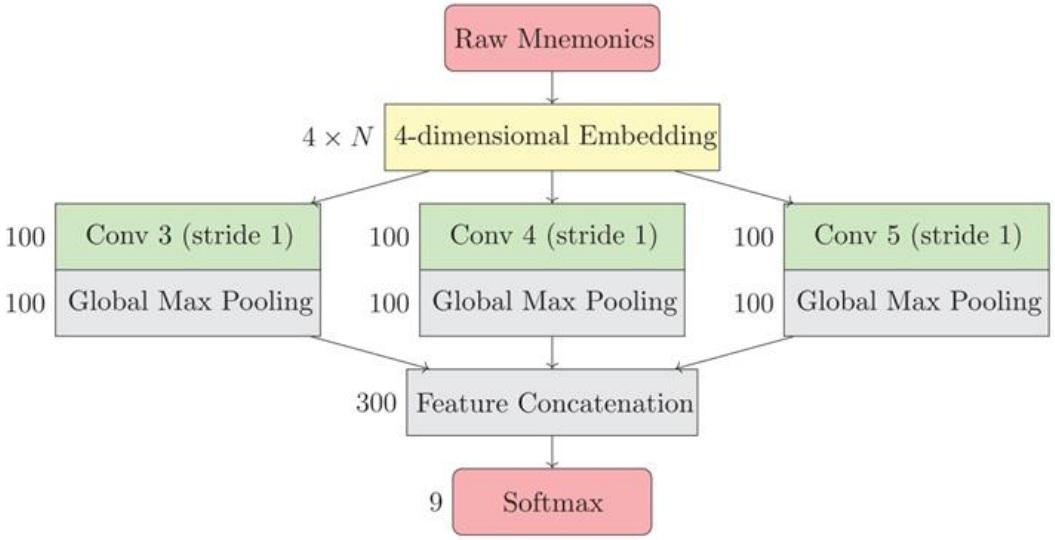
F. Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo:

- Bài báo có đề cập tới các phương thức API function call, Mnemonic và opcode-byte.
- Kĩ thuật API windows call: Kĩ thuật Windows API call là việc sử dụng các hàm của hệ điều hành Windows để thực hiện các tác vụ cụ thể. Bằng cách gọi các hàm API từ các thư viện, ta có thể tiếp cận và sử dụng các tính năng của hệ thống như tệp tin, giao diện người dùng, mạng và nhiều tính năng khác. Kĩ thuật này được sử dụng trong lập trình Windows để tạo ứng dụng đa nền tảng và tương tác với hệ thống



Hình 3: Luồng dữ liệu truyền thống của một hệ thống phân loại phần mềm độc hại dựa trên API.

- Phân tích Mnemonics: trong đó sử dụng Mạng nơ-ron tích chập như một lựa chọn thay thế cho N-gram:

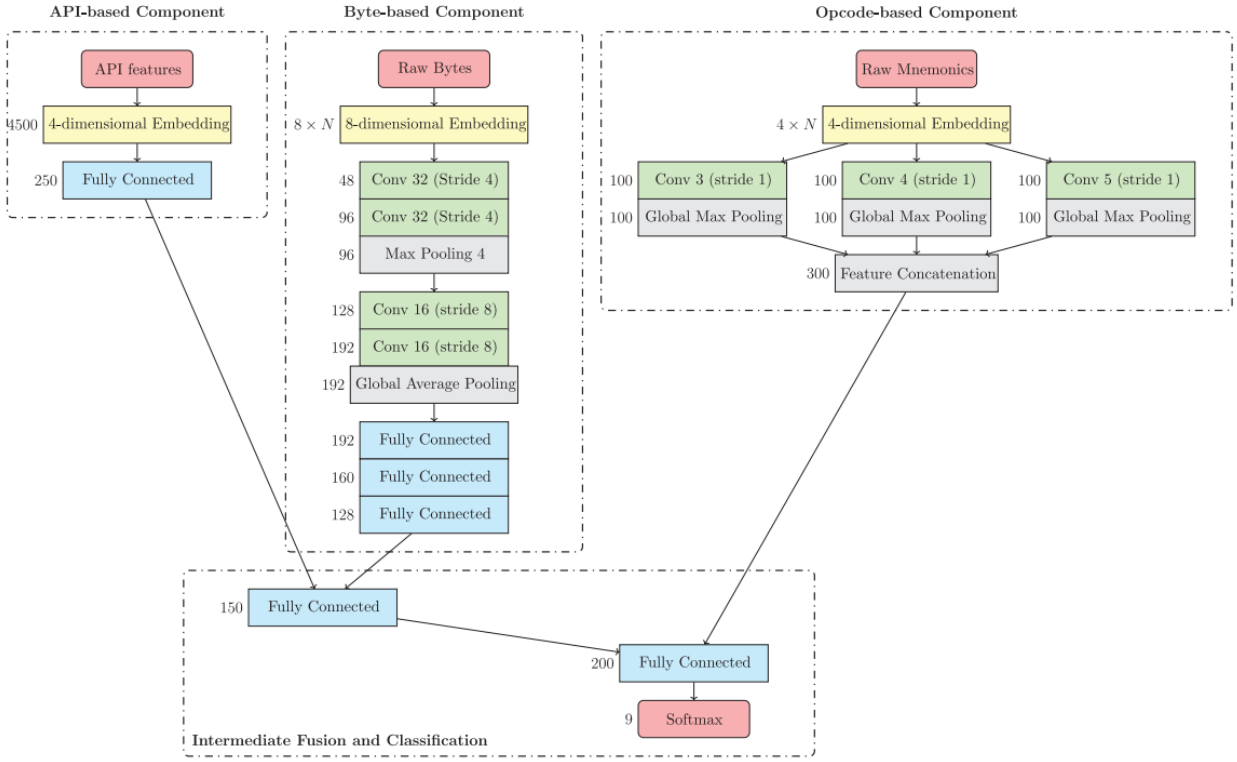


Hình 5: Mạng nơ-ron tích chập để phân loại phần mềm độc hại từ các chuỗi ký tự nhớ.

- Chức năng các lớp: Mạng neural nhận đầu vào là một file thực thi và chuyển đổi mỗi lệnh hợp ngữ thành một vector one-hot. Lớp embedding chập các bộ lọc khác nhau qua các chuỗi mnemonics và trích xuất các tính năng giống như n-gram. Lớp global max-pooling tổng hợp tối đa toàn cầu được áp dụng để trích xuất các tính năng. Lớp softmax đưa ra phân phối xác suất trên các dòng phần mềm độc hại.

- Kiến trúc của HYDRA:
  - Kiến trúc HYDRA được xây dựng trên các mạng neural đa tác vụ (multi-task neural networks) để tích hợp nhiều dữ liệu đầu vào khác nhau. Gồm 3 phần chính:
  - Input module: Các dữ liệu đầu vào khác nhau được xử lý bởi các module đầu vào khác nhau và được kết hợp lại thành một biểu diễn đa phương tiện của malware. Các module đầu vào bao gồm: mô-đun hình ảnh, mô-đun mã nhị phân và mô-đun chuỗi ký tự.
  - Multimodal feature extractor: Biểu diễn đa phương tiện của malware được đưa vào một mô hình trích xuất đặc trưng đa phương tiện để tạo ra các đặc trưng đại diện cho malware.
  - Classifier: Các đặc trưng được trích xuất từ bước trước đó được đưa vào một mô hình phân loại để đưa ra dự đoán về loại malware. Mô hình phân loại được xây dựng bằng cách sử dụng mạng neural đa lớp (Multi-layer neural network) với lớp kết nối đầy đủ (Fully connected layer).
- Bài báo sử dụng kĩ thuật HYDRA Multimodal Deep Learning: Nó bao gồm 4 thành phần chính: thành phần dựa trên API, thành phần dựa trên mnemonics, thành phần dựa trên byte và thành phần hợp nhất đặc trưng và phân loại. Mỗi

thành phần trích xuất đặc trưng từ một biểu diễn trừu tượng khác nhau của mã độc. Thành phần cuối cùng kết hợp các đặc trưng học được bởi mỗi thành phần vào một biểu diễn chia sẻ và cho ra các dự đoán phân loại



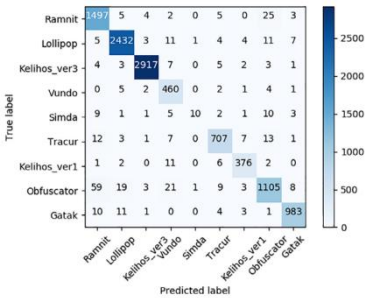
- Đánh giá hiệu xuất thành phần dựa trên API, Mnemonic, byte, hiệu quả của mô hình Multimodal Deep Learning, so sánh với state-of-the-art.

