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: Lập trình an toàn và khai thác lỗ hổng phần mềm

Tên chủ đề: PHƯƠNG PHÁP KẾT HỢP HỌC MÁY DỰA TRÊN ĐỒ THỊ VỚI THU THẬP DỮ LIỆU TỰ ĐỘNG TRONG PHÁT HIỆN LỖ HỔNG PHẦN MỀM

Mã nhóm: ***G08*** *Mã đề tài:* ***CK33.***

Lớp: **NT521.O11.ATCL**

1. THÔNG TIN THÀNH VIÊN NHÓM:

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

STT	Họ và tên	MSSV	Email
1	Nguyễn Mạnh Cường	20520421	20520421@gm.uit.edu.vn
2	Hoàng Văn Anh Đức	20520890	20520890@gm.uit.edu.vn
3	Phan Võ Thiên Trường	20522091	20522091@gm.uit.edu.vn

2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

A. Chủ đề nghiên cứu trong lĩnh vực An toàn phần mềm:

☑ Phát hiện lỗ hổng bảo mật phần mềm

☐ Khai thác lỗ hổng bảo mật phần mềm

☐ Sửa lỗi bảo mật phần mềm tự động

☐ Lập trình an toàn

☐ Khác: ……..

B. Tên bài báo tham khảo chính:

H. Wang et al., "Combining Graph-Based Learning With Automated Data Collection for Code Vulnerability Detection," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1943-1958, 2021, doi: 10.1109/TIFS.2020.3044773.

C. Dịch tên Tiếng Việt cho bài báo:

Phương Pháp Kết Hợp Học Máy Dựa Trên Đồ Thị Với Thu Thập Dữ Liệu Tự Động Trong Phát Hiện Lỗ Hổng Phần Mềm

[1] Ghi nội dung tương ứng theo mô tả

D. Tóm tắt nội dung chính:

FUNDED (Flow-sensitive vUlNerability coDE Detection) là một framework nghiên cứu về vấn đề phát hiện lỗ hổng phần mềm dựa trên học máy. FUNDED được xây dựng dựa trên mạng nơ-rôn đồ thị (GNN – Graph Neural Network), với mục đích đồ thị hóa chương trình ở mức độ mã nguồn (cụ thể hơn là mức độ hàm). Với việc mã nguồn được biểu diễn ở dạng đồ thị, nó thể hiện được chi tiết nhiều khía cạnh khác nhau của mã nguồn, bao gồm các luồn thực thi, các quan hệ giữa các đối tượng trong chương trình, từ đó cải thiện khả năng phát hiện và phân loại các lỗ hổng.

Để có thể phát huy được hết khả năng của mô hình phát hiện dựa trên GNN, yêu cầu phải có một tập dữ liệu huấn luyện đủ lớn với độ chính xác cao, trong khi những kho dữ liệu lỗ hổng có sẵn còn hạn chế. Vì thế, framework bao gồm thêm một quá trình tự động thu thập dữ liệu huấn luyện từ những commit liên quan đến vá lỗ hổng bảo mật của các chương trình mã nguồn mở. Quá trình thu thập bao gồm kết quả phân loại từ nhiều mô hình học máy, điều đó giúp khái quát hóa quá trình xử lý để phù hợp với nhiều dạng dữ liệu commit, tối ưu chất lượng cho tập huấn luyện. Ứng dụng thêm phương pháp đo độ chính xác để chọn lọc ra những dự đoán có độ tin cậy cao, từ đó nâng cao uy tín cho đầu ra và tự cải thiện các mô hình xử lý.

FUNDED cho thấy được sự toàn diện khi đưa ra được giải pháp từ khâu thu thập dữ liệu đến kết quả đầu ra. Kết quả so sánh thực nghiệm vượt trội hơn 6 phương pháp đề xuất tương tự.

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

FUNDED framework bao gồm 2 phần, phần phát hiện lỗ hổng và phần thu thập dữ liệu:

- Phát hiệu lỗ hổng: đồ thị hóa mã nguồn, phát hiện lỗ hổng
  - Đồ thị hóa mã nguồn được mở rộng từ AST (Abtract Syntax Tree) và PCDG (Program Control and Dependence Graph)
  - Mô hình học máy GNN được sử dụng là GGNN (Gated Graph Neural Network) của một phương pháp đề xuất trước đó.
- Thu thập dữ liệu: hổ trợ bổ sung cho tập dữ liệu huấn luyện mô hình phát hiện
  - Sử dụng loạt các mô hình học máy: SVM (Support Vector Machine), RF (Random Forests), KNN (K-nearest Neighbor), LR (Logistic Regression), GB (Gradient Boosting) cho việc đưa ra các dự đoán phân loại commit.

- Ứng dụng Conformal Prediction (CP) để đánh giá độ chính xác của các dự đoán, lọc ra các dự đoán kém uy tín và tự cải thiện các mô hình học máy phân loại.

F. Môi trường thực nghiệm của bài báo:

- Cấu hình máy tính: Ubuntu 22.04, 10gb RAM, 6 core
- Các công cụ hỗ trợ sẵn có: … <dùng cho giai đoạn nào>
- Ngôn ngữ lập trình để hiện thực phương pháp: python
- Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có): FUNDED/dataset: 73000 samples cho training, 17000 samples cho testing
- Tiêu chí đánh giá tính hiệu quả của phương pháp: Accruacy, Precision, Recall, F1-score

G. Kết quả thực nghiệm của bài báo:

Kết quả thu về không được như mong đợi, có thể vì dataset tác giả đưa không có được phân bố rõ ràng.

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

Tiền xử lý dataset, train và test model

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

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

100%

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Đọc hiểu bài báo	Cả nhóm
2	Phân tích phương pháp đề xuất	Mạnh Cường, Anh Đức

3	Triển khai thực nghiệm	Thiên Trường, Anh Đức
4	Viết báo cáo	Cả nhóm
5	….	

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.

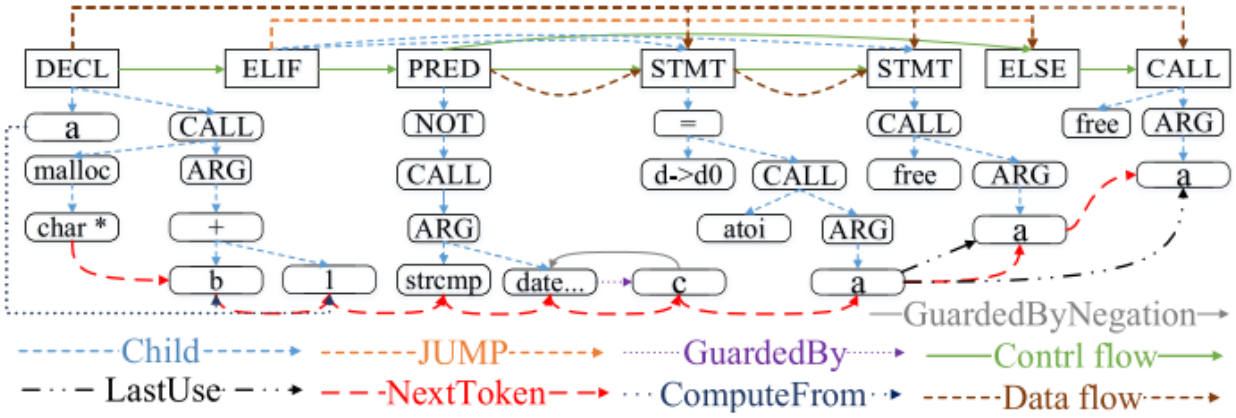
Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)

A. Phương pháp thực hiện

FUNDED framework bao gồm 2 phần, phần phát hiện lỗ hỏng và phần thu thập dữ liệu:

- **Phát hiện lỗ hỏng:**
  - Dữ liệu đầu vào là một đoạn code (thường là một hàm) cần được kiểm tra lỗ hỏng.
  - Code đầu vào được tiền xử lý, viết lại tên biến và hàm nhằm đồng bộ hóa và tránh làm nhiễu khi được xử lý bởi mô hình phát hiện.
  - Đồ thị biểu diễn code (Program graph) được xây dựng từ AST (Abtract Syntax Tree - gồm syntax node, syntax token và parent-child edge (cạnh thể hiện quan hệ cha-con)) và mở rộng thêm 8 loại edge khác, bao gồm:
    - Dataflow và Controlflow: được trích xuất từ PCDG (Program Control and Dependence Graph).
    - GuardedBy: thể hiện sự bao bọc của các biểu thức.
    - Jump: thể hiện cho các phân nhánh chương trình.
    - ComputedFrom: thể hiện toán tử gán (=) và tính toán.
    - NextToken: thể hiện thứ tự các syntax token.
    - LastUse: kết nối các lần sử dụng của cùng một biến.
    - LastLexicalUse: kết nối các lần sử dụng của cùng một biến trong câu lệnh IF.

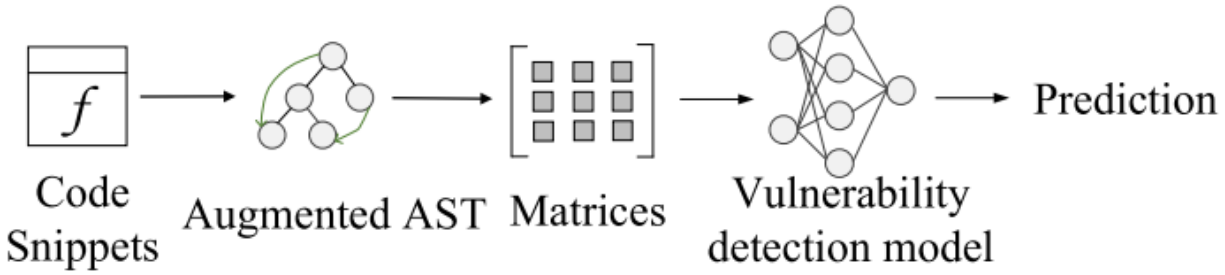
```
1   a = (char*)malloc(b+1);
2   ...
3   else if(!strcmp(c,"dateadded")) {
4     d->d0 = atoi(a);
5     free(a);
6   } else {
7     free(a);
8   }
```



Hình 1: *Đoạn code mẫu và thể hiện đồ thi hóa (Program graph) của nó.*

- Mỗi loại edge được biểu diễn trên mỗi đồ thị riêng, từ đó tạo ra được 9 đồ thị mối quan hệ thể hiện cho 9 loại edge (Relation graph - đồ thị có hướng). Đồ thị có các điểm là các AST syntax node và token, và các cạnh là các edge đã đề cập.
- Sau đó, sự kết nối của các cạnh được biểu diễn dưới dạng ma trận (Program graph matrix).
- GNN được sử dụng là GGNN (Gated Graph Neural Network) của một phương pháp đề xuất trước đó. GGNN bao gồm 4 mô hình nhúng được xếp chồng dựa trên GRU (Gated Recurrent Unit). GGNN được điều chỉnh để phù hợp với ngữ cảnh xử lý code được biểu diễn ở dạng đồ thị nhiều mối quan hệ.
- Mô hình học máy GGNN được huấn luyện bằng tập dữ liệu lấy từ các kho lưu trữ lỗ hổng công khai (như CVE, SARD, NVD) đã được xác thực, gán nhãn và đầu ra của phần thu thập dữ liệu trong framework.

- Mô hình nhận đầu vào là Program graph matrix và các AST syntax node và token. Từ đó, code được biểu diễn bằng một vector nhúng có giá trị số bằng word2vector network. Cuối cùng, dữ liệu được đưa vào mạng nơ-rôn đồ thị để phát hiện và phân loại lỗ hỏng.
- Mô hình triển khai có dạng mô hình học máy đa lớp (multi-class), đầu ra sẽ cho kết quả phát hiện code có lỗ hỏng hay không và nếu có, thì đó là phân loại lỗ hỏng nào.
- *Lưu ý:* Mục tiêu của framework là đồ thị hóa mã nguồn và tập trung vào phân loại các lỗ hỏng phổ biến đã được biết từ trước trên những đoạn mã nguồn hoàn toàn mới, chứ không tập trung vào tìm ra được loại lỗ hỏng mới.



Hình 2: *Khái quát phần phát hiện lỗ hỏng*

- **Thu thập dữ liệu:**
  - Dữ liệu đầu vào là những commit của các chương trình mã nguồn mở, bao gồm phần message và patch.
  - Qua một bước lọc bởi các quy tắc Regular Expression (RE - biểu thức chính quy) để chọn lọc ra những commit có liên quan hơn đến vá lỗ hỏng bảo mật. Sau đó, được đưa vào mỗi mô hình xử lý phân loại.

Code revisions	C1: Vulnerability-relevant commit	C2: Vulnerab. irrelevant commit
Message	Add NULL check to avoid null pointer access.	Check err when partial == NULL is meaningless because partial == NULL means getting branch successfully without error.
Patch	*4 addition lines , 2 deletion lines* -sap_ctx− >csa_reason = reason; +if (sap_ctx) +sap_ctx− >csa_reason = reason; -hdd_ap_ctx− >... +if (hdd_ap_ctx − > sap_context) +hdd_ap_ctx− >...	*3 addition lines, 2 deletion lines* -if (err) -goto cleanup; +if (err) +mutex_unlock(ei− >... +goto cleanup;

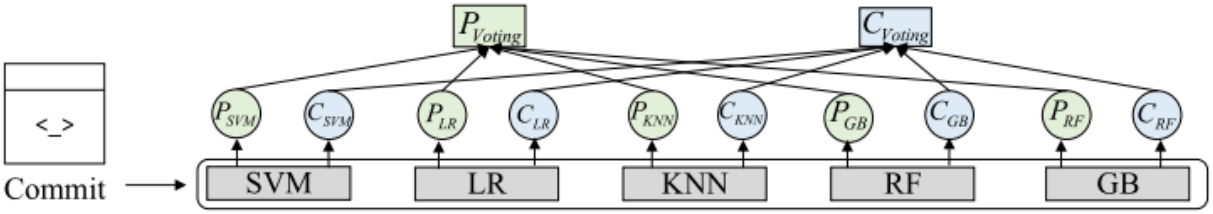
Bảng 1: *Mẫu commit có và không liên quan đến lỗ hổng bảo mật.*

- Bao gồm 5 mô hình học máy phân loại, tất cả được huấn luyện và xử lý độc lập:
  - SVM (Support Vector Machine);
  - RF (Random Forests);
  - KNN (K-nearest Neighbor);
  - LR (Logistic Regression);
  - GB (Gradient Boosting).
- 5 mô hình được huấn luyện trên cũng một tập dữ liệu. Dữ liệu huấn luyện được lấy từ các kho lưu trữ lỗ hổng công khai (như CVE, SARD, NVD) đã được xác thực, gán nhãn và các commit từ chương trình mã nguồn mở được gán nhãn thủ công dựa trên thông tin từ CVE.
- Các feature cho quá trình phân loại bao gồm:
  - Project quality and activities: là những thông tin và hoạt động từ dự án mã nguồn mở của commit.
  - Code commit description và Code patch: là phần message và patch của commit.

Category	Features
Project quality and activities	(1) #stars; (2) #commits; (3) # releases; (4) #contributors; (5) contribution rate; (6) #branches
Code commit description	commit message;
Code patch	code changes;

Bảng 2: *Các feature phân loại commit.*

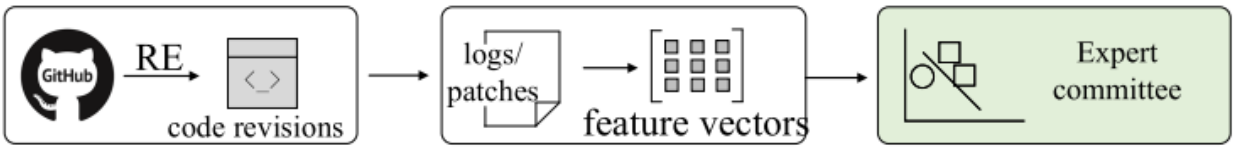
- Kết quả dự đoán phân loại của tất cả 5 mô hình được tổng hợp, sau đó được đo lường và đánh giá độ chính xác bằng phương pháp Conformal Prediction (CP). Các dự đoán có độ chính xác cao đạt trên ngưỡng thiết lập sẽ được gia tăng giá trị bỏ phiếu cho gán nhãn đầu ra. Các dự đoán có độ chính xác thấp hơn có thể được sử dụng để cải thiệt chất lượng cho các mô hình phân loại, từ đó ngày càng nâng cao hoàn thiện cho quá trình thu thập dữ liệu huấn luyện.
- Những dự đoán (gồm kết quả dự đoán và độ chính xác đã đo lường) được dựa trên kết quả bỏ phiếu đa số để thống nhất kết quả gán nhãn cuối cùng.



Hình 3: *Quá trình bỏ phiếu gán nhãn cho commit (P là dự đoán, C là độ chính xác)*

- Vì đầu vào là những commit vá lỗ hỏng (gồm message và patch), cần trích xuất ra những đoạn code (hàm) trước khi commit chỉnh sửa vá lỗ hỏng, được đề cập trong patch (tức là phần code có lỗ hỏng bảo mật), để phù hợp với dữ liệu đầu vào của phần phát hiện lỗ hỏng trong framework.

- Đoạn code trên kết hợp với kết quả gán nhãn là dữ liệu đầu ra cho quá trình thu thập dữ liệu, hỗ trợ bổ sung tập dữ liệu cho quá trình huấn luyện mô hình của phần phát hiện lỗ hỏng.
- *Lưu ý:* Phần thu thập dữ liệu chỉ phân loại những lỗ hỏng liên quan đến các vấn đề bảo mật là "có lỗ hỏng", chứ không bao gồm những vấn đề liên quan đến hiệu năng và tối ưu.



Hình 4: *Khái quát phần thu thập dữ liệu*

Với khả năng có thể tự tạo ra những tập dữ liệu huấn luyện mới có độ chính xác cao và tự cải thiện chất lượng xử lý gán nhãn ở phần thu thập dữ liệu, framework cho thấy được sự toàn diện và có thể phát triển, học tập theo thời gian để cải thiện khả năng phát hiện và phân loại lỗ hỏng dựa trên GNN.

Nhóm triển khai thực nghiệm FUNDED framework với mô hình tương tự bài báo.

B. Chi tiết cài đặt, hiện thực

Trong mô tả của tác giả đã có sẵn requirement cho model FUNDED nên nhóm chỉ việc chuẩn bị cấu hình máy tính cho việc thực hiện việc thu thập, xử lý dữ liệu và train, test model

- Cấu hình máy tính: HĐH: Ubuntu 22.04, Ram: 32GB, CPU: 6 cores, Storage: 120GB
- Data thu thập:
  - Train: 73000 samples
  - Test: 17000 samples

Thông tin về data

CWE-20	**Improper Input Validation**
CWE-74	**Improper Neutralization of Special Elements in Output Used by a D**
CWE-77	**Improper Neutralization of Special Elements used in a Command (**
CWE-78	**Improper Neutralization of Special Elements used in an OS Co Injection')**
CWE-119	**Improper Restriction of Operations within the Bounds of a Memor**
CWE-138	**Improper Neutralization of Special Elements**
CWE-190	**Integer Overflow or Wraparound**
CWE-191	**Integer Underflow (Wrap or Wraparound)**
CWE-200	**Exposure of Sensitive Information to an Unauthorized Actor**
CWE-287	**Improper Authorization**
CWE-362	**Concurrent Execution using Shared Resource with Improper Condition')**
CWE-369	**Divide By Zero**

CWE-400	**Uncontrolled Resource Consumption**
CWE-404	**Improper Resource Shutdown or Release**
CWE-467	**Use of sizeof() on a Pointer Type**
CWE-476	**NULL Pointer Dereference**
CWE-573	**Improper Following of Specification by Caller**
CWE-610	**Externally Controlled Reference to a Resource in Another Sphere**
CWE-665	**Improper Initialization**
CWE-666	**Operation on Resource in Wrong Phase of Lifetime**
CWE-668	**Exposure of Resource to Wrong Sphere**
CWE-670	**Always-Incorrect Control Flow Implementation**
CWE-676	**Use of Potentially Dangerous Function**
CWE-704	**Incorrect Type Conversion or Cast**

CWE-754	**Improper Following of Specification by Caller**
CWE-758	**Externally Controlled Reference to a Resource in Another Sphere**
CWE-770	**Improper Initialization**
CWE-772	**Operation on Resource in Wrong Phase of Lifetime**

- Cài đặt Requirement:

absl-py==0.9.0
astor==0.8.1
attrs==19.3.0
azure-common==1.1.25
azure-nspkg==3.0.2
azure-storage==0.36.0
azureml==0.2.7
backcall==0.1.0
bleach==3.1.4
boto3==1.12.37
botocore==1.15.37
bypy==1.6.8
cachetools==4.0.0
certifi==2020.6.20
cffi==1.14.0
chardet==3.0.4
colorama==0.4.3
contextlib2==0.5.5
coverage==5.0.4
cryptography==2.9
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
dill==0.3.1.1
docopt==0.6.2
docutils==0.15.2
dpu-utils==0.2.11
entrypoints==0.3
et-xmlfile==1.0.1
future==0.18.2
gast==0.2.2
gensim==3.8.1
google-api-core==1.16.0
google-auth==1.13.1
google-auth-oauthlib==0.4.1
google-cloud-core==1.3.0

google-cloud-storage==1.27.0
google-pasta==0.2.0
google-resumable-media==0.5.0
googleapis-common-protos==1.51.0
grpcio==1.28.1
h5py==2.10.0
hyperopt==0.1.2
idna==2.9
importlib-metadata==1.6.0
ipykernel==5.2.0
ipython==7.9.0
ipython-genutils==0.2.0
ipywidgets==7.5.1
jdcal==1.4.1
jedi==0.16.0
Jinja2==2.11.1
jmespath==0.9.5
joblib==0.14.1
json-tricks==3.15.0
jsonlines==1.2.0
jsonschema==3.2.0
jupyter==1.0.0
jupyter-client==6.1.2
jupyter-console==6.1.0
jupyter-core==4.6.3
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
kiwisolver==1.2.0
Markdown==3.2.1
MarkupSafe==1.1.1
matplotlib==3.3.2
mistune==0.8.4
multiprocess==0.70.9
nbconvert==5.6.1
nbformat==5.0.5
networkx==2.4
notebook==6.0.3
numpy==1.18.2
oauthlib==3.1.0
openpyxl==3.0.5
opt-einsum==3.2.1
pandas==1.0.5
pandocfilters==1.4.2
parso==0.6.2
pexpect==4.8.0
pickleshare==0.7.5
Pillow==6.2.2
prometheus-client==0.7.1
prompt-toolkit==2.0.10
protobuf==3.11.3
psutil==5.7.0
ptyprocess==0.6.0
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycparser==2.20
Pygments==2.6.1
pymongo==3.10.1
pyparsing==2.1.10
pyrsistent==0.16.0

python-dateutil==2.8.1
pytz==2019.3
pyzmq==19.0.0
qtconsole==4.7.2
QtPy==1.9.0
requests==2.23.0
requests-oauthlib==1.3.0
requests-toolbelt==0.9.1
rsa==4.0
s3transfer==0.3.3
scikit-learn==0.23.1
scipy==1.4.1
Send2Trash==1.5.0
SetSimilaritySearch==0.1.7
six==1.14.0
sklearn==0.0
smart-open==1.10.0
tensorboard==2.0.2
tensorflow-estimator==2.0.1
tensorflow-gpu==2.0.0
termcolor==1.1.0
terminado==0.8.3
testpath==0.4.4
threadpoolctl==2.1.0
tornado==6.0.4
tqdm==4.45.0
traitlets==4.3.3
urllib3==1.25.8
wcwidth==0.1.9
webencodings==0.5.1
Werkzeug==1.0.1
widgetsnbextension==3.5.1
wrapt==1.12.1
zipp==1.2.0
nni==1.9.0

- Xử lý data: trích xuất code block để xây dựng AST, các edge, joern (joern để bắt các flows về control và data trong code)
- Chạy thực nghiệm.
- Training

- Thông số mặc định của model

cli_utils > default_hypers > {} GraphRegression_GNN_Edge_MLP.json

```
{
    "task_params": {
        "max_nodes_per_batch": 10000
    },
    "model_params": {
        "gnn_num_edge_MLP_hidden_layers": 0,
        "gnn_hidden_dim": 64,
        "gnn_num_layers": 12,
        "gnn_residual_every_num_layers": 2,
        "gnn_global_exchange_every_num_layers": 40,
        "gnn_dense_every_num_layers": 10000,
        "gnn_use_inter_layer_layernorm": true,
        "gnn_layer_input_dropout_rate": 0.1,
        "gnn_message_activation_function": "gelu",
        "gnn_aggregation_function": "sum",
        "gnn_initial_node_representation_activation": "tanh",
        "gnn_dense_intermediate_layer_activation": "tanh",
        "graph_aggregation_hidden_layers": [64],
        "graph_aggregation_size": 32,
        "graph_aggregation_num_heads": 4,
        "learning_rate_decay": 0.98,
        "momentum": 0.85,
        "learning_rate": 0.0001,
        "gradient_clip_value": 1.0,
        "optimizer": "Adam"
    }
}

```

  - max_nodes_per_batch: Số lượng node tối đa trong mỗi batch được sử dụng trong quá trình huấn luyện. Điều này có thể giúp kiểm soát kích thước của các batch trong quá trình huấn luyện.
  - gnn_num_edge_MLP_hidden_layers: Số lớp ẩn trong mô hình MLP (Multilayer Perceptron) được sử dụng để xử lý thông tin về cạnh (edge) của đồ thị.
  - gnn_hidden_dim: Số lượng chiều của biểu diễn ẩn cho mỗi node trong mạng GNN.
  - gnn_num_layers: Số lớp trong mạng GNN.

- gnn_residual_every_num_layers: Tần suất thêm residual connections trong quá trình xây dựng lớp GNN.
- gnn_global_exchange_every_num_layers: Tần suất thực hiện trao đổi thông tin toàn cục giữa các node trong đồ thị.
- gnn_dense_every_num_layers: Tần suất thêm lớp dense trong mạng GNN.
- gnn_use_inter_layer_layernorm: Sử dụng layer normalization giữa các lớp GNN.
- gnn_layer_input_dropout_rate: Tỷ lệ dropout được áp dụng cho đầu vào của mỗi lớp GNN.
- gnn_message_activation_function: Hàm kích hoạt được sử dụng cho thông điệp giữa các node.
- gnn_aggregation_function: Phương pháp tổng hợp thông tin của các node láng giềng (neighbor) trong mỗi lớp GNN (ví dụ: "sum" - tổng hợp theo tổng).
- gnn_initial_node_representation_activation: Hàm kích hoạt được sử dụng cho biểu diễn ban đầu của mỗi node.
- gnn_dense_intermediate_layer_activation: Hàm kích hoạt được sử dụng cho lớp dense giữa các lớp GNN.
- graph_aggregation_hidden_layers: Số lớp ẩn trong mô hình MLP được sử dụng trong quá trình tổng hợp đồ thị.
- graph_aggregation_size: Kích thước biểu diễn đồ thị sau quá trình tổng hợp.
- graph_aggregation_num_heads: Số lượng đầu ra (heads) trong tầng tổng hợp đồ thị.
- learning_rate_decay: Hệ số giảm learning rate sau mỗi epoch.
- momentum: Hệ số momentum được sử dụng trong quá trình tối ưu hóa.
- learning_rate: Tốc độ học của mô hình.

- gradient_clip_value: Giá trị tuyến tính được sử dụng để cắt tỉa (clip) gradient.
- optimizer: Thuật toán tối ưu hóa được sử dụng (ví dụ: "Adam").

- Khi model chạy thì đầu tiên sẽ load config nhờ vào 2 hàm này
  - Nni_config: load parameters từ file GraphBinaryClassification_GGNN.json

```
CodiumAI: Options | Test this function
def nni_config():
    json_file = "../cli_utils/default_hypers/GraphBinaryClassification_GGNN.json"

    params = {"model_params":{\
     "gnn_aggregation_function": "sum",\
     "gnn_message_activation_function": "relu",\
     "gnn_hidden_dim": 16
     }}
    try:
        model_params = nni.get_next_parameter()
        params["model_params"].update(model_params)
        if "graph_aggregation_hidden_layers" in model_params.keys():
            params["model_params"]["graph_aggregation_hidden_layers"] = \
                [model_params["graph_aggregation_hidden_layers"]]
    except Warning:
        pass

    with open(json_file, 'w') as f:
        json.dump(params, f)
        print("Parameter Config:", params)

```

- get_train_cli_arg_parser: dựng thông số

```
CodiumAI: Options | Test this function
def get_train_cli_arg_parser():
    import argparse

    parser = argparse.ArgumentParser(description="Train a GNN model.")
    if "--model" in sys.argv:
        model_param_name, task_param_name, data_path_param_name = "--model", "--task", "--data_pat
    else:
        model_param_name, task_param_name, data_path_param_name = "model", "task", "data_path"

    parser.add_argument(
        "--task_model_default_hypers_filePath",
        dest="task_model_default_hypers_filePath",
        type=str,
        default="../default_hypers/GraphBinaryClassification_GGNN.json",
        help="load the task_model_default_hypers",
    )
    parser.add_argument(
        "--storedModel_path",
        dest="storedModel_path",
        type=str,
        default="../cli/trained_model/GGNN_GraphBinaryClassification__2023-02-01_05-36-00_f1 = 0.8
        help="load the model which is trained before",
    )
    parser.add_argument(
        model_param_name,
        type=str,
        choices=sorted(get_known_message_passing_classes()),
        help="GNN model type to train.",
    )
    parser.add_argument(
        task_param_name,
        type=str,
```

- Sau đó model sẽ được dựng lên nhờ vào các thông số đã nêu bằng việc sử dụng hàm run_train_from args

```
231 def run_train_from_args(args, hyperdrive_hyperparameter_overrides: Dict[str, str] = {}) -> Nor
232     # Get the housekeeping going and start logging:
233     os.makedirs(args.save_dir, exist_ok=True)
234     run_id = make_run_id(args.model, args.task)
235     log_file = os.path.join(args.save_dir, f"{run_id}.log")
```

Khi chạy sẽ có 2 loại data path (ast, cdfg) sẽ được dựng lên từ 2 loại dataset để từ đó 2 model được dựng lên

```
#data split

DataSplit.Preprocess(args.data_path)
data_path = RichPath.create(os.path.split(args.data_path)[0]+'/tem_'+os.path.split(args.data_path)[1]+'/as
args.azure_info)
#second path
data_path_2 = RichPath.create(os.path.split(args.data_path)[0]+'/tem_'+os.path.split(args.data_path)[1]+'/
args.azure_info)
```

```
try:
    dataset, model = get_model_and_dataset(
        msg_passing_implementation=args.model,
        task_name=args.task,
        data_path=data_path,
        trained_model_file=args.load_saved_model,
        cli_data_hyperparameter_overrides=args.data_param_override,
        cli_model_hyperparameter_overrides=args.model_param_override,
        hyperdrive_hyperparameter_overrides=hyperdrive_hyperparameter_overrides
        folds_to_load={DataFold.TRAIN, DataFold.VALIDATION},
        load_weights_only=args.load_weights_only,
    )
    #second
    dataset2, model_2 = get_model_and_dataset(
        msg_passing_implementation=args.model,
        task_name=args.task,
        data_path=data_path_2,
        trained_model_file=args.load_saved_model,
        cli_data_hyperparameter_overrides=args.data_param_override,
        cli_model_hyperparameter_overrides=args.model_param_override,
        hyperdrive_hyperparameter_overrides=hyperdrive_hyperparameter_overrides
        folds_to_load={DataFold.TRAIN, DataFold.VALIDATION},
        load_weights_only=args.load_weights_only,
    )
```

Khi dựng model sẽ cần tới hàm get_model_and_dataset, trong đây sẽ xét 2 trường hợp đã có model và chưa có model

```
def get_model_and_dataset(
    task_name: Optional[str],
    msg_passing_implementation: Optional[str],
    data_path: RichPath,
    trained_model_file: Optional[str],
    cli_data_hyperparameter_overrides: Optional[str],
    cli_model_hyperparameter_overrides: Optional[str],
    hyperdrive_hyperparameter_overrides: Dict[str, str] = {},
    folds_to_load: Optional[Set[DataFold]] = None,
    load_weights_only: bool = False,
):
    # case of a trained model file being passed, where the entire model should be loaded,
    # a new class and dataset type are not required
    print(f"this is trained_model_file:{trained_model_file},load_weights_only:{load_weights_only} ")
    if trained_model_file and not load_weights_only:
        with open(get_model_file_path(trained_model_file, "pkl"), "rb") as in_file:
            data_to_load = pickle.load(in_file)
        model_class = data_to_load["model_class"]
        dataset_class = data_to_load["dataset_class"]
        default_task_model_hypers = {}
    # case 1: trained_model_file and loading weights only -- create new dataset and class of the type specified by
    # task to be trained, but use weights from another corresponding model
    # case 2: no model to be loaded; make fresh dataset and model classes
    elif (trained_model_file and load_weights_only) or not trained_model_file:
        data_to_load = {}
        model_class, dataset_class = None, None

        # Load potential task-specific defaults:
        default_task_model_hypers = {}
        task_model_default_hypers_file = os.path.join(
            os.path.dirname(__file__),
            "default_hypers",
```

Nếu có sẵn model thì chỉ việc tiến hành tạo dataset và task mới cho model để tiến hành huấn luyện, còn không có sãn model thì tạo dataset để tiến hành train model mới

- Model được tạo nên

```
model = get_model(
    msg_passing_implementation,
    task_name,
    model_class,
    dataset,
    dataset_model_optimised_default_hyperparameters=default_task_model_hypers.get(
        "model_params", {}
    ),
    loaded_model_hyperparameters=data_to_load.get("model_params", {}),
    cli_model_hyperparameter_overrides=json.loads(
        cli_model_hyperparameter_overrides or "{}"
    ),
    hyperdrive_hyperparameter_overrides=hyperdrive_hyperparameter_overrides or {},
)
```

```
116 def get_model(
117     msg_passing_implementation: str,
118     task_name: str,
119     model_cls: Type[GraphTaskModel],
120     dataset: GraphDataset,
121     dataset_model_optimised_default_hyperparameters: Dict[str, Any],
122     loaded_model_hyperparameters: Dict[str, Any],
123     cli_model_hyperparameter_overrides: Dict[str, Any],
124     hyperdrive_hyperparameter_overrides: Dict[str, str],
125 ) -> GraphTaskModel:
126     if not model_cls:
127         model_cls, model_default_hyperparameter_overrides = task_name_to_model_class(
128             task_name
129         )
130         model_params = model_cls.get_default_hyperparameters(msg_passing_implementation)
131         print(f" Model default parameters: {model_params}")
132         model_params.update(model_default_hyperparameter_overrides)
133         if len(model_default_hyperparameter_overrides):
134             print(
135                 f"  Model parameters overridden by task defaults: {model_default_hyperparameter_overri
136             )
137         model_params.update(dataset_model_optimised_default_hyperparameters)
138         if len(dataset_model_optimised_default_hyperparameters):
139             print(
140                 f"  Model parameters overridden by task/model defaults:
                    {dataset_model_optimised_default_hyperparameters}"
141             )
142     else:
143         model_params = loaded_model_hyperparameters
144     model_params.update(cli_model_hyperparameter_overrides)
145     if len(cli_model_hyperparameter_overrides):
146         print(
147             f"  Model parameters overridden from CLI: {cli_model_hyperparameter_overrides}"
148         )
```

- Sau khi dựng xong model thì sẽ chạy tới hàm train

```
CodiumAI: Options | Test this function
def train(
    model: GraphTaskModel,
    dataset: GraphDataset,
    dataset2:GraphDataset,
    log_fun: Callable[[str], None],
    run_id: str,
    max_epochs: int,
    patience: int,
    save_dir: str,
    quiet: bool = False,
    aml_run=None,
```

Trong vòng tối đa 25 epochs nếu như không có cải thiện về metrics thì sẽ ngưng

```
    # Save if good enough.
    if best_val_strf1 > BestValidf1:
        log_fun(
            f"  (Best epoch so far, target metric decreased to {best_val_strf1} from {BestValidf1}.)",
        )
        save_file = os.path.join(save_dir, f"{run_id}_{best_val_strf1}_best.pkl")
        save_model(save_file, model, dataset)
        BestValidf1 = best_val_strf1
        best_valid_epoch = epoch
    elif epoch - best_valid_epoch >= patience:
        total_time = time.time() - train_time_start
        log_fun(
            f"Stopping training after {patience} epochs without "
            f"improvement on validation metric.",
        )
        log_fun(f"Training took {total_time}s. Best validation metric: {best_valid_ACC}", )
        break
return save_file
```

```
    )
    parser.add_argument(
        "--patience",
        dest="patience",
        type=int,
        default=25,
        help="Maximal number of epochs to continue training without improvemen
    )
```

- Testing: tương tự training ở các bước đầu nhưng thay vì chạy hàm run_train_from_args thì chương trình sẽ chạy hàm loadModuleAndPredict. Hàm này tương tự các bước nạp dataset và model xong tới việc test bằng chạy hàm prediction để tính metrics

-
```
在data_preprocess.py中数据处理将输入的全部数据都放入test.json中,所以只需要对DataFold.TEST 预测
dataset.load_data(data_path, {DataFold.TEST})
dataset2.load_data(data_path_2, {DataFold.TEST})
test_data_1 = dataset.get_tensorflow_dataset(DataFold.TEST)
test_data_2 = dataset2.get_tensorflow_dataset(DataFold.TEST)
model.prediction(
    test_data_1, test_data_2
)
```

-
```
CodiumAI: Options | Test this method
def prediction(
        self, dataset: tf.data.Dataset, dataset2: tf.data.Dataset, quiet: bool = False, training: bool = F
) -> Tuple[float, float, List[Any]]:

    task_outputs=[]
    for ((step, (batch_features, batch_labels)), (step_2, (batch_features_2, batch_labels_2))) in zip(
            enumerate(dataset), enumerate(dataset2)):
        task_output = self(batch_features, batch_features_2, training=training)
        task_outputs.append(np.argmax(task_output, axis=1))
    print(f"task_outputs:{task_outputs}")

    __, _, test_results = self.run_one_epoch_new(dataset, dataset2, training=False, quiet=False)

    valid_ACC, val_stracc, \
    best_valid_Pre, best_val_strpre, \
    best_valid_metric_RE, best_val_strre, \
    best_valid_metric_f1, best_val_strf1, \
    best_valid_metric_TPR, best_val_strtpr, \
    best_valid_metric_FPR, best_val_strfpr, \
    best_valid_metric_TNR, best_val_strtnr, \
    best_valid_metric_FNR, best_val_strfnr, = self.compute_epoch_metrics(test_results)
    print(
        f"task_metric:|{val_stracc}|{best_val_strpre} | {best_val_strre} | {best_val_strf1} |"
        # f"{best_val_strtpr} | {best_val_strfpr} | {best_val_strtnr} | {best_val_strfnr} |",
    )
```

C. Kết quả thực nghiệm

<mô tả hình ảnh về thực nghiệm, bảng biểu số liệu thống kê từ thực nghiệm, nhận xét về kết quả thu được.>

- Kết quả của tác giả

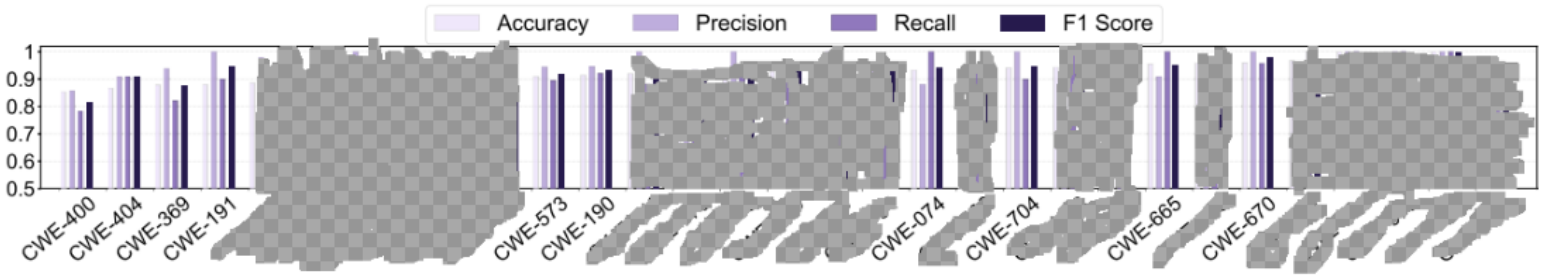


Fig. 8. FUNDED delivers on average, an accuracy of 92%, for detecting C functions with the top-30 CWE vulnerabilities.

- Kết quả thu được

	CWE-074	CWE-190	CWE-191	CWE-369	CWE-400	CWE-404
Accruacy	49.9%	(1)53.6% (2)47.4%	(1)55.1% (2)44.5% (3)45.4%	(1)60.8% (2)61.3% (3)62.1% (4)60.5% (5)59.5% (6)56.8%	52.8%	51.4%
Precision	50.5%	(1)61.7% (2)39.7%	(1)67.2% (2)38.3% (3)39.2%	(1)69.2% (2)73% (3)72.8% (4)68.7% (5)69.6% (6)70.1%	52%	51%
Recall	76%	(1)66% (2)78.2%	(1)67.6% (2)62.3% (3)74.1%	(1)78.4% (2)74% (3)75.3%	78.1%	76%

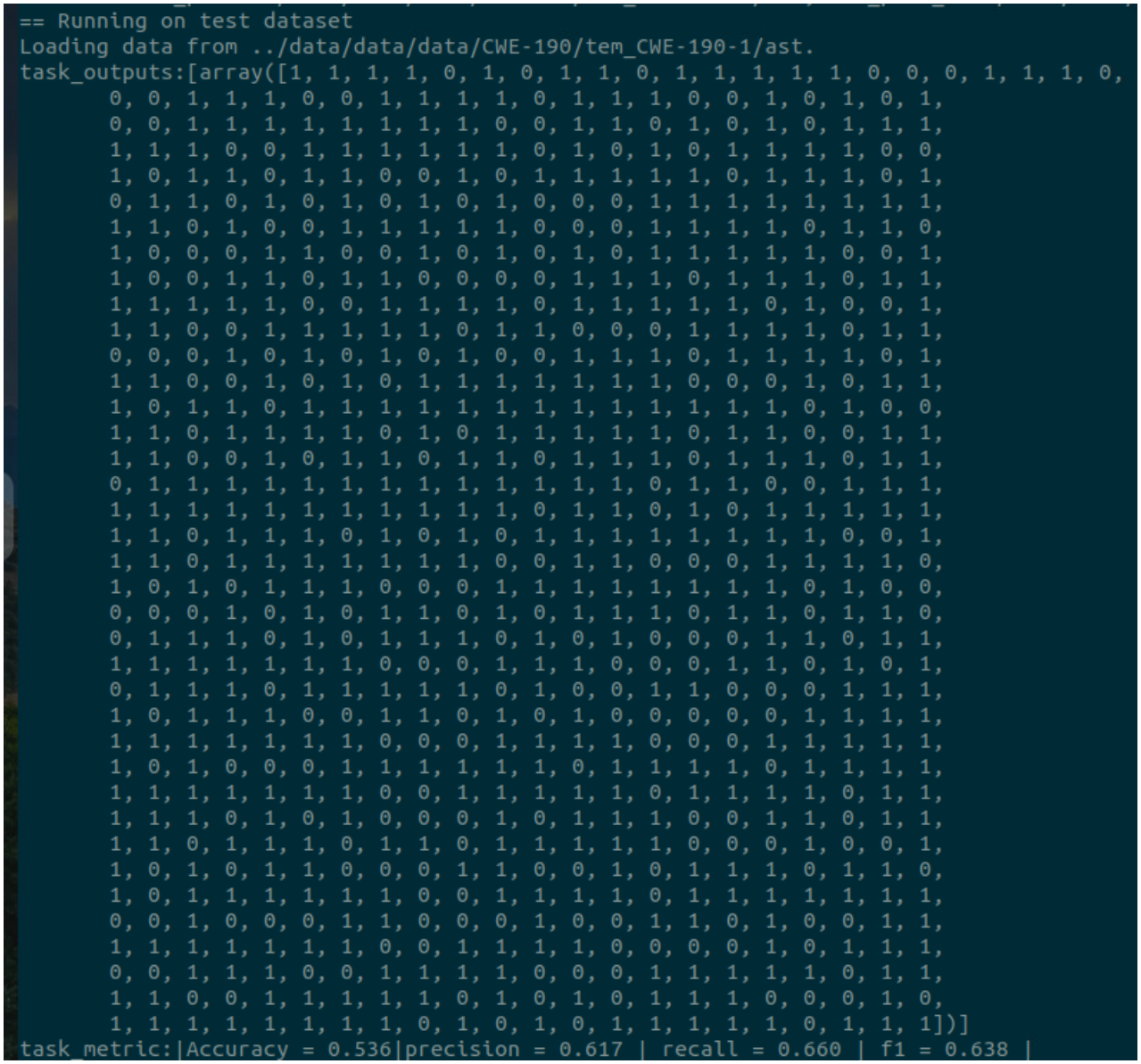
				(4)78.6% (5)74.1% (6)69.7%		
F1	60.7%	(1)63.8% (2)52.6%	(1)67.4% (2)47.5% (3)51.3%	(1)73.5% (2)73.5% (3)74% (4)73.3% (5)71.7% (6)69.9%	62.4%	61.1%

CWE-573	CWE-665	CWE-670	CWE-704	Badall
45.6%	(1)48.6% (2)53.8%	49.4%	49.4%	73.5%
46.7%	(1)41% (2)59%	49.3%	49.6%	100%
72.4%	(1)60.2% (2)70.5%	75.5%	83%	73.5%
56.8%	(1)48.8% (2)64.3%	59.6%	62.1%	84.7%

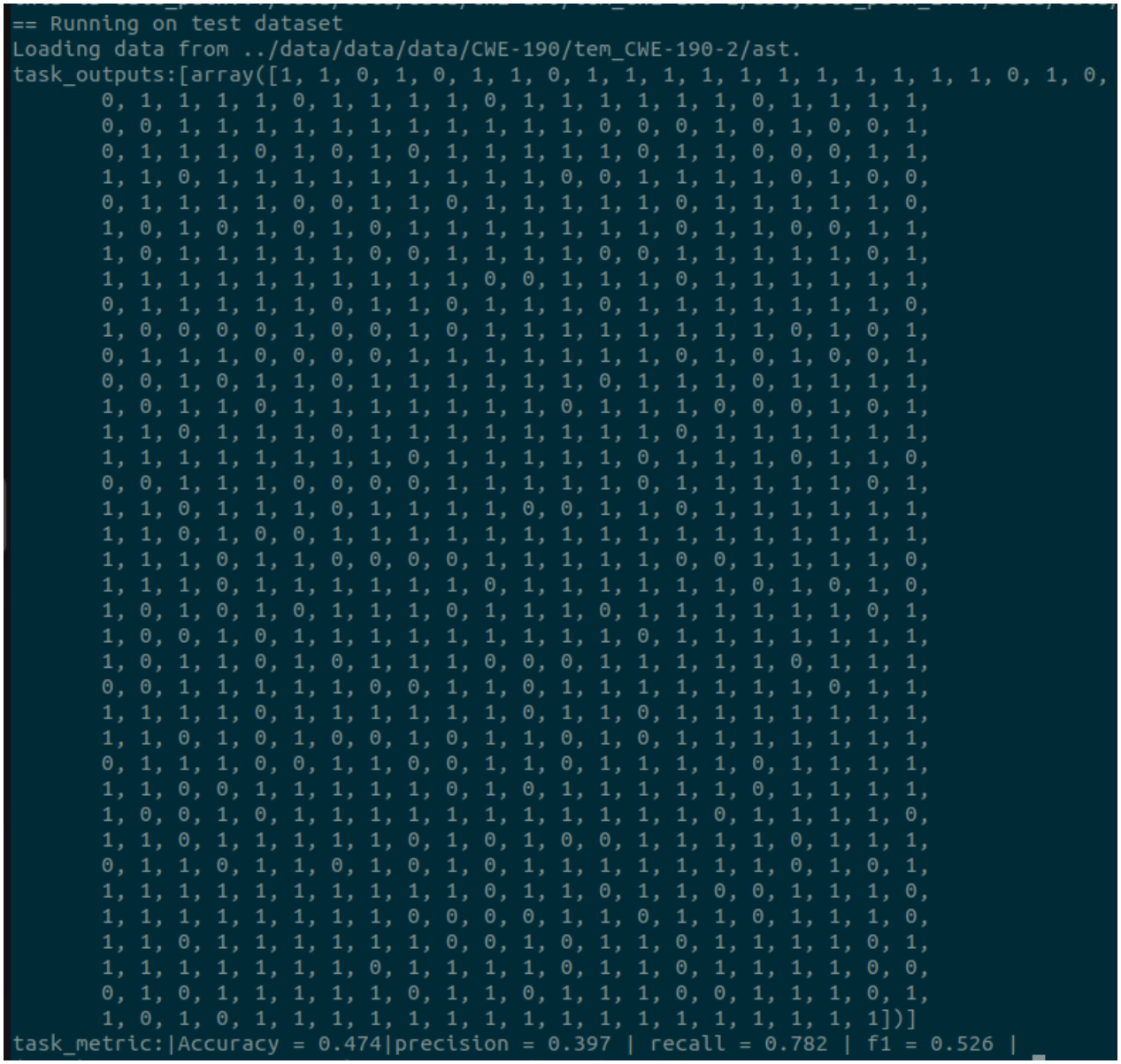
CWE-074

```
this is data_path:../data/data/data/sample_test/tem_CWE-074/ast,data_path_2:../data/data/data/sample_test/tem_CWE-074/cdfg
== Running on test dataset
Loading data from ../data/data/data/sample_test/tem_CWE-074/ast.
task_outputs:[array([1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1])]
task_metric:|Accuracy = 0.499|precision = 0.505 | recall = 0.760 | f1 = 0.607 |
```

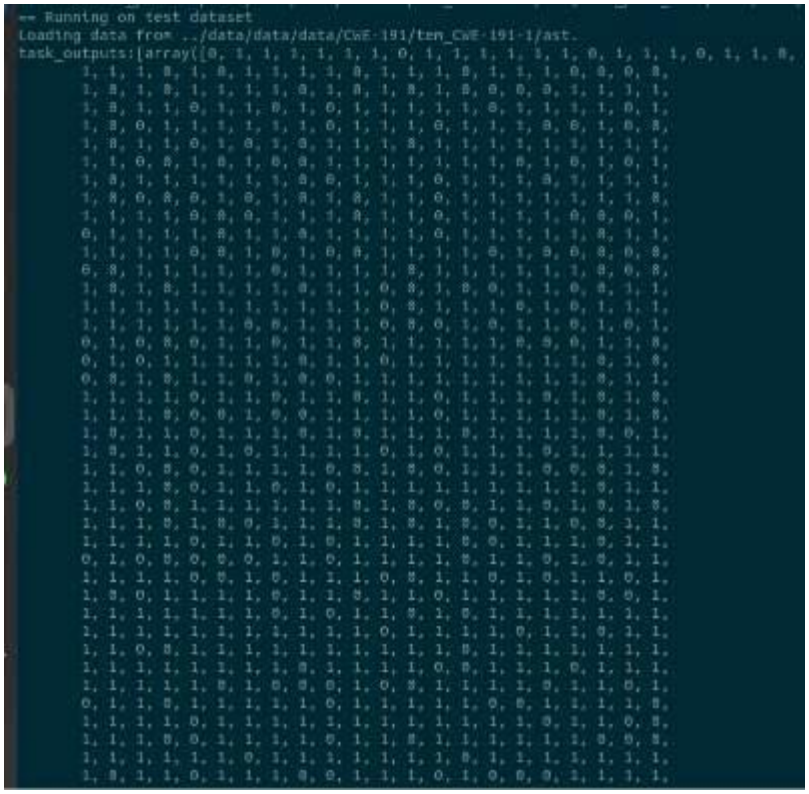
CWE-190-1

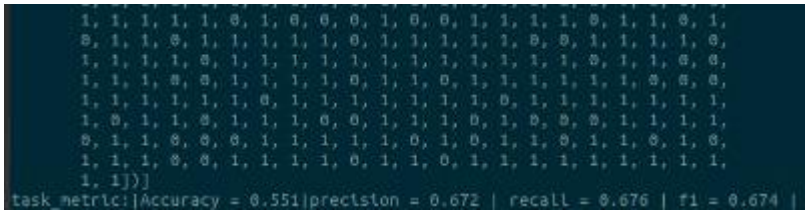

== Running on test dataset
Loading data from ../data/data/data/CWE-190/tem_CWE-190-1/ast.
task_outputs:[array([1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1])]
task_metric:|Accuracy = 0.536|precision = 0.617 | recall = 0.660 | f1 = 0.638 |

CWE-190-2

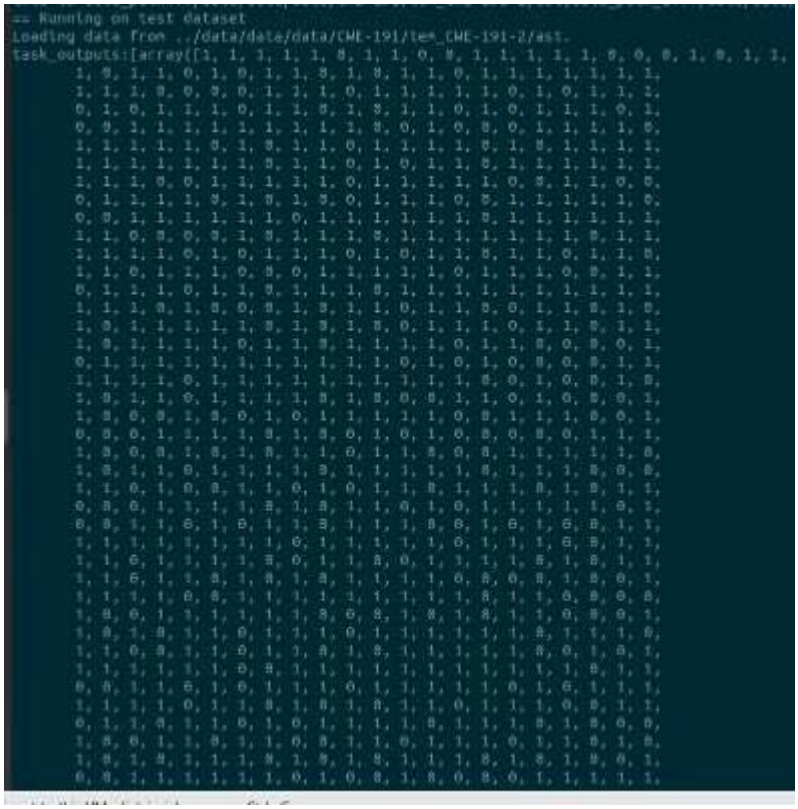

== Running on test dataset
Loading data from ../data/data/data/CWE-190/tem_CWE-190-2/ast.
task_outputs:[array([1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])]
task_metric:|Accuracy = 0.474|precision = 0.397 | recall = 0.782 | f1 = 0.526 |

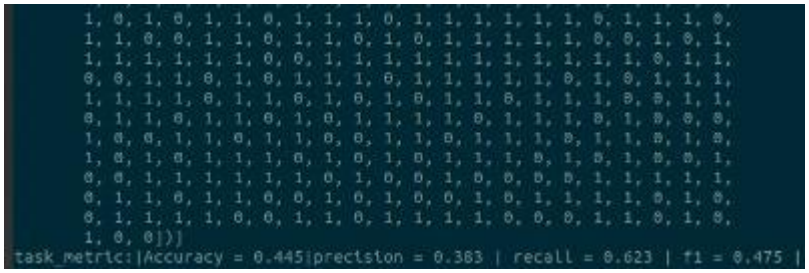
CWE-191-1


-- Running on test dataset
Loading data from ../data/data/data/CWE-191/ten_CWE-191-1/ast.
task_outputs:[array([0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,

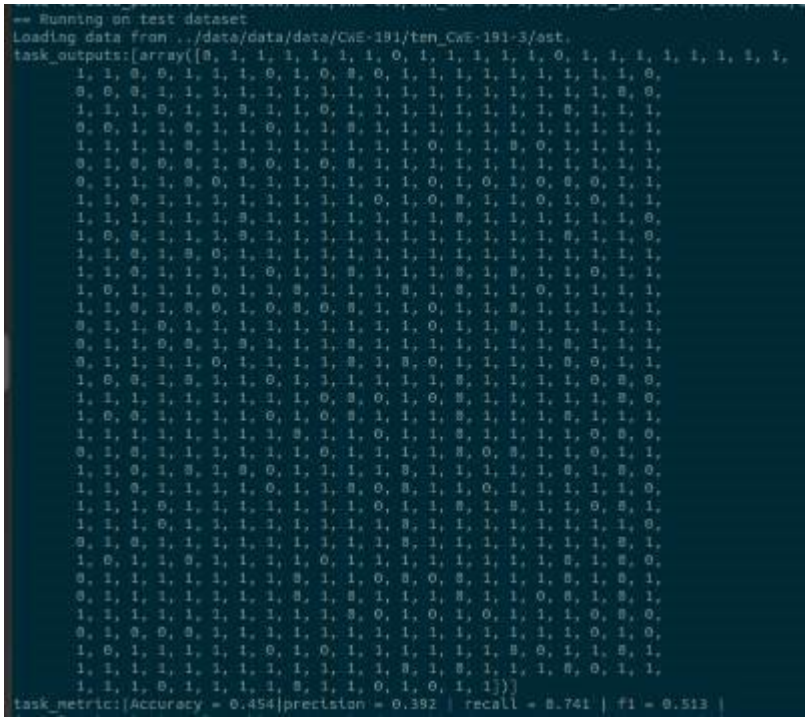

task_metric:|Accuracy = 0.551|precision = 0.672 | recall = 0.676 | f1 = 0.674 |

CWE-191-2

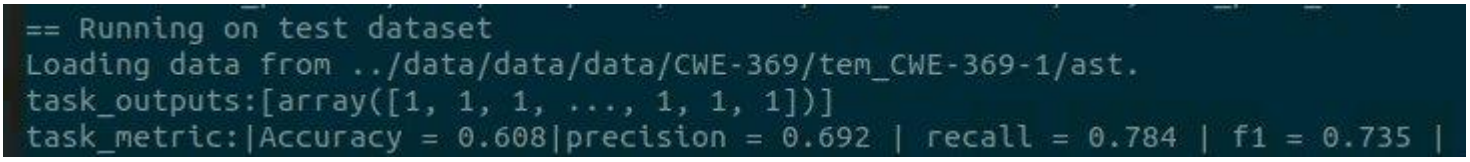

== Running on test dataset
Loading data from ../data/data/data/CWE-191/len_CWE-191-2/ast.


task_metric:|Accuracy = 0.445|precision = 0.383 | recall = 0.623 | f1 = 0.475 |

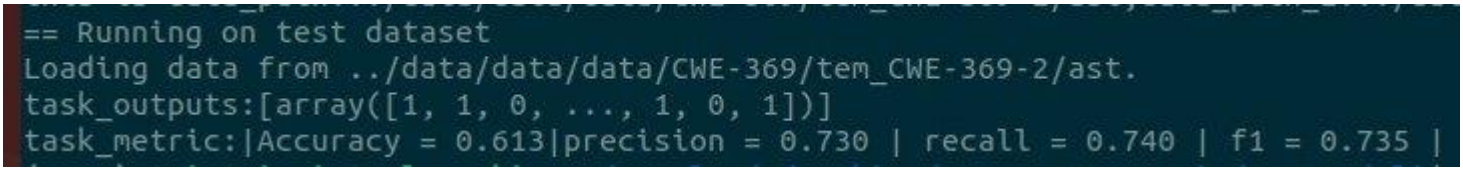
CWE-191-3



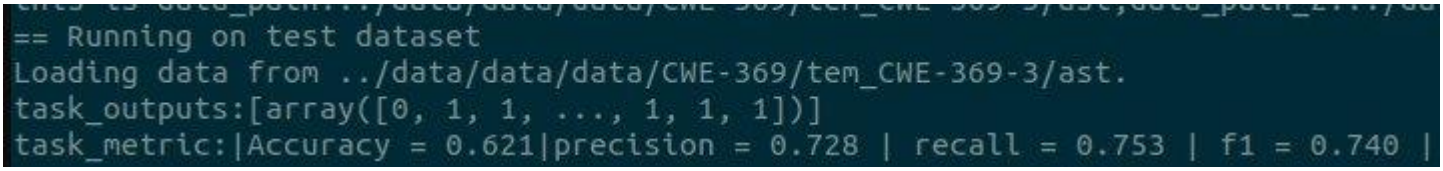
CWE-369-1



CWE-369-2



CWE-369-3



CWE-369-4

```
== Running on test dataset
Loading data from ../data/data/data/CWE-369/tem_CWE-369-4/ast.
2023-12-19 16:25:23.326252: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 172694000 exceeds 10% of system memory.
2023-12-19 16:25:31.419657: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 108548800 exceeds 10% of system memory.
2023-12-19 16:25:40.282076: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 113077056 exceeds 10% of system memory.
2023-12-19 16:25:40.744047: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 113077056 exceeds 10% of system memory.
2023-12-19 16:25:41.304532: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 113077056 exceeds 10% of system memory.
task_outputs:[array([1, 1, 0, ..., 1, 1, 1])]
task_metric:|Accuracy = 0.605|precision = 0.687 | recall = 0.786 | f1 = 0.733 |
```

CWE-369-5

```
== Running on test dataset
Loading data from ../data/data/data/CWE-369/tem_CWE-369-5/ast.
task_outputs:[array([0, 1, 1, ..., 1, 1, 1])]
task_metric:|Accuracy = 0.595|precision = 0.696 | recall = 0.741 | f1 = 0.717 |
```

CWE-369-6

```
== Running on test dataset
Loading data from ../data/data/data/CWE-369/tem_CWE-369-6/ast.
task_outputs:[array([1, 1, 1, ..., 0, 1, 1])]
task_metric:|Accuracy = 0.568|precision = 0.701 | recall = 0.697 | f1 = 0.699 |
```

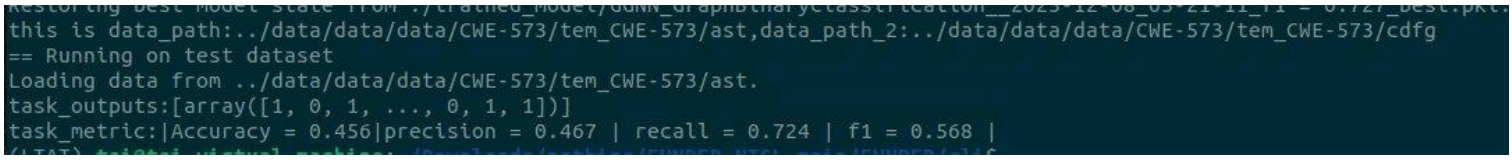
CWE-400

```
this is data_path:../data/data/data/CWE-400/tem_CWE-400/ast,data_path_2:../data/data/data/CWE-400/tem_CWE-400/cdfg
== Running on test dataset
Loading data from ../data/data/data/CWE-400/tem_CWE-400/ast.
2023-12-19 11:22:57.516047: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 116948800 exceeds 10%
2023-12-19 11:23:02.818346: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 68381200 exceeds 10%
2023-12-19 11:23:09.898135: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 56135424 exceeds 10%
2023-12-19 11:23:09.899753: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 37324032 exceeds 10%
2023-12-19 11:23:09.989409: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 37324032 exceeds 10%
task_outputs:[array([1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])]
task_metric:|Accuracy = 0.528|precision = 0.520 | recall = 0.781 | f1 = 0.624 |
```

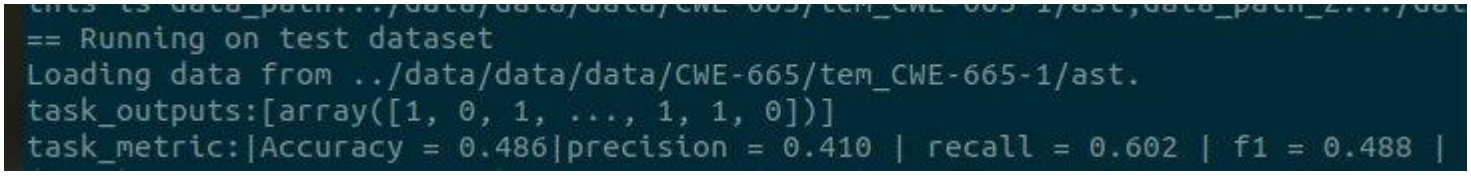
CWE-404

```
this is data_path:../data/data/data/CWE-404/tem_CWE-404/ast,data_path_2:../data/data/data/CWE-404/
== Running on test dataset
Loading data from ../data/data/data/CWE-404/tem_CWE-404/ast.
task_outputs:[array([1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1])]
task_metric:|Accuracy = 0.514|precision = 0.510 | recall = 0.760 | f1 = 0.611 |
```

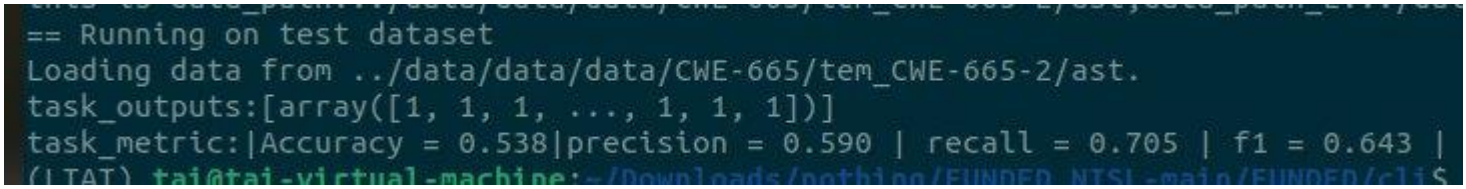
CWE-573



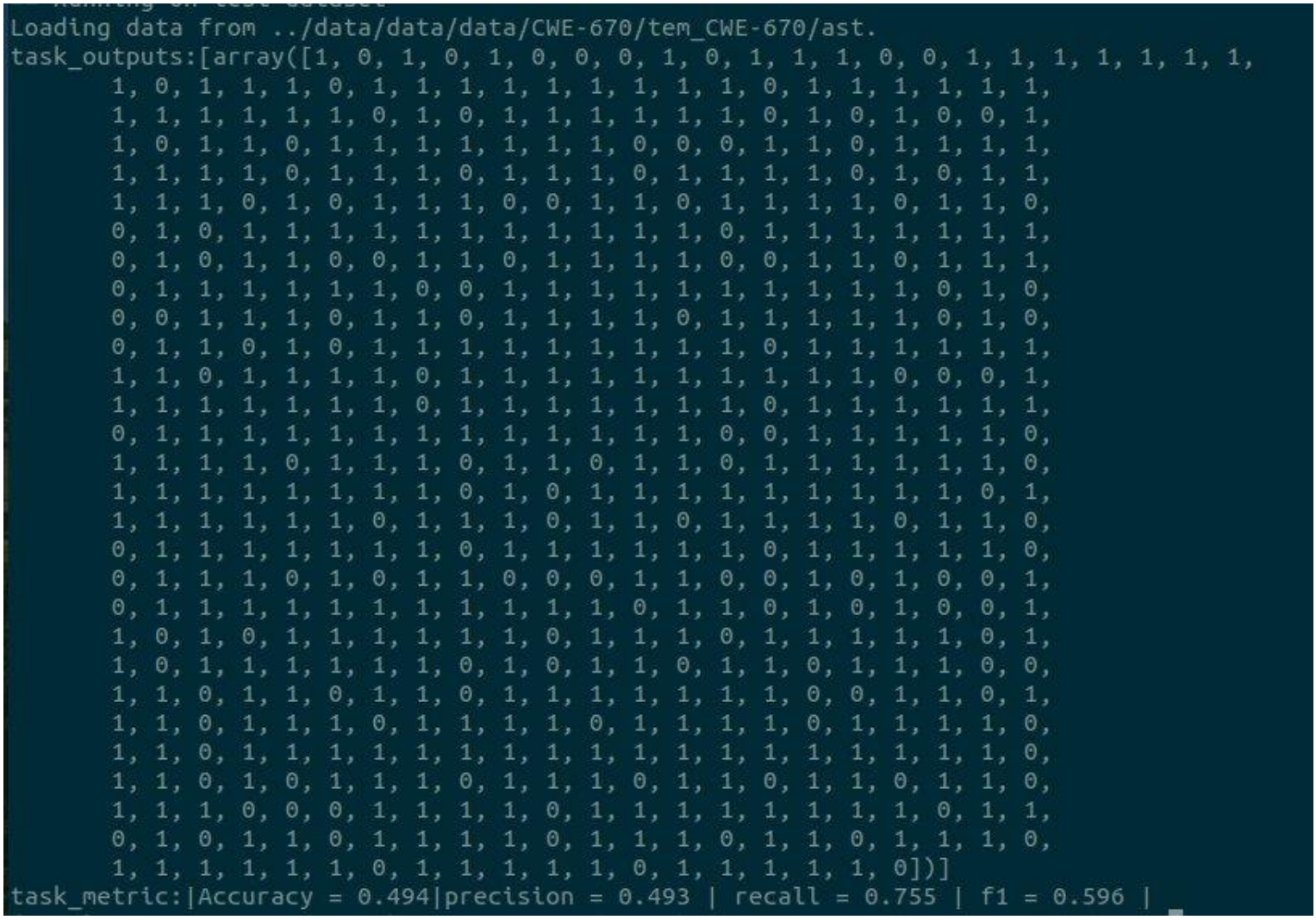
CWE-665-1



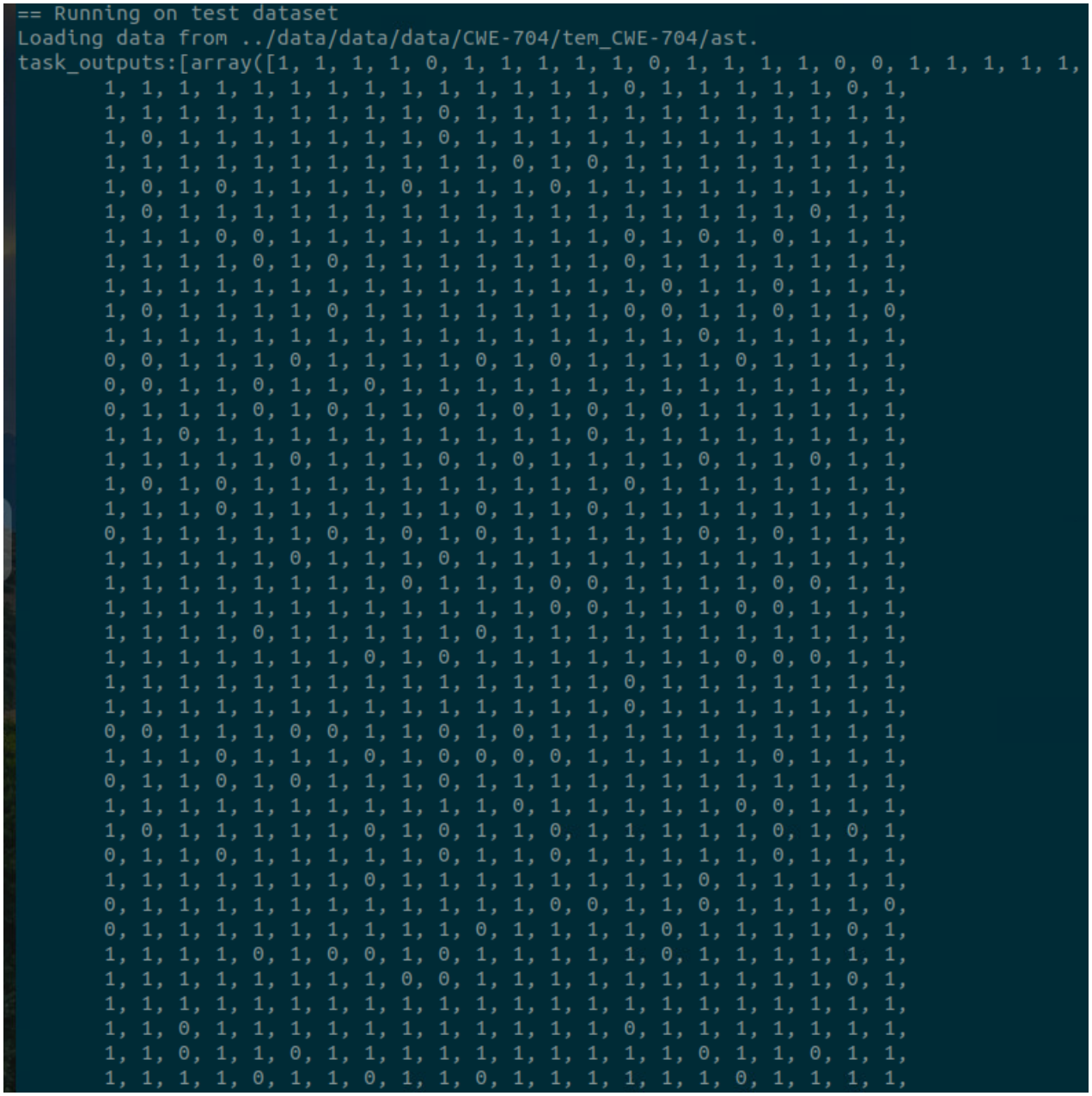
CWE-665-2



CWE-670


Loading data from ../data/data/data/CWE-670/tem_CWE-670/ast.
task_outputs:[array([1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0])]
task_metric:|Accuracy = 0.494|precision = 0.493 | recall = 0.755 | f1 = 0.596 |

CWE-704


== Running on test dataset
Loading data from ../data/data/data/CWE-704/tem_CWE-704/ast.
task_outputs:[array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0])]
task_metric:|Accuracy = 0.494|precision = 0.496 | recall = 0.830 | f1 = 0.621 |

Badall

```
Restoring best model state from ./trained_model/GGNN_GraphBinaryClassification__2023-12-08_03-21-11_f1 = 0.727_best.pkl.
this is data_path:../data/data/data/cve/tem_badall/ast,data_path_2:../data/data/data/cve/tem_badall/cdfg
== Running on test dataset
Loading data from ../data/data/data/cve/tem_badall/ast.
task_outputs:[array([1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 0])]
task_metric:|Accuracy = 0.735|precision = 1.000 | recall = 0.735 | f1 = 0.847 |
```

- Nhận xét:

Trong quá trình training tới 73000 samples nên khi ra kết quả nhóm được model với F1 = 0.727 so với model chính của tác giả có F1 = 0.80 thì sự khác biệt về kết quả không có lớn nhưng nói chung kết quả của cả 2 model đều thấp như nhau. Lý do vì khi chạy test data thì framework chỉ có thể chạy với input có dưới 1500 samples nên khi chạy chúng em phải chia nhỏ sample ra.

Link source code của tác giả - github: Link1

Link dataset đã train của nhóm (đăng nhập bằng acc onedrive cung cấp bởi UIT): Link2

D. Hướng phát triển

Có thể ứng dụng vào các phần mềm anti-virus. Nhưng vì đây là phiên bản tối ưu hơn so với các model quét source code truyền thống nên không có gì đặc biệt hơn.
