ƯỚC LƯỢNG NỒNG ĐỘ RƯỢU SỬ DỤNG DEEP TENSOR NEURAL NETWORKS

ĐỒ ÁN KỸ SƯ

Nguyễn Trí Hiếu 1711298

Giảng viên hướng dẫn: TS. Phạm Quang Thái

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ, BỘ MÔN VIỄN THÔNG

07 – 2020

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

TRƯỜNG ĐẠI HỌC BÁCH KHOA **Độc lập – Tự do – Hạnh Phúc**

Số: _____ /BKĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Viễn Thông**

NHIỆM VỤ ĐỒ ÁN MÔN HỌC

1. Họ và tên: Nguyễn Trí Hiếu MSSV: 1711298
2. Ngành: Điện – Điện tử Chuyên ngành: Kỹ thuật Điện tử - Truyền thông
3. Đề tài: Ước lượng nồng độ rượu dùng deep tensor neural networks
4. Nhiệm vụ:
   - Sử dụng phân tích quang phổ của hai loại rượu có sẵn.
   - Xây dựng mô hình deep tensor neural networks, với ngõ vào là đặc trưng của 2 loại rượu, ngõ ra có thể dự đoán được nồng độ của nó và nồng độ của hỗn hợp 2 loại rượu đó.
5. Ngày giao nhiệm vụ luận văn: 10/3/2020
6. Ngày hoàn thành nhiệm vụ: 17/07/2020
7. Họ và tên người hướng dẫn: Phần hướng dẫn

TS. Phạm Quang Thái,
BM Viễn Thông, Khoa Điện – Điện Tử 100%
Nội dung và yêu cầu đồ án đã được thông qua Bộ Môn.

TP.HCM, ngày tháng năm 2020

CHỦ NHIỆM BỘ MÔN **NGƯỜI HƯỚNG DẪN CHÍNH**

Học hàm. Học vị. Họ tên Học hàm. Học vị. Họ tên

PHẦN DÀNH CHO KHOA, BỘ MÔN:
Người duyệt (chấm sơ bộ):
Đơn vị:
Ngày bảo vệ:
Điểm tổng kết:
Nơi lưu trữ đồ án:

1. LỜI CÁM ƠN

Để hoàn thành đồ án này, em xin chân thành gửi lời cảm ơn đến:

Đầu tiên và sâu sắc nhất là thầy TS. Phạm Quang Thái, người đã giúp đỡ em xuyên suốt trong quá trình hoàn thành đồ án. Từ những ngày đầu tiên, khi em gặp khó khăn về điều kiện làm việc, thầy đã sẵn sàng góp ý, thay đổi đề tài giúp em có điều kiện tốt hơn để hoàn thành. Trong lúc thực hiện đồ án, thầy cũng đã tận tình giúp đỡ em, chỉ ra những điểm thiếu sót, gợi ý phương pháp làm bài, nguồn tài liệu, …Đó là những đóng góp quý báu giúp em có thể hoàn thành đồ án.

Kế đến cũng không thể không nhắc đến công lao của quý thầy cô khoa Điện-Điện tử, đại học Bách Khoa thành phố Hồ Chí Minh, những người đã chỉ dạy em trong những năm vừa qua.

Cuối cùng là anh Thái Văn Quang, người đã cung cấp cho em những số liệu để hoàn thành đồ án này.

Xin chúc những điều tốt đẹp nhất sẽ luôn đồng hành cùng quý thầy cô.

TP. HCM, ngày 17, tháng 7 năm 2020

Nguyễn Trí Hiếu

2. LỜI CAM ĐOAN

Tôi tên Nguyễn Trí Hiếu là sinh viên chuyên ngành Kỹ thuật Điện tử - Truyền thông, khóa 2017, tại Đại học Quốc gia thành phố Hồ Chí Minh – Trường Đại học Bách Khoa. Tôi xin cam đoan những nội dung sau đều là sự thật: (i) Công trình nghiên cứu này hoàn toàn do chính tôi thực hiện; (ii) Các tài liệu và trích dẫn trong luận văn này được tham khảo từ các nguồn thực tế, có uy tín và độ chính xác cao; (iii) Các số liệu và kết quả của công trình này được tôi tự thực hiện một cách độc lập và trung thực.

TP. HCM, ngày 17, tháng 07 năm 2020

Nguyễn Trí Hiếu

3. TÓM TẮT ĐỒ ÁN

Trong đồ án này tôi sẽ tiến hành xây dựng mô hình ước lượng nồng độ rượu bằng bước sóng thông qua mô hình deep tensor neural networks. Đây là một phương pháp mới được cải tiến từ mô hình DNN thông thường, nhằm mục tiêu cải thiện độ chính xác, giảm mất mát. Dung dịch sử dụng là hai loại rượu (nếp và Vodka) để làm mẫu. Kết quả đo đạt có sẵn từ luận văn trước đó, mỗi dung dịch sẽ được đo ở 6 nồng độ khác nhau. Ngoài ra còn có thêm 4 mẫu hỗn hợp ở nồng độ bất kỳ. Hai mẫu rượu trên sẽ được dùng để xây dựng mô hình và ước lượng kiểm tra. Mẫu hỗn hợp sẽ được ước lượng nồng độ thông qua hai mô hình trên.

4. ABSTRACT

In this thesis, I built a deep tensor neural network model, which can predict an alcohol concentration. That is a new method, which is an upgraded version of a DNN model to improve accuracy and reduce loss. I use two alcohol samples (glutinous-rice wine and Vodka) for a dataset. The measurement of these samples from an old thesis, each sample comprises six different concentrations. Besides, four mixed samples of two samples above are also used to test my model. Two alcohol samples use to train the model and four mixed samples to predict concentration.

5. DANH SÁCH TỪ VIẾT TẮT:

- DNN: Deep neural networks
- DTNN: Deep tensor neural networks
- ANN: Artificial neural networks
- DB: Double projection
- BP: backpropagation

6. DANH SÁCH HÌNH VẼ:

7. DANH SÁCH BẢNG THÔNG TIN:

8. MỤC LỤC:

Contents

1. GIỚI THIỆU:

1.1 Đặt vấn đề:

Rượu có nhiều loại khác nhau, không những khác về tên, màu sắc, … mà còn khác nhau về độ cồn ethyl và các thành phần có trong rượu. Loài người biết đến rượu cách đây hàng ngàn năm. Ở nước ta ngành nghề làm rượu cũng có từ lâu đời, chủ yếu là do nấu. Từ lâu rượu đã gắn liền với phong tục tập quán, tín ngưỡng tôn giáo của người Việt Nam. Trong các dịp lễ hội, nhất là vào dịp Tết, dù giàu hay nghèo thì người Việt Nam cũng không thể thiếu được rượu.