G. Môi trường thực nghiệm của bài báo:

- Cấu hình máy tính như sau : một máy tính với bộ vi xử lý Intel Core i7-7700k, 4 GPU Geforce GTX 1080Ti và RAM 64 Gb
- Ngôn ngữ lập trình Python và các thư viện học máy
- Đối tượng nghiên cứu: Tập dữ liệu của Microsoft 2015
- Tiêu chí đánh giá tính hiệu quả của phương pháp: Lọc và phân loại dữ liệu thu thập được, sau đó đưa vào cấu trúc HYDRA để đưa ra những thông số và so sánh với các thông số của các phương pháp phân loại khác

H. Kết quả thực nghiệm của bài báo:

- Bài báo đánh giá các mô hình trên bao gồm các phương thức dựa trên API. Byte, Mnemonic và hiệu quả của mô hình Multimodal Deep Learning.
- Ba kiến trúc API, byte bà Mnemonic
  - Đánh giá mô hình byte:
    - Trong phần này, tác giả so sánh hiệu suất của mô hình MalConv và DeepConv trên bộ dữ liệu Microsoft Malware Classification Challenge (Ronen et al., 2018). Hình 13 và 14 hiển thị ma trận nhầm lẫn được báo cáo từ cross-validation 10 lần, với macro F1-score lần lượt là 0.8902



Hình 13: Ma trận MalConv confusion.

và 0.9071 cho các mô hình MalConv và DeepConv (Xem bảng 7). Do đó, thành phần dựa trên byte của hệ thống đa dạng sẽ bao gồm kiến trúc DeepConv, do hiệu suất vượt trội hơn của nó.

Approaches	Accuracy	Macro F1-score
Random guess	0.1755	–
Zero rule	0.2707	–
MalConv	0.9641	0.8902
DeepConv	0.9756	0.9071

Bảng 7: So sánh các phương pháp dựa trên Bytes.

- Đánh giá mô hình API
  - Kết quả đánh giá cho thấy mô hình API đạt được độ chính xác 97,8% và 97,9% trên tập huấn luyện và tập kiểm tra, tương ứng với tỷ lệ phát hiện và macro F1-score lần lượt là 98,6% và 96,2%. Điều này chứng tỏ mô hình API đạt được kết quả tốt trong việc phân loại phần mềm độc hại trên tập dữ liệu Microsoft Malware Classification Challenge.

Algorithm	Accuracy	Macro F1-score
Random guess	0.1755	–
Zero rule	0.2707	–
Logistic regression	0.9810	0.9573
NN 4500 × 30	0.9824	0.9566
NN 4500 × 50	0.9826	0.9570
NN 4500 × 100	0.9829	0.9602
NN 4500 × 250	**0.9833**	**0.9621**
NN 4500 × 500	0.9829	0.9617
NN 4500 × 2000	0.9828	0.9602
NN 4500 × 2500	0.9825	0.9603
NN 4500 × 30 × 20	0.9823	0.9612
NN 4500 × 50 × 20	0.9822	0.9564
NN 4500 × 50 × 30	0.9824	0.9580
NN 4500 × 1000 × 100	0.9820	0.9603

Bảng 5: Tìm kiếm lưới mạng nơ-ron Feed-forward. Kiến trúc của một mạng nơ-ron Feed-forward được định nghĩa như sau: NN F × H1 × H2, trong đó F là kích thước của vector đặc trưng đầu vào, H1 và H2 là số lượng nơ-ron trong các lớp ẩn thứ nhất và thứ hai, tương ứng.

- Đánh giá mô hình Mnemonic.
  - Mô hình Mnemonic sử dụng một mạng neural tích chập để trích xuất đặc trưng từ mã opcode và một mạng neural dựa trên LSTM để trích xuất đặc trưng từ hình ảnh. Sau đó, các đặc trưng được kết hợp và đưa vào một mạng neural để phân loại phần mềm độc hại.Kết quả đánh giá cho thấy mô hình Mnemonic đạt được độ chính xác 98,4% và 98,8% trên tập huấn luyện và tập kiểm tra, tương ứng với tỷ lệ phát hiện và macro F1-score lần lượt là 98,3% và 98,2%. Điều này chứng tỏ mô hình Mnemonic đạt được kết quả tốt trong việc phân loại phần mềm độc hại trên tập dữ liệu Microsoft Malware Classification Challenge.

Approach	Accuracy	Macro F1-score
Random guess	0.1755	–
Zero rule	0.2707	–
CNN-rand	0.9917	0.9856
CNN-skipgram	0.9899	0.9770
CNN-cbow	0.9886	0.9717

Bảng 6: So sánh các phương pháp CNN dựa trên Opcode.

- Kết quả của mô hình Multimodal Deep Learning
  - Tác giả chứng minh tính hiệu quả của HYDRA bằng cách so sánh với các mô hình được đào tạo trên từng phương thức độc lập. Hình 15 cho thấy các độ chính xác được báo cáo chung của quá trình xác nhận chéo 10 lần để ước tính hiệu suất của HYDRA. Kết quả của tất cả các mô hình được hiển thị trong Bảng 8. Chúng tôi có thể quan sát thấy rằng độ chính xác toàn bộ của HYDRA là 0,9975 và macro F1-score là 0,9954, vượt trội hơn so với các mô hình chỉ tập trung vào một phương thức cụ thể. Việc tiền đào tạo từng phương thức và dropout đa phương thức đã là quan trọng đối với sự thành công của HYDRA. Một mặt, việc tiền đào tạo từng phương thức tránh việc overfitting một tập con của các tính năng thuộc về một phương thức và underfitting các tính năng thuộc về các phương thức khác. Mặt khác, dropout đa phương thức ngăn chặn sự đồng thời tối ưu của các mạng con cho một loại tính năng hoặc phương thức dữ liệu cụ thể.

Model	Accuracy	Macro F1-score
API-based Feedforward Network	0.9833	0.9621
Assembly-based Shallow CNN	0.9917	0.9856
Bytes-based DeepConv	0.9756	0.8902
HYDRA	0.9871	0.9695
HYDRA (Pretraining, Modality Dropout)	**0.9975**	**0.9954**

Bảng 8: So sánh độ chính xác và macro F1-score của mô hình dựa trên phương pháp và mô hình HYDRA với kiểm định chéo 10 lần.

- So sánh với state-of the-art: Bài báo đánh giá hiệu suất của phương pháp đa dạng hóa HYDRA trong việc phân loại phần mềm độc hại trên tập dữ liệu được cung cấp cho Cuộc thi Phân loại Phần mềm độc hại Microsoft của Kaggle. Bài báo so sánh HYDRA với các phương pháp tiên tiến trong văn bản đã được đề xuất trong các nghiên cứu trước đó. Các phương pháp trong văn bản được chia thành các nhóm khác nhau tùy thuộc vào loại tính năng được sử dụng làm đầu vào. Bài báo chứng minh rằng HYDRA đạt được kết quả tốt hơn hoặc tương đương với các phương pháp trong văn bản khác với ít modalities đầu vào hơn và đạt được tỷ lệ phát hiện và macro F1-score cao hơn. Bài báo cũng nhấn mạnh rằng hệ thống học end-to-end có thể được bổ sung thành công bằng các đặc trưng được thiết kế bằng tay để đạt được kết quả tốt nhất trong nhiệm vụ phân loại malware, trong khi kiến thức đặc thù vẫn là cách tiếp cận thông thường để xây dựng hệ thống.

Approach	Feature Type	Classification Algorithm	Accuracy	Macro F1
Random guess	–	–	0.1755	–
Zero rule	–	–	0.2707	–
Grayscale image				
Gibert et al. (2018c)	128 × 128 Grayscale Image	CNN	0.9750	0.9400
Ahmadi et al. (2016)	Haralick features	XGBoost	0.9690	0.9282
Ahmadi et al. (2016)	Local Binary Pattern features	XGBoost	0.9724	0.9530
Narayanan et al. (2016)	PCA features	1-NN	0.9660	0.9102
Entropy				
Gibert et al. (2018b)	Structural Entropy	Dynamic Time Warping +K-NN	0.9894	0.9813
Gibert et al. (2018b)	Structural Entropy	CNN	0.9828	0.9636
Ahmadi et al. (2016)	Entropy Statistical Measures	XGBoost	0.9900	0.9766
Sequence of opcodes				
Ahmadi et al. (2016)	1-Gram	XGBoost	0.9929	0.9906
McLaughlin et al. (2017)	Opcode sequence	CNN	0.9903	0.9743
Gibert et al. (2019)	Opcode sequence	Hierarchical CNN	0.9913	0.9830
OPCODE-based component	Opcode sequence	CNN	0.9917	0.9856
Sequence of bytes				
Ahmadi et al. (2016)	1-Gram	XGBoost	0.9850	0.9678
Raff et al. (2018)	Bytes sequence	MalConv	0.9641	0.8894
BYTE-based component (Krčál et al., 2018)	Bytes sequence	DeepConv	0.9756	0.9089
Le et al. (2018)*	Scaled bytes sequence	CNN	0.9647	0.9341
Le et al. (2018)*	Scaled bytes sequence	CNN+Unidirectional LSTM	0.9800	0.9577
Le et al. (2018)*	Scaled bytes sequence	CNN+Bidirectional LSTM	0.9814	0.9662
Gibert et al. (2018a)	Bytes sequence	Denoising Autoencoder + Dilated Residual Network	0.9861	0.9719
Yousefi-Azar et al. (2017)	1-Gram	Autoencoder + XGBoost	0.9309	0.8664
API invocations				
Ahmadi et al. (2016)	API feature vector (796)	XGBoost	0.9868	0.9638
API-based component	API feature vector (4500)	Feed-forward network	0.9833	0.9621
Multiple features				
Zhang et al. (2016)	Total lines of each Section, Operation Code Count, API Usage, Special Symbols Count, Asm File Pixel Intensity Feature, Bytes File Block Size Distribution, Bytes File N-Gram	Ensemble Learning (XGBoost)	0.9974	0.9938
Ahmadi et al. (2016)	ENT, Bytes 1-G, STR, IMG1, IMG2, MD1, MISC, OPC, SEC, REG, DP, API, SYM, MD2	Ensemble Learning (XGBoost)	0.9976	0.9931
Mays et al. (2017)	IMG and Opcode N-Grams	Ensemble Learning (CNN and NN)	0.9724	0.9618
HYDRA	**APIs, Bytes sequence, Opcode sequence**	**Multimodal Deep Neural Network**	**0.9975**	**0.9951**

Bảng 9: Độ chính xác của phương pháp được đánh giá bằng phương pháp đánh giá chéo 10 lần trên thử thách phân loại phần mềm độc hại của Microsoft.

I. Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:

- Để demo được các thực nghiệm của bài báo thì cần phải đáp ứng như sau:
  - Tài nguyên thô .Dữ liệu đầu phải lớn vào để train cho Deep Learning và Machine Learning. Sử dụng các kĩ thuật học máy và học sâu để tính toán và phân loại dữ liệu đầu vào.
- Triển khai lại được cấu trúc HYDRA - Multimodal Deep Learning Framework
  - Đã triển khai được mô hình deep learning của kiến trúc HYDRA, đọc dữ liệu từ file byte và asm vào TFrecord và phân loại dữ liệu thô đầu vào.
  - Chưa làm được: sau khi đọc được dữ liệu file vào Tfrecord thì tới đưa dữ liệu này vào mô hình HYDRA thì báo lỗi chưa fix được(sẽ giải thích chi tiết ở phần triển khai)

J. Các khó khăn, thách thức hiện tại khi thực hiện:

- Dữ liệu đầu vào khá lớn, cần có tài nguyên đủ mạnh để cấp cho việc Deep Learning và Machine Learning.
- Cần tài nguyên phần cứng yêu cầu khá lớn (Ram 64 gb và phải có GPU vì nó là yếu tố quan trọng để tăng tốc độ tính toán của thuật toán mạng nơ-ron đa phương tiện)

- Không quá bao quát, bài báo chỉ đưa ra những thông số chung chung

3. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:

80.9999%

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Dịch và tìm hiểu bài báo – Tóm tắt nội dung bài báo	Trần Xuân Nam
2	Thuyết trình (giữa và cuối kì) – Làm và sửa Slide thuyết trình – Viết báo cáo đồ án	Nguyễn Phúc Hải
3	Thực hiện Demo – Viết báo cáo tổng kết đồ án	Đỗ Minh Thọ

BÁO CÁO TỔNG KẾT CHI TIẾT

Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.

Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)

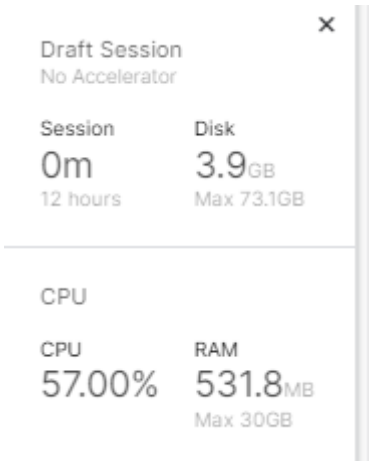
A. Phương pháp thực hiện

- Kiến trúc HYDRA:Kiến trúc HYDRA được xây dựng trên các mạng neural đa tác vụ (multi-task neural networks) để tích hợp nhiều dữ liệu đầu vào khác nhau. Gồm 3 phần chính:
    - ***Input module:*** Các dữ liệu đầu vào khác nhau được xử lý bởi các module đầu vào khác nhau và được kết hợp lại thành một biểu diễn đa phương tiện của malware. Các module đầu vào bao gồm: mô-đun hình ảnh, mô-đun mã nhị phân và mô-đun chuỗi ký tự.
    - ***Multimodal feature extractor***: Biểu diễn đa phương tiện của malware được đưa vào một mô hình trích xuất đặc trưng đa phương tiện để tạo ra các đặc trưng đại diện cho malware.
    - ***Classifier:*** Các đặc trưng được trích xuất từ bước trước đó được đưa vào một mô hình phân loại để đưa ra dự đoán về loại malware. Mô hình phân loại được xây dựng bằng cách sử dụng mạng neural đa lớp (Multi-layer neural network) với lớp kết nối đầy đủ (Fully connected layer).
- Mô hình HYDRA trong bài báo đã được nêu ở phía trên.

B. Chi tiết cài đặt, hiện thực

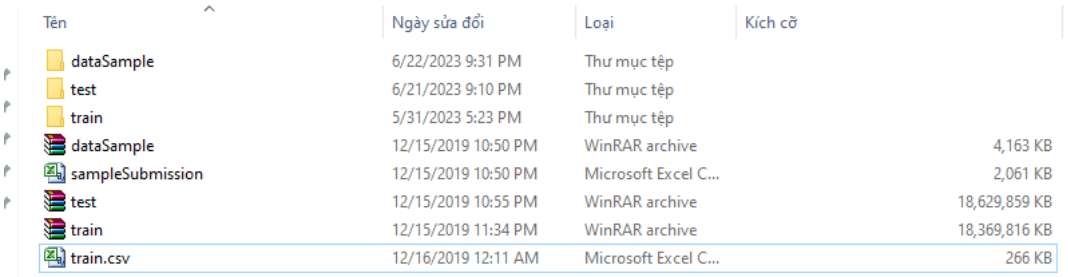
<cách cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu, v.v>

- Cách triển khai:
  - Cấu hình máy tính sử dụng triển khai mô hình:
    - Đối với việc phân loại dataset: Sử dụng kaggle của Microsoft(ánh xạ data). Và triển khai phân loại trên đây

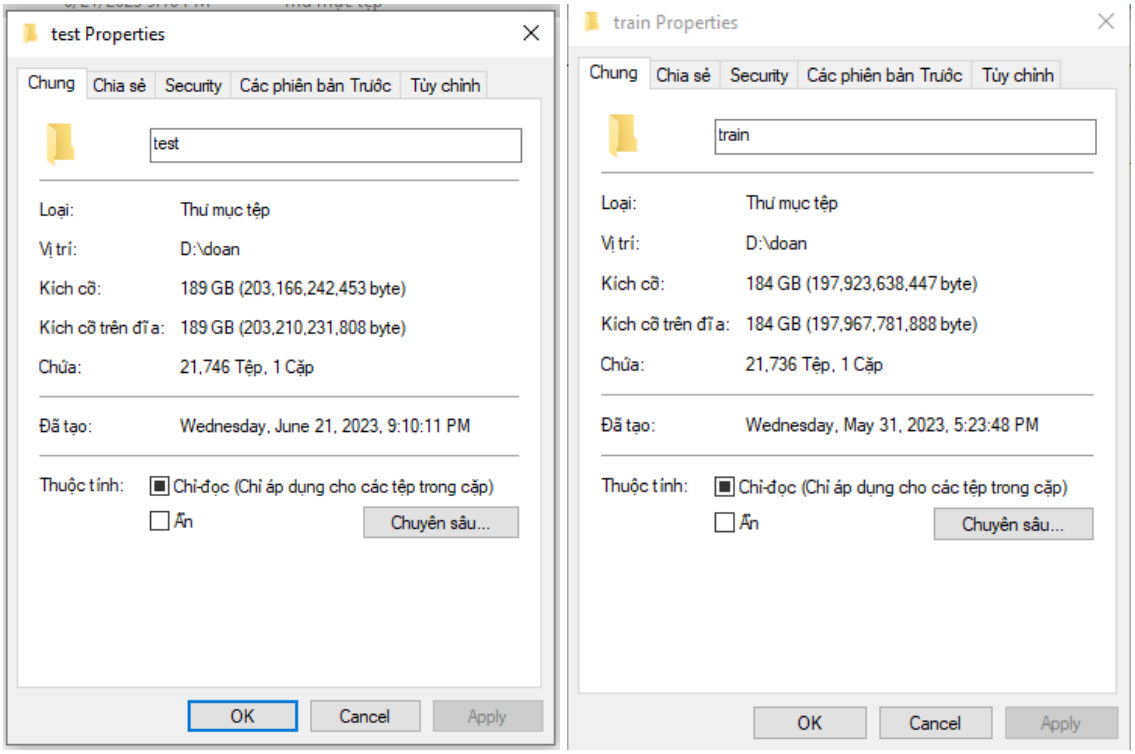


    - Đối với việc triển khai mô hình HYDRA: Triển khai trên máy tính Intel có vi xử lí là i7-10700U, 64GB ram và GPU1 của máy là NVIDIA GexForce RTX 2060

  - Đối với dữ liệu khi train và phân loại xong sẽ có 4 file như sau: 1 thư mục test chứa data để test, 1 thư mục train chứa data để train, 1 file csv chứa thông tin lớp và tên của tệp data.



Tên	Ngày sửa đổi	Loại	Kích cỡ
dataSample	6/22/2023 9:31 PM	Thư mục tệp	
test	6/21/2023 9:10 PM	Thư mục tệp	
train	5/31/2023 5:23 PM	Thư mục tệp	
dataSample	12/15/2019 10:50 PM	WinRAR archive	4,163 KB
sampleSubmission	12/15/2019 10:50 PM	Microsoft Excel C...	2,061 KB
test	12/15/2019 10:55 PM	WinRAR archive	18,629,859 KB
train	12/15/2019 11:34 PM	WinRAR archive	18,369,816 KB
train.csv	12/16/2019 12:11 AM	Microsoft Excel C...	266 KB



- Triển khai phân loại data: sử dụng các kĩ thuật thuật toán học máy để phân loại data như sau (vì nó là phần phụ nên chỉ đi ngang qua):
  - Random Forest
  - Gradient Boosting Decision Trees
  - Logistic Regression
  - Support Vector Machines (SVM)
  - Neural Networks (Multilayer Perceptron)

- Triển khai mô hình HYDRA Multimodal Deep Learning như sau: tổng quan các thành phần và khi ghép lại để thành mô hình hoàn chỉnh thì đều có 4 tệp py có chức năng như sau:
  - base_architecture.py: sử dụng các thư viện học máy và học sâu để huấn luyện các mô hình trên nền data có sẵn (thư viện sử dụng như là Tensorflow,...)
  - custom_training.py: Đưa dữ liệu cần train vào trong mô hình và phân loại sau đó tính các thông số như là F1 score, độ chính xác,... và vẽ ma trận
  - tfreader.py: nhận dữ liệu từ tfwriter.py và định nghĩa chúng lại chuẩn để sử dụng trong mô hình học máy.
  - tfwriter.py: có chức năng là chuyển đổi dữ liệu thô (byte và asm) sang định dạng Tfrecord để có thể sử dụng được trong mô hình học máy huấn luyện

- Các mô hình thành phần triển khai như sau:
  - Mô hình API based- component: Định nghĩa mô hình học máy PIsNN được xây dựng bằng Keras trong TensorFlow phân loại các ứng dụng dựa tên các api. Kiến trúc xây dựng là neural truyền thẳng (feedforward neural network).

```python
mlw_classification_hydra-main > src > method > api_component > base_architecture.py > APIsNN > build
import tensorflow as tf

class APIsNN(tf.keras.Model):
    def __init__(self, parameters):
        super(APIsNN, self).__init__()
        self.parameters = parameters

    def build(self, input_shapes):
        self.input_dropout = tf.keras.layers.Dropout(self.parameters["input_dropout_rate"],
                                                     input_shape=(None, self.parameters["features"]))

        self.h1 = tf.keras.layers.Dense(self.parameters['hidden'],
                                        activation="relu",
                                        input_shape=(None, self.parameters["features"]))


        self.output_dropout = tf.keras.layers.Dropout(self.parameters["hidden_dropout_rate"],
                                                      input_shape=(None, self.parameters["hidden"]))

        self.out = tf.keras.layers.Dense(self.parameters['output'],
                                         activation="softmax",
                                         input_shape=(None, self.parameters["hidden"]))

    def call(self, input_tensor, training=False):
        input_dropout = self.input_dropout(input_tensor, training=training)
        hidden1 = self.h1(input_dropout)
        output_dropout = self.output_dropout(hidden1, training=training)
        out = self.out(output_dropout)
        return out
```

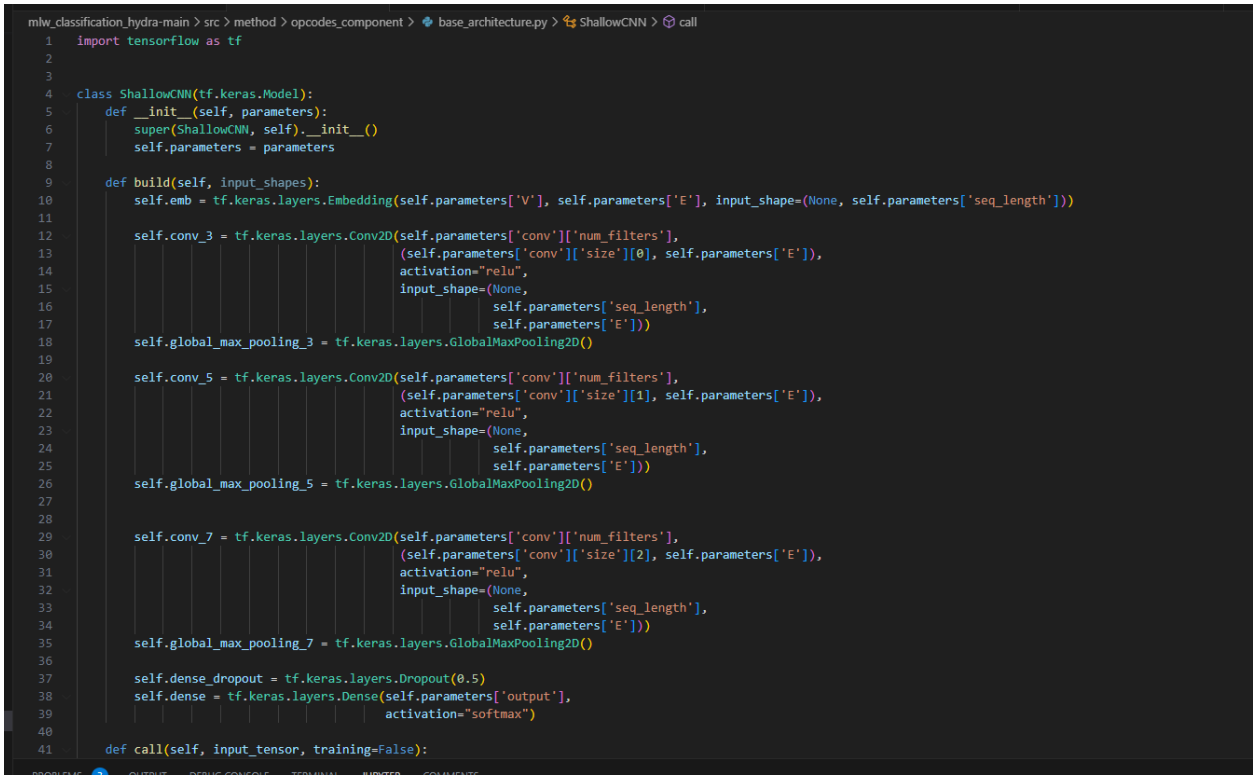
- Mô hình byte-based-component: Đây là một lớp mô hình Keras được định nghĩa bằng cách kế thừa lớp tf.keras.Model. Lớp mô hình này sử dụng một kiến trúc mạng neural tích chập (Convolutional Neural Network) để phân loại các văn bản. Bên cạnh đó sẽ có những từ khóa(vocabulary) để có thể train và phân loại một cách dễ dàng hơn).

```python
mlw_classification_hydra-main > src > method > bytes_component > base_architecture.py > DeepConv > build
import tensorflow as tf


class DeepConv(tf.keras.Model):
    def __init__(self, parameters):
        super(DeepConv, self).__init__()
        self.parameters = parameters

    def build(self, input_shapes):
        self.emb = tf.keras.layers.Embedding(self.parameters['V'], self.parameters['E'],
                                             input_shape=(None, self.parameters['max_bytes_values']))

        self.conv_1 = tf.keras.layers.Conv2D(filters=self.parameters['num_filters'][0],
                                             kernel_size=[self.parameters['kernel_sizes'][0],
                                                          self.parameters['E']],
                                             strides=(self.parameters['strides'][0],1),
                                             data_format='channels_last',
                                             use_bias=True,
                                             activation="relu")

        self.conv_2 = tf.keras.layers.Conv2D(filters=self.parameters['num_filters'][1],
                                             kernel_size=[self.parameters['kernel_sizes'][1],
                                                          1],
                                             strides=(self.parameters['strides'][1],1),
                                             data_format='channels_last',
                                             use_bias=True,
                                             activation="relu")

        self.max_pool_1 = tf.keras.layers.MaxPooling2D(pool_size=(self.parameters['max_pool_size'], 1))

        self.conv_3 = tf.keras.layers.Conv2D(filters=self.parameters['num_filters'][2],
                                             kernel_size=[self.parameters['kernel_sizes'][2],
                                                          1],
                                             strides=(self.parameters['strides'][2], 1),
                                             data_format='channels_last',
                                             use_bias=True,
                                             activation="relu")
```

- Mô hình opcode-based-component: Đây là một lớp mô hình Keras được định nghĩa bằng cách kế thừa lớp tf.keras.Model. Lớp mô hình này cũng

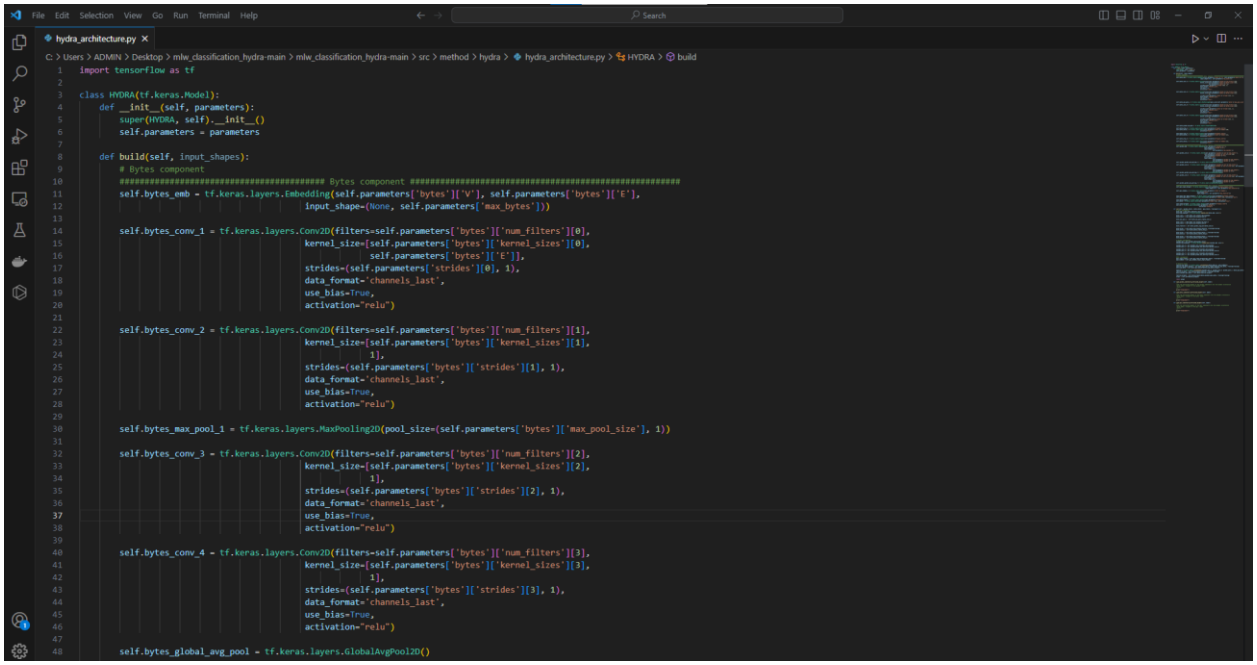
sử dụng một kiến trúc mạng neural tích chập để phân loại các văn bản.Các thành phần chính như sau:

- Đối số đầu vào: sẽ tham chiếu qua vocabulary kèm theo, số chiều embedding, bộ lọc, độ dài,...
- Lớp nhúng: chuyển dữ liệu thành vector theo chiều của embedding
- Các lớp tích chập và pooling...




```python
import tensorflow as tf


class ShallowCNN(tf.keras.Model):
    def __init__(self, parameters):
        super(ShallowCNN, self).__init__()
        self.parameters = parameters

    def build(self, input_shapes):
        self.emb = tf.keras.layers.Embedding(self.parameters['V'], self.parameters['E'], input_shape=(None, self.parameters['seq_length']))

        self.conv_3 = tf.keras.layers.Conv2D(self.parameters['conv']['num_filters'],
                                             (self.parameters['conv']['size'][0], self.parameters['E']),
                                             activation="relu",
                                             input_shape=(None,
                                                          self.parameters['seq_length'],
                                                          self.parameters['E']))
        self.global_max_pooling_3 = tf.keras.layers.GlobalMaxPooling2D()

        self.conv_5 = tf.keras.layers.Conv2D(self.parameters['conv']['num_filters'],
                                             (self.parameters['conv']['size'][1], self.parameters['E']),
                                             activation="relu",
                                             input_shape=(None,
                                                          self.parameters['seq_length'],
                                                          self.parameters['E']))
        self.global_max_pooling_5 = tf.keras.layers.GlobalMaxPooling2D()


        self.conv_7 = tf.keras.layers.Conv2D(self.parameters['conv']['num_filters'],
                                             (self.parameters['conv']['size'][2], self.parameters['E']),
                                             activation="relu",
                                             input_shape=(None,
                                                          self.parameters['seq_length'],
                                                          self.parameters['E']))
        self.global_max_pooling_7 = tf.keras.layers.GlobalMaxPooling2D()

        self.dense_dropout = tf.keras.layers.Dropout(0.5)
        self.dense = tf.keras.layers.Dense(self.parameters['output'],
                                           activation="softmax")

    def call(self, input_tensor, training=False):
```

- Tổng hợp kiến trúc thành kiến trúc HYDRA: sẽ sử dụng và kế thừa các mô hình trên và tổng hợp lại thành kiến trúc HYDRA



```python
import tensorflow as tf

class HYDRA(tf.keras.Model):
    def __init__(self, parameters):
        super(HYDRA, self).__init__()
        self.parameters = parameters

    def build(self, input_shapes):
        # Bytes component
        ################################### Bytes component ##########################################
        self.bytes_emb = tf.keras.layers.Embedding(self.parameters['bytes']['V'], self.parameters['bytes']['E'],
                                                   input_shape=(None, self.parameters['max_bytes']))

        self.bytes_conv_1 = tf.keras.layers.Conv2D(filters=self.parameters['bytes']['num_filters'][0],
                                                   kernel_size=[self.parameters['bytes']['kernel_sizes'][0],
                                                                self.parameters['bytes']['E']],
                                                   strides=(self.parameters['strides'][0], 1),
                                                   data_format='channels_last',
                                                   use_bias=True,
                                                   activation="relu")

        self.bytes_conv_2 = tf.keras.layers.Conv2D(filters=self.parameters['bytes']['num_filters'][1],
                                                   kernel_size=[self.parameters['bytes']['kernel_sizes'][1],
                                                                1],
                                                   strides=(self.parameters['bytes']['strides'][1], 1),
                                                   data_format='channels_last',
                                                   use_bias=True,
                                                   activation="relu")

        self.bytes_max_pool_1 = tf.keras.layers.MaxPooling2D(pool_size=(self.parameters['bytes']['max_pool_size'], 1))

        self.bytes_conv_3 = tf.keras.layers.Conv2D(filters=self.parameters['bytes']['num_filters'][2],
                                                   kernel_size=[self.parameters['bytes']['kernel_sizes'][2],
                                                                1],
                                                   strides=(self.parameters['bytes']['strides'][2], 1),
                                                   data_format='channels_last',
                                                   use_bias=True,
                                                   activation="relu")

        self.bytes_conv_4 = tf.keras.layers.Conv2D(filters=self.parameters['bytes']['num_filters'][3],
                                                   kernel_size=[self.parameters['bytes']['kernel_sizes'][3],
                                                                1],
                                                   strides=(self.parameters['bytes']['strides'][3], 1),
                                                   data_format='channels_last',
                                                   use_bias=True,
                                                   activation="relu")

        self.bytes_global_avg_pool = tf.keras.layers.GlobalAvgPool2D()
```

- Bytes Component: Đây là thành phần xử lý dữ liệu bytes. Nó bao gồm các lớp như Embedding, Conv2D, MaxPooling2D, GlobalAvgPool2D, Dropout và Dense. Các lớp này được sử dụng để ánh xạ dữ liệu bytes vào không gian nhúng (embedding) và thực hiện các phép tích chập và gộp dữ liệu trên không gian 2D.
- Opcodes Component: Đây là thành phần xử lý dữ liệu opcodes. Nó cũng bao gồm các lớp như Embedding, Conv2D, GlobalMaxPooling2D. Tương tự như thành phần bytes, thành phần này ánh xạ dữ liệu opcodes vào không gian nhúng và thực hiện các phép tích chập và gộp dữ liệu trên không gian 2D.
- APIs Component: Đây là thành phần xử lý dữ liệu APIs. Nó bao gồm các lớp như Dropout và Dense. Thành phần này xử lý dữ liệu APIs bằng cách áp dụng các lớp kết nối đầy đủ (fully connected layers) trên dữ liệu đầu vào.
- Features Fusion: Sau khi xử lý dữ liệu trong các thành phần riêng biệt, dữ liệu được kết hợp lại thông qua việc ghép các đặc trưng (features) lại với nhau. Cụ thể, các đặc trưng từ thành phần bytes, opcodes và APIs được ghép lại bằng cách kết nối chúng và sau đó áp dụng lớp kết nối đầy đủ (fully connected layer) để tạo ra đầu ra dự đoán.

C. Kết quả thực nghiệm

- Phân loại malware theo bộ dữ liệu của Microsoft :

```
classes=labels.groupby('Class').count()
classes=classes.rename(columns={'Id':'count'})
classes['percentage']=classes['count']/classes['count'].sum()
classes
```

Class	count	percentage
1	1541	0.141792
2	2478	0.228009
3	2942	0.270703
4	475	0.043706
5	42	0.003865
6	751	0.069102
7	398	0.036621
8	1228	0.112992
9	1013	0.093209

1.1 Cuộc thử thách Phân loại Phần mềm độc hại của Microsoft.

Family name	#Samples	Type
Ramnit	1541	Worm
Lollipop	2478	Adware
Kelihos_ver3	2942	Backdoor
Vundo	475	Trojan
Simda	42	Backdoor
Tracur	751	TrojanDownloader
Kelihos_ver1	398	Backdoor
Obfuscator.ACY	1228	Any kind of obfuscated malware
Gatak	1013	Backdoor

Bảng 1: Phân bố các lớp trong bộ dữ liệu thử thách phân loại phần mềm độc hại của Microsoft.

- Kết quả thống kê khi đã phân loại trùng khớp với dữ liệu đầu vào của bài báo

```
count_df
```

	filename	Class	size(MB)	00	01	02	03	04	05	06	...	f6	f7	f8	f9	fa	fb	fc	fd	fe	ff
0	2RmXnKGVYksyxd89fTMU.bytes	2	0.668160	15362	1061	721	615	810	363	330	...	275	7872	493	179	149	122	562	406	618	6986
1	cgjqJU4OnlGLavkNKQmX.bytes	1	0.683008	43285	788	643	553	1122	425	465	...	522	371	503	356	357	328	436	370	417	2353
2	cRmPXUIkyYxFfNT0v2IS.bytes	1	0.772096	27937	4133	838	1005	1704	818	700	...	596	617	717	497	499	485	696	492	607	4560
3	1IqjvX7dTxCu54M0yIBE.bytes	2	3.489280	264506	4509	2981	7938	3619	2868	4168	...	642	325	4812	242	171	281	4795	449	3421	43649
4	g7dyvFke26HpiqVs8Afc.bytes	1	0.994816	47629	2382	1179	1045	2270	896	717	...	701	651	871	578	572	651	1019	593	1057	8321
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10863	kg24YRJTB8DNdKMXpwOH.bytes	9	4.825600	91302	9127	2852	2620	9379	123519	8953	...	2497	2647	2355	2535	2316	2306	2776	2400	3213	50424
10864	cFa4U6xqeVW7IiJzMQk0.bytes	9	0.608768	17319	1215	429	321	1138	1053	1061	...	315	355	287	289	315	275	346	439	510	7210
10865	CZ2gJmqpoi0PuKQnxfHw.bytes	9	3.088384	124706	7124	2957	3449	3310	4099	3222	...	2261	2338	2489	2465	2383	2439	2638	2744	2785	9587
10866	IbALBFiNrcgEW0h3Ow9k.bytes	9	0.876032	87099	2025	826	752	605	651	487	...	505	491	561	482	472	426	445	415	479	2284
10867	hEetyRmw4z6TGj23791i.bytes	9	0.549376	19370	1157	644	539	675	502	397	...	287	328	420	324	339	379	666	385	400	2851

10868 rows × 259 columns

➔ Giải thích sơ lược: thể hiện được tên của file data, thuộc lớp malware nào (có 9 loại malware) , cột tiếp theo là thể hiện dung lượng của file, tiếp là số lượng dòng, 3 cột tiếp theo có thể là thể hiện cho API Windowns call, byte và opcode và những thông tin khác.

- Phân loại dữ liệu malware

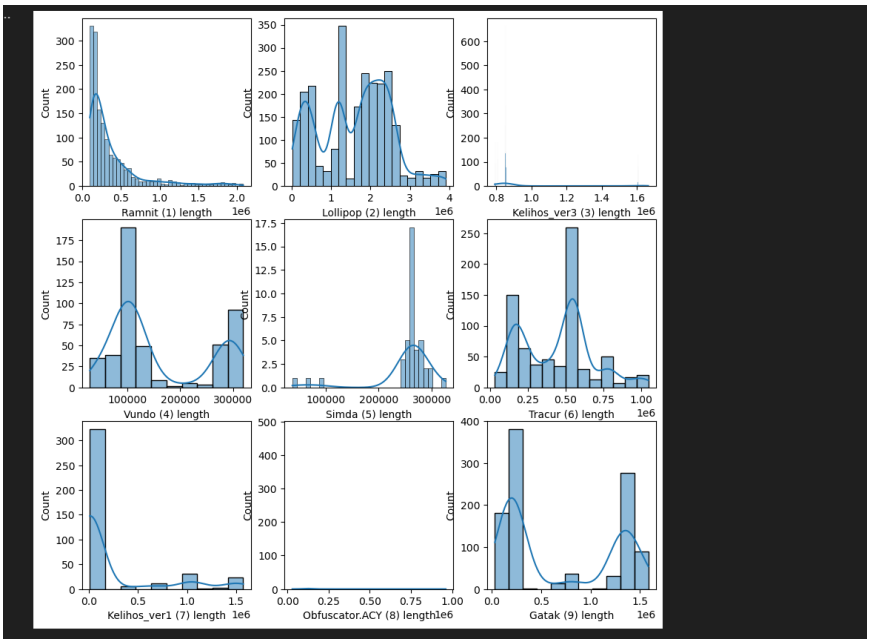
```
tfidf_trans_df
```

Python

	filename	size(MB)	Class	00	01	02	03	04	05	06	...	f6	f7	f8	f9	fa	fb	fc	fd	fe	ff
0	2RmXnKGVYksyxd89fTMU.bytes	0.668160	3	0.573824	0.039632	0.026992	0.022972	0.030256	0.013559	0.012327	...	0.010272	0.294047	0.018415	0.006697	0.005566	0.004558	0.020993	0.015167	0.023084	0.260951
1	cgjqJU4OnlGLavkNKQmX.bytes	0.683008	1	0.981613	0.017870	0.014582	0.012541	0.025445	0.009638	0.010545	...	0.011838	0.008413	0.011407	0.008086	0.008096	0.007439	0.009888	0.008392	0.009457	0.053361
2	cRmPXUlkyYxFfNT0v2IS.bytes	0.772096	1	0.888243	0.131407	0.026644	0.031953	0.054178	0.026008	0.022256	...	0.018950	0.019617	0.022797	0.015827	0.015865	0.015422	0.022129	0.015644	0.019299	0.144983
3	1lqjvX7dTxCu54M0ylBE.bytes	3.489280	2	0.828920	0.014130	0.009342	0.024876	0.011341	0.008988	0.013062	...	0.002012	0.001018	0.015080	0.000760	0.000536	0.000881	0.015027	0.001407	0.010721	0.136789
4	g7dyvFke26HpiqVs8Afc.bytes	0.994816	1	0.928599	0.046441	0.022986	0.020374	0.044257	0.017469	0.013979	...	0.013667	0.012692	0.016981	0.011287	0.011152	0.012693	0.019867	0.011562	0.020608	0.162230
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10863	kg24YRJTB8DNdKMXpwOH.bytes	4.825600	9	0.371008	0.037088	0.011589	0.010646	0.038112	0.501922	0.036381	...	0.010147	0.010756	0.009570	0.010317	0.009411	0.009371	0.011280	0.009753	0.013056	0.204899
10864	cFa4U6xqeVW7IiJzMQk0.bytes	0.608768	9	0.608434	0.042684	0.015071	0.011277	0.039979	0.036993	0.037274	...	0.011066	0.012472	0.010083	0.010169	0.011066	0.009662	0.012155	0.015424	0.017917	0.253295
10865	CZ2gJmqpoi0PuKQnxfHw.bytes	3.088384	9	0.930038	0.053130	0.022053	0.025722	0.024685	0.030570	0.024029	...	0.016862	0.017436	0.018563	0.018412	0.017772	0.018191	0.019674	0.020466	0.020770	0.071498
10866	IbALBFiNrcgEW0h3Ow9k.bytes	0.876032	9	0.991056	0.023041	0.009399	0.008557	0.006884	0.007407	0.005541	...	0.005746	0.005587	0.006383	0.005493	0.005371	0.004848	0.005063	0.004723	0.005450	0.025988
10867	hEetyRmw4z6TGj23791i.bytes	0.549376	9	0.920014	0.054954	0.030588	0.025601	0.032060	0.023843	0.018856	...	0.013632	0.015579	0.019949	0.015413	0.016101	0.018003	0.031633	0.018288	0.018999	0.135413

0868 rows × 259 columns

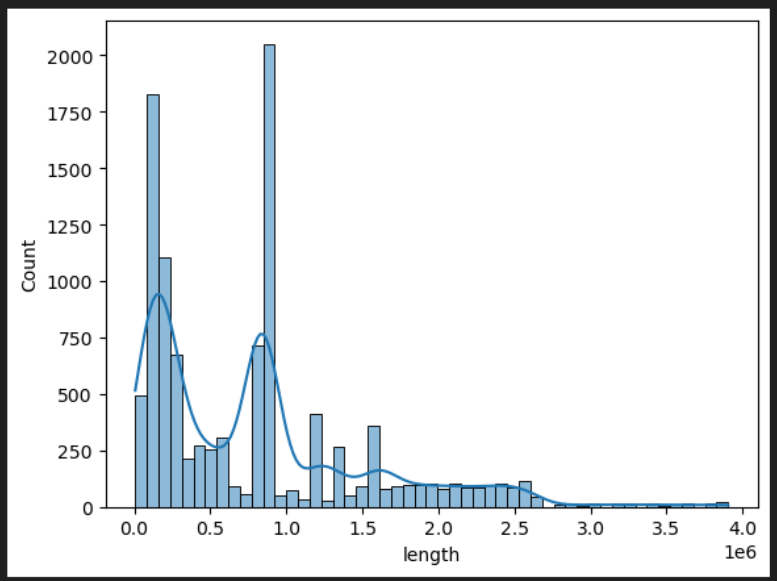
- Mỗi hàng của DataFrame này đại diện cho một tệp tin chứa mã độc. Các cột đầu tiên của DataFrame là filename, size(MB), và Class, lần lượt là tên tệp tin, kích thước tệp tin (đơn vị là MB), và lớp được gán cho tệp tin đó (0 hoặc 1). Các cột tiếp theo là các giá trị tf-idf của các từ vựng trong danh sách vocab_ls cho mỗi tệp tin. Các giá trị tf-idf này được tính toán bằng cách sử dụng mô hình TfidfTransformer được huấn luyện từ dữ liệu trong tmp_count_df.
- Từ dữ liệu đã phân loại trên, vẽ histogram biểu thị phân bố độ dài của chuỗi byte trong các tệp nhị phân trong tập dữ liệu trực quan mô hình phân loại.



- Biến đổi byte được chuyển đổi thành hình ảnh. Đoạn mã này chuyển mã byte thành mảng numpy và sau đó chia thành các khối có kích thước vuông. Kích thước của mỗi hình ảnh được tính toán dựa trên độ dài của mã byte. Biểu đồ histogram cho biết phân phối tần suất của các giá trị trong cột 'length' trên trục x, trong khi ước lượng KDE cung cấp một cái nhìn tổng quan về hình dạng của phân bố dữ liệu.

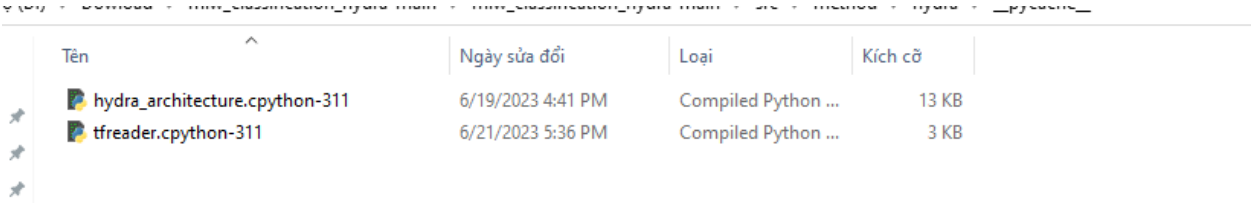
```
[40] sns.histplot(data=data_df,x='length',kde=True)
```

```
<AxesSubplot:xlabel='length', ylabel='Count'>
```

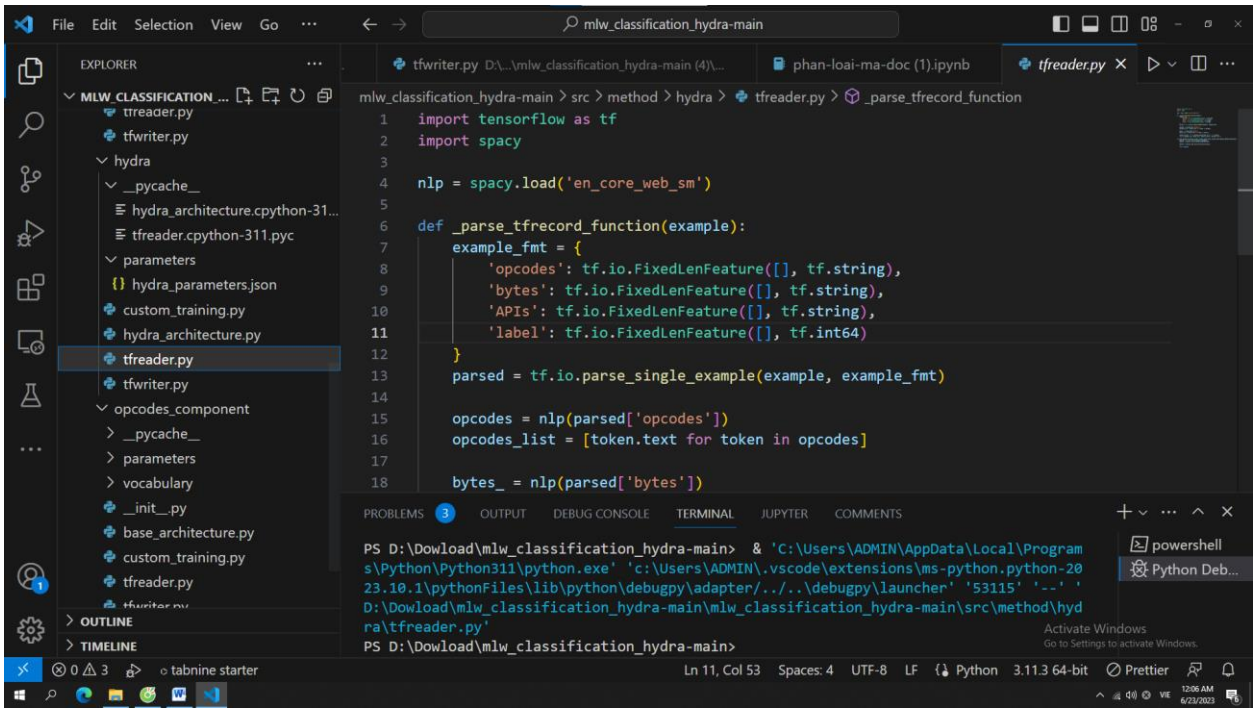


Chạy kiến trúc HYDRA:

  - Nhóm chỉ mới đạt được đọc dữ liệu từ các vùng TFrecord (tức là dữ liệu thô đã được phân loại và tinh chỉnh) vào cho mô hình huấn luyện và chạy được mô hình huấn luyện.
  - Còn chưa đạt được là chưa đọc chuẩn được từ dữ liệu thô đã phân loại ở phía trên vào trong TFrecord và khi cho chạy mô hình train thì vẫn gặp lỗi.
- Đây là những kết quả đã đạt được và chưa đạt được.
  - Đạt được: khi chạy thành công thì sẽ xuất hiện những file exe của python để có thể chạy mô hình (mô hình sẽ train và đưa ra kết quả thành công thì phải có cả 4 file nhưng nhóm chỉ mới chạy được 2).

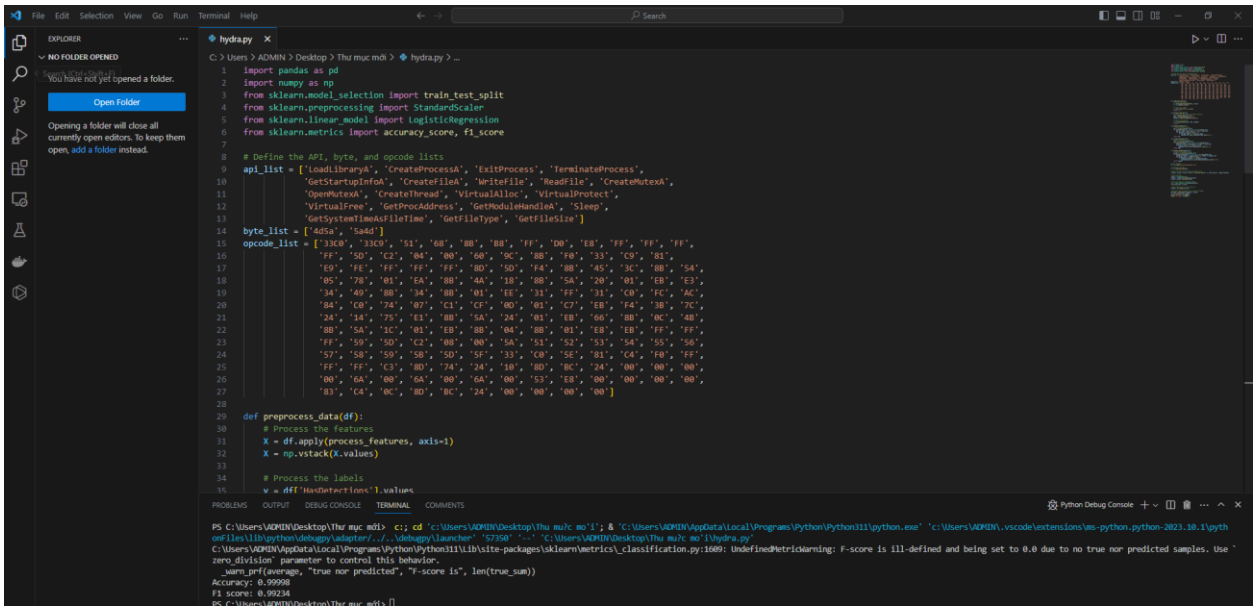


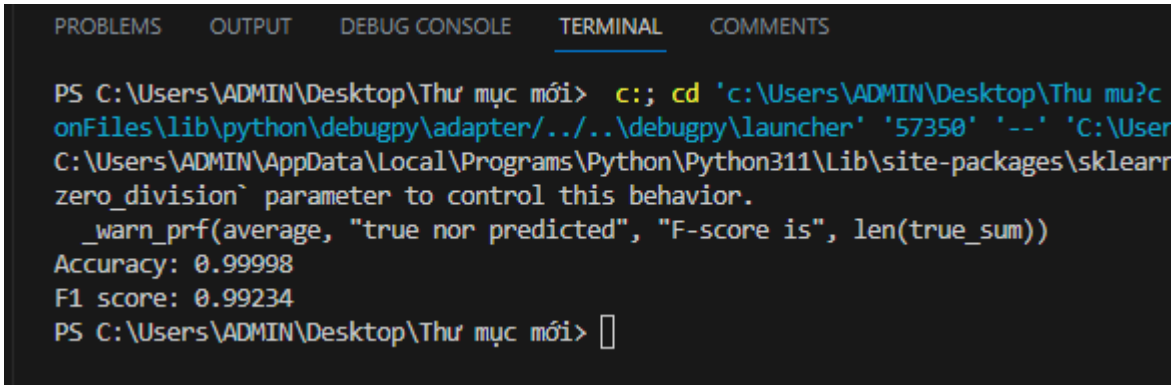
- Hình ảnh trực quan khi chạy load các dữ liệu từ TFrecord vào trong mô hình huấn luyện.



- Nhóm có đề xuất là xây dựng một mô hình HYDRA đơn giản hơn nhưng vẫn đầy đủ các tiêu chí như API, opcde và byte để có thể đánh giá được độ chính xác là f1 score.
- Ý tưởng như sau:
  - xây dựng một mô hình phân loại dựa trên các đặc trưng liên quan đến các hành vi độc hại của phần mềm độc hại. Ý tưởng chính của code là sử dụng các đặc trưng API , byte và opcode (được cấu hình vocabulary có sẵn) để biểu diễn các hành vi của phần mềm độc hại. Các đặc trưng này được xử lý để tạo ra một ma trận đặc trưng, mỗi hàng của ma trận đặc trưng tương ứng với một ví dụ dữ liệu.
  - Sau đó, một mô hình phân loại Logistic Regression được huấn luyện trên tập dữ liệu đã được xử lý để phân loại các ví dụ dữ liệu thành hai lớp: "có chứa phần mềm độc hại" và "không chứa phần mềm độc hại".
  - Cuối cùng, độ chính xác và độ F1 score của mô hình được tính toán trên tập dữ liệu kiểm tra để đánh giá hiệu suất của mô hình. Độ F1 score được sử dụng để đánh giá hiệu suất của mô hình trên cả hai lớp và được tính toán bằng cách kết hợp độ chính xác và độ phủ của mô hình trên cả hai lớp.

- Khi chạy code dưới thì có ra được kết quả mô hình có độ tương đối chính xác khá cao(với một dữ liệu data nhỏ 1/200 so với bộ dữ liệu data trên). Vì vậy muốn đánh giá tổng quan hơn thì phải cần thời gian huấn luyện và một lượng tài nguyên phần cứng đủ lớn để có thể train và test và rút ra được độ chính xác trung bình và f1 score trung bình.





- Em có chỉnh sửa và xây dựng một mô hình HYDRA phức tạp hơn và chưa chạy được có tên là HYDRA-2 được đính kèm theo ☺.

D. Hướng phát triển

Cuối bài báo, tác giả có đưa ra những hướng phát triển và cải thiện như sau:

- Sử dụng các mô hình học sâu phức tạp hơn: Để cải thiện độ chính xác, có thể sử dụng các mô hình học sâu phức tạp hơn như Transformer hay GPT. Mô hình Transformer sử dụng cơ chế chú ý để hiệu quả hơn trong việc xử lý các chuỗi dữ liệu dài và phức tạp, trong khi mô hình GPT được huấn luyện trước trên một tập dữ liệu rất lớn và đã đạt được kết quả rất tốt trong các cuộc thi đánh giá bài toán xử lý ngôn ngữ tự nhiên. Sử dụng các mô hình này trong HYDRA có thể cải thiện độ chính xác của phân loại phần mềm độc hại, nhưng cần có một tập dữ liệu đủ lớn và đa dạng để huấn luyện và tinh chỉnh các siêu tham số của mô hình.

- Sử dụng các phương pháp tăng cường: Phương pháp tăng cường dữ liệu là một kỹ thuật quan trọng trong việc giải quyết vấn đề về quá khớp và tăng tính đa dạng của tập dữ liệu huấn luyện, đặc biệt là trong bài toán phân loại phần mềm độc hại. Các phương pháp tăng cường dữ liệu như đảo ngược, cắt ghép hoặc xoay ảnh giúp tạo ra nhiều phiên bản dữ liệu khác nhau từ mẫu dữ liệu ban đầu, giúp mô hình học được các đặc trưng khác nhau và giảm thiểu vấn đề quá khớp. Tuy nhiên, cần kiểm tra và tinh chỉnh các tham số của phương pháp tăng cường dữ liệu để đạt được hiệu quả tối ưu, và tránh việc tăng cường dữ liệu quá mức dẫn đến mất mát thông tin và giảm độ chính xác của mô hình
- Tối ưu hóa các siêu tham số: Để đạt được độ chính xác cao nhất của mô hình, ta cần phải tối ưu hóa các siêu tham số một cách cân bằng và xác định được sự phù hợp của mô hình với dữ liệu huấn luyện.
- Kết hợp HYDRA với các phương pháp phân loại khác: Kết hợp HYDRA với các phương pháp phân loại khác là một phương pháp cải tiến hiệu quả để tăng độ chính xác của mô hình phân loại malware. Kỹ thuật voting sử dụng nhiều mô hình phân loại khác nhau và tính toán kết quả bằng cách lấy phần đông phiếu. Kỹ thuật stacking sử dụng HYDRA làm mô hình cơ sở và sử dụng các phương pháp phân loại khác làm mô hình meta để học và kết hợp các đầu ra của các mô hình phân loại khác nhau để cho ra kết quả cuối cùng.

Sinh viên báo cáo các nội dung mà nhóm đã thực hiện, có thể là 1 phần hoặc toàn bộ nội dung của bài báo. Nếu nội dung thực hiện có khác biệt với bài báo (như cấu hình, tập dữ liệu, kết quả,...), sinh viên cần chỉ rõ thêm khác biệt đó và nguyên nhân.

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Đặt tên theo định dạng: [Mã lớp]-Project_Final_NhomX_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

  Ví dụ: [NT521.N11.ANTT]-Project_Final_Nhom03_CK01.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT