BÁO CÁO TIẾN ĐỘ ĐỒ ÁN MÔN HỌC

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

Tên chủ đề: CD-VulD: Cross-Domain Vulnerability Discovery based on Deep Domain Adaptation

***Mã nhóm**: G12* ***Mã đề tài**: CK11*

Lớp: NT521.N11.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	Phan Hữu Luân	20521585	20521585@gm.uit.edu.vn
2	Phạm Ngọc Lợi	20521560	20521560@gm.uit.edu.vn
3	Nguyễn Trần Đức An	20520373	20520373@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

- **Chủ đề nghiên cứu trong lĩnh vực An toàn phần mềm:**

þ 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: ……

- **Tên bài báo tham khảo chính:**

CD-VulD: Cross-Domain Vulnerability Discovery based on Deep Domain Adaptation

- **Dịch tên Tiếng Việt cho bài báo:**

CD-VulD: Tìm Kiếm Lỗ Hổng Bảo Mật Trên Cross-Domain dựa trên Deep Domain Adaptation

- **Tóm tắt nội dung chính:**

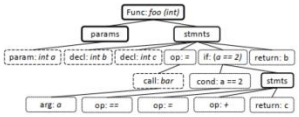
<mô tả nội dung tóm tắt của bài báo/chủ đề trong vòng 350 từ>
Ngày nay, nguyên nhân chính của các sự cố bảo mật như tấn công mạng bắt nguồn từ các **lỗ hổng phần mềm**. Các phương pháp tiếp cận dựa trên Machine Learning đạt được hiệu suất tiên tiến nhất trong việc nắm bắt các lỗ hổng. Các Model dự đoán được train trên một tập hợp dữ liệu thật , trong đó training data và test data được giả định là rút ra từ cùng một phân phối xác suất. Tuy nhiên, trong thực tế, test data thường khác với training data về dịch chuyển phân phối và các không gian đặc trưng khác nhau vì chúng đến từ các dự án khác nhau hoặc chúng khác nhau về các loại lỗ hổng. Trong bài báo này, chúng tôi trình bày một hệ thống mới cho Cross Domain Software Vulnerability Discovery (CD-VulD) sử dụng Deep Learning (DL) và Domain Adaptation (DA). Chúng tôi sử dụng DL vì nó có khả năng tự động hóa xây dựng các biểu diễn tính năng trừu tượng cấp cao của các chương trình, có khả năng hữu ích trên cross-domain hơn là các tính năng thủ công. Sự khác nhau giữa các miền dữ liệu được giảm bớt bằng cách cho model học các biểu diễn cross-domain. Thứ nhất, CD-VulD chuyển đổi chúng thành chuỗi token và thêm token embeddings để tổng quát hóa các tokens. Tiếp theo, CD-VulD sử dụng một mô hình deep learning có tính năng đặc thù để xây dựng các cách biểu diễn của các chuỗi token trên. Sau đó, the Metric Transfer Learning Framework (MTLF) được sử dụng để học các biểu diễn cross-domain bằng cách giảm thiểu sự khác nhau giữa miền nguồn và miền đích. Cuối cùng, các biểu diễn cross-domain được sử dụng để xây dựng một bộ phân loại (classifier) để phát hiện lỗ hổng. Kết quả thử nghiệm cho thấy CD-VulD làm tốt hơn các phương pháp phát hiện lỗ hổng bảo mật tiên tiến nhất nhờ một biên độ rộng.

- **Các kỹ thuật chính được mô tả sử dụng trong bài báo:**

<mô tả các kỹ thuật chính được dùng trong nghiên cứu của bài báo>(liệt kê tóm tắt vai trò của các kỹ thuật đó – kèm hình ảnh minh họa về hệ thống/kỹ thuật/phương pháp)
Kỹ thuật chính được nghiên cứu: CD-VulD
+ **Tiền xử lý** : function => dạng Abstract Synax Tree => token sequence : đưa function của chương tình về dạng input cho mô hình deep learning
Dưới dạng AST:

```
1  int foo (int a)
2  {
3     int c = 0;
4     /* Invoke function bar() */
5     int b = bar (a);
6     if (a == 2)
7     {
8        c = a + b;
9        return c;
10    }
11    return b;
12 }
```

(a) An Example of a C Function.



(b) An AST of function 'foo'.

# Type	Depth	Value 1	Value 2
func	0	int	foo
params	1		
param	2	int	a
stmnts	1		
decl	2	Int	c
op	2	=	
decl	2	int	b
op	2	=	
call	3	bar	
arg	4	a	
if	2	(a == 2)	
cond	3	a == 2	
op	4	==	
stmts	3		
op	4	=	
op	4	+	
return	4	c	
return	2	b	

(c) A serialized AST of function 'foo'.

Dưới dạng Code Gadget :

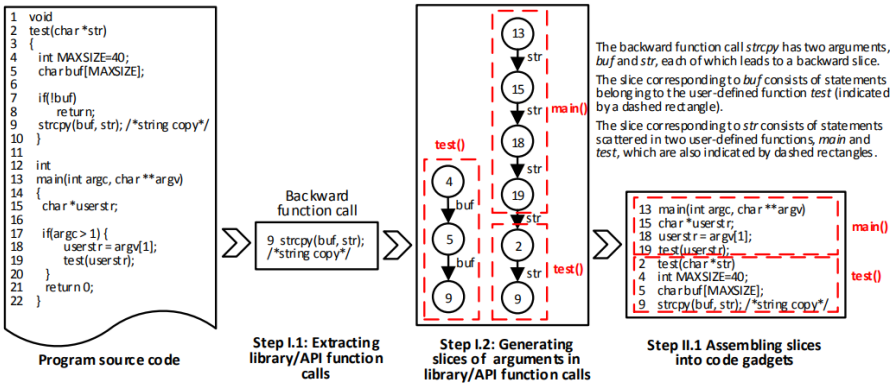


Figure 3. Illustrating the extraction of library/API function calls (Step I.1) from a (training) program, which contains a backward function call (i.e., *strcpy*) that is also used as an example to demonstrate the extraction of program slices (Step I.2) and the assembly of program slices into code gadgets (Step II.1).

+ **Train Deep Model** : ánh xạ token thành các vector đại diện cho chúng.

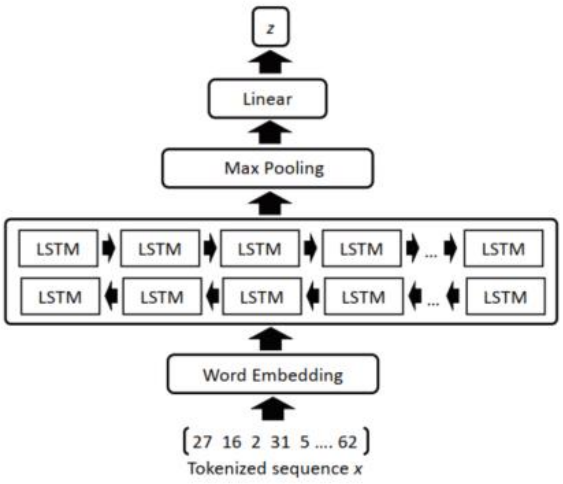
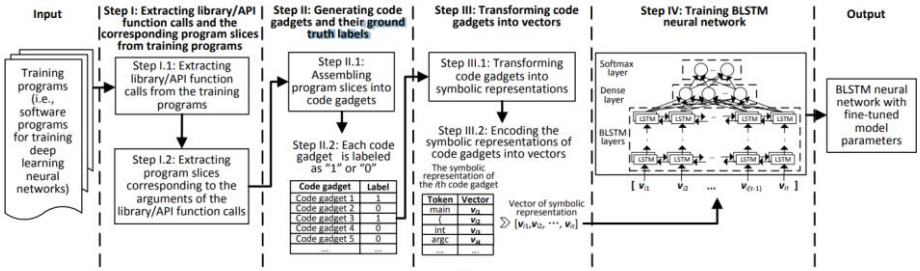
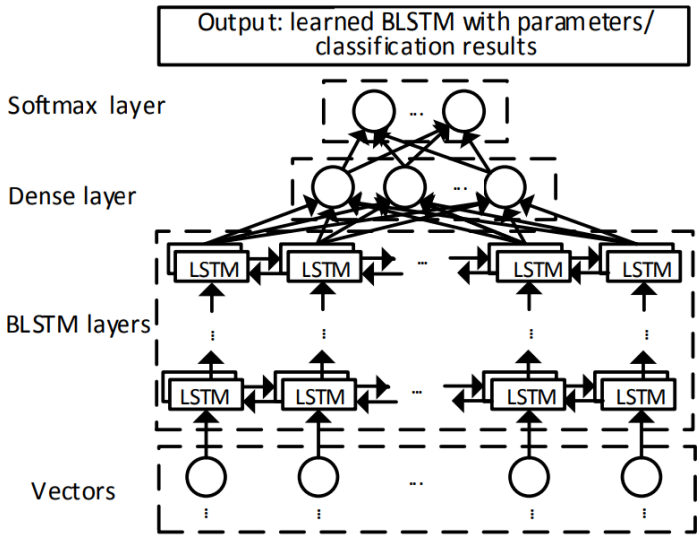


Fig. 4: Architecture of the deep feature model.

+ Ngoài ra với dạng **Code Gadget** ta có mô hình như sau :



(a) Learning phase



+ **Domain Adaptation** : **Metric Transfer Learning Framework**

Mục tiêu của domain adaptation là giảm sự khác biệt về phân phối miền giữa miền dữ liệu nguồn (training data) và miền dữ liệu (testing data) đích để kiến thức đã học từ miền nguồn có thể được áp dụng thêm cho miền đích.

- Với bài toán có $\mathbf{D}_s \# \mathbf{D}_t, \mathbf{T}_s = \mathbf{T}_s$, deep domain adaptation được sử dụng để chuyển tri thức, thông tin từ dữ liệu nguồn sang dữ liệu đích. Nói chung, mục tiêu của domain adatation là học một hàm ánh xạ F để giảm sự phân kỳ miền giữa $\mathbf{D}_s$ và $\mathbf{D}_t$ bao gồm dịch chuyển phân phối và các không gian đặc trưng khác nhau. Cụ thể, nhận miền nguồn là $\mathbf{D}_s$ cho nhiệm vụ $\mathbf{T}_s$ và $\mathbf{D}_t$ cho nhiệm vụ $\mathbf{T}_t$, trong đó $\mathbf{D}_s \# \mathbf{D}_t$. Domain adaptation có mục tiêu học $\mathbf{P}_t(Y_t|X_t)$ trong khi tri thức từ $\mathbf{T}_s$ và $\mathbf{D}_t$ có thể sử dụng để nâng cao $\mathbf{P}_t(Y_t|X_t)$.

CD-VulnD chia làm 2 giai đoạn:

- Train: tiền xử lý function thành các token sequences sau đó train deep model để ánh xạ token thành các vector đại diện cho chúng, cuối cùng triển khai

MTLF để model học ma trận chuyển đổi các vector vào miền biểu diễn cross-domain.

– Test: CD-VulD sẽ áp dụng các kỹ thuật tiền xử lý để biến đổi function thành các symbolic token sequences, sau đó ánh xạ các token vào miền biểu diễn cross-domain bằng deep model đã được train trước đó và ma trận chuyển đổi. Hệ thống phân loại đã được train dữ liệu mã nguồn domain được áp dụng để dự đoán xem mã nguồn đó có dễ bị tấn công hay không.

- **Môi trường thực nghiệm của bài báo:**

<mô tả môi trường thực nghiệm của nhóm tác giả, bao gồm cấu hình máy tính (phần cứng, phần mềm), các chương trình hỗ trợ để hiện thực phương pháp, ngôn ngữ lập trình được sử dụng, các chương trình phần mềm dùng để kiểm tra khả năng/tính năng của phương pháp>

- **Cấu hình máy tính:**

- CentOS Linux 7, 2 CPU Intel Xeon® E5-2690 v3 2.6 GHz
- RAM 128 GB

- **Công cụ:**

- Keras(2.0.8) with TensorFlow(1.3.0) (OpenSource Library)
- Gensim (3.0.1) (Opensoure Library)

- **Ngôn ngữ lập trình thực hiện phương pháp :**

- C/C++ (Keras, Gensim) (Chính)
- Java , C# (trong tương lai)

- **Đối tượng nghiên cứu**:

- dataset : sử dụng bộ dữ liệu CWE119, CWE399 mỗi bộ chứa một loại lỗ hổng là lỗi bộ đệm và quản lí tài nguyên, mã nguồn mở LibTIFF, Ffmpeg, LibPNG dự trên cơ sở dữ liệu NVD, CVE.
- In-domain: sử dụng CWE-ALL và LFL-ALL để dựng model deep learning
- Cross-domain: chia dataset thành 3 set nhỏ, 5% cho dữ liệu phát triển, 5% cho dữ liệu của labeled domain và phần còn lại cho test set
- Baselines

- **Tiêu chí đánh giá tính hiệu quả của phương pháp :**

- Chỉ số về độ chính xác (Precision)
- Số lần thu hồi (TPR)
- Độ đo F1 (F1)
- Tỉ lệ nhận diện sai (FPR)
- Độ chính xác top K
- Hệ số tương quan Matthews (MCC)

- **Kết quả thực nghiệm của bài báo:**

<mô tả ngắn gọn kết quả thực nghiệm của bài báo, tự nhận xét về khả năng, ưu và nhược điểm của phương pháp được đề cập trong bài báo>

Ưu điểm:

- CD-VulD có thể phát hiện các lỗ hổng của các loại mới trong cùng một dự án
- CD-VulD có thể phát hiện các lỗ hổng thuộc loại mới từ các dự án khác với domain nguồn
- Hiệu suất của CD-VulD tương đối cao trên các dự án khác nhau, bao gồm nhiều loại lỗ hổng bảo mật
- Hiệu suất của CD-VulD không thay đổi với các cấu trúc khác nhau của input
- CD-VulD có thể phát hiện các lỗ hổng mới

Nhược điểm:

- CD-VulD tập trung vào việc phát hiện các lỗ hổng trong mã nguồn, tuy nhiên cũng có một số lượng lớn các lỗ hổng trong mã nhị phân mà không có mã nguồn nào có sẵn
- Chọn LDA làm trình phân loại hoạt động tốt nhất trong số tất cả các trình phân loại ứng viên, nhưng vẫn chưa rõ ràng liệu các đại diện cross-domain của tất cả các loại lỗ hổng có thể phân tách tuyến tính hay không
- CD-VulD hiện bị giới hạn trong việc sử dụng kiến trúc dựa trên Recurrent Neural Network (RNN) và domain adaptation

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

<liệt kê kế hoạch của nhóm, các công việc mà nhóm dự định thực hiện cho đề tài dựa trên phân tích phương pháp/hệ thống được sử dụng trong bài báo đã tham khảo>

Từng thành viên phải tìm hiểu từng khái niệm mới như Deep Learning, Cross Domain, Domain Adaption, sau đó tiến hành phân tích kĩ bài báo.

- Tuân theo tiến trình framework của bài báo :

+ Đầu tiên, xử lí các code function thành dạng token, sau đó dưới dạng vector cho mô hình Deep Learning

+ Pre-Train dữ liệu với Deep Feature Model

+ Áp dụng Domain Adaptation : cụ thể Metric Transfer Learing Framework để giảm thiểu độ phân tán giữa training data và testing data

+ Train bộ máy phân loại (Classifier)

=> Cuối cùng, thực hiện mô hình hoàn chỉnh để phát hiện các lỗ hổng của chương trình code.

- Ngoài ra cũng tìm hiểu thêm bộ dữ liệu CWE119,... khá lớn và phức tạp.

<liệt kê các công việc đã thực hiện+ kèm theo kết quả của công việc này>

- Pre-processing dataset :

Asbtract Syntax Tree : construct lại các function dưới dạng ASTs, có khá nhiều tool và thư viện hỗ trợ cho việc này.

```
 8
 9 # Display the sum
10 print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
11
```

```c
int foo(int y)
{
    int n = bar(y);

    if (n == 0)
        return 1;

    return (n + y);
}
```

(a) Exemplary C function

```
# type  depth  value1   value2
func    0      int      foo
params  1
param   2      int      y
stmts   1
decl    2      int      n
op      2      =
call    3      bar
arg     4      y
if      2      (n == 0)
cond    3      n == 0
op      4      ==
stmts   3
return  4      1
return  2      (n + y)
op      3      +
```

(b) Serialized AST

Figure 2: Example of a C function and a serialized AST

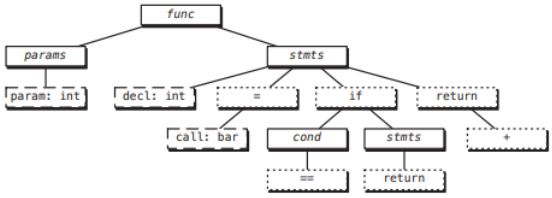


Figure 3: Abstract syntax tree with API nodes (dashed) and syntax nodes (dotted).

```c
int foo () {
    return 2;
}
int main() {
    int val;
    val = foo();
    return 0;
}
```

```
ahueck@sys:~/astprint/install$ ./bin/astprinter ../test.c --
ast-printer> l
foo:~/astprint/install/../test.c:1:3->3:4
main:~/astprint/install/../test.c:4:3->8:4

ast-printer> p foo
FunctionDecl 0x3977120 <test.c:1:3, line:3:3> line:1:7 used foo 'int ()'
`-CompoundStmt 0x3977238 <col:14, line:3:3>
  `-ReturnStmt 0x3977220 <line:2:7, col:14>
    `-IntegerLiteral 0x3977200 <col:14> 'int' 2
~/astprint/install/../test.c:1:3->3:4

ast-printer> 2
ReturnStmt 0x1ba3220 <test.c:2:7, col:14>
`-IntegerLiteral 0x1ba3200 <col:14> 'int' 2
~/astprint/install/../test.c:2:7->2:15
```

+ Ngoài ra ta có thể biểu diễn dưới dạng code gadget :

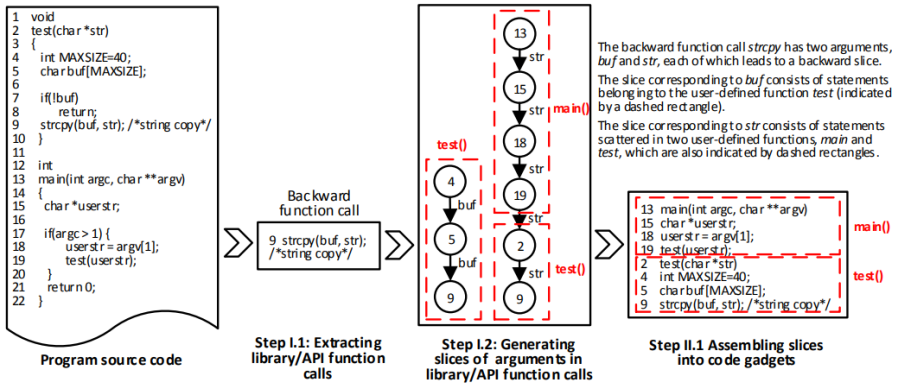


Figure 3. Illustrating the extraction of library/API function calls (Step I.1) from a (training) program, which contains a backward function call (i.e., *strcpy*) that is also used as an example to demonstrate the extraction of program slices (Step I.2) and the assembly of program slices into code gadgets (Step II.1).

- Nhóm đã tham khảo nhiều tài liệu về AST, nhưng chưa hoàn thiện được quá trình này.

+ **Tokenize**: tách thành từng word sau đó dùng layer TextVectorize của Keras biến thành các vector sau đó ánh xạ vào các token

- Nhóm đã thử tiến hành thực thi một số lệnh để chuyển code dưới dạng text sang token

CHUẨN BỊ DATASET CHO TRAINING

Dùng layer TextVectorization để

- Standardize: remove dấu chấm câu hoặc HTML elements
- Tokenization: tách chuỗi thành các token (bằng whitespace)
- Vectorization: chuyển token thành num để fed cho network

Ta sẽ viết 1 hàm để remove tag `<br />` do layer TextVectorization không remove

Note: Để tránh tình trạng skew, để tạo điều kiện tiền xử lý data lúc train và test giống nhau, ta có thể bao gồm layer TextVectorization vào model như dưới đây

```
[14] def custom_standardization(input_data):
       lowercase = tf.strings.lower(input_data)
       stripped_html = tf.strings.regex_replace(lowercase, '<br />', ' ')
       return tf.strings.regex_replace(stripped_html,
                                       '[%s]' % re.escape(string.punctuation),
                                       '')
```

Tiếp theo, ta sẽ tạo TextVectorization layer dùng để standardize, tokenize, vectorize data. Set output_mode thành int để tạo số nguyên duy nhất đại diện cho mỗi token.

Note: ta sẽ dùng hàm split mặc định, và hàm custom_standardization đã viết ở trên. Ta sẽ định nghĩa 1 số hằng số cho model, như maximum sequence_length, để bắt layer add thêm padding hoặc cắt bớt để đạt được sequence_length

```
[15] max_features = 10000
     sequence_length = 250

     vectorize_layer = layers.TextVectorization(
         standardize = custom_standardization,
         max_tokens = max_features,
         output_mode = 'int',
         output_sequence_length = sequence_length
     )
```

Ta gọi hàm `adapt` để fit state của layer TextVectorization vào dataset. Nó sẽ làm cho model tạo ra an index of strings to integers (?)

Note: chỉ call adapt để tập data train vì dùng tập test sẽ bị leak info

```
[16] train_text = raw_train_ds.map(lambda x, y: x)
     vectorize_layer.adapt(train_text)
```

```
[17] def vectorize_text(text, label):
         text = tf.expand_dims(text, -1)
         return vectorize_layer(text), label
```

```
[18] text_batch, label_batch = next(iter(raw_train_ds))
     first_review, first_label = text_batch[1], label_batch[1] #chọn review đầu tiên
     print("Review", first_review)  #in ra review dạng text
     print("Label", raw_train_ds.class_names[first_label]) #in ra label của review
     print("Vectorized review", vectorize_text(first_review, first_label)) #in ra dạng token của review
```

```
Review tf.Tensor(b"I am shocked. Shocked and dismayed that the 428 of you IMDB users who voted before me have not given this film a rating of higher than 7. 7?!?? - that's a C!.
Label pos
Vectorized review (<tf.Tensor: shape=(1, 250), dtype=int64, numpy=
array([[  10,  237, 2350, 2350,    3,    1,   12,    2,    1,    5,   22,
         922, 5790,   36, 5633,  153,   69,   25,   21,  340,   11,   19,
           4,  693,    5, 1718,   70, 1131, 1131,  177,    4, 1885,   45,
          10,   98,  193,    1,    4,  950,  450,    1,   82,    9,   11,
          19, 3798,  331,    1,    2,    1,    5,  709,  220, 4475,  363,
           1,    3,    1,   14,   28,    5,    2,   88,  615,   94,    5,
          30,   58,   45,   22,  118,  228,   42, 3662,  222,   22,  219,
          67,   11,   45,   22,  118,  155,   42, 3662,  222,  817,  138,
           3,   92,   67,   11, 5471,    6, 5572, 4139, 1760,    6, 1111,
           2, 1534, 1209,    5,   11,  953,   19,   45,   22,  414, 1956,
        2985,  483,  115,   11,  103,    9,    3, 2187,    9,    4,  296,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0]])>, <tf.Tensor: shape=(), dtype=int32, numpy=1>)
```

+ Hiện tại, nhóm cũng có tìm hiểu một số nguồn tham khảo về mô hình pre-training deep learning.

```python
def BiLSTM_network(MAX_LEN, EMBEDDING_DIM, word_index, embedding_matrix, use_dropout=False):
    inputs = Input(shape=(MAX_LEN,))

    sharable_embedding = Embedding(len(word_index) + 1,
                                   EMBEDDING_DIM,
                                   weights=[embedding_matrix],
                                   input_length=MAX_LEN,
                                   trainable=False)(inputs)
    bilstm_1 = Bidirectional(LSTM(64, activation='tanh', return_sequences=True))(sharable_embedding)
    if use_dropout:
        droput_layer_1 = Dropout(0.5)(bilstm_1)
        bilstm_2 = Bidirectional(LSTM(64, activation='tanh', return_sequences=True))(droput_layer_1)
    else:
        bilstm_2 = Bidirectional(LSTM(64, activation='tanh', return_sequences=True))(bilstm_1)

    #gmp_layer = GlobalMaxPooling1D()(bilstm_2)

    """
    Replace the GlobalMaxPool layer with the attention layer.

    """
    atten_layer = attention_3d_block(bilstm_2, MAX_LEN)
    flatten_layer = Flatten()(atten_layer)

    if use_dropout:
        dropout_layer_2 = Dropout(0.5)(flatten_layer)
        dense_1 = Dense(64, activation='relu')(dropout_layer_2)
    else:
        dense_1 = Dense(64, activation='relu')(flatten_layer)

    dense_2 = Dense(32)(dense_1)
    dense_3 = Dense(1, activation='sigmoid')(dense_2)

    model = Model(inputs=inputs, outputs = dense_3, name='BiLSTM_network')

    model.compile(loss=LOSS_FUNCTION,
            optimizer=OPTIMIZER,
```

Cũng như về MTLF :
https://github.com/scikit-learn-contrib/metric-learn
<các công việc cần thực hiện tiếp theo>
+ Tìm bộ dữ liệu phù hợp để thực hiện demo ở cuối kì
+ Thực hiện hoàn chỉnh bước “tiền xử lí dữ liệu”
+ Khởi tạo được mô hình Pre-Training Deep Learning
+ Khởi tạo được Metric Transfer Learning Framework.
+ Khởi tạo được Train Classifier

- **Các khó khăn, thách thức hiện tại khi thực hiện:**

- Nhiều định nghĩa mới gây khó khăn cho việc đọc hiểu
- Khá nhiều kiến thức mới vẫn còn mơ hồ
- Theo như bài báo, thì các dataset khá lớn, nhóm cần phải tạo hoặc lọc lại bộ dataset phù hợp để demo cuối kì.
- Từng thành viên phải tìm hiểu từng khái niệm mới như Deep Learning, Cross Domain, Domain Adaption, việc tìm kiếm tốn khá nhiều thời gian, sau khi tìm hiểu, nhóm tiến hành phân tích kĩ bài báo.

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

40%

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Tìm hiểu về Deep Learning, các khái niệm cơ bản của bài báo	Tất cả thành viên
2	Hoàn thành báo cáo về tên đề tài, nội dung chính, môi trường thực nghiệm	Phạm Ngọc Lợi
3	Hoàn thành báo cáo về các kỹ thuật chính được sử dụng, thực hiện quá trình Pre-Processing cụ thể Abstract Syntax Tree	Phan Hữu Luân
4	Hoàn thành báo cáo về kết quả thực nghiệm, thực hiện thử tokenize dạng văn bản text	Nguyễn Trần Đức An
5	Phân tích bài báo, mô hình CD-VuLD	Tất cả thành viên

BÁO CÁO CHI TIẾT

<phần này chỉ dùng cho báo cáo đề tài môn học ở giai đoạn cuối kì>

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

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)

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

YÊU CẦU CHUNG

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

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- 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.

Đánh giá:

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

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

HẾT

[1] Ghi nội dung tương ứng theo mô tả