HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

Học phần: **Mật mã học nâng cao**

ỨNG DỤNG HỌC SÂU PHÁT HIỆN LIÊN KẾT ĐỘC HẠI TRÊN TRÌNH DUYỆT

Giảng viên hướng dẫn: TS. Đỗ Xuân Chợ

Sinh viên thực hiện: **Nhóm lớp 2**

Nguyễn Thanh Tùng	B16DCAT172
Lê Ngọc Linh	B16DCAT092
Nguyễn Duy Hoàng	B16DCAT067
Hạ Viết Huy	B16DCAT076

Hà Nội, 2019

Mục lục

DANH SÁCH CÁC THUẬT NGỮ TIẾNG ANH VÀ VIẾT TẮT

Từ viết tắt	**Thuật ngữ tiếng Anh/Giải thích**	**Thuật ngữ tiếng Việt/Giải thích**
	Flask	Một framework trong ngôn ngữ Python, được sử dụng để tạo máy chủ web
CNN	Convolutional Neural Networks	Mạng nơ ron tích chập
ReLU	Rectified Linear Unit	Đơn vị chỉnh lưu tuyến tính
URL	Uniform Resource Locator	Đường dẫn dùng để tham chiếu đến các tài nguyên trên Internet

DANH MỤC CÁC HÌNH VẼ

LỜI MỞ ĐẦU

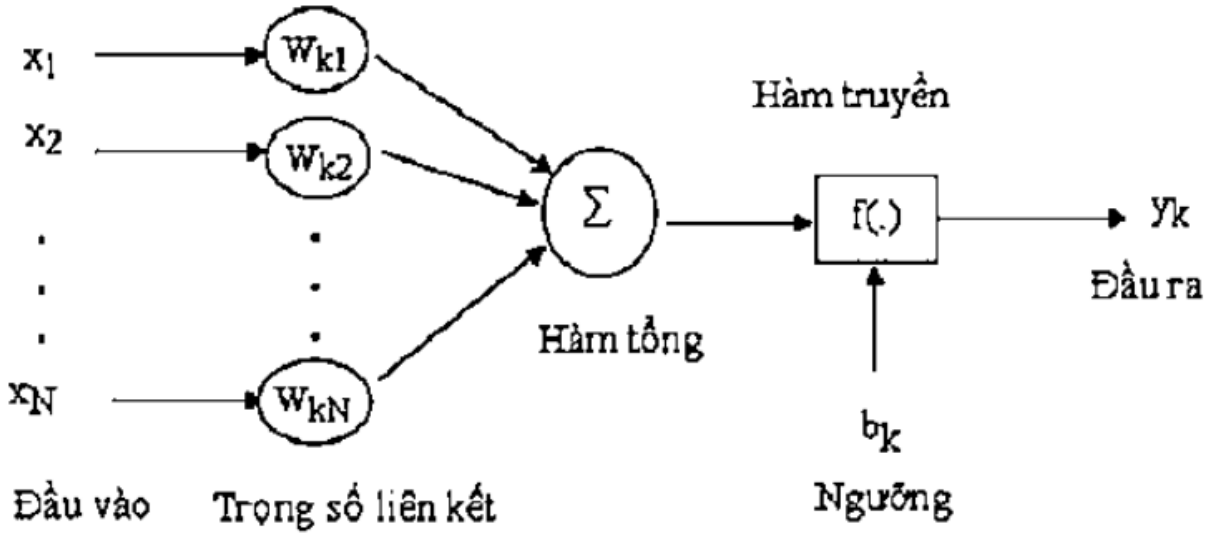
Học sâu là một thuật toán dựa trên một số ý tưởng từ não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu. Học sâu được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên. Hiện nay rất nhiều các bài toán nhận dạng sử dụng học sâu để giải quyết do học sâu có thể giải quyết các bài toán với số lượng lớn, kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống.

Đặc biệt, với mạng nơ ron tích chập (Convolutional Neural Networks - CNN), các chuyên gia đã đạt được nhiều thành tựu trong việc phân loại văn bản trong những năm gần đây. Kế thừa từ những thành công trước đó, trong bài báo cáo này chúng em sẽ trình bày về phương pháp áp dụng CNN với mức ký tự và từ vựng để phát hiện các liên kết độc hại. Đồng thời cũng trình bày về mô hình phát hiện liên kết độc hại theo thời gian thực trên trình duyệt sử dụng phần mở rộng (extension) kết hợp phương pháp nêu trên.

Chương 1. Mạng nơ ron tích chập

1.1. Giới thiệu về mạng nơ ron

Mạng nơ ron nhân tạo, Artificial Neural Network (ANN) là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơ ron sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (nơ ron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơ ron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu,...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơ ron.



Hình 1: Cấu tạo của một nơ ron

Các thành phần cơ bản của một nơron nhân tạo bao gồm:

• Tập các đầu vào: Là các tín hiệu vào (input signals) của nơ ron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.

• Tập các liên kết: Mỗi liên kết được thể hiện bởi một trọng số liên kết – Synaptic weight. Trọng số liên kết giữa tín hiệu vào thứ j với nơ ron k thường được kí hiệu là wkj. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.

• Hàm tổng (Summing function): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.

• Ngưỡng (còn gọi là một độ lệch - bias): Ngưỡng này thường được đưa vào như một thành phần của hàm truyền.

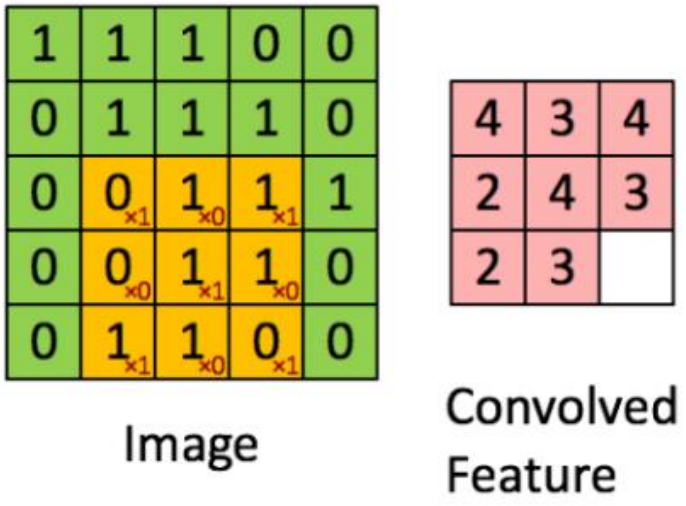
• Hàm truyền (Transfer function): Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơ ron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng.

• Đầu ra: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.

1.2. Mạng nơ ron tích chập

1.2.1. Tích chập là gì

Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số. Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận.



Hình 2: Minh hoạ tích chập

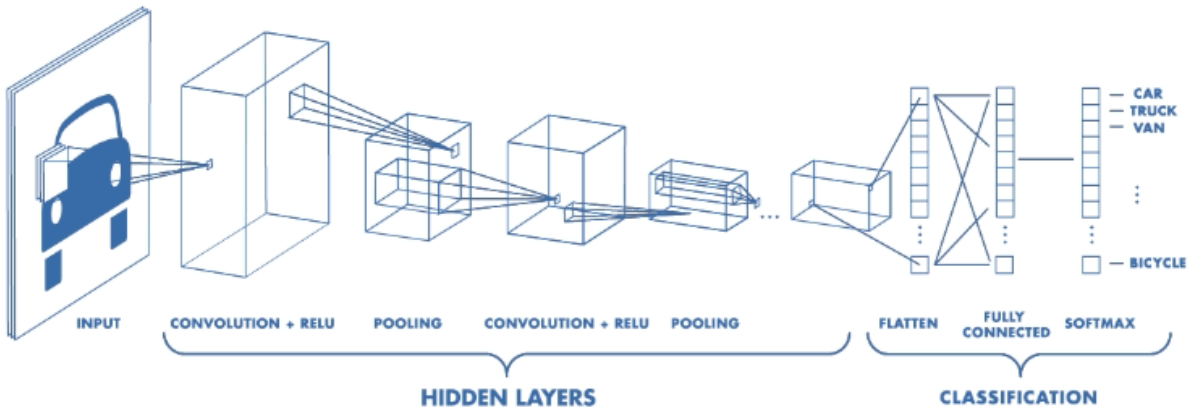
Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng. Sliding window còn có tên gọi là filter. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận (convoled feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái.

1.2.2. Mô hình mạng nơ ron tích chập

Bây giờ, Chúng ta đã biết thế nào là convolution. Vậy CNN là gì? CNN chỉ đơn giản gồm một vài layer của convolution kết hợp với các hàm kích hoạt phi tuyến

(nonlinear activation function) như ReLU hay tanh để tạo ra thông tin trừu tượng hơn (abstract/higher-level) cho các layer tiếp theo.

Trong mô hình CNN, các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Nghĩa là mỗi nơ-ron ở layer tiếp theo sinh ra từ filter áp đặt lên một vùng ảnh cục bộ của nơ-ron layer trước đó. Mỗi layer như vậy được áp đặt các filter khác nhau, thông thường có vài trăm đến vài nghìn filter như vậy. Một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các filter.



Hình 3: Mô hình mạng nơ ron tích chập

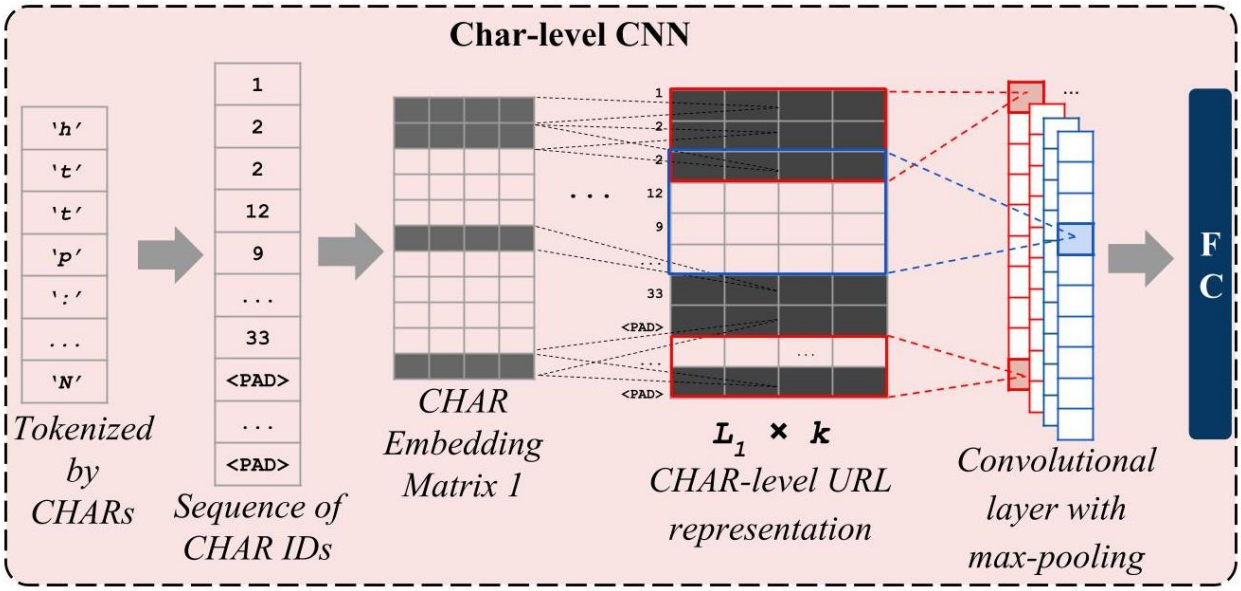
Chương 2: Phương pháp áp dụng CNN mức ký tự và từ vựng

2.1. Áp dụng CNN mức ký tự (Character-level CNN)

Mục tiêu của việc áp dụng CNN mức ký tự là lọc ra được các thuộc tính về các chuỗi ký tự xuất hiện trong văn bản (cụ thể ở đây là URL) [1].

Trước tiên ta cần xác định tất cả các ký tự xuất hiện trong bộ dữ liệu huấn luyện và những ký tự xuất hiện hiếm khi xuất hiện trong bộ dữ liệu (ví dụ trong 200 000 mẫu thì xuất hiện ít hơn 10 lần). Những ký tự hiếm khi xuất hiện này sẽ được thay thế bởi một token <UNK>. Tiếp theo, ta cố định chiều dài của một URL là L1 = 200 ký tự. Những URL dài hơn 200 ký tự sẽ bị cắt đi chỉ còn 200 ký tự. Những URL ít hơn 200 ký tự sẽ được đệm thêm token <PAD> cho đến khi đủ 200 ký tự.

Mỗi ký tự sẽ được biểu diễn bằng một vector k chiều, ở đây chọn ngẫu nhiên k = 32. Vector này sẽ được khởi tạo ngẫu nhiên cho các ký tự và được học trong quá trình huấn luyện. Bằng cách này, mỗi URL sẽ được biểu diễn là một ma trận 200 x 32, như vậy có thể áp dụng CNN để học các thuộc tính của URL [2].



Hình 4: Mô hình CNN mức ký tự

Với phương pháp này, CNN ở mức ký tự có thể dễ dàng phân tích được các URL mới chưa bao giờ xuất hiện trong bộ dữ liệu huấn luyện, từ đó giảm thiểu tối đa việc không thể nhận biết được các mẫu URL mới như các phương pháp học sâu cũ. Bên cạnh đó, số lượng ký tự là có hạn vì vậy dung lượng của mô hình là cố định, khác với việc áp dụng CNN ở mức từ vựng có dung lượng tăng theo kích thước dữ liệu.

Tuy có nhiều ưu điểm, nhưng CNN ở mức ký tự cũng có một số hạn chế. Điển hình là khi thay đổi một vài ký tự từ URL sạch [3], CNN ở mức ký tự sẽ không thể phát hiện được điểm bất thường này, ví dụ: facebook -> facebock. Nguyên nhân là do với đa số các ký tự giống nhau thì CNN mức ký tự sẽ cho ra các kết quả gần giống nhau. Vì vậy, CNN mức ký tự đứng một mình vẫn chưa đảm bảo việc phát hiện URL độc hại một cách chính xác, chúng ta cần thêm CNN mức từ vựng để giải quyết những vấn đề nêu trên.

2.2. Áp dụng CNN mức từ vựng (Word-level CNN)

Về cơ bản, CNN mức từ vựng cũng giống như làm với CNN mức ký tự, nhưng lần này mỗi hàng của ma trận thay vì là một biểu diễn của một ký tự, thì sẽ là một biểu diễn của một từ.

Trước tiên, chúng ta xác định các từ có trong tập dữ liệu huấn luyện. Không giống như CNN mức ký tự có số lượng cố định, CNN mức từ vựng có số lượng từ tăng theo kích thước của dữ liệu, bởi vì từ mới có thể xuất hiện trong các URL khác nhau. Việc xác định các từ phân tách bởi một vài ký tự đặc biệt như: '.', '/',… [3][4]. Những từ hiếm gặp (xuất hiện chỉ một lần trong toàn bộ dữ liệu huấn luyện) sẽ được thay thế bằng token <UNK>. Một URL được cố định chiều dài là L2 = 200 từ. Tương tự Char-level CNN, các URL không đủ 200 từ sẽ được đệm thêm token <PAD> cho đủ và các URL nhiều hơn sẽ bị cắt bớt từ đi.

Mỗi từ sẽ được biểu diễn bằng vector k = 32 chiều. Từ đó, ta có một URL sẽ được biểu diễn bằng một ma trận 200 x 32, nơi mà có thể áp dụng CNN để học các thuộc tính về từ của URL.

2.3. Thu thập dữ liệu huấn luyện

Một mẫu dữ liệu nằm trên 1 dòng theo định dạng "URL, nhãn". Trong đó nhãn nhận giá trị 0 hoặc 1, tương ứng với URL sạch (0) hoặc độc hại (1).

Việc thu thập dữ liệu diễn ra trong vòng 1 tháng với các công việc sau:

- Trước tiên thu thập các tập dữ liệu có sẵn trên mạng và định lại.
- Dữ liệu URL độc hại được thu thập từ các trang web cung cấp dữ liệu theo thời gian thực: OpenPhish, PhishTank, …
- Dữ liệu URL sạch được thu thập bổ sung từ lịch sử duyệt web: Các thành viên tạo máy ảo để sử dụng trình duyệt, chỉ truy cập các trang web mạng xã hội, tin tức nổi tiếng,…

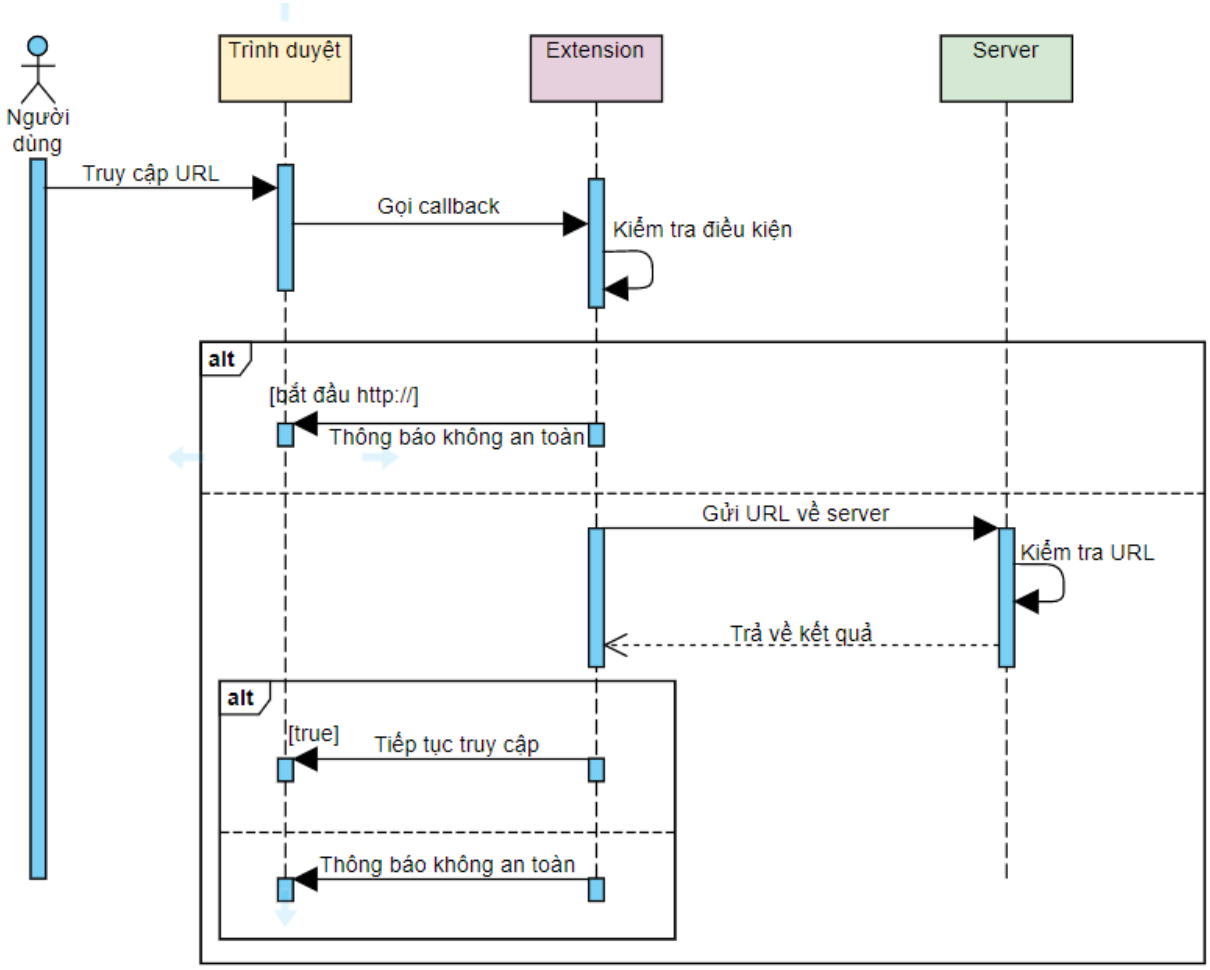
Kết quả sau khi thu thập và xử lý, thu được khoảng 250 000 URL với tỷ lệ sạch/độc là 50/50.

Chương 3: Mô hình phát hiện URL độc hại trên trình duyệt theo thời gian thực

3.1. Tổng quan

Mô hình sẽ được chia làm 2 phần đó là client và server. Trong đó client (ở đây là extension) sẽ có nhiệm vụ thu thập URL được người dùng truy cập và gửi về server. Server có nhiệm vụ phân tích URL nhận được từ client, sử dụng mô hình học sâu đã được huấn luyện, và trả về kết quả phân tích cho client để quyết định chặn hay không.

Ngoài ra, phía client cũng thực hiện một số thao tác kiểm tra đơn giản nhằm tăng tốc độ xử lý.



Hình 5: Sơ đồ hoạt động

3.2. Client

Client là một phần mở rộng của trình duyệt Chrome (extension). Cấu trúc của extension gồm 2 file chính: manifest.js và backgound.js.

Manifest.js là file đăng ký extension với trình chuyệt. Trong file này cần khai báo quyền chặn trang web và file background.jsđể có thể hoạt động.

```
"permissions": [
    "webRequest", "webRequestBlocking", "<all_urls>"
],
"background": {
    "scripts": ["background.js"],
    "persistent": true
}
```

Background.js là file xử lý chính của extension. Trong file này đăng ký callback Listener với sự kiện onBeforeRequest để kiểm tra URL trước khi nó được truy cập.

```
chrome.webRequest.onBeforeRequest.addListener(
  listener,
  {
    urls: ['<all_urls>'],
    types: ["main_frame"],
  },
  ['blocking']
);
```

Khai báo *types: ["main_frame"]* để loại bỏ các URL không cần kiểm tra được tải kèm khi truy cập trang web (như script, ảnh, video).

Việc giao tiếp giữa client server sử dụng đối tượng XMLHttpRequest trong Javascript.

```
var req = new XMLHttpRequest();
req.open('GET', 'http://localhost:5000/?url=' + requestDetails.url, false);
req.send(null);
```

3.3. Server

Server sử dụng thư viện Python Flask để nhận URL gửi từ extension về. Trước khi xử lý, loại bỏ phần giao thức (https, ftp) và "www." trong URL để việc dự đoán chính xác nhất. Nguyên nhân là khi huấn luyện dữ liệu cũng đã loại bỏ những thành phần này.

```
if url.startswith("http://"):
    url=url[7:]
if url.startswith("https://"):
    url=url[8:]
if url.startswith("ftp://"):
    url=url[6:]
if url.startswith("www."):
    url = url[4:]
```

Server sẽ nạp lên model mới nhất được huấn luyện trong thư mục checkpoints và thực hiện dự đoán URL là độc hại hay không, sau đó trả về cho extension kết quả là một xâu "true", tương ứng với URL sạch, hoặc "false" tương ứng với URL độc hại.

```
checkpoint_file = tf.train.latest_checkpoint(FLAGS["log.checkpoint_dir"])
saver = tf.train.import_meta_graph("{}.meta".format(checkpoint_file))
saver.restore(sess, checkpoint_file)
...
if (int)(batch_predictions[0]) == 0:
    return "true"
else:
    return "false"
```

Kết luận

Trong bài báo cáo, chúng em đã trình bày tổng quát về mạng nơ ron và mô hình mạng nơ ron tích chập. Đồng thời, báo cáo cũng đưa ra phương pháp mạng nơ ron tích chập đối với mức ký tự và mức từ vựng. Từ đó, áp dụng cả hai phương pháp vào việc huấn luyện mô hình phát hiện URL độc hại để cho độ chính xác cao nhất, thông qua việc học các thuộc tính về ký tự và từ vựng của URL. Với những lý thuyết đó, mang vào thực tế đã tạo ra một mô hình extension-server có thể phát hiện được những URL không an toàn trong trình duyệt Chrome theo thời gian thực, để kịp thời cảnh báo tới người dùng.

Tài liệu tham khảo

[1] X. Zhang, J. Zhao, và Y. Lecun, "Character-level convolutional networks for text classification", *Advances in Neural Information Processing Systems*, vol 2015-Janua, tr 649–657, 2015.

[2] J. Saxe và K. Berlin, "eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys", 2017.

[3] W. Chu, B. B. Zhu, F. Xue, X. Guan, và Z. Cai, "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs", *IEEE International Conference on Communications*, tr 1990–1994, 2013.

[4] A. Blum, B. Wardman, T. Solorio, và G. Warner, "Lexical feature based phishing URL detection using online learning", *Proceedings of the ACM Conference on Computer and Communications Security*, tr 54–60, 2010.

[5] J. Ma, L. K. Saul, S. Savage, và G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs", *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, tr 1245–1253, 2009.

Và còn rất nhiều bài viết tham khảo khác trên các trang chuyên ngành, diễn đàn.