Trong những năm gần đây thị trường rượu nước ta phát triển rất phong phú đa dạng về số lượng, chủng loại,…Tuy nhiên, với cơ chế thị trường nên trong quá trình sản xuất, một số cơ sở sản xuất đã sử dụng hóa chất, hương liệu, bột màu thực phẩm, chất phụ gia, …quá giới hạn cho phép, hoặc pha chế cồn công nghiệp vào rượu làm cho rượu kém chất lượng. Bên cạnh đó, với công nghệ sản xuất lạc hậu, nguyên liệu đầu vào chưa được kiểm định chặt chẽ, …cũng là nguyên nhân làm cho rượu kém chất lượng. Những loại rượu kém chất lượng thường được bán trà trộn vào những loại rượu chất lượng cao làm cho người tiêu dùng khó phân biệt.

Theo số liệu thống kê thì người dân Việt Nam thuộc nhóm cao nhất thế giới với trung bình 9 lít đồ uống có cồn/1 người/ 1 năm (theo tạm chí y khoa LanCet (Anh) năm 2017). Cùng với sự tiêu thụ lớn như vậy thì vấn đề vệ sinh an toàn thực phẩm trong ngành sản xuất trở thành mối quan tâm hàng đầu của xã hội, vì nó ảnh hưởng trực tiếp đến sức khỏe của người dân. Ngoài ra việc sử dụng rượu bia cũng có ảnh hưởng đến sức khóe thể chất, sức khỏe tâm thần gây ra tai nạn giao thông hay bạo lực gia đình, …

Kết quả của việc xác định nồng độ rượu sẽ giúp chúng ta có cái nhìn tổng quát về chất lượng rượu, lượng rượu đưa vào cơ thể, …

1.2 Phạm vi và phương pháp nghiên cứu:

- ***Phạm vi:*** dựa trên số liệu đo đạc có sẵn từ luận văn trước, để xây dựng hai mô hình có khả năng ước lượng nồng độ rượu của chính loại đó và nồng độ của mẫu hỗn hợp hai loại.
- ***Phương pháp nghiên cứu:*** mô hình được làm hoàn toàn trên máy tính bằng cách áp dụng công nghệ trí tuệ nhân tạo, cụ thể hơn là mô hình deep tensor neural network, với platform là thư viện tensorflow. Mô hình mới này sẽ được so sánh với một mô hình cũ trước đây là deep neural networks DNN. Ngoài ra trong bài còn áp dụng một số phương pháp machine learning, phân tích để tối ưu hóa.

1.3 Các đóng góp của đồ án:

Phát triển một phương pháp mới, ít được sử dụng trước đây để cải thiện độ chính xác của mô hình. Tìm hiểu các machine learning phù hợp cho việc phân tách dữ liệu, chọn lọc đặc trưng.

2. LÝ THUYẾT:

2.1 Phân tích quang phổ:

Phân tích quang phổ là một trong những phương pháp chính đề nghiên cứu về thế giới thực. Một số ứng dụng như:

- Phát hiện quy luật của tự nhiên
- Khám phá được những hiện tượng, tính chất của những vật chất và thực thể

Nhiều kỹ thuật quang phổ khác nhau được sử dụng để khám phá ra những đặc tính đa dạng của vật chất. Những kỹ thuật hiện tại thường được sử dụng là: hấp thụ, phát xạ, scanning tunneling, Raman or electron ‑ paramagnetic resonance, …

Tuy nhiên, các thí nghiệm thường tốn nhiều thời gian và đôi khi đòi hỏi nhiều yêu cầu, tiêu tốn chi phí lên đến hàng triệu Euro, chẳng hạn như synchrotron. Các phương pháp phổ lý thuyết bổ sung dựa trên các nguyên lý đầu tiên của cơ học lượng tử cũng tốn thời gian tương tự và đòi hỏi các cơ sở tính toán hiệu suất cao, với quy mô lớn.

2.2 Ứng dụng của Deep learning vào phân tích quang phổ:

Trong những năm gần đây với sự phát triển của công nghệ trí tuệ nhân tạo, và đặc biệt là công nghệ Machine learning/Deep Learning đã tác động đến rất nhiều lĩnh vực, trong đó có phân tích quang phổ.

Những kỹ thuật machine learning trước đây cho phân tích phổ: kernel ridge regression, support vector machine, reduced ‑ error pruning trees and rotation forests, reduced ‑ error pruning trees and rotation forests, Bayesian optimization.

Tuy nhiên, độ chính xác của những phương pháp trên không cao, chúng ta cần một phương pháp có độ chính xác cao hơn như Deep Learning.

2.3 Giới thiệu Deep Neural Networks (DNN):

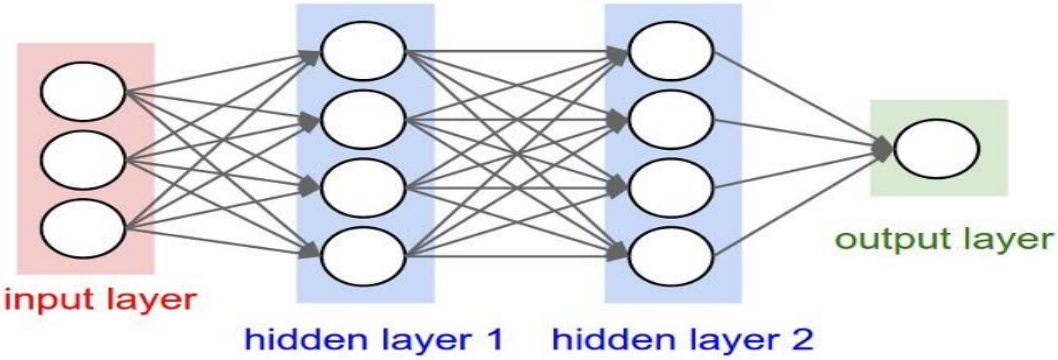
Loài chim truyền cảm hứng cho chúng ta tạo ra máy bay, loài cá truyền cảm hứng cho chúng ta làm ra những con thuyền, thiên nhiên đã truyền cảm hứng cho rất nhiều phát minh khác. Với cùng một logic như vậy, cấu trúc bộ não của con người đã truyền cảm hứng cho việc xây dựng machine learning. Đó là tư tưởng cốt lõi của artificial neural networks (ANNs).

