MỘT GIẢI PHÁP ỨNG DỤNG TRÍ TUỆ NHÂN TẠO CHO NHÀ THÔNG MINH

AN APPROACH FOR APPLYING ARTIFICIAL INTELLIGENCE TO SMART-HOME

Nguyễn Tất Bảo Thiện[1], Trương Tiến Sỹ[2]
[1]Trường Đại học Thủy lợi, Việt Nam
[2]Học viện Công nghệ Bưu chính Viễn thông, Cơ sở tại TP. Hồ Chí Minh, Việt Nam

Ngày toà soạn nhận bài 20/3/2019, ngày phản biện đánh giá 9/04/2019, ngày chấp nhận đăng 20/5/2019

TÓM TẮT

Trong những năm gần đây, cùng với sự phát triển của khoa học và công nghệ, đặc biệt là lĩnh vực trí tuệ nhân tạo đã len lỏi vào trong mọi lĩnh vực của cuộc sống mà ngôi nhà chúng ta đang ở cũng không phải là một ngoại lệ. Các căn nhà mà ở đó các thiết bị được điều khiển bằng giọng nói đã và đang là xu hướng phát triển của nhà thông minh. Trong bài báo này, chúng tôi có ứng dụng thêm các mô hình máy học vào các căn nhà điều khiển thiết bị bằng giọng nói nhằm tăng cường tính chính xác cũng như tính linh hoạt của hệ thống (ví dụ như khi người điều khiển có thể nói những "ý định điều khiển" gần giống với những gì đã được lập trình thì hệ thống điều khiển vẫn hiểu và thực hiện). Với việc ứng dụng các mô hình máy học như mạng nơ ron nhiều lớp, véc-tơ hỗ trợ một lớp thì chúng tôi đã xây dựng một bộ xử lý trung tâm làm nhiệm vụ phân loại dữ liệu và đưa ra quyết định điều khiển thiết bị cho nhà thông minh.

Từ khóa: *máy học; mạng nơron nhiều lớp; nhà thông minh; nhận diện giọng nói; véc-tơ hỗ trợ một lớp.*

ABSTRACT

In recent years, along with the development of science and technology, especially the techniques of artificial intelligence has appeared in many areas of life in that smart-home is not an exception. The houses of which devices were controlled through the voice recognition has been the trend of smart-home development. In this article, we applied machine learning models to a smart-home with voice-controlled devices to enhance the accuracy and flexibility of the system (for example, when the operator says "control intents" which are similar to the programmed keywords, the control system still understands and executes). With the applications of machine learning models such as multi-layer neural networks and one-class support vector machine, we have already built a processing centre to classify data and make control decisions for smart-homes.

Keywords: *machine learning; multi-layer perceptron; one-class support vector; smart-home; speech recognition.*

1. ĐẶT VẤN ĐỀ

Nhà thông minh là một căn nhà được tích hợp các phương pháp, hệ thống nhằm vận hành và kiểm soát các thiết bị điện tử để giảm thiểu hoạt động của con người ở mức tối thiểu. Nhà thông minh được thiết kế và phát triển để điều khiển và giám sát các thiết bị khác nhau như đèn, cảm biến nhiệt độ, độ ẩm, thiết bị phát hiện khói và cháy cũng như hệ thống an ninh [1]. Một trong những ưu điểm nổi bật nhất của nhà thông minh là các thiết bị trong nhà có thể được điều khiển và quản lý dễ dàng bằng điện thoại thông minh, máy tính bảng hay máy tính xách tay [2].

Hiện nay, các công ty công nghệ lớn của thế giới đã và đang phát triển các ứng dụng phục vụ cho nhà thông minh như Google Assistant của Google, HomeKit của Apple v.v. Tại Việt Nam, các công ty công nghệ cũng đã phát triển các sản phẩm như Bkav Smarthome, nhà thông minh của Lumi. Các sản phẩm này thường tập trung vào việc điều khiển các thiết bị từ xa thông qua thiết bị di động, công tắc cảm biến hay qua các ứng dụng trên nền tảng di động.

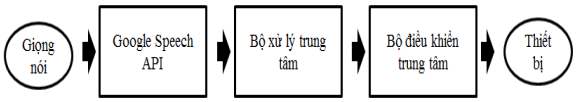
Đã có những nghiên cứu về việc ứng dụng trí tuệ vào nhà thông minh mà điển hình là việc sử dụng giọng nói để điều khiển các thiết bị trong nhà [3] [4]. Aml A. Arriany và Mohamed S. Musbah đã đề xuất giải pháp điều khiển các thiết bị trong nhà bằng giọng nói thông qua bộ tự động nhận diện giọng nói, bộ xử lý trung tâm và bộ điều khiển trung tâm để điều khiển các thiết bị như đèn, quạt. Noriyuki Kawarazaki và Tadashi Yoshidome đã xây dựng hệ thống điều khiển thiết bị bằng giọng nói thông qua bộ PMRC (Programmable multi remote controller). Khi người điều khiển gửi lệnh bằng giọng nói đến hệ thống, PMRC gửi tín hiệu đến các thiết bị điện dân dụng để điều khiển chúng. Điểm chung của các nghiên cứu này là việc tập trung xây dựng mô hình điều khiển các thiết bị từ xa bằng giọng nói. Tức là khi con người điều khiển thiết bị trong nhà bằng giọng nói thì chúng sẽ được chuyển về dạng văn bản để so sánh với các từ khóa đã được lập trình sẵn từ trước (như tắt đèn, mở đèn, …).

Trong bài báo này, chúng tôi tập trung xây dựng các mô hình máy học để xử lý dữ liệu văn bản thu được từ bộ nhận dạng giọng nói nhằm tăng cường sự chính xác cũng như sự linh hoạt của hệ thống điều khiển thiết bị thông qua giọng nói. (ví dụ khi chuỗi điều khiển nhận được là "bật điện lên" thì đối với các hệ thống trên, chuỗi thu được này không trùng với bất kỳ "từ khóa" nào hết. Vì thế hệ thống sẽ không điều khiển thiết bị, việc này sẽ gây khó khăn cho người sử dụng khi họ phải nhớ các từ khóa để điều khiển thiết bị). Với việc áp dụng các mô hình máy học vào việc phân tích dữ liệu văn bản thu được, chúng tôi có thể giúp hệ thống điều khiển "hiểu" được ý muốn của con người nhằm tăng tính chính xác và sự linh hoạt cho hệ thống.

2. MÔ HÌNH ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI

Mô hình điều khiển các thiết bị trong nhà bằng giọng nói bao gồm 3 bộ xử lý dữ liệu được mô tả trong Hình 1:

- Bộ chuyển đổi giọng nói thành văn bản với nhiệm vụ chính là biến đổi giọng nói thành văn bản.
- Bộ xử lý trung tâm gồm các mô hình máy học có chức năng phân loại các văn bản thu được từ bộ nhận dạng giọng nói.
- Bộ điều khiển trung tâm là nơi kết nối với các thiết bị đèn, quạt, cửa và nhận lệnh điều khiển từ bộ xử lý trung tâm.



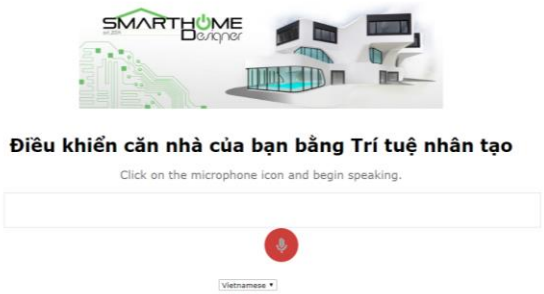
Hình 1. *Mô hình điều khiển thiết bị trong nhà bằng giọng nói*

2.1. Bộ chuyển đổi văn bản thành giọng nói

Hệ thống nhận diện giọng nói là hệ thống có khả năng nhận và dịch các lệnh thu được từ giọng nói con người. Nhận dạng giọng nói gồm 2 thuật ngữ chính:

- Voice Recognition liên quan đến việc xác định giọng nói chính xác của một cá nhân nào đó, tương tự một phương pháp sinh trắc học.
- Speech Recognition là việc xác định những từ ngữ trong câu rồi dịch chúng sang ngôn ngữ máy tính.

Với bài báo này, chúng tôi xây dựng một bộ chuyển đổi giọng nói thành văn bản dựa trên nền tảng của Google Cloud Speech API (Google Speech API là một dịch vụ cho phép chuyển đổi giọng nói thành văn bản và hiện tại đã hỗ trợ ngôn ngữ Tiếng Việt). Nhiệm vụ chính của của bộ nhận diện giọng nói này chuyển đổi giọng nói đầu vào thành văn bản và gửi chúng lên bộ xử lý trung tâm bằng phương thức GET.



Hình 2. *Google Speech API*

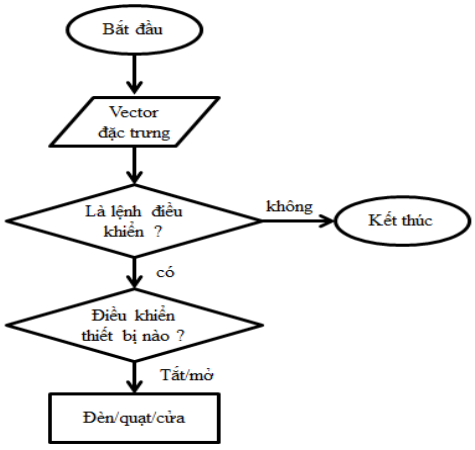
2.2. Bộ xử lý trung tâm

Chúng tôi sử dụng server có cài hệ điều hành Linux để xử lý thông tin nhận được từ bộ chuyển đổi giọng nói thành văn bản. Các văn bản nhận được này sẽ được phân loại theo các tiêu chí như sau:

- Các văn bản thu được có phải là các câu lệnh điều khiển thiết bị hay chỉ là một câu nói thông thường.
- Nếu chúng là câu lệnh điều khiển thiết bị thì chúng điều khiển thiết bị nào?

Để giải quyết các câu hỏi trên, chúng tôi lựa chọn hai mô hình máy học là Oneclass-SVM và Multi-layer Perceptron. Việc tại sao lại lựa chọn hai mô hình này sẽ được chúng tôi trình bày chi tiết ở mục 3.2 và 3.3.

Sau khi dữ liệu văn bản đầu vào được tiền xử lý và trích rút đặc trưng thì sẽ được đưa vào hai mô hình máy học là Oneclass-SVM và Multi-layer Perceptron để phân loại dữ liệu theo mô hình giải thuật Hình 3.



Hình 3. *Mô hình lưu đồ thuật toán trong bộ xử lý trung tâm.*

2.3. Bộ điều khiển trung tâm

Máy tính nhúng là một thiết bị, hệ thống được thiết kế để phục vụ cho một yêu cầu, ứng dụng hoặc một chức năng nhất định nào đó và được ứng dụng nhiều trong lĩnh vực công nghiệp, tự động hóa điều khiển hay truyền tin …. Một số ưu điểm của máy tính nhúng như chi phí thấp, kích thước nhỏ gọn, dễ sử dụng. Chính vì thế chúng tôi đã sử dụng Raspberry Pi để làm bộ điều khiển trung tâm vì nó đủ mạnh để có thể chạy như một máy chủ hoạt động đa nhiệm, có hỗ trợ kết nối internet và có thể cài hệ điều hành để chạy một số ứng dụng phần mềm phù hợp cho việc thiết kế và nghiên cứu. Ngoài ra, chúng tôi còn sử dụng các đèn LED để mô hình hóa các thiết bị điện trong nhà như đèn, quạt, cửa.

Hình 4. *Raspberry Pi.*

Raspberry Pi là một bo mạch máy tính đơn giản được phát triển tại Anh bởi Raspberry Pi Foundation với mục đích giảng dạy khoa học máy tính tại trường [5].Trên bo mạch của Raspnerry Pi có các thành phần: CPU, GPU, RAM, khe cắm, thẻ microSD, Wifi, Bluetooth và 4 cổng USB. Sau khi cài đặt hệ điều hành vào thẻ SD và gắn vào SD card, chúng tôi thực hiện kết nối với mạng thông qua cổng LAN của Raspberry Pi và đặt IP tĩnh dễ dàng kết nối và quản lý về sau.

3. XÂY DỰNG MÔ HÌNH MÁY HỌC CHO VIỆC PHÂN LOẠI DỮ LIỆU ĐẦU VÀO

Để xây dựng các mô hình máy học thì chúng ta cần có những phép biến đổi để có thể có những véc-tơ đặc trưng phù hợp từ dữ liệu thô ban đầu (các dữ liệu thô này có thể là văn bản, hình ảnh v.v). Quá trình này được gọi là Feature Engineering (trích chọn đặc trưng) bởi vì khi thực hiện những bài toán

máy học thực tế thì hầu hết dữ liệu đầu vào chỉ là những dữ liệu thô chưa qua xử lý và chọn lọc. Chúng ta cần phải tìm một phép biến đổi để loại ra nhiễu (noise) và đưa dữ liệu thô với số chiều khác nhau về cùng một dạng chuẩn (cùng là các vector hoặc ma trận). Dữ liệu chuẩn mới này phải đảm bảo giữ được những thông tin đặc trưng (features) của dữ liệu thô ban đầu. Ngoài ra, tùy thuộc vào từng bài toán khác nhau thì chúng ta cần thiết kế những phép biến đổi để có thể có những đặc trưng phù hợp.

Chúng tôi có một tập dữ liệu thô với khoảng hơn 200 câu văn bản tiếng Việt dùng để điều khiển tắt/mở ba thiết bị đèn, quạt và cửa. Tập dữ liệu thô này được mô tả trong Bảng 1.

Bảng 1. *Bảng tổng quát tập dữ liệu huấn luyện và dữ liệu kiểm thử*

Tập dữ liệu	**File chứa dữ liệu**
Tập dữ liệu huấn luyện	*batden.txt* (chứa các câu có "ý nghĩa" bật đèn) Ví dụ: Bật đèn, mở đèn đi, tắt điện chi vậy…
	tatden.txt (chứa các câu có "ý nghĩa" tắt đèn) Ví dụ: tắt đèn, sáng rồi tắt đèn đi.
	batquat.txt (chứa các câu có "ý nghĩa" bật quạt) Ví dụ: bật quạt, mở quạt …
	tatquat.txt (chứa các câu có "ý nghĩa" tắt quạt) Ví dụ: tắt quạt đi, tắt cái quạt đi.
	mocua.txt (chứa các câu có "ý nghĩa" mở cửa) Ví dụ: mở cửa, sao không mở cửa đi, mở cửa giúp tôi …
	dongcua.txt (chứa các câu có "ý nghĩa" đóng cửa) Ví dụ: đóng cửa, đóng cửa đi, khép cửa lại đi …
Tập dữ liệu kiểm thử	*test_accept.txt* (chứa các tập dữ liệu thuộc loại câu lệnh điều khiển thiết bị) Ví dụ: tắt cái bóng đèn kia giúp tôi, sao không tắt điện …
	test_fail.txt (chứa tập dữ liệu không phải là câu lệnh điều khiển) Ví dụ: cậu có khỏe không, trời lạnh quá, xin chào …

Ở giai đoạn huấn luyện, chúng tôi thực hiện một số bước sau:

- Tập dữ liệu huấn luyện (data training) lấy ngẫu nhiên ¾ từ tập dữ liệu thô ban đầu.
- Tiền xử lý văn bản tiếng Việt.
- Trích chọn đặc trưng: tùy thuộc vào từng mô hình sử dụng mà chúng ta có thể lựa chọn ra những đặc trưng phù hợp. Đối với bài báo này, chúng tôi sử dụng hai mô hình máy học là OCSVM và Multi-layer Perceptron. Quá trình trích chọn đặc trưng của từng mô hình sẽ được chúng tôi trình bày tại mục 3.2 và 3.3.
- Mô hình OCSVM: đây là mô hình máy học có giám sát. Tuy nhiên do đặc trưng của OCSVM nên dữ liệu đầu vào chỉ là các véc-tơ đặc trưng (features vector).
- Mô hình Multi-layer Perceptron: đây là mô hình máy học có giám sát vì vậy dữ liệu đầu vào sẽ là một cặp (feature vector, label) với quy định như sau:

Bảng 2. *Ý nghĩa nhãn của mô hình MLP*

Nhãn	**Nhóm**
1	Bật đèn
2	Tắt đèn
3	Bật quạt
4	Tắt quạt
5	Mở cửa
6	Đóng cửa

Trong giai đoạn kiểm thử (testing), tập dữ liệu kiểm thử (data testing) lấy ngẫu nhiên ¼ dữ liệu thô ban đầu và thêm vào một số câu văn bản tiếng Việt không liên quan đến việc điều khiển các thiết bị đèn, quạt và cửa. Tập dữ liệu kiểm thử này sẽ được đưa qua các bước tiền xử lý, trích chọn đặc trưng và sau đó đưa qua các mô hình máy học đã được thiết kế ở giai đoạn huấn luyện. Kết quả là các văn bản tiếng Việt sẽ được phân loại theo yêu cầu của bài toán. Mô hình OCSVM: nhãn đầu ra sẽ là 1 nếu dữ liệu kiểm thử thuộc loại là câu lệnh điều khiển thiết bị và các dữ liệu này sẽ đi tiếp vào mô hình Multi-layer Perceptron. Nếu nhãn đầu ra là -1 thì dữ liệu kiểm thử là dữ liệu không liên quan đến việc điều khiển thiết bị. Mô hình Multilayer-Perceptron: nhãn đầu ra sẽ tương ứng với các lệnh điều khiển được giới thiệu trong bảng 2.

3.1. Tiền xử lý và véc-tơ hóa dữ liệu Tiếng Việt

Tiền xử lý dữ liệu là bước đầu tiên trong việc xử lý ngôn ngữ tự nhiên bao gồm một số bước cơ bản như sau:

- Làm sạch: loại bỏ nhiễu trong dữ liệu đầu vào (nhiễu này có thể là các ký tự đặc biệt, các thẻ html, v.v).

- Tách từ trong câu: đối với tiếng Việt thì có một số từ là từ đơn thì không có nghĩa hoặc mang một ý nghĩa khác nhưng nếu chúng được ghép với một từ khác thì nó sẽ có nghĩa mới.

- Chuẩn hóa từ: chuẩn hóa dạng ký tự như viết hoa, viết thường, kiểu chữ v.v.

- Loại bỏ stop words: loại bỏ các từ không cần thiết, quan trọng. Trong Tiếng Việt, nếu chỉ xét đến các câu lệnh điều khiển thiết bị thì chúng ta có thể loại bỏ một số từ "tôi", "làm ơn", "với", v.v.

Trong bài báo này, chúng tôi thực hiện tiền xử lý các văn bản tiếng Việt ở tập dữ liệu huấn luyện dựa trên các thư viện có sẵn của ngôn ngữ lập trình Python. Từ đó xây dựng mô hình tiền xử lý cho các dữ liệu đầu vào của hệ thống điều khiển thiết bị.

Ví dụ: tập văn bản cần tiền xử lý gồm các câu thuộc tập dữ liệu data_input:

data_input= ['Tắt đèn', 'trời sáng rồi, sao không tắt điện đi!', 'tắt đèn giúp tôi với',…].

Chúng tôi sẽ tiến hành loại bỏ nhiễu và chuẩn hóa từ thuộc tập dữ liệu này dựa vào thư viện gensim có sẵn trong Python. Thư viện này sẽ bỏ các dấu câu như (,.#! …) và chuẩn hóa các từ trong câu. Do tập dữ liệu data_input thuộc dạng danh sách nên chúng tôi sử dụng hàm for để xử lý từng phần tử của danh sách này. Kết quả được chúng tôi lưu lại trong biến temp. Sau đó, chúng tôi tiếp tục sử dụng thư viện *pyvi. ViTokenizer* để nối các từ ghép có nghĩa trong tiếng Việt. Các từ nối có nghĩa sẽ có dấu gạch nối giữa các từ (ví dụ: *tắt_đèn,…*).

```
data_train = [];
for i in range(0,len(data_input)):
temp=gensim.utils.simple_preprocess(data_input[i]);
data_train.append(ViTokenizer.tokenize(' '.join(temp)));
```

Kết quả của quá trình loại bỏ nhiễu, chuẩn hóa và tách từ trong câu sẽ là tập dữ liệu data_train.

data_train = ['tắt_đèn', 'trời sáng rồi sao không tắt điện đi', 'tắt_đèn giúp tôi với',…];

Do trong câu có rất nhiều từ không liên quan đến việc phân loại điều khiển nên chúng ta sẽ bỏ bớt các từ này. Chúng tôi có một tập tin chứa các stopword của tiếng Việt như (tôi, tớ, cậu, ….). Dựa vào tập tin stopword này, chúng tôi sẽ bỏ bớt đi các từ không cần thiết cho quá trình phân loại dữ liệu.

```
stopword = ['tôi','rồi','với','giúp',...];
for i in range(0,len(data_train)):
words_segment =  data_train[i];
for j in stopword:
words_segment=words_segment.replace(j," ");
words_segment=".join(words_segment.split());
data_train[i] = words_segment;
```

Kết quả sẽ cuối cùng sẽ là một tập dữ liệu được chuẩn hóa làm đầu vào cho việc véc-tơ hóa dữ liệu.

data_train= ['tắt_đèn', 'trời sáng sao không tắt điện đi', 'tắt_đèn',…]

- Véc-tơ hóa dữ liệu: chuyển câu, từ thành dạng véc-tơ. Đây là bước rất quan trọng trong viêc tiền xử lý ngôn ngữ tự nhiên, kết quả của quá trình này là các điểm dữ liệu sẽ được biểu diễn bằng các véc-tơ, các véc-tơ này còn được gọi là feature vector (véc-tơ đặc trưng) có độ dài như nhau. Để chuyển đổi văn bản tiếng Việt thành các véc-tơ thì chúng tôi sử dụng phương pháp *Bag of Words* để đưa các dữ liệu này về dạng véc-tơ có độ dài bằng nhau [6].

Phương pháp Bag-of-words là phương pháp tạo ra một từ điển gồm tất cả các từ xuất hiện trong tập các câu dữ liệu đầu vào. Các từ này sẽ đi kèm với một số index. Dựa vào từ điển này, mô hình sẽ tiến hành tạo véc-tơ lưu trữ số lần xuất hiện của từ trong từ điển tương ứng với mỗi câu. Số chiều của vec-tơ tương ứng với số lượng từ trong từ điển. Ví dụ:

data_train = ['tắt_đèn', 'trời sáng rồi sao không tắt điện đi', 'tắt_đèn giúp tôi với',…];

Chúng tôi đã sử dụng các thư viện sklearn trong Python để thực hiện vec-tơ hóa. Đây là một thư viện có các module hỗ trợ cho việc véc-tơ hóa dữ liệu văn bản.

vectorizer= CountVectorizer()

vectorizer.fit(data_train)

Module này sẽ tạo ra một từ điển bao gồm tất cả các từ xuất hiện trong dữ liệu đầu vào *data_train*.

*vectorizer.vocabulary_*

{'không': 0,'sao': 1,'sáng': 2,'trời': 3,'tắt': 4,'tắt_đèn': 5,'đi': 6,'điện': 7, …}

Véc-tơ hóa các câu trong tập *data_train*, chúng ta sẽ được ma trận các véc-tơ đặc trưng tương ứng. Các véc-tơ này có số chiều bằng với số phần tử có trong từ điển.

vector=vectorizer.transform(data_train).todense()

matrix([[0, 0, 0, 0, 0, 1, 0, 0],[1, 1, 1, 1, 1, 0, 1, 1],[0, 0, 0, 0, 0, 1, 0, 0],…]

Tùy thuộc vào từng loại mô hình máy học mà chúng ta có thể lựa chọn đặc trưng sao cho phù hợp với nhu cầu của mô hình đó. Cách lựa chọn véc-tơ đặc trưng cho mô hình máy học One-class SVM và Neural Network được chúng tôi trình bày trong phần xây dựng mô hình máy học ở phần 4.2.

Như vậy, từ dữ liệu văn bản thô ban đầu, chúng tôi đã chuyển đổi các dữ liệu này về cùng một kích thước để làm dữ liệu đầu vào cho các thuật toán máy học.

3.2. Mô hình One-class SVM

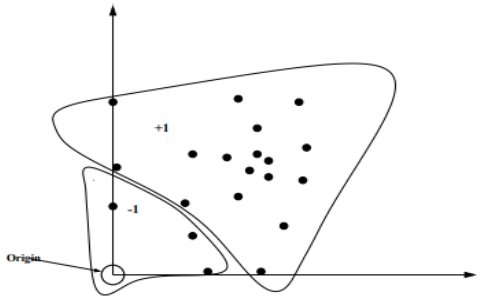
- **Phát biểu bài toán**

Phân loại câu lệnh đầu vào có phải là câu lệnh điều khiển các thiết bị hay không?

- **Giải quyết bài toán**

Yêu cầu của bài toán là phân loại dữ liệu đầu vào có phải là dữ liệu điều khiển thiết bị hay không. Đây là bài toán phân loại một lớp, vì thế chúng ta phải tìm một thuật toán mà khi dữ liệu mới đưa vào, nó sẽ so sánh với dữ liệu đã được huấn luyện (ở đây là các câu lệnh điều khiển thiết bị). Nếu cùng loại, thì mô hình máy học sẽ trả lời "1", còn nếu không thì sẽ là "-1". Chính vì thế chúng tôi đề xuất lựa chọn mô hình máy học One-class Support Vector Machine (OCSVM). Đây là một mô hình máy học theo phương thức học có giám sát do Scholkopf và các cộng sự đề xuất và rất phù hợp cho các bài toán phân loại một lớp [7].

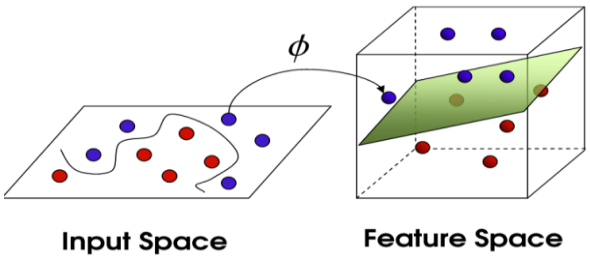
Chúng tôi sử dụng mô hình này với mục đích loại đi những dữ liệu đầu vào không liên quan đến việc điều khiển thiết bị trong nhà như "hôm nay trời nắng quá nhỉ" hay "tắt xe máy kìa" v.v. Tập dữ liệu huấn luyện của mô hình này sẽ là một số câu lệnh điều khiển thiết bị như "tắt điện đi", "bật quạt lên nào" v.v. Sau khi quá trình huấn luyện kết thúc thì mô hình sẽ tạo ra một "đường bao" bao quanh các điểm dữ liệu huấn luyện.



Hình 5. *Phân loại One-class SVM [7].*

Như vậy, khi một dữ liệu mới đi vào mô hình OCSVM này, nếu dữ liệu nằm trong đường bao được tạo ra trong lúc huấn luyện (khu vực mang nhãn là +1) thì có nghĩa là dữ liệu này thuộc loại dữ liệu điều khiển thiết bị, còn nếu dữ liệu nằm ngoài đường bao thì dữ liệu này không quan trọng đối với hệ thống điều khiển thiết bị (sẽ mang nhãn -1) và được bỏ qua.

Cơ sở toán học của thuật toán này được trình bày ở [8][9][10]. Giả sử ta có tập huấn luyện với n điểm $x_1,\ldots, x_n$ ở không gian R^n. Sử dụng hạt nhân kernel để ánh xạ các điểm này tới không gian mới mà ở không gian này dữ liệu có thể được tách ra bởi một siêu phẳng (hyperplane).



Hình 6. *Ánh xạ điểm dữ liệu lên không gian mới*

Sau đó, chúng ta thực hiện phân chia các điểm dữ liệu này trong không gian mới từ điểm gốc (origin) và tối đa hóa khoảng cách từ siêu phẳng (hyperplane) đến điểm gốc (origin) bằng hàm tối thiểu hóa bậc hai:

$$\min_{\omega,\xi_i,\rho} \frac{1}{2}\|\omega\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \qquad (1)$$

Trong đó $(\omega.\Phi(x_i)) \geq \rho - \xi_i$, $\forall i = 1,\ldots,n$, $\xi_i \geq 0$, $\forall i = 1,\ldots,n$. ω là vector của siêu mặt phẳng (hyperplane), $\|\omega\| = \sqrt{\sum_{i=1}^{d}\omega_i^2}$ với d là số chiều của không gian mới, ξ_i là biến slack, ρ là bias là khoảng cách từ gốc (origin) đến siêu phẳng (hyperplane), ν∈ (0,1]. Hàm (1) sẽ trả về một không gian con chứa các điểm thuộc tập huấn luyện. Sau khi xác định được siêu mặt phẳng (hyperplane) thì sử dụng hàm quyết định (2) để xác định các điểm dữ liệu có thuộc tập huấn luyện hay không. Nếu kết quả của hàm quyết định là 1 thì dữ liệu đó cùng loại với nhóm tập dữ liệu huấn luyện, còn nếu là -1 thì điểm dữ liệu đó không cùng loại với dữ liệu huấn luyện.

$$f(x) = \text{sgn}((\omega.\Phi(x_i)) - \rho) \qquad (2)$$

3.3. *Mô hình Multi-layer Perceptron*

- **Phát biểu bài toán**

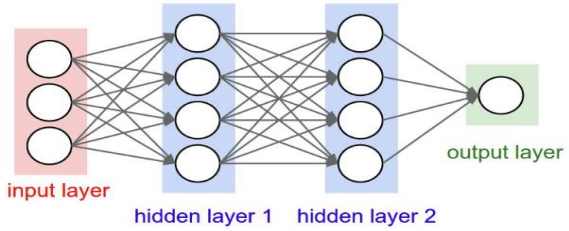
Với dữ liệu đầu vào là các câu lệnh điều khiển thiết bị thì làm thế nào để phân loại các dữ liệu này điều khiển các thiết bị nào?

- **Giải quyết bài toán**

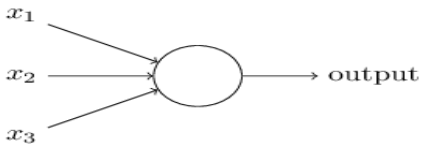
Đây là một bài toán phân loại dữ liệu vào một trong các lớp (trong bài báo này là 6 lớp bao gồm: tắt đèn, bật đèn, tắt quạt, bật quạt, đóng cửa và mở cửa). Đầu vào của bài toán này là các câu lệnh điều khiển, chúng ta phải tìm ra một mô hình để phân loại các dữ liệu này là điều khiển thiết bị gì (đèn, quạt hay cửa) và hành động sẽ là gì (tắt hay mở). Có rất nhiều mô hình có thể đáp ứng được những yêu cầu trên mà một trong số đó là mô hình Multi-layer Perceptron. Đây là một mô hình mạng Neural nhân tạo truyền thẳng nhiều lớp. Mạng Neural nhân tạo (*Artificial Neural Network - ANN*) xuất phát từ việc mô phỏng mạng nơ-ron sinh học của bộ não con người [11] [12]. Mạng nơ-ron nhân tạo hiện đang được áp dụng cho nhiều lĩnh vực khác nhau trong các trường đại học, học viện hay các ngành công nghiệp [13] [14].

Đặc trưng của các mô hình ANN là các nơ-ron (hay còn gọi là perceptron). Một mạng nơ-ron nhân tạo thường có 3 lớp: Input layer, Hidden layer và Output layer. Ở mỗi tầng, số lượng các nơ-ron có thể khác nhau tùy thuộc vào cách giải quyết bài toán. Nhưng thường thì các tầng ẩn sẽ có số lượng nơ-ron bằng nhau. Ngoài ra, nơ-ron ở các

tầng thường được liên kết đôi một với nhau để tạo thành mạng kết nối đầy đủ (full-connected network). Các liên kết này sẽ kèm theo một trọng số (weight) nào đó đặc trưng cho tính kích hoạt giữa các nơ-ron.



Hình 7. *Mô hình mạng Nơ-ron nhân tạo*



Hình 8. *Mô hình Perceptron*

Mỗi nơ-ron tính toán tổng các input đi vào bằng cách nhân các giá trị input này với trọng số (weight) tương ứng và cộng thêm một giá trị *bias (b)* bằng công thức:

$$o = \sum_i \omega_i x_i + b \tag{3}$$

Cho $x_0 = 1, \omega_0 = b$ thì công thức (3) có thể được viết lại như sau:

$$o = \omega^T x \tag{4}$$

Nếu giá trị o này vượt qua một ngưỡng nào đó thì nơ-ron này sẽ phát ra một output và ngược lại thì không. Để quyết định ngưỡng này thì mô hình máy học sẽ sử dụng một hàm số được gọi chung là hàm kích hoạt (activation function), hàm kích hoạt này có thể là hàm sigmoid, tanh, ReLU [14]. Trong bài báo này, chúng tôi sử dụng hàm sigmoid để làm hàm quyết định, hàm sigmoid có công thức như sau:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{5}$$

Như vậy, giá trị output của perceptron sẽ được biểu diễn qua một hàm kích hoạt $f(z)$ như sau:

$$o = f(z) = f(\omega^T x) = \frac{1}{1+\exp(-\omega^T x)} \tag{6}$$

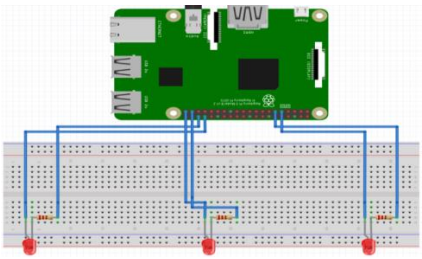
4. HỆ THỐNG THỰC NGHIỆM

4.1. Chuẩn bị phần cứng

Để xây dựng nhà thông minh điều khiển bằng giọng nói chúng ta cần chuẩn bị như sau:

- 1 server Linux: là bộ xử lý trung tâm và webserver cho ứng dụng chuyển đổi văn bản thành giọng nói.
- 1 Raspberry Pi: là bộ điều khiển trung tâm kết nối với các thiết bị.
- 3 đèn LED: đại diện cho các thiết bị đèn, quạt, cửa.

Trong mô hình này, các thiết bị trong nhà thông minh được mô tả đơn giản bằng các LED. Việc đấu nối các thiết bị (LED) vào Raspberry Pi theo thứ tự được cho trong hình 9: Đèn đấu nối với chân GPIO 17, quạt đấu nối với chân GPIO 26 và cửa đấu nối với chân GPIO 16.



Hình 9. *Sơ đồ đấu nối thiết bị*

4.2. Xây dựng mô hình xử lý dữ liệu

4.2.1. Mô hình One-class SVM

Tập dữ liệu huấn luyện: là tập dữ liệu gồm hơn 200 câu lệnh điều khiển tắt/mở thiết bị đèn, quạt, cửa được lưu trữ dưới dạng văn bản vào 6 file tương được trình bày trong Bảng 1. Chúng tôi đã xây dựng một hàm *readfile* để đọc các file chứa dữ liệu huấn luyện này và lưu chúng vào biến *data_train.*

data_train = ['bật điện','bật bóng đèn','bật bóng điện', 'trời tối quá bật điện đi', 'trời tối quá bật đèn đi','trời tối rồi sao không bật điện','tối rồi bật điện lên đi', 'bật điện lên đi',....]

Từ tập dữ liệu này, tiến hành tiền xử lý theo các bước được trình bày ở mục 3.1.

Trích chọn đặc trưng: sử dụng phương pháp Bag-of-word để chuyển đổi dữ liệu văn

bản thành các véc-tơ. Phương pháp Bag-of-word sẽ tạo ra một từ điển gồm tất cả các từ xuất hiện trong tập data_train.

{'bóng': 0,'bật': 1,'chi': 2,'chói': 3,'coi': 4,'cái': 5,...,'đang': 43,'đi': 44,'điện': 45,'đèn': 46,'đóng': 47,'ơn': 48}

Trong đó, chúng tôi quan sát thấy có một số đặc trưng không cần thiết cho việc phân loại dữ liệu điều khiển thiết bị như "tối", "rồi" v.v, và một số đặc trưng rất quan trọng trong việc phân loại là "tắt", "mở", "đèn", v.v. Dựa vào việc tính toán tần suất xuất hiện của các từ trong từ điển thì chúng tôi đã loại bỏ bớt đi các đặc trưng không cần thiết để tạo ra một từ điển mới gồm các đặc trưng quan trọng cho việc phân loại dữ liệu điều khiển.

vectorlizer_svm.vocabulary_ = {'bật': 6,'cửa': 7,'khép': 9,'khóa': 8,'mở': 1,'quạt': 4,'tắt': 0,'điện': 3,'đèn': 2,'đóng': 5}

Sau khi đã có những đặc trưng tốt nhất cho việc phân loại dữ liệu, chúng tôi tiến hành chuyển đổi các dữ liệu văn bản thành các véc-tơ với hàm:

X_train=vectorizer_svm.transform(data_train).todense()

Kết quả của hàm này sẽ trả về một ma trận X_train chứa các véc-tơ đặc trưng tương ứng với các dữ liệu huấn luyện đầu vào.

X_train = [[0, 0, 1, ..., 0, 0, 0],[0, 0, 0, ..., 0, 0, 0],[0, 0, 1, ..., 0, 0, 0],...,[0, 0, 0, ..., 1, 0, 0],[0, 0, 0, ..., 1, 1, 0],[0, 0, 0, ..., 1, 1, 0]])

Mô hình One-class SVM là một loại mô hình máy học có giám sát và dữ liệu đầu vào của mô hình sẽ là các véc-tơ đặc trưng còn dữ liệu đầu ra là các nhãn 1 hoặc -1 đã được giải thích ở phần 3. Với các cơ sở toán học của mô hình đã được trình bày ở mục 3.2. Chúng tôi sử dụng thư viện *Scikit Learning* của ngôn ngữ lập trình Python để xây dựng mô hình One-class SVM. Trong quá trình huấn luyện, chúng tôi đã điều chỉnh các thông số của mô hình One-class SVM để đáp ứng được việc phân loại dữ liệu mới có phải là dữ liệu điều khiển thiết bị hay không với các kernel và thông số khác nhau để chọn ra mô hình giúp cho tập dữ liệu của chúng tôi đạt kết quả tốt. Kết quả đối với từng kernel được trình bày trong bảng 3.

Bảng 3. *Kết quả huấn luyện mô hình One-class SVM với các kernel khác nhau*

	Rbf	**Poly**	**Sigmod**	**Linear**
Tol	0.001	0.001	0.001	0.001
Nu	0.002	0.216	0.052	0.026
Degree(d)	X	3	X	X
gamma(γ)	0.01	0.01	0.01	X
coef0 (r)	X	0.0	0.0	X
Kết quả	64%	88%	98%	96.7%

Trong đó, các thông số *tol, degree, gamma, coef0* là được giữ nguyên ở các mô hình. Chúng tôi thực hiện vòng lặp *for* để tìm ra giá trị *nu* giúp cho mô hình đạt kết quả tốt nhất khi đưa ma trận véc-tơ đặc trưng vào huấn luyện với hàm *clf.fit (X_train)*. Sử dụng tập nhãn đã được tạo từ trước của tập dữ liệu và so sánh với kết quả đầu ra của mô hình. Chúng tôi đã tìm được các thông số giúp cho mô hình có kết quả phân loại tốt nhất.

clf = svm.OneClassSVM(nu=0.052, kernel="sigmoid", tol =0.001, gamma=0.01)

Với *nu* là giới hạn trên của *training errors* và giới hạn dưới của *support vectors*, *tol* là dung sai cho quá trình dừng, *gamma* là hệ hố kernel.

Chạy thử tập dữ liệu kiểm thử với mô hình OCSVM vừa được xây dựng. Tập dữ liệu này bao gồm một số câu lệnh thuộc loại câu lệnh điều khiển và một số câu không thuộc loại này lưu ở 2 file *test_fail* và *test_accept*. Sau quá trình tiền xử lý và trích chọn đặc trưng giống với tập dữ liệu huấn luyện thì kết quả sẽ là một ma trận X_test chứa các véc-tơ đặc trưng. Để kiểm tra sự chính xác của mô hình thì chúng tôi tạo một ma trận chứa các nhãn phân loại của tập dữ liệu kiểm thử bằng bằng cách dán nhãn 1 nếu dữ liệu đó thuộc file *test_accept* và -1 nếu dữ liệu đó thuộc *file_fail* với hàm *np.concatenate*:

*label=np.concatenate((1*np.ones(len(test_fail)),np.ones(len(test_accept))))*

Sau khi đưa tập dữ liệu kiểm thử qua mô hình OCSVM với hàm *clf.predict(X_test)* thì

kết quả trả về sẽ là một ma trận *y_pred* chứa các nhãn 1 hoặc -1 do mô hình dự đoán và số phần tử bằng với số lượng các câu lệnh thuộc tập dữ liệu kiểm thử. Từ ma trận này, chúng tôi tiến hành so sánh với ma trận nhãn *label* bằng hàm *accuracy_score (label, y_pred)*100.* Kết quả ở bảng 3 cho thấy mô hình OCSVM có sự phân loại chính xác gần như 100% đối với bài toán phân loại dữ liệu này.

4.2.2. Mô hình Multi-layer Perceptron

Tập dữ liệu huấn luyện hơn 200 câu lệnh điều khiển thiết bị được chia theo 6 file tương ứng như bảng 1. Với các bước tiền xử lý đã được trình bày từ mục 3.1.

Trích chọn đặc trưng: sự trích chọn đặc trưng đối với từng loại mô hình máy học là khác nhau. Với dữ liệu đầu vào của mô hình MLP đã là các câu lệnh điều khiển thì các từ trong câu lệnh đó sẽ có một ý nghĩa nào đó giúp cho việc phân nhóm dữ liệu dễ dàng hơn. Tương tự như mô hình OCSVM, chúng tôi sử dụng phương pháp *Bag-of-words* để véc-tơ hóa dữ liệu đầu vào, việc lựa chọn các đặc trưng của mô hình này cũng khác một ít so với mô hình OCSVM, các từ "tối", "sáng", "lạnh" … thì không có nhiều ý nghĩa đối với mô hình OCSVM nhưng nó lại giúp mô hình MLP phân loại chính xác hơn ở một số trường hợp. Chính vì thế, véc-tơ đặc trưng của mô hình MLP có số chiều lớn hơn véc-tơ đặc trưng của mô hình OCSVM.

X_train = [[0, 1, 0, ..., 1, 0, 0],[0, 1, 0, ..., 0, 0, 0],[1, 1, 0, ..., 1, 0, 0],...,[0, 0, 0, ..., 0, 1, 0]]

Mô hình MLP là một mô hình máy học có giám sát, tức là dữ liệu đi vào sẽ là một cặp *(feature vector, label).* Ma trận *label*s được tạo ra bằng cách nhân một ma trận *np.one* (ma trận 1) có số chiều bằng số dữ liệu trong từng tập dữ liệu với giá trị tương ứng từ 1 đến 6 được trình bày trong bảng 2:

*np.concatenate((np.ones(len(batden)),2*np.ones(len(tatden)),3*np.ones(len(batquat)),4*np.ones(len(tatquat)),5*np.ones(len(mocua)),6*np.ones(len(dongcua))))*

Với các cơ sở toán học của mô hình đã được trình bày ở mục 3.3. Chúng tôi xây dựng mô hình MLP này với thư viện *Scikit-learning* được hỗ trợ sẵn trong ngôn ngữ Python. Trong quá trình huấn luyện mô hình với ma trận véc-tơ đặc trưng *X_train* và ma trận *labels* tương ứng bằng hàm *mlp.fit(X_train,lables)* thì chúng tôi đã điều chỉnh các thông số để giúp cho mô hình phân loại chính xác các câu lệnh thuộc nhóm lệnh nào cũng như đáp ứng được hiệu suất xử lý của hệ thống.

MLPClassifier(hidden_layer_sizes=20,tol=0.001,activation='logistic',max_iter=100)

Với *hidden_layer_sizes* là đại diện cho số lượng tế bào nơ-ron trong một lớp ẩn (hidden layer), *tol* là dung sai cho việc tối ưu hóa, *activation* là hàm hoạt động của lớp ẩn, *max_iter* là số lần lặp tối đa.

Khi dữ liệu mới đi vào mô hình sẽ được dán nhãn đầu ra theo các nhóm mà chúng được phân loại dựa trên bảng 2. Các nhãn đầu ra này sẽ được gửi xuống bộ điều khiển trung tâm để thực hiện lệnh điều khiển.

Chạy thử tập dữ liệu kiểm thử với mô hình MLP vừa được xây dựng. Chúng tôi lấy tập dữ liệu kiểm thử từ file *test_accept* để tiền xử lý và trích chọn đặc trưng tương tự như tập dữ liệu huấn luyện thì kết quả sẽ là một ma trận X_test chứa các véc-tơ đặc trưng. Sau đó chúng tôi tạo ra một ma trận *label* nhãn "chuẩn" để so sánh với ma trận *y_pred* (là ma trận chứa các nhãn điều khiển thiết bị tương ứng) thu được khi đưa dữ liệu kiểm thử qua mô hình MLP. Kết quả cho thấy mô hình MLP có sự phân loại chính xác gần như 100%.

4.3. Kết quả thực nghiệm

Chuẩn bị

Chúng tôi xây dựng 2 mô hình điều khiển thiết bị bằng giọng nói.

- Mô hình thứ nhất sử dụng phương pháp từ khóa: xây dựng dựa trên nguyên lý của bài báo [2] và các mô hình thực tế. Mô hình này bao gồm một bộ chuyển đổi giọng nói thành văn bản và một bộ điều khiển trung tâm. Bộ điều khiển trung tâm này sẽ nhận dữ liệu từ bộ chuyển đổi giọng nói thành văn bản và so sánh với các "từ khóa" đã được lập trình sẵn để có thể điều khiển thiết bị.

- Mô hình thứ hai sử dụng các thuật toán máy học: mô hình này bao gồm bộ chuyển đổi giọng nói thành văn bản, bộ xử lý trung tâm và bộ điều khiển trung tâm. Chức năng và nhiệm vụ các các bộ xử lý này đã được trình bày ở phần 2.

Thực nghiệm

Chúng tôi thực hiện chạy song song hai mô hình này với đầu vào là bộ chuyển đổi giọng nói thành văn bản (Google Speed to Text) được sử dụng chung cho hai mô hình nhằm tăng tính khách quan trong việc đánh giá kết quả.

Thực hiện lần lượt 50 câu lệnh điều khiển thiết bị đã được chuẩn bị từ trước trong môi trường yên tĩnh (trong phòng kín, không có người xung quanh) và môi trường có tiếng ồn (môi trường làm việc hàng ngày). Sau khi qua bộ chuyển đổi giọng nói thành văn bản thì các văn bản điều khiển này cùng lúc được đưa vào hai mô hình để điều khiển thiết bị. Kết quả thực nghiệm được chúng tôi trình bày tại bảng 4.

Bảng 4. *Kết quả thực nghiệm*

Mô hình	**Môi trường**	**Nhận dạng giọng nói đúng**	**Điều khiển đúng**	**Tỷ lệ (50 mẫu)**
Sử dụng so sánh từ khóa	Yên tĩnh	48	37	74%
Sử dụng các thuật toán máy học			47	94%
Sử dụng so sánh từ khóa	Có tiếng ồn	40	29	58%
Sử dụng các thuật toán máy học			38	76%

Từ kết quả ở bảng 4, đối với cùng một bộ chuyển đổi văn bản thành giọng nói cho ra các dữ liệu đầu vào giống nhau thì với mô hình sử dụng phương pháp so sánh từ khóa sẽ cho ra các kết quả sai ở những câu điều khiển không trùng với từ khóa được lập trình từ trước. Ví dụ: văn bản nhận được là "tắt cái đèn đi" hay "sáng rồi sao không tắt điện đi" trong khi từ khóa là "tắt đèn". Cả hai văn bản nhận được đều có ý nghĩa là "tắt đèn" nhưng không trùng khớp với từ khóa dẫn tới việc hệ thống không điều khiển thiết bị theo ý muốn của người điều khiển. Trong khi đó, với mô hình áp dụng các thuật toán máy học thì cho ra kết quả điều khiển chính xác. Điều này giúp cho việc giao tiếp giữa người điều khiển và thiết bị trở nên bớt nhàm chán khi người sử dụng có thể điều khiển thiết bị bằng những câu nói khác nhau mà không cần quan tâm câu nói đó đã đúng từ khóa chưa.

Ngoài ra, kết quả điều khiển thiết bị bằng giọng nói cũng bị ảnh hưởng rất lớn khi bộ chuyển đổi giọng nói thành văn bản nhận dạng sai nhất là đối với phương pháp sử dụng từ khóa vì chỉ cần nhận diện sai một từ khóa thôi thì kết quả điều khiển thiết bị sẽ không chính xác.

5. KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

Trong nghiên cứu này, chúng tôi xây dựng một hệ thống điều khiển thông minh cho ngôi nhà dựa trên công nghệ nhận dạng giọng nói và máy học. Các mô hình máy học được vận dụng hiệu quả để xử lý các câu lệnh điều khiển, giúp cho hệ thống hoạt động thông minh và linh hoạt hơn so với các giải pháp tự động hóa cứng. Ngoài ra, giải pháp chúng tôi đề xuất cho thấy chính xác cao trong việc điều khiển thiết bị bằng giọng nói cũng như giúp cho con người có thể dễ dàng tiếp cận và sử dụng hệ thống (nhất là đối với những người già, người khuyết tật). Tuy nhiên, nghiên cứu này cũng còn một số hạn chế như bộ dữ liệu huấn luyện chưa đủ lớn nhất là đối với các dữ liệu tiếng Việt vì thế hệ thống vẫn có thể bị "đánh lừa" với các câu điều khiển thiết bị "lạ". Với việc xây dựng thành công bộ xử lý trung tâm, hệ thống có thể kiểm soát các thông số của căn nhà thông qua các kết quả được thu thập từ các cảm biến, camera. Từ đó, hệ thống sẽ đưa ra các cảnh báo, lệnh điều khiển để giúp cho các thiết bị trong nhà hoạt động một cách ổn định. Ngoài ra, chúng tôi có thể xây dựng các ứng dụng khác như ứng dụng trả lời tự động bằng Google text to Speech (tương tự như hệ thống chatbox nhưng trả lời lại bằng giọng nói), nhận diện khuôn mặt khách khi khách ở trước cửa để thông báo cho chủ nhà v.v. Để làm được những ứng

dụng này chúng ta cần có tập dữ liệu đủ lớn cho việc huấn luyện mô hình.

LỜI CẢM ƠN

Công trình này được hỗ trợ bởi Học Viện Công nghệ Bưu chính Viễn thông và Trường Đại học Thủy Lợi.

TÀI LIỆU THAM KHẢO

[1] Felix, C. and Raglend, I.J., *Home automation using GSM*, International Conference on Signal Processing, Communication, Computing and Networking Technology, 21-22 July 2011, Thuckafay, India.

[2] Ramlee, R.A., Othman, M.A., Leong, M.H., Ismail, M.M. and Ranjit, S.S.S., *Smart home system using android application*, International Conference of Information and Communication Technology (ICoICT), 20-22 March 2013, Bandung, Indonesia.

[3] Arriany, A.A. and Musbah, M.S., *Applying voice recognition technology for smart home networks*, International Conference on Engineering & MIS (ICEMIS), 22-24 Sept 2016, Agadir, Marocco.

[4] Kawarazaki, N. and Yoshidome, T., *Remote control system of home electrical appliances using speech recognition*, IEEE International Conference on Automation Science and Engineering (CASE), 20-24 Aug 2012, Seoul, South Korea.

[5] Jain, S., Vaibhav, A. and Goyal, L., *Raspberry Pi based interactive home automation system through E-mail*, International Conference on Reliability Optimization and Information Technology (ICROIT), 6-8 Feb 2014, Faridabad, India.

[6] D. V. Opdenbosch, M. Oelsch, A. Garcea, and E. Steinbach, *A joint compression scheme for local binary feature descriptors and their corresponding bag-of-words representation*, IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 2017.

[7] Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R., *Estimating the support of a high-dimensional distribution*, Neural Computation, 13(7):1443-1472, 2001.

[8] Bounsiar, A. and Madden, M.G., *Kernels for One-class Support Vector Machine*, International Conference on Information Science & Applications (ICISA), 6-9 May 2014.

[9] Gomez-Verdejo, V., Arenas-Garcia, J., Lazaro-Gredilla, M. and Navia-Vazquez, A., *Adaptive One-Class Support Vector Machine*, IEEE Transaction on Signal Processing, pp.2975-2981, June 2011.

[10] Manevitz, L.M. and Yousef, M., *One-class SVMs for Document Classification*, Journal of Machine Learning Research 2.(1):139-154, 2002.

[11] Wilson, E. and Tufts, D.W., *Multilayer perceptron design algorithm*, Proceedings of IEEE Workshop on Neural Networks for Signal Processing, 6-8 Sept 1994, Ermioni, Greece.

[12] Alsmadi, M.K., Omar, K.B., Noah, S.A. and Almarsashdah, I., *Performance Comparison of Multi*-layer *Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks*, IEEE International Advance Computing Conference, 6-7 March 2009, Patiala, India.

[13] S. D. Patil and P. S. Sanjekar, *Musical Instrument Identification Using SVM , MLP& AdaBoost with Formal Concept Analysis*, 1st International Conference on Intelligent Systems and Information Management (ICISIM), 2017, Aurangabad, India.

[14] Haykin, *Neural* Networks*: A Comprehensive Foundation*. New York, 1994.

Tác giả chịu trách nhiệm bài viết:
TS. Nguyễn Tất Bảo Thiện
Trường Đại học Thủy lợi.
Email: baothien@tlu.edu.vn