

# BÁO CÁO THỰC HÀNH

Môn học: Phương pháp học máy cho an toàn thông tin

Lab 3: Advanced Malware Detection

GVHD: Nguyễn Hữu Quyền

**1. THÔNG TIN CHUNG:**

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT522.021.ATCL.2

Nhóm: 7

STT	Họ và tên	MSSV	Email
1	Hồ Ngọc Thiện	21522620	21522620@gm.uit.edu.vn

**2. NỘI DUNG THỰC HIỆN:<sup>1</sup>**

STT	Công việc	Kết quả tự đánh giá
1	Bài tập 1	100%
2	Bài tập 2	100%
3	Bài tập 3	100%
4	Bài tập 4	100%
5	Bài tập 5	100%
6	Bài tập 6	100%
7	Bài tập 7	100%
8	Bài tập 8	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

<sup>1</sup> Ghi nội dung công việc, các kịch bản trong bài Thực hành

## BÁO CÁO CHI TIẾT

### ® Bài tập (yêu cầu làm)

1. Cho biết kết quả accuracy và confusion matrix.

```
import os
from sklearn.feature_extraction.text import HashingVectorizer, TfidfTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.pipeline import Pipeline

js_path = "JavascriptSamples"
obfuscated_js_path = "JavascriptSamplesObfuscated"
corpus = []
labels = []
file_types_and_labels = [(js_path, 0), (obfuscated_js_path, 1)]

for files_path, label in file_types_and_labels:
    files = os.listdir(files_path)
    for file in files:
        file_path = files_path + "/" + file
        try:
            with open(file_path, "r") as myfile:
                data = myfile.read().replace("\n", "")
                data = str(data)
                corpus.append(data)
                labels.append(label)
        except:
            pass

X_train, X_test, y_train, y_test = train_test_split( corpus, labels, test_size=0.33, random_state=42 )
text_clf = Pipeline([
    ("vect", HashingVectorizer(input="content", ngram_range=(1,3))),
    ("tfidf", TfidfTransformer(use_idf=True,)),
    ("rf", RandomForestClassifier(class_weight="balanced")),
])

text_clf.fit(X_train, y_train)
y_test_pred = text_clf.predict(X_test)
print(accuracy_score(y_test, y_test_pred))
print(confusion_matrix(y_test, y_test_pred))
```

-Code:

```
import os

from sklearn.feature_extraction.text import HashingVectorizer, TfidfTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
from sklearn.pipeline import Pipeline

js_path = "JavascriptSamples"
obfuscated_js_path = "JavascriptSamplesObfuscated"
corpus = []
labels = []
file_types_and_labels = [(js_path, 0), (obfuscated_js_path, 1)]

for files_path, label in file_types_and_labels:
    files = os.listdir(files_path)
    for file in files:
        file_path = files_path + "/" + file
        try:
            with open(file_path, "r") as myfile:
                data = myfile.read().replace("\n", "")
                data = str(data)
                corpus.append(data)
                labels.append(label)
        except:
            pass

X_train, X_test, y_train, y_test = train_test_split( corpus, labels, test_size=0.33,
random_state=42 )
text_clf = Pipeline([
    ("vect", HashingVectorizer(input="content", ngram_range=(1,3))),
    ("tfidf", TfidfTransformer(use_idf=True,)),
    ("rf", RandomForestClassifier(class_weight="balanced")),
])

text_clf.fit(X_train, y_train)
y_test_pred = text_clf.predict(X_test)
print(accuracy_score(y_test, y_test_pred))
print(confusion_matrix(y_test, y_test_pred))
```

-Kết quả lần lượt là accuracy và ma trận confusion:

```
(kali㉿kali)-[~/NT522/Lab3]
$ python3 ex1.py
0.9703770197486535
[[622 13]
 [ 20 459]]
```

EX2

® Bài tập (yêu cầu làm)

2. Cho biết kết quả vector X.

```
1 import subprocess
2 from os import listdir
3
4 def PDF_to_FV(file_path):
5     """Featurize a PDF file using pdfid."""
6     result = subprocess.run(["pdfid", file_path], capture_output=True, text=True)
7     out = result.stdout
8     out1 = out.split("\n")[2:-2]
9     return [int(x.split()[-1]) for x in out1]
10
11 PDFs_path = "PDFSamples/"
12 X = []
13 files = listdir(PDFs_path)
14 for file in files:
15     file_path = PDFs_path + file
16     X.append(PDF_to_FV(file_path))
17
18 print(X)
19 |
```

-Code:

```
import subprocess
```

```
from os import listdir
```

```
def PDF_to_FV(file_path):
```

```
    """Featurize a PDF file using pdfid."""
```

```
    result = subprocess.run(["pdfid", file_path], capture_output=True, text=True)
```

```
    out = result.stdout
```

```
    out1 = out.split("\n")[2:-2]
```

```
    return [int(x.split()[-1]) for x in out1]
```

```
PDFs_path = "PDFSamples/"
```

```
X = []
files = listdir(PDFs_path)
for file in files:
    file_path = PDFs_path + file
    X.append(PDF_to_FV(file_path))
```

```
print(X)
```

-Kết quả vector x

```
(kali@kali)-[~/NT522/Lab3]
$ python3 ex2.py
[[1096, 1095, 1061, 1061, 0, 0, 2, 32, 0, 43, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0], [153, 153, 82, 82, 2, 2, 2, 7, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0]]
```

EX3:

**Bài tập (yêu cầu làm)**  
**3. Cho biết kết quả vector X.**

-Code:

```
from os import listdir
from nltk import ngrams
import hashlib

directories = ["Benign 3", "Malware 3"]
N = 2

def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
    with open(file_path, "rb") as binary_file:
        data = binary_file.read()
    return data

def byte_sequence_to_Ngrams(byte_sequence, N):
    """Creates a list of N-grams from a byte sequence."""
    return ngrams(byte_sequence, N)

def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
```

```
with open(file_path, "rb") as binary_file:
    data = binary_file.read()
return data

def byte_sequence_to_Ngrams(byte_sequence, N):
    """Creates a list of N-grams from a byte sequence."""
    return ngrams(byte_sequence, N)

def hash_input(inp):
    """Compute the MD5 hash of an input."""
    return int(hashlib.md5(inp).hexdigest(), 16)

def make_ngram_hashable(Ngram):
    """Convert N-gram into bytes to be hashable."""
    return bytes(Ngram)

def hash_file_Ngrams_into_dictionary(file_Ngrams, T):
    """Hashes N-grams in a list and then keeps track of the counts in a dictionary."""
    for Ngram in file_Ngrams:
        hashable_Ngram = make_ngram_hashable(Ngram)
        hashed_and_reduced = hash_input(hashable_Ngram) % B
        T[hashed_and_reduced] = T.get(hashed_and_reduced, 0) + 1

B = 65521
T = {}



for dataset_path in directories:
    samples = [f for f in listdir(dataset_path)]
    for file in samples:
        file_path = dataset_path + "/" + file
        file_byte_sequence = read_file(file_path)
        file_Ngrams = byte_sequence_to_Ngrams(file_byte_sequence, N)
        hash_file_Ngrams_into_dictionary(file_Ngrams, T)

K1 = 1000
```

```
import heapq
K1_most_common_Ngrams_Using_Hash_Grams = heapq.nlargest(K1, T)

def featurize_sample(file, K1_most_common_Ngrams_Using_Hash_Grams):
    """Takes a sample and produces a feature vector. The features are the counts of the
    K1 N-grams we've selected."""
    K1 = len(K1_most_common_Ngrams_Using_Hash_Grams)
    fv = K1 * [0]
    file_byte_sequence = read_file(file_path)
    file_Ngrams = byte_sequence_to_Ngrams(file_byte_sequence, N)
    for Ngram in file_Ngrams:
        hashable_Ngram = make_ngram_hashable(Ngram)
        hashed_and_reduced = hash_input(hashable_Ngram) % B
        if hashed_and_reduced in K1_most_common_Ngrams_Using_Hash_Grams:
            index =
K1_most_common_Ngrams_Using_Hash_Grams.index(hashed_and_reduced)
            fv[index] += 1
    return fv
X = []
for dataset_path in directories:
    samples = [f for f in listdir(dataset_path)]
    for file in samples:
        file_path = dataset_path + "/" + file
        X.append(featurize_sample(file_path,
K1_most_common_Ngrams_Using_Hash_Grams))
print(X)
```

-Sử dụng 2 file dataset:

 DA Logs Malware 3	4/21/2024 1:05 PM	WinRAR archive	23,273 KB
 DA Logs Benign 3	4/21/2024 1:05 PM	WinRAR archive	14,981 KB

-Kết quả ta có được là chuỗi vector x



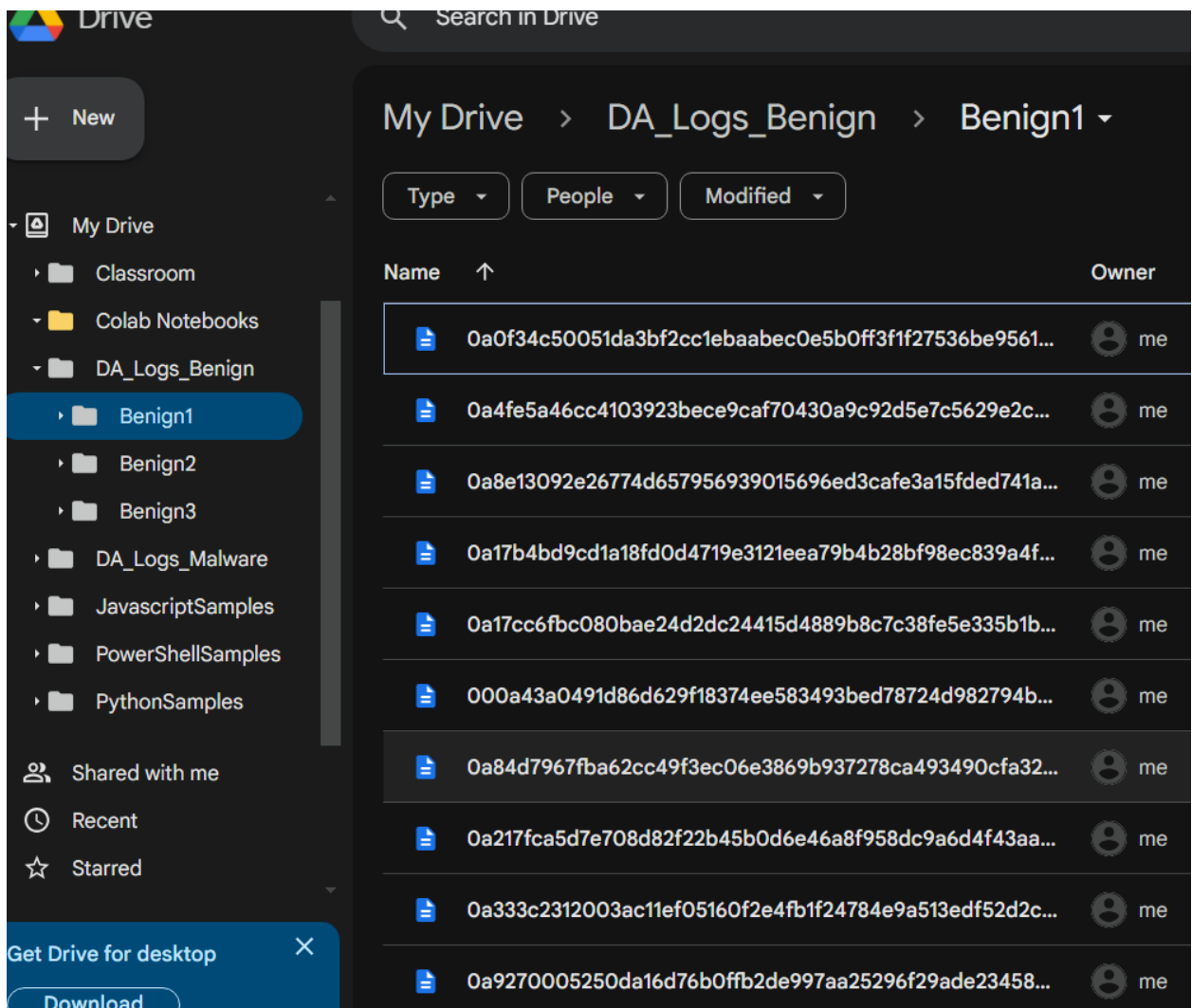


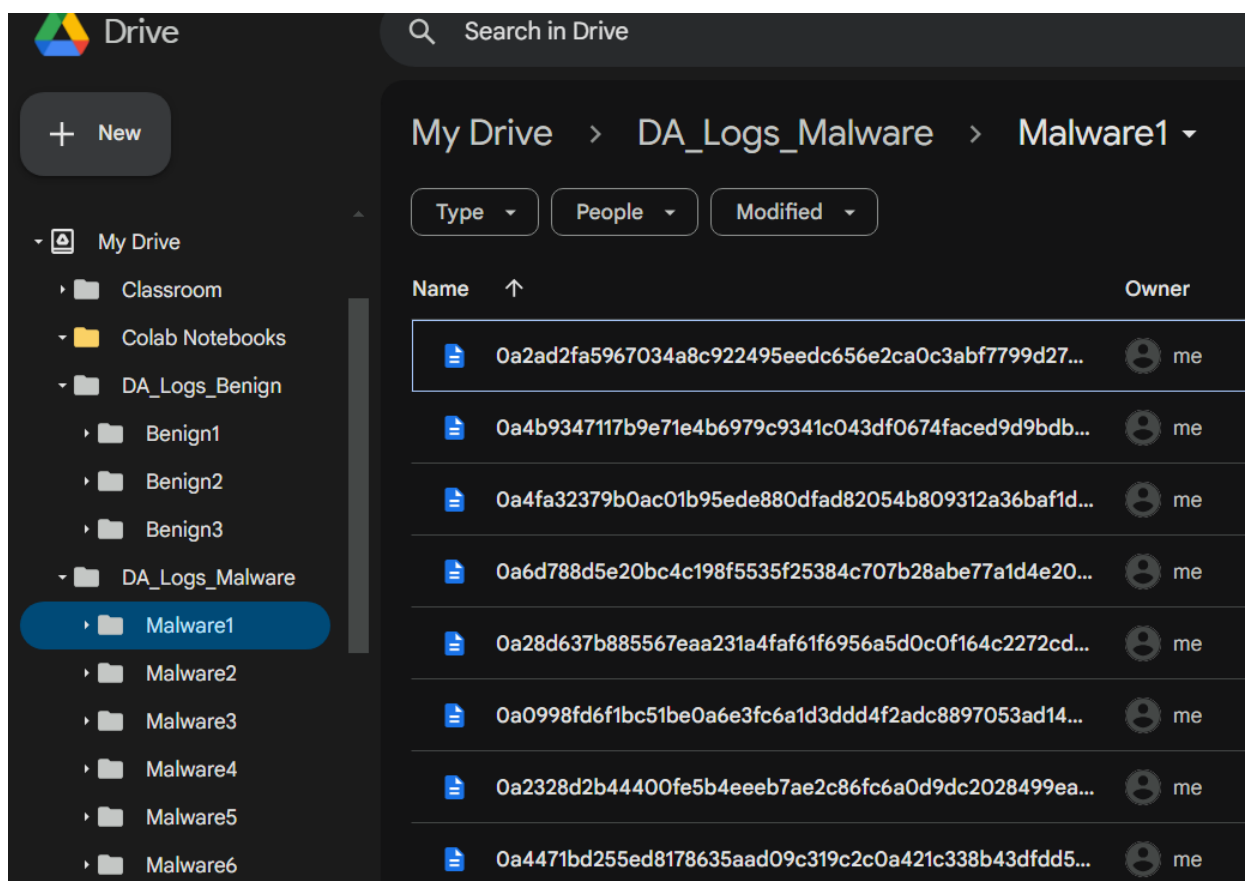
```
✓ [23] import numpy as np
0s      import os
      import json
```

```
✓ [24] from google.colab import drive
2s      drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount,

```
✓ [25] DA_Logs_Benign = "/content/drive/MyDrive/DA_Logs_Benign/Benign1/"
0s      DA_Logs_Malware = "/content/drive/MyDrive/DA_Logs_Malware/Malware1/"
      directories_with_labels = [(DA_Logs_Benign, 0), (DA_Logs_Malware, 1)]
```





- Hàm parse nhật ký JSON và trích xuất class, method và type từ API call.

```
[43] def get_API_class_method_type_from_log(log):  
    """Parses out API calls from behavioral logs."""  
    API_data_sequence = []  
    with open(log) as log_file:  
        json_log = json.load(log_file)  
        api_calls_array = "[" + json_log["api_calls"] + "]"  
        api_calls = json.loads(api_calls_array)  
        for api_call in api_calls:  
            data = api_call["class"] + ":" + api_call["method"] + ":" + api_call["type"]  
            API_data_sequence.append(data)  
    return API_data_sequence
```

- Trích xuất dữ liệu và phân loại từ các tập dữ liệu có sẵn và gán nhãn tương ứng, nhằm tạo ra tập dữ liệu huấn luyện.

```
data_corpus = []
labels = []
for directory, label in directories_with_labels:
    logs = os.listdir(directory)
    for log_path in logs:
        file_path = directory + "/" + log_path
        try:
            data_corpus.append(get_API_class_method_type_from_log(file_path))
            labels.append(label)
        except:
            pass
print(data_corpus[0])
```

```
['java.lang.reflect.Method:invoke:reflection', 'libcore.io.IoBridge:open:file', 'andro:
```

- Chia tập dữ liệu thành tập train và tập test với `corpus_train` và `corpus_test` là tập dữ liệu train và test chứa các mẫu dữ liệu, `y_train` và `y_test` là nhãn tương ứng với tập train và tập test.

```
from sklearn.model_selection import train_test_split

corpus_train, corpus_test, y_train, y_test = train_test_split(
    data_corpus, labels, test_size=0.2, random_state=11
)
```

- Hàm `read_file` đọc nội dung của file ở dạng nhị phân và trả về giá trị nhị phân tương ứng.
- Hàm `text_to_Ngrams` tạo ra danh sách các N-grams.
- Cuối cùng hàm `get_Ngram_counts` tính toán tần suất xuất hiện các N-grams trong 1 văn bản đã cho.

✓  
1s

```
import collections
from nltk import ngrams
import numpy as np

def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
    with open(file_path, "rb") as binary_file:
        data = binary_file.read()
    return data

def text_to_Ngrams(text, n):
    """Produces a list of N-grams from a text."""
    Ngrams = ngrams(text, n)
    return list(Ngrams)

def get_Ngram_counts(text, N):
    """Get a frequency count of N-grams in a text."""
    Ngrams = text_to_Ngrams(text, N)
    return collections.Counter(Ngrams)
```

- Tính tổng số lần xuất hiện các N-grams với số từ của 1 N-grams là 4 ( $N = 4$ ) từ tập train.

✓  
41s

```
N = 4
total_Ngram_count = collections.Counter([])
for file in corpus_train:
    total_Ngram_count += get_Ngram_counts(file, N)
```

- Tính toán danh sách các N-gram phổ biến nhất trong tập huấn luyện được sắp xếp theo thứ tự giảm dần số lần xuất hiện và có  $K1 = 3000$  phần tử.

✓  
0s

```
[54] K1 = 3000
K1_most_frequent_Ngrams = total_Ngram_count.most_common(K1)
K1_most_frequent_Ngrams_list = [x[0] for x in K1_most_frequent_Ngrams]
```

- Hàm chuyển đổi mỗi mẫu dữ liệu (văn bản) thành một vector đặc trưng, mà mỗi phần tử của vector đó đại diện cho số lần xuất hiện của một N-gram đã chọn trong mẫu dữ liệu.

```
✓ 0s ▶ def featurize_sample(file, Ngrams_list):  
    """Takes a sample and produces a feature vector.  
    The features are the counts of the K1 N-grams we've selected.  
    """  
    K1 = len(Ngrams_list)  
    feature_vector = K1 * [0]  
    fileNgrams = get_Ngram_counts(file, N)  
    for i in range(K1):  
        feature_vector[i] = fileNgrams[Ngrams_list[i]]  
    return feature_vector
```

- Sử dụng hàm trên để tạo tập train và tập test.

```
✓ 5s ▶ X_train = []  
for sample in corpus_train:  
    X_train.append(featurize_sample(sample,  
    K1_most_frequent_Ngrams_list))  
X_train = np.asarray(X_train)  
  
X_test = []  
for sample in corpus_test:  
    X_test.append(featurize_sample(sample,  
    K1_most_frequent_Ngrams_list))  
X_test = np.asarray(X_test)
```

- Tạo ra một pipeline để chọn ra K2 đặc trưng tốt nhất sử dụng mutual information và sau đó huấn luyện một mô hình XGBoost.

```
✓ 1s ▶ from sklearn.feature_selection import SelectKBest, mutual_info_classif  
from sklearn.pipeline import Pipeline  
from xgboost import XGBClassifier  
  
K2 = 500  
mi_pipeline = Pipeline(  
    [  
        ("mutual_information", SelectKBest(mutual_info_classif, k=K2)),  
        ("xgb", XGBClassifier()),  
    ]  
)
```

- Cuối cùng là huấn luyện pipeline và đánh giá hiệu suất.

```
✓ 31s [58] mi_pipeline.fit(X_train, y_train)
      print("Training accuracy:")
      print(mi_pipeline.score(X_train, y_train))
      print("Testing accuracy:")
      print(mi_pipeline.score(X_test, y_test))
```

```
Training accuracy:
0.9357749469214437
Testing accuracy:
0.8237791932059448
```

\* Bài tập (yêu cầu làm)

#### 4. Cho biết kết quả đánh giá.

Kết quả đánh giá : training accuracy khoảng 0.93577 và testing accuracy khoảng 0.82378.

® Bài tập (yêu cầu làm)

5. Cho biết kết quả đánh giá mô hình qua tập test.

- Import thư viện numpy để tính toán vector và tqdm để theo dõi tiến trình trong vòng lặp.

```
✓ 0s [8] import numpy as np
      from tqdm import tqdm
```

- Định nghĩa hàm để chuyển byte thành vector.

```

✓ [9] def embed_bytes(byte):
    Os binary_string = "{0:08b}".format(byte)
    vec = np.zeros(8)
    for i in range(8):
        if binary_string[i] == "1":
            vec[i] = float(1) / 16
        else:
            vec[i] = -float(1) / 16
    return vec

```

- Đọc các tin PE mẫu và dán nhãn cho chúng.

```

✓ [10] import os
    Os from os import listdir

✓ [11] Benign_PE_Samples = "/content/drive/MyDrive/Benign_PE_Samples"
    Os Malicious_PE_Samples = "/content/drive/MyDrive/Malicious_PE_Samples"
    directories_with_labels = [(Benign_PE_Samples, 0), (Malicious_PE_Samples, 1)]

✓ [12] list_of_samples = []
    Os labels = []
    for dataset_path, label in directories_with_labels:
        samples = [f for f in listdir(dataset_path)]
        for file in samples:
            file_path = os.path.join(dataset_path, file)
            list_of_samples.append(file_path)
            labels.append(label)

```

- Hàm đọc chuỗi byte trong tập tin.

```

✓ [13] def read_file(file_path):
    Os """Read the binary sequence of a file."""
    with open(file_path, "rb") as binary_file:
        return binary_file.read()

```

- Đặt độ dài tối đa, maxSize byte, để đọc cho mỗi mẫu, lấy tất cả byte của mẫu đưa vào X.

✓  
55s

```
max_size = 15000
num_samples = len(list_of_samples)
X = np.zeros((num_samples, 8, max_size))
Y = np.asarray(labels)
file_num = 0
for file in tqdm(list_of_samples):
    sample_byte_sequence = read_file(file)
    for i in range(min(max_size, len(sample_byte_sequence))):
        X[file_num, :, i] = embed_bytes(sample_byte_sequence[i])
    file_num += 1
```

100% |██████████| 260/260 [00:55<00:00, 4.71it/s]

- Thiết lập trình tối ưu.

✓  
0s

```
[23] from keras import optimizers
```

```
initial_learning_rate = 0.01
lr_schedule = optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=10000,
    decay_rate=1e-5,
    staircase=True)

my_opt = optimizers.SGD(learning_rate=lr_schedule, nesterov=True)
```

- Sử dụng API của Keras để thiết lập màn thần kinh học sâu.



✓  
1s

```
[24] from keras import Input
      inputs = Input(shape=(8, max_size))

      from keras.layers import Conv1D
      conv1 = Conv1D(kernel_size=(128), filters=32, strides=(128),
padding="same")(inputs)
      conv2 = Conv1D(kernel_size=(128), filters=32, strides=(128),
padding="same")(inputs)

      from keras.layers import Activation
      a = Activation("sigmoid", name="sigmoid")(conv2)

      from keras.layers import multiply
      mul = multiply([conv1, a])
      b = Activation("relu", name="relu")(mul)

      from keras.layers import GlobalMaxPool1D
      p = GlobalMaxPool1D()(b)

      from keras.layers import Dense
      d = Dense(16)(p)
      predictions = Dense(1, activation="sigmoid")(d)

      from keras import Model
      model = Model(inputs=inputs, outputs=predictions)
```

- Biên dịch mô hình và chọn batch size.

✓  
0s

```
▶ model.compile(optimizer=my_opt, loss="binary_crossentropy",
metrics=["acc"])
model.summary()

batch_size = 16
num_batches = int(num_samples / batch_size)
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 8, 15000)]	0	[]
conv1d_1 (Conv1D)	(None, 1, 32)	6144003 2	['input_1[0][0]']
conv1d (Conv1D)	(None, 1, 32)	6144003 2	['input_1[0][0]']
sigmoid (Activation)	(None, 1, 32)	0	['conv1d_1[0][0]']
multiply (Multiply)	(None, 1, 32)	0	['conv1d[0][0]', 'sigmoid[0][0]']
relu (Activation)	(None, 1, 32)	0	['multiply[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 32)	0	['relu[0][0]']
dense (Dense)	(None, 16)	528	['global_max_pooling1d[0][0]']
dense_1 (Dense)	(None, 1)	17	['dense[0][0]']

Total params: 122880609 (468.75 MB)  
 Trainable params: 122880609 (468.75 MB)  
 Non-trainable params: 0 (0.00 Byte)

- Chia tập train, tập test.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

- Huấn luyện mô hình.

```
[39] for batch_num in tqdm(range(num_batches)):
    batch = X_train[batch_num * batch_size : (batch_num + 1) * batch_size]
    model.train_on_batch(
        batch, Y_train[batch_num * batch_size : (batch_num + 1) * batch_size]
    )
```

100%|██████████| 16/16 [02:06<00:00, 7.88s/it]

\* Bài tập (yêu cầu làm)

## 5. Cho biết kết quả đánh giá mô hình qua tập test.

- Kết quả đánh giá mô hình qua tập test.

```
print(model.evaluate(X_test, Y_test))
```

2/2 [=====] - 0s 167ms/step - loss: 0.2021 - acc: 1.0000  
[0.20210742950439453, 1.0]

**Bài tập (yêu cầu làm)**

6. Cài đặt UPX từ <https://github.com/1upx/upx/releases>, và tiến hành đóng gói các tập tin pe tại Benign PE Samples UPX

Tiến hành tạo đường dẫn đến Benign PE Samples UPX folder

```
[2] from google.colab import drive
drive.mount("/content/drive", force_remount=True)

Mounted at /content/drive

[4] import os
files_path = "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/"
files = os.listdir(files_path)
file_paths = [files_path + x for x in files]

file_paths
['/content/drive/MyDrive/Lab3/Benign PE Samples UPX/ieUnatt.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/logagent.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/CameraSettingsUIHost.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/bfsvc.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/FsIso.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/CIDiag.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/licensingdiag.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/dusmtask.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/bcdboot.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/bootsect.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/DTUHandler.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/DataUsageLiveTileTask.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/DataStoreCacheDumpTool.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/dstokenclean.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/LicenseManagerShellExt.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/dxgiadaptercache.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/dispdiag.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/IcsEntitlementHost.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/EDPCleanup.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/dasHost.exe',
'/content/drive/MyDrive/Lab3/Benign PE Samples UPX/coredpussvr.exe',
```

Sau đó, ta kiểm tra hệ điều hành và tải bản UPX tương ứng

```
[6] !cat /etc/os-release # Check the version of OS for appropriate downloading

PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy

%cd "/content/drive/MyDrive/Lab3/Tool"
!wget https://github.com/upx/upx/releases/download/v4.0.0/upx-4.0.0-amd64_linux.tar.xz -P "." # '-P' Output path (if used)

/content/drive/MyDrive/Lab3/Tool
--2024-04-18 04:07:04-- https://github.com/upx/upx/releases/download/v4.0.0/upx-4.0.0-amd64_linux.tar.xz
Resolving github.com (github.com)... 140.82.116.4
Connecting to github.com (github.com)|140.82.116.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/67031040/67157722-5558-4d57-aabd-9cea504ecbaa?X-Amz-Algorithm=AWS4-HMAC-SHA256
--2024-04-18 04:07:04-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/67031040/67157722-5558-4d57-aabd-9cea504ecbaa?X-Amz-Algorithm=
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 509584 (498K) [application/octet-stream]
Saving to: './upx-4.0.0-amd64_linux.tar.xz'

upx-4.0.0-amd64_lin 100%[=====] 497.64K --.-KB/s in 0.03s

2024-04-18 04:07:05 (14.3 MB/s) - './upx-4.0.0-amd64_linux.tar.xz' saved [509584/509584]
```

Ta giải nén và kiểm tra xem file thực thi có hoạt động chính xác không

```
[8] !mkdir upx-4.0.0-linux
!tar -xvf upx-4.0.0-amd64_linux.tar.xz -C upx-4.0.0-linux # '-C': output path (if used)

upx-4.0.0-amd64_linux/
upx-4.0.0-amd64_linux/COPYING
upx-4.0.0-amd64_linux/LICENSE
upx-4.0.0-amd64_linux/NEWS
upx-4.0.0-amd64_linux/README
upx-4.0.0-amd64_linux/THANKS
upx-4.0.0-amd64_linux/upx
upx-4.0.0-amd64_linux/upx.doc.html
upx-4.0.0-amd64_linux/upx.doc.txt
upx-4.0.0-amd64_linux/upx.1

! /content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx # Check if the executable file works fine

Ultimate Packer for eXecutables
Copyright (C) 1996 - 2022
UPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022

Usage: upx [-123456789dhtv] [-qvk] [-o file] file..

Commands:
  -1 compress faster          -9 compress better
  -d decompress               -l list compressed file
  -t test compressed file     -V display version number
  -h give more help           -L display software license

Options:
  -q be quiet                  -v be verbose
  -oFILE write output to 'FILE'
  -f force compression of suspicious files
  -k keep backup files
  file.. executables to (de)compress

Type 'upx --help' for more detailed help.
```

Sau đó tiến hành pack các tập tin lại

```
import os
from subprocess import Popen, PIPE

cmd = "/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx" # Path to "upx"
for path in file_paths:
    #cmd2 = cmd + " \"" + path + "\""
    print(cmd2)
    #res = Popen(cmd2, stdout=PIPE).communicate() #--> This line has something wrong, producing errors.
    res = Popen([cmd, path], stdout=PIPE).communicate() # Output errors as the files are already packed
    print(res)
    if "error" in str(res[0]):
        print(path)
        os.remove(path)

/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
/content/drive/MyDrive/Lab3/Tool/upx-4.0.0-linux/upx-4.0.0-amd64_linux/upx "/content/drive/MyDrive/Lab3/Benign PE Samples UPX/junction.exe"
(b' Ultimate Packer for eXecutables\n Copyright (C) 1996 - 2022\nUPX 4.0.0 Markus Oberhumer, Laszlo Molnar & John Reiser Oct 28th 2022\n\nUsage: upx [-123456789dhtv] [-qvk] [-o file] file..\n\nCommands:\n  -1 compress faster          -9 compress better\n  -d decompress               -l list compressed file\n  -t test compressed file     -V display version number\n  -h give more help           -L display software license\n\nOptions:\n  -q be quiet                  -v be verbose\n  -oFILE write output to 'FILE'\n  -f force compression of suspicious files\n  -k keep backup files\n  file.. executables to (de)compress\n\nType 'upx --help' for more detailed help.\n')
```

### ® Bài tập (yêu cầu làm)

7. Cho biết kết quả đánh giá.

```

[16] import os
from os import listdir
directories_with_labels = [
    ("/content/drive/MyDrive/Lab3/Benign PE Samples", 0),
    ("/content/drive/MyDrive/Lab3/Benign PE Samples UPX", 1),
    ("/content/drive/MyDrive/Lab3/Benign PE Samples Amber", 2),
]
list_of_samples = []
labels = []
N=2

for dataset_path, label in directories_with_labels:
    samples = [f for f in listdir(dataset_path)]
    for file in samples:
        file_path = os.path.join(dataset_path, file)
        list_of_samples.append(file_path)
        labels.append(label)

[6] from sklearn.model_selection import train_test_split
samples_train, samples_test, labels_train, labels_test = train_test_split(
    list_of_samples,
    labels,
    test_size=0.3,
    stratify=labels,
    random_state=11
)

[7] import collections
from nltk import ngrams
import numpy as np

[8] def read_file(file_path):
    """Reads in the binary sequence of a binary file."""
    with open(file_path, "rb") as binary_file:
        data = binary_file.read()
    return data

[9] def byte_sequence_to_Ngrams(byte_sequence, N):
    """Creates a list of N-grams from a byte sequence."""
    Ngrams = ngrams(byte_sequence, N)
    return list(Ngrams)

def extract_Ngram_counts(file, N):
    """Takes a binary file and outputs the N-grams counts of its binary
    sequence."""
    filebyte_sequence = read_file(file)
    file_Ngrams = byte_sequence_to_Ngrams(filebyte_sequence, N)
    return collections.Counter(file_Ngrams)

[11] def featurize_sample(sample, K1_most_frequent_Ngrams_list):
    """Takes a sample and produces a feature vector.
    The features are the counts of the K1 N-grams we've selected.
    """
    K1 = len(K1_most_frequent_Ngrams_list)
    feature_vector = K1 * [0]
    file_Ngrams = extract_Ngram_counts(sample, N)
    for i in range(K1):
        feature_vector[i] = file_Ngrams[K1_most_frequent_Ngrams_list[i]]

N_gram = 2
total_Ngram_count = collections.Counter([])
for file in samples_train:
    total_Ngram_count += extract_Ngram_counts(file, N_gram)
K1 = 100
K1_most_common_Ngrams = total_Ngram_count.most_common(K1)
K1_most_common_Ngrams_list = [x[0] for x in K1_most_common_Ngrams]

[17] Ngram_features_list_train = []
y_train = []
for i in range(len(samples_train)):
    file = samples_train[i]
    Ngram_features = featurize_sample(file, K1_most_common_Ngrams_list)
    Ngram_features_list_train.append(Ngram_features)
    y_train.append(labels_train[i])
X_train = Ngram_features_list_train

[18] from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
clf = clf.fit(X_train, y_train)

[19] Ngram_features_list_test = []
y_test = []
for i in range(len(samples_test)):
    file = samples_test[i]
    Ngram_features = featurize_sample(file, K1_most_common_Ngrams_list)
    Ngram_features_list_test.append(Ngram_features)
    y_test.append(labels_test[i])
X_test = Ngram_features_list_test

```

```
[34] y_pred = clf.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, f1_score, precision_score, roc_auc_score

print("Confusion matrix:\n %s" % confusion_matrix(y_test, y_pred))

# Additional metrics (Not required)
print("Accuracy: %s" % accuracy_score(y_test, y_pred))
print("Recall: %s" % recall_score(y_test, y_pred, average='micro'))
print("Precision: %s" % precision_score(y_test, y_pred, average='micro'))
print("F1 Score: %s" % f1_score(y_test, y_pred, average='micro'))
```

Confusion matrix:  
[[101 1 0]  
[ 0 60 0]  
[ 0 0 18]]  
Accuracy: 0.9944444444444445  
Recall: 0.9944444444444445  
Precision: 0.9944444444444445  
F1 Score: 0.9944444444444445

### ® Bài tập (yêu cầu làm)

8. Cho biết kết quả đánh giá mẫu mới trong việc đánh lừa bộ nhận diện.

```
# Move to Python executable folder
%cd "/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN"
# Original reference: https://www.kaggle.com/code/fanbyprinciple/malgan-can-we-create-malware-on-the-fly/notebook
```

/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN

```
import os
import pandas as pd
from keras.models import load_model

# ===== FIXING PYTHON LIB =====
# Line 5:
# "from tf.compat.v1.keras.backend import set_session"
# Line 34:
# "set_session(tf.compat.v1.Session(config=config))"
import MalGAN_utils

# In file "MalGAN_preprocess.py"
# Line 6, 7:
# "import tensorflow as tf"
# "from tensorflow.keras.utils import pad_sequences"

import MalGAN_gen_adv_examples
```

```
save_path = "/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_output"
model_path = "/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_input/malconv.h5"
log_path = "/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_output/adversarial_log.csv"
pad_percent = 0.1
threshold = 0.6
step_size = 0.01
limit = 0.
input_samples = "/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_input/samplesIn.csv"
```

```
# In line 31 of "MalGAN_utils.py", change into:
# config = tf.compat.v1.ConfigProto()
MalGAN_utils.limit_gpu_memory(limit)
```

```
df = pd.read_csv(input_samples, header=None)
fn_list = df[0].values
list(enumerate(fn_list))
```



## Lab 1: Memory Forensic

```
Out[ ]: [(0,
          '0778a070b283d5f4057aeb3b42d58b82ed20e4eb_f205bd9628ff8dd7d99771f13422a665a70bb916'),
         (1,
          'fbd1a4b23eff620c1a36f7c9d48590d2fccda4c2_cc82281bc576f716d9a0271d206beb81ad078b53'),
         (2,
          'c095da034535f15a27c073dce54212a28e1af683_8e86441bc4f6a7fc492779caf280f1d769e0cd4d'),
         (3,
          '488e5eea345e24440f7d0d2a32fbafda314ee6ca_df473c0493d503828157e32664e28357a4094f7a'),
         (4,
          '7a359bcc1c7ac5f18eff7c3459dadedfa9f9e4610_3b7ac6b0a7a720460526c78628c8616dad8c6a1f'),
         (5,
          '509038aad80431b8aa0c9b29bfce07fe7134fc7a_263fbe72e691a0c047f75ce75a585ba0af84ac94'),
         (6,
          '05202b7ebc42f2a159154f99cec58fc1bcfe2c17_8cc75062dbe1ddc363fa5178312845709f669e37'),
         (7,
          'c47ed37c0e1e8be110f889f5989aea6b1bb7fda4_42c3033e34d27c951f7db7ae6aa5f45b8ef472ac'),
         (8,
          '18a62e8ee8522c26e9970373895209ee15a56841_4f0f6e9d21bbbe1842e8e8d6d911561108389662'),
         (9,
          '12113db281913797b6d58079b45089d4d057f766_ce1c8adb310886deb3ace648d4152ddeb1ed32fb'),
         (10,
          '19c625db4021d4934916f42b9777098b0b8cf15_0192ba11326fe2298c8cb4de616f4d4140213838'),
         (11,
          '2a7a2505e43decf3583ec11bbad6ead816ef1861_2906bffe07800fe9ff93d544bfff8a46e56aa64d9'),
         (12,
          '3f3fc4ed6a51a4385de9a38b164bcc14a4334eac_20002faf28adfd94ca98cf6ced46f14334b53684'),
         (13,
          'ead2a1884a52c93c654853d2891a1208ac6ee2b8_b75311ed885fb252554ab6dc228326d388043630'),
         (14,
          '6bc4d5ba3fa9c0f1c1431de8d214cc90bcfe4b55_4f53fe0260bec1166add1d1977e461984c448473'),
         (15,
          '3138cef67d651745c2db3744a70d6bf861e2a728_7b5ec86f006dc6d35d48821ca1311561bd86157a'),
         (16,
          '80913fa3c7d18d89c47b42e929cf9c98729aa565_2dcb3285f944bc0b6e7cb7612dad360b9458ccdc'),
         (17,
          '5746b9ed6b78d8f1cddb0b8bb9016624a46430346_007439fd0f1bfc25e540a2b5ddbc96d7ea09afd'),
         (18,
```

```
model = load_model(model_path)
model.summary() # Have an overview of model
# Use 'layers[0]' as input and 'layer[1]' as output for the next snippet code
```

Model: "model\_1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 80000)]	0	[]
embedding_1 (Embedding)	(None, 80000, 8)	2048	['input_1[0][0]']
conv1d_2 (Conv1D)	(None, 160, 128)	512128	['embedding_1[0][0]']
conv1d_1 (Conv1D)	(None, 160, 128)	512128	['embedding_1[0][0]']
activation_1 (Activation)	(None, 160, 128)	0	['conv1d_2[0][0]']
multiply_1 (Multiply)	(None, 160, 128)	0	['conv1d_1[0][0]', 'activation_1[0][0]']
global_max_pooling1d_1 (Global MaxPooling1D)	(None, 128)	0	['multiply_1[0][0]']
dense_1 (Dense)	(None, 64)	8256	['global_max_pooling1d_1[0][0]']
dense_2 (Dense)	(None, 1)	65	['dense_1[0][0]']
Total params: 1,034,625			
Trainable params: 1,034,625			
Non-trainable params: 0			

```
model.layers[0].input
```

```
<KerasTensor: shape=(None, 80000) dtype=float32 (created by layer 'input_1')>
```

```
model.layers[1].output
```

```
<KerasTensor: shape=(None, 80000, 8) dtype=float32 (created by layer 'embedding_1')>
```

```
adv_samples, log = MalGAN_gen_adv_examples.gen_adv_samples(
    model,
    fn_list,
    pad_percent,
    step_size,
    threshold
)
```

```
0778a070b283d5f4057aeb3b42d58b82ed20e4eb_f205bd9628ff8dd7d99771f13422a665a70bb916 not exist
```

```
inp: []
```

```
len_List: 80000
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-7-67fde16be76d> in <module>
      4     pad_percent,
      5     step_size,
----> 6     threshold
      7 )

/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_gen_adv_examples.py in gen_adv_samples(model, fn_list, pad_per
cent, step_size, thres)
     35     print("inp: ", inp)
     36     print("len_List: ", max_len)
----> 37     inp_emb = np.squeeze(np.array(inp2emb([inp, False])), 0)
     38
     39     pad_idx = len_list[0]

/usr/local/lib/python3.7/dist-packages/keras/backend.py in func(model_inputs)
    4329     wrap_outputs = isinstance(outputs, list) and len(outputs) == 1
```

Đến bước này nhóm chúng em gặp lỗi, nhóm sẽ nghiên cứu thêm để sửa được lỗi này



```
with open('/content/drive/MyDrive/Shared Drive/Lab3/Dataset/MalGAN/MalGAN_gen_adv_examples.py', 'r') as f:
    print(f.read())
```

```
from sklearn.neighbors import NearestNeighbors
from tensorflow.keras import backend as K
import MalGAN_utils
from MalGAN_preprocess import preprocess
import numpy as np

def gen_adv_samples(model, fn_list, pad_percent=0.1, step_size=0.001, thres=0.5):

    ### search for nearest neighbor in embedding space ###
    def emb_search(org, adv, pad_idx, pad_len, neigh):
        out = org.copy()
        for idx in range(pad_idx, pad_idx+pad_len):
            target = adv[idx].reshape(1, -1)
            best_idx = neigh.kneighbors(target, 1, False)[0][0]
            out[0][idx] = best_idx
        return out

    max_len = int(model.input.shape[1])
    emb_layer = model.layers[1]
    emb_weight = emb_layer.get_weights()[0]
    inp2emb = K.function([model.layers[0].input], [model.layers[1].output]) # [function] Map sequence to embedding

    # Build neighbor searches
    neigh = NearestNeighbors()
    neigh.fit(emb_weight)

    log = MalGAN_utils.logger()
    adv_samples = []

    for e, fn in enumerate(fn_list):

        ### run one file at a time due to different padding length, [slow]
        inp, len_list = preprocess([fn], max_len)
        inp_emb = np.squeeze(np.array(inp2emb([inp, False])), 0)

        pad_idx = len_list[0]
        pad_len = max(min(int(len_list[0]*pad_percent), max_len-pad_idx), 0)

        log.save(log_path)
        for fn, adv in zip(fn_list, adv_samples):
            _fn = fn.split('/')[-1]
            dst = os.path.join(save_path, _fn)
            print(dst)
            with open(dst, 'wb') as f:
                f.write(adv)
```

---

*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 **.DOCX và .PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-ExeX\_GroupY. (trong đó X là Thứ tự Bài tập, Y là mã số thứ tự nhóm trong danh sách mà GV phụ trách công bố).
- Ví dụ: [NT101.K11.ANTT]-Exe01\_Group03.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài nộp.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại [courses.uit.edu.vn](https://courses.uit.edu.vn).

### Đánh giá:

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

*Bài sao chép, trộm, ... sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**