ANNs là cốt lõi của Deep learning. Chúng đa dạng, mạnh mẽ, giúp chúng ta tạo ra những ý tưởng lớn, cực kỳ phức tạp về các ứng dụng Machine learning. Ví dụ như là phân loại hàng tỷ bức ảnh (Google Image), dịch vụ nhân dạng giọng nói đầy mạnh mẽ (Apple’s Siri), giới thiệu những videos tốt nhất cho hàng trăm triệu người dùng (Youtube), …

DNN là một artificial neural network (ANN) với nhiều layers giữa các layers ngõ vào và ngõ ra. DNN là sự vận dụng toán học để chuyển từ ngõ vào sang ngõ ra, có thể là mối quan hệ tuyến tính hoặc không tuyến tính. Network giữa qua các layers để tính toán xác xuất sau mỗi ngõ ra.

Ví dụ DNN train để nhận ra tiếng sủa của giống chó khác nhau. Người dùng có thể xem lại và chọn xác xuất nên hiển thị và trả về nhãn đề xuất. Mỗi sự vận dụng tính toán như là xem xét layer, DNN phức tạp có thể có nhiều layers.

DNN có thể mô hình hóa những mối quan hệ không tuyến tính. Cấu trúc DNN cấu thành những models, nơi mà những đối tượng có thể được xem là nguyên thủy.



Hình 1: Cấu trúc mô hình deep neural networks đơn giản

2.4 Giới thiệu Deep Neural Tensor Network (DTNN):

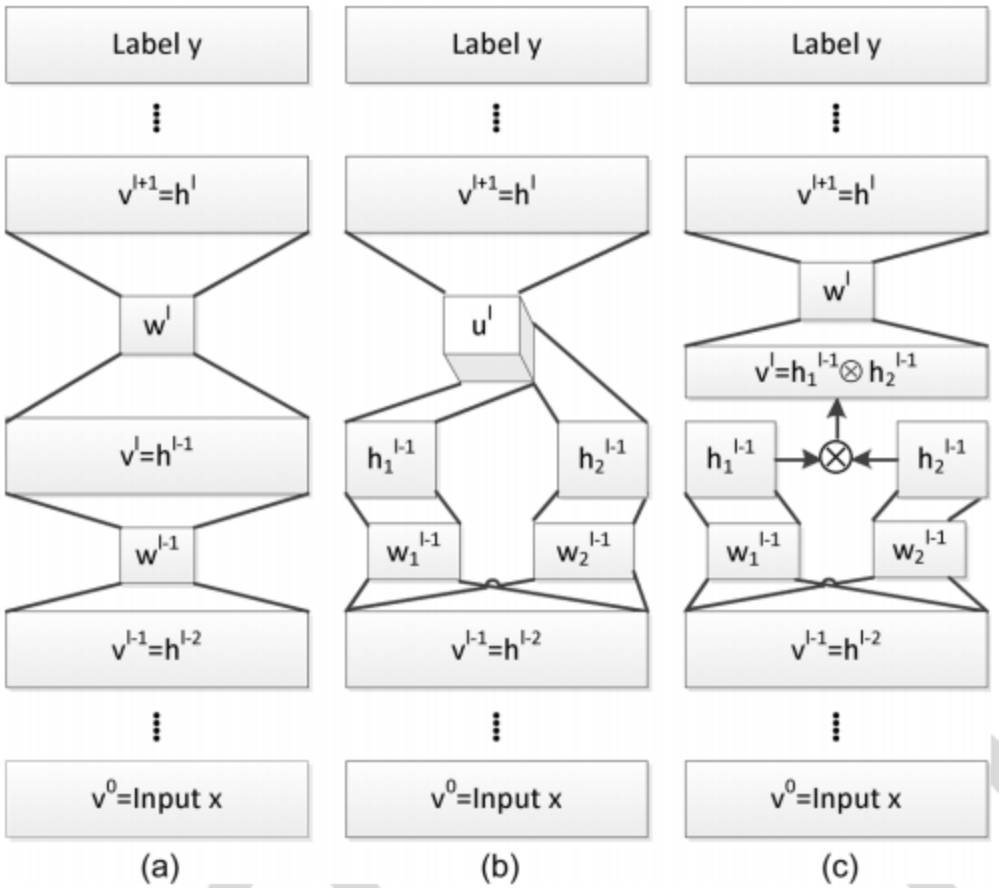
2.4.1 Giới thiệu:

Trong nghiên cứu này, chúng ta sẽ phát triển sâu hơn về một Deep Neural Network (DNN) mới gọi là Deep Tensor Neural Network (DTNN)

DTNN tăng độ lợi bằng cách lặp lại một hay nhiều layers với double-projection (DP) layer, mỗi vector ngõ vào được chiếu thành hai không gian con không tuyến tính và tensor layer. Hai không gian con đó tác động qua lại lẫn nhau và cùng nhau dự đoán được layer tiếp theo trong deep architecture. Thêm vào đó chúng ta còn mô tả phương pháp lập ra tensor layers thành sigmoid layers thông thường. Vì vậy những cái trước đó có thể được treated và trained bằng cách giống nhau sau cùng.

Với cách sắp xếp này ta có thể xem DTNN như là DNN gia tăng thêm DP layers, vì vậy nó không chỉ là thuật toán BP learning của DTNNs có thể thu nhận được mà còn là DTNN mới có thể dễ dàng phát triển. Ước lượng trên Switchboard chỉ ra rằng DTNN có thể có độ chính xác cao hơn DNN khoảng 4-5%.

2.4.2 Kiến trúc DTNN cơ bản:



Hình 2: Kiến trúc DNN thông thường và mối quan hệ với DTNN. a) DNN. b) hidden layer h^{l-1} bao gồm hai phần: h_1^{l-1} và h_2^{l-1}. Hidden layer h^1 là tensor layer kết nối với weights u^l từ 3-way tensor. c) Cách biểu diễn thay thế của *b)*: tensor u^l được thay thế bởi ma trận w^l khi v^l được định nghĩa bởi tích cross product của h_1^{l-1} và h_2^{l-1}.

- Quy ước của DNN: ngõ vào là *x*, với *I* x *1* vector, output là *y* với *C* x *1* vector. Với l là layer index. Mỗi hidden layer h^{l-1} kết nối với layer tiếp theo h^1 với weight matrix w^l và bias a^l

$$h_{(k)}^{l} = \sigma\left(\sum_i w_{(i,k)}^{l} h_{(i)}^{l-1} + a_{(k)}^{l}\right)$$

  Trong đó: *i, k* là những indexes trong hidden units của h^{l-1} và h^l
- DTNN với hidden layer được chia ra thành h_1^{l-1} (K_1^{l-1} x *1* vector) và h_2^{l-1} (K_2^{l-1} x *1* vector). Những phần đó kết nối với h^1 (K^l x*1*vector) thông qua 3-way tensor u^l có chiều là (K_1^{l-1} x K_2^{l-1} x K^l)

$$h^l_{(k)} = \sigma\left(\sum_{i,j} u^l_{(i,j,k)}\, h^{l-1}_{1(i)} h^{l-1}_{2(j)} + a^l_{(k)}\right)$$

- Chúng ta gọi hidden layer h^{l-1} là double projection (DP) với thông tin nhận được từ layer trước đó là h^{l-2} chiếu lên hai không gian con là h_1^{l-1} và h_2^{l-1}

$$h^{l-1} = \begin{bmatrix} h_1^{l-1} \\ h_2^{l-1} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_1^{l-1} \\ w_2^{l-1} \end{bmatrix} h^{l-2} + \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \end{bmatrix}\right)$$

- Hidden layer h^l gọi là tensor layer. DP layer trước đó là h^{l-1} liên kết với h^l thông qua weight tensor là u^l

Ở hình 2c, là một cách nhìn khác về DTNN, giống như hình b, bằng cách định nghĩa

$$v^l = h_1^{l-1} \otimes h_2^{l-1} = vec(h_1^{l-1}(h_2^{l-1})^T)$$

Trong đó, $\otimes$ là tích *cross product,* và *vec(.)* là column vectorize representation của ma trận.

Chúng ta có thể tổ chức là viết lại tensor u^l như là w^l giống như hình chữ nhật trong hình 1

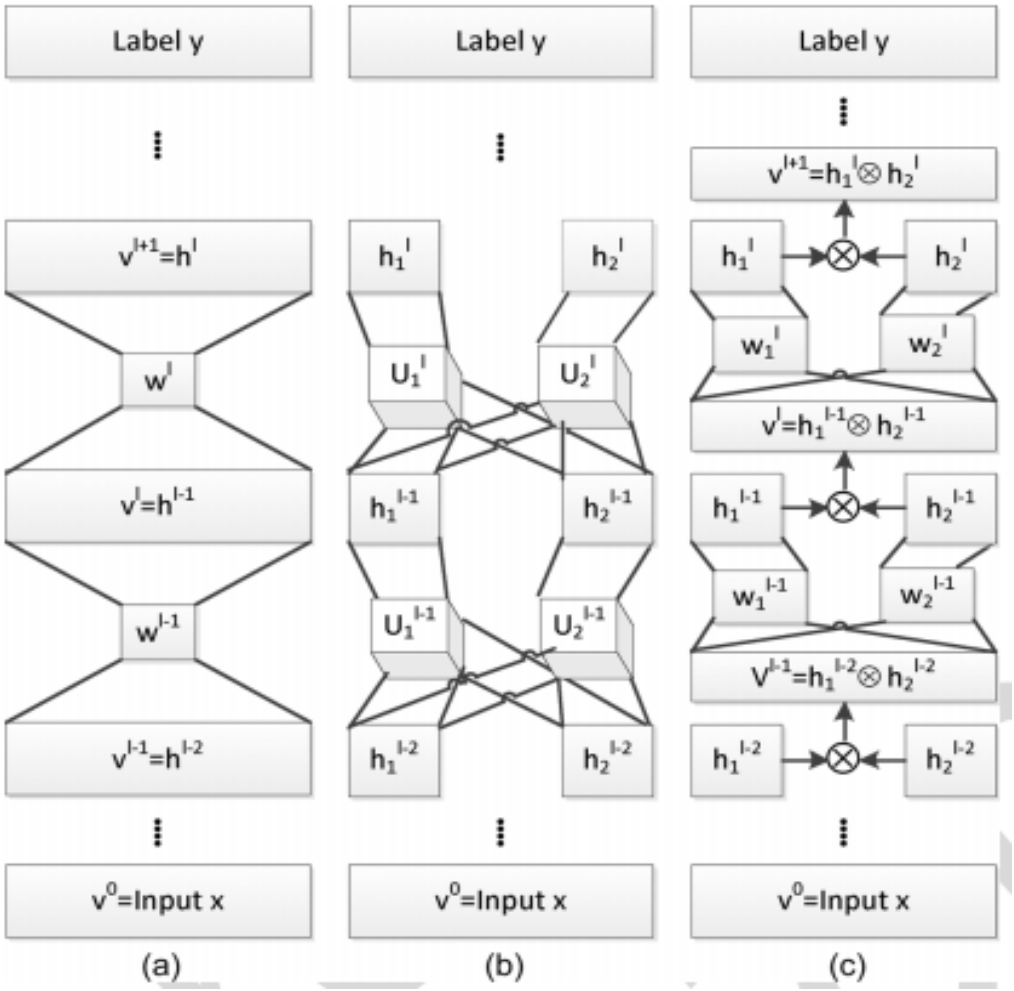
$$h^l{}_{(k)} = \sigma(\sum_i w^l{}_{(i,k)} v^l{}_{(i)} + a^l{}_{(k)}).$$

Bằng cách biết lại trên cho phép ta giảm và đổi tensor layer thành matrix layers thông thường bằng cách định nghĩa cùng loại giao diện để mô tả hai loại layers.

Ví dụ: trong hình *2c* ta có thể thấy hidden layer h^l có thể được xem xét như là một layer thông thường như trong hình *1a,* và có thể học được bằng cách sử dụng thuật toán backpropagation (BP) như bình thường.

Cách viết trên còn có thể chỉ ra rằng tensor layer có thể được xem như là layer thông thường mà ngõ vào gồm có cross product của những giá trị đi qua layer trước đó.

2.4.3 Kiến trúc DTNN mở rộng



Hình 3: So sánh DNN thông thường và 2 cách nhìn tương đương của DTNN trong đó tất cả hidden layers là DP tensor layers. a) DNN. b) DTNN: cách nhìn bình thường nơi mà hidden layers được kết nối với 3-way tensor tensors. c) DTNN: cách nhìn thay thế trong đó tensors được thay thế bằng ma trận khi v^l được định nghĩa là tích Cross product của h_1^{l-1} và h_2^{l-1}.

- Hidden layer là h^{l-1}, tuy nhiên DP layer vẫn còn chứa 2 phần là h_1^{l-1} và h_2^{l-1}, mà lần lượt phải xác định 2 weight là w_1^{l-1} và w_2^{l-1} bằng cách như nhau như trong hình 2b và 2c.
- DTNN ở hình 1 chỉ chứa một DP layer. Tuy nhiên không gì có thể ngăn layer khác trở thành DP layer.
- Hình 3b là một ví dụ về DTNN mà ở đó tất cả các hidden layers là DP layers. Ví dụ hidden layer h^{l-2} chia ra 2 thành phần h_1^{l-2} và h_2^{l-2} được kết nối với h_1^{l-1} và h_2^{l-1} trong qua tensor u_1^{l-1} và u_2^{l-1}.
- Trong DTNN mỗi DP layer chiếu thành 2 vùng subspace không tuyến tính là h_1^l và h_2^l. Quan hệ song tuyến tính của hai hình chiếu này sau đó được tổng hợp như là input feature của layer cao hơn liền kề.

- Để xác định input v^{l-1} từ hidden layer h^{l-1} như là 2 tensor u_1^{l-1} và u_2^{l-1} được viết lại dưới dạng ma trận w_1^{l-1} và w_2^{l-1} như hình 3c.

Tổng kết: *chúng ta có thể biểu diễn DTNNs bằng hai loại hidden layers: sigmoid layer thông thường và DP layer. Mỗi hidden layer có thể sắp xếp linh hoạt vị trí trong DTNN.*

Đối với nhiệm vụ classification, softmax layer kết nối hidden layer cuối cùng để dán nhãn có thể được sử dụng trong DTNN giống như cách có trong quy ước của DNN.

$$v^{l-1} = h_1^{l-2} \otimes h_2^{l-2} = vec(h_1^{l-2}(h_2^{l-2})^T)$$

TABLE I
FORWARD COMPUTATIONS IN DTNNS

Condition	**Input v^l**
first layer	$v^l = v^0 = x$
h^{l-1}: normal layer	$v^l = h^{l-1}$
h^{l-1}: DP layer	$v^l = vec(h_1^{l-1} \otimes h_2^{l-1})$
Condition	**Output h^l**
normal layer	$h^l = \sigma(z^l(v^l)) = \sigma((w^l)^T v^l + a^l)$
DP layer, $i \in \{1,2\}$	$h_i^l = \sigma\left(z_i^l(v^l)\right) = \sigma\left((w_i^l)^T v^l + a_i^l\right)$
softmax layer	$p(y\|v^L) = \frac{exp\left((w_y^L)^T v^L + a_y^L\right)}{\sum_{y'} exp\left((w_{y'}^L)^T v^L + a_{y'}^L\right)}$

Hình 4: tổng kết lại tất cả các forward computation

2. 5 Phương pháp tìm những đặc trưng quan trọng (feature importance):

2.5.1 Tổng quan về feature importance:

Feature importance kỹ thuật tìm score của dữ liệu ngõ vào dựa trên việc chúng hữu dụng như thế nào trong việc dự đoán biến mục tiêu.

Có rất nhiều loại cho feature importance, phổ biến nhất có thể kể đến như là thống kê correlation score, tính toán coefficient như là một phần của mô hình tuyến tính decision tree, permutation importance score, …

Feature importance có thể được sử dụng cho những bài toán như regression hay classification.

2.5.2 Logistic Regression:

a) Giới thiệu:

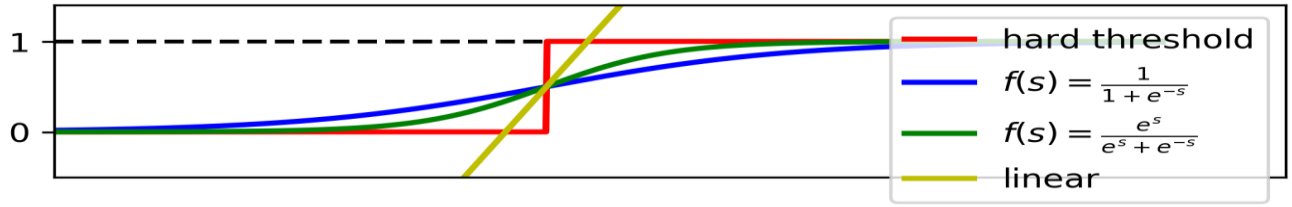
Logistic regression đã được ứng dụng trong sinh học từ đầu thế kỷ 20, sau đó được ứng dụng trong rất nhiều trong các lĩnh vực khoa học xã hội khác. Ví dụ như dự đoán xem email là spam (1) hay (0), khối u là ác tính (1) hay (0).

Trong thống kê, mô hình logistic được sử dụng trong việc mô hình hóa xác suất của một lớp chắc chắn hay sự kiện tồn tại như là đậu/rớt, thắng/thua, sống/chết, …Nó có thể được mở rộng thành mô hình nhiều lớp như là bức ảnh có chứa những con chó, mèo, sư tử hay không. Mỗi đối tượng sẽ có xác suất giữa 0 và 1.

Đầu ra của dự đoán logistic regression thường được viết dưới dạng:

$$f(x) = \theta(w^T x)$$

Trong đó: θ là logistic function. Một số activation cho mô hình tuyến tính cho bởi hình bên dưới:



Hàm activation cần phải thỏa mãn các điều kiện sau:

- Là hàm số liên tục nhận giá trị thực bị chặn trong khoảng (0, 1).
- Nếu coi điểm có tung độ là 1/2 làm điểm phân chia thì các điểm càng xa điểm này về phía bên trái có giá trị càng gần 0. Ngược lại, các điểm càng xa điểm này về phía phải có giá trị càng gần 1.
- Nên có đạo hàm tại mọi nơi, thuận lợi cho việc tối ưu

Một số hàm thỏa mãn 3 điều kiện trên thường được sử dụng như sau:

- Hàm sigmoid: $f(x) = \frac{1}{1+e^{-s}} \triangleq \sigma(s)$
- Hàm Tanh: $\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$

b) Coefficient:

Sau khi fit xong model, chúng ta sẽ muốn kiểm tra những đóng góp của những thành phần trong việc dự đoán. Để làm điều đó chúng ta có thể sử dụng regression coefficient.

Regression coefficient mô tả kích thước và mối quan hệ giữa predictor và đáp ứng biến. Sử dụng coefficient có thể xác định được sự thay đổi ở biến dự đoán có làm cho sự kiện thay đổi nhiều hay ít không.

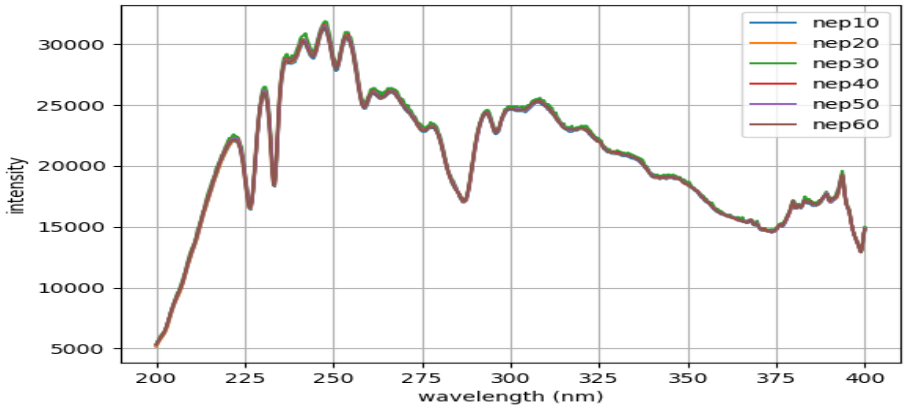
Trong logistic regression có nhiều phương pháp để kiểm định lại mức độ quan trọng của những thành phần dự đoán. Hai phương pháp thường được sử dụng là Wald test và likelihood ratio test.

3. PHÂN TÍCH VÀ KẾT QUẢ:

3.1 Tiền xử lý dữ liệu:

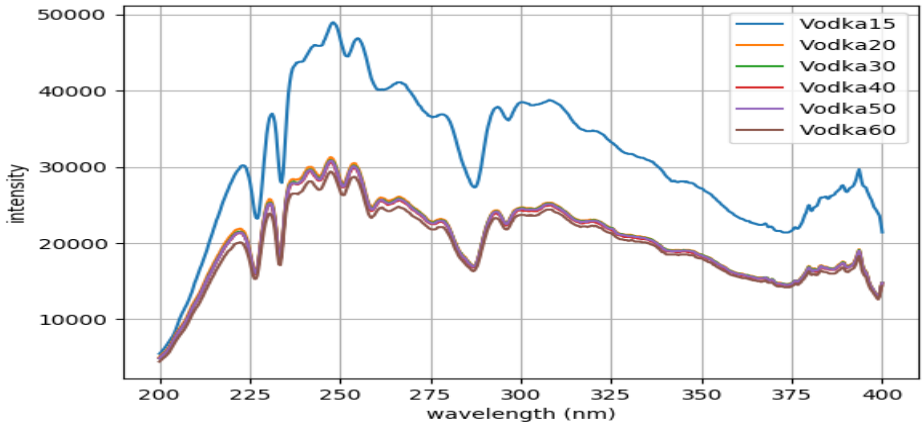
a) Dữ liệu đầu vào:

- Bao gồm 2 loại rượu, mỗi loại được pha trộn với nước cất để tạo ra 6 mẫu mới ở các nồng độ khác nhau
- Ngoài ra, còn có thêm 4 mẫu rượu hỗn hợp được pha trộn từ 2 mẫu trên.
- Kết quả đo được thể hiện ở 2 hình bên dưới:



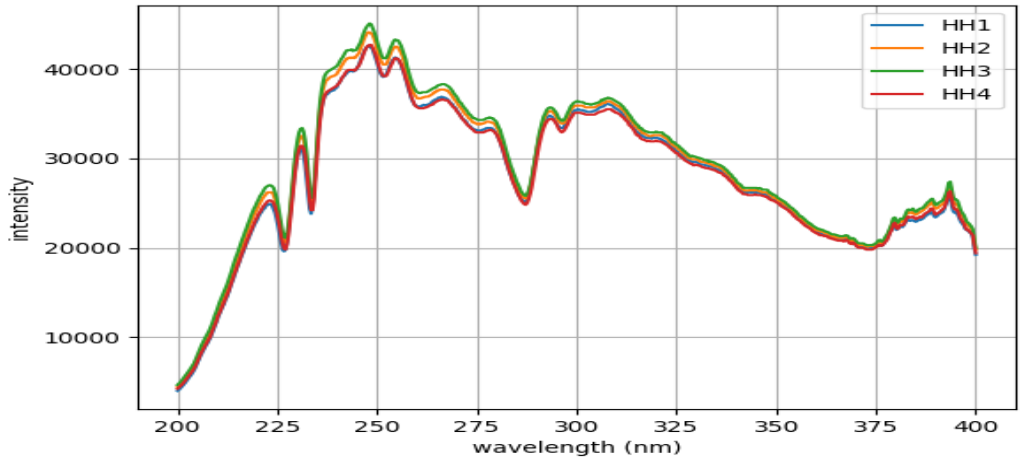
Hình 5: Cường độ rượu nếp

Cường độ đo được tại 6 nồng độ khác nhau là 10/60, 20/60, 30/60, 40/60, 50/60 và 60/60.



Hình 7: Cường độ Vodka

Cường độ đo được tại 6 nồng độ khác nhau là 15/60, 20/60, 30/60, 40/60, 50/60 và 60/60.



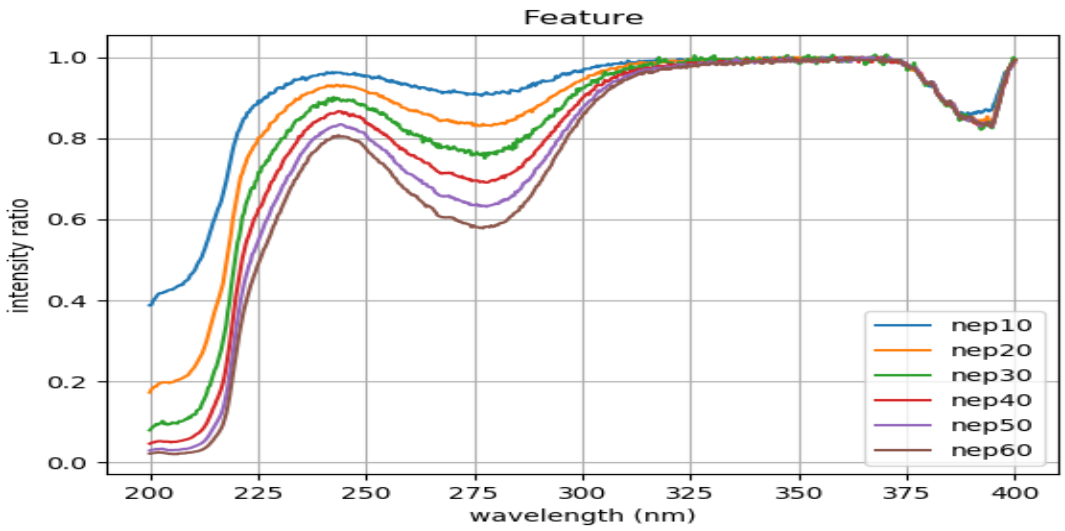
Hình 8: Cường độ mẫu hỗn hợp

Cường độ đo được tại 4 mẫu hỗn hợp với nồng độ và thể tích tương ứng là:

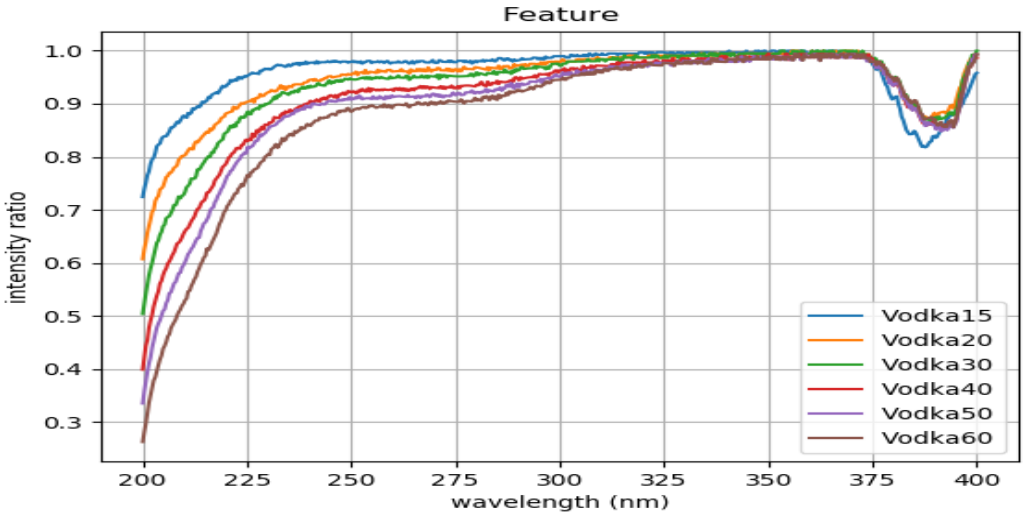
- HH1 = 21ml vodka 50/60 (35/60) + 9ml nếp 60/60 (18/60)
- HH2 = 25ml vodka 30/60 (25/60) + 5ml nếp 60/60 (10/60)
- HH3 = 27ml vodka 50/60 + 3ml nước cất (45/60)
- HH4 = 26ml nếp 60/60 (52/60) + 4ml nước cất

b) Trích xuất đặc trưng:

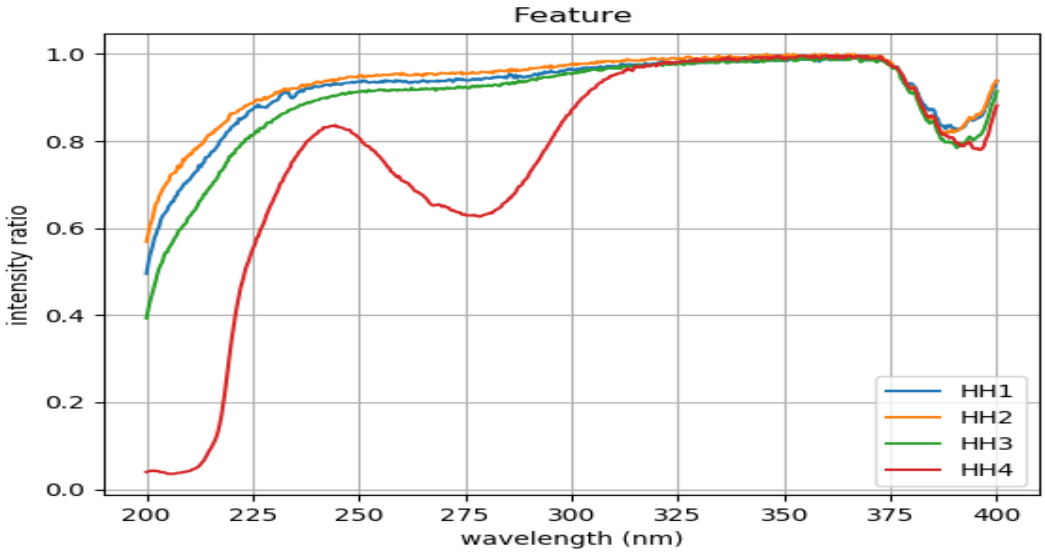
- Theo như hình trên, cường độ của các chất (ngoại trừ của Vodka ở nồng độ 15/60) là rất gần nhau, mắt thường rất khó có thể phân biệt được. Vì vậy, cần phải tiền xử lý dữ liệu trích xuất ra feature của các mẫu.
- Các đường đặc trưng được lấy bằng cách giới hạn bước sóng trong khoảng từ 200 đến 400 nm. Sau đó, chia cho cường độ mẫu nước cất. Ta có các đường đặc trưng như hình bên dưới:



Hình 8: Đường đặc trưng của rượu nếp



Hình 9: Đường đặc trưng của Vodka

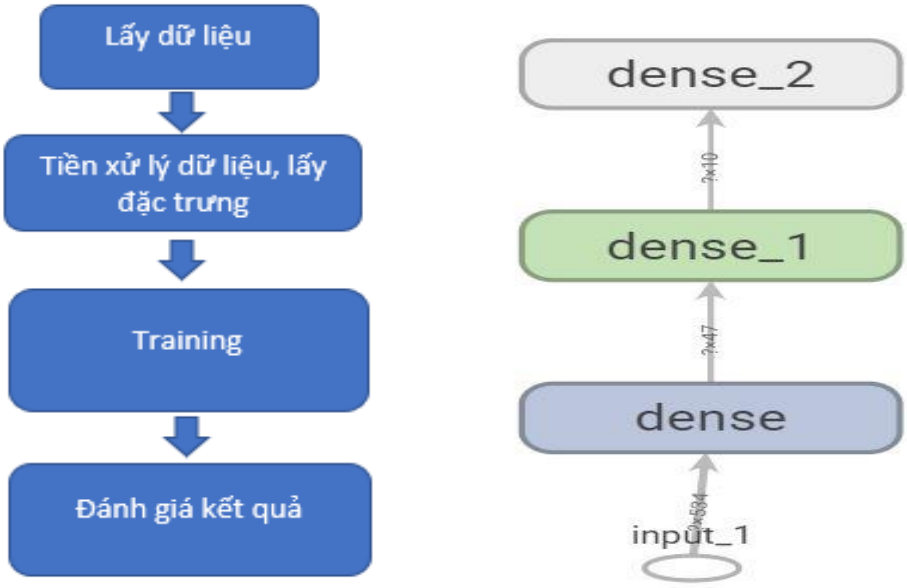


Hình 10: Đường đặc trưng của các hỗn hợp

3.2 Xây dựng mô hình:

a) Deep Neural Networks:

Xây dựng mô hình DNN cơ bản gồm 1 input layer, 2 hidden layers và 1 output layer như hình bên dưới:

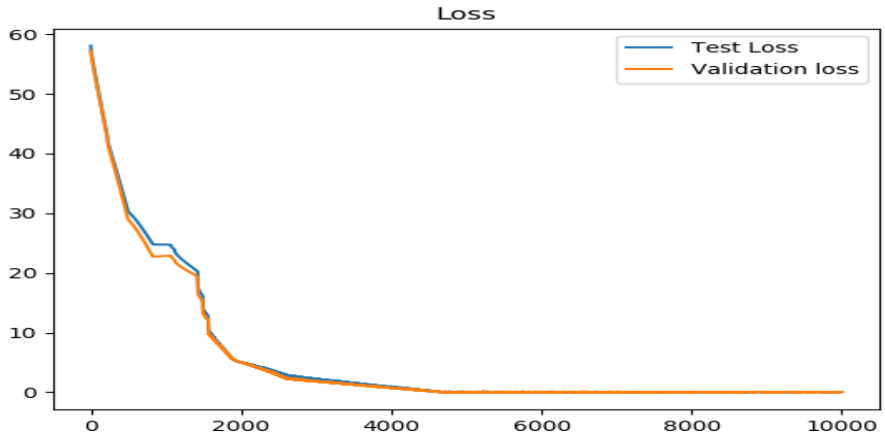


Hình 11: Lưu đồ giải thuật và mô hình DNN dùng cho việc ước lượng nồng độ rượu

Trong đó:

- Tất cả layers đều sử dụng hàm sigmoid activation.
- Số epochs là 10000
- Tối ưu hóa Adam với learning rate là 0.0008.

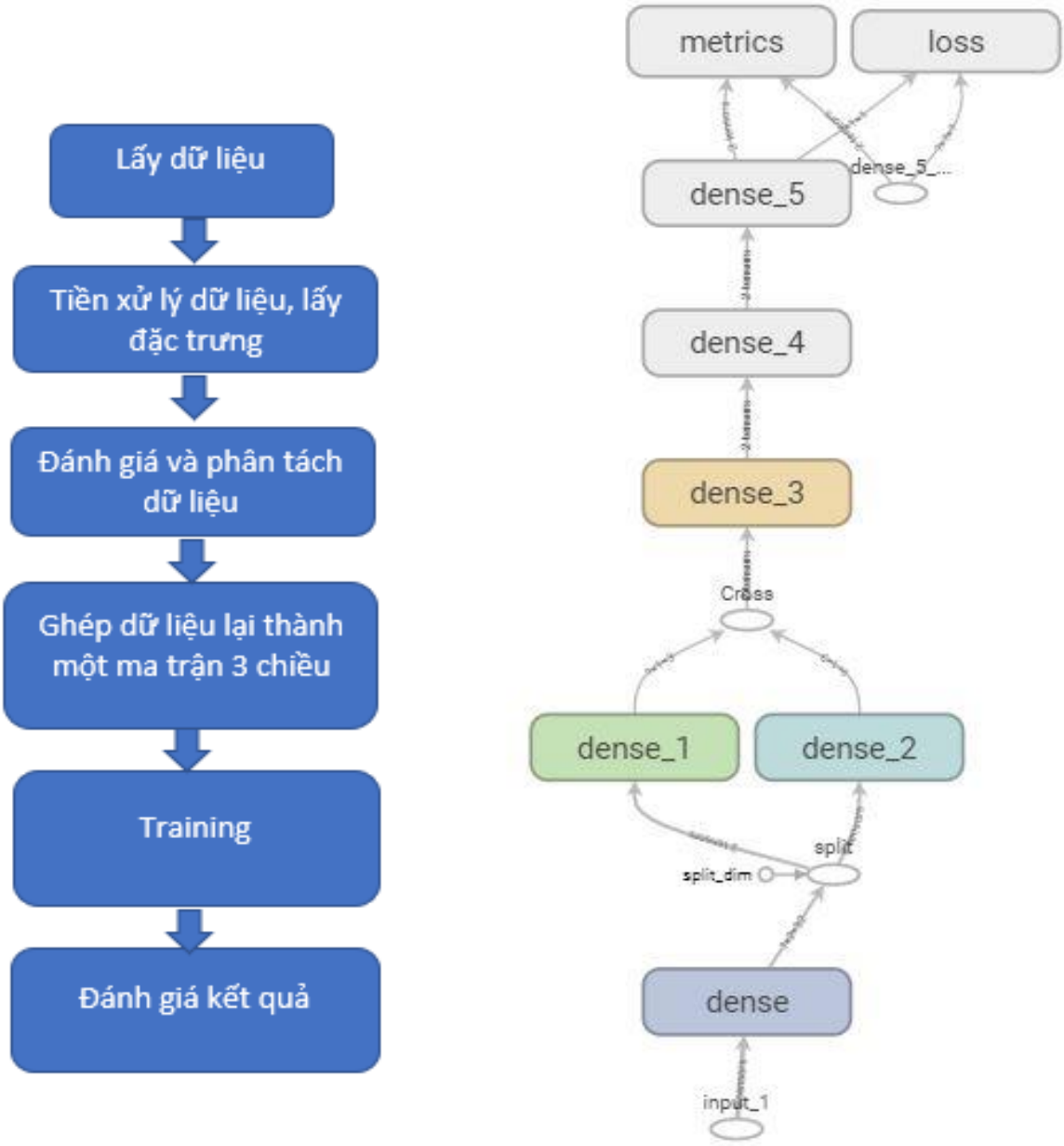
Kết quả training được biểu diễn qua hình bên dưới:



Hình 12: Đường cong training của DNN

b) Deep Tensor Neural Network:

Xây dựng mô hình DTNN cơ bản gồm 1 input layer, 3 hidden layers thông thường, 1 double projection layer và 1 output layer như hình bên dưới:



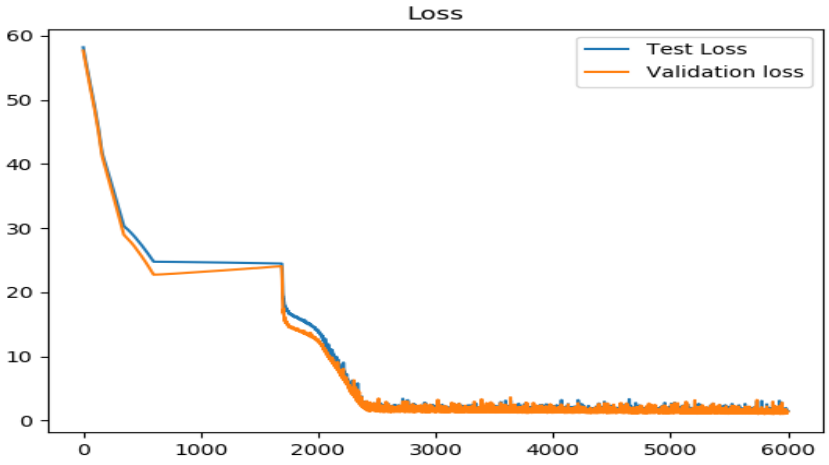
Hình 13: Lưu đồ giải thuật và mô hình DTNN sử dụng cho việc ước lượng nồng độ rượu

- Dữ liệu ban đầu được tách thành 2 features.
- Sau khi đi qua hidden layer thứ nhất, thì được tách ra làm 2 phần để training riêng biệt.
- Sau đó được ghép lại với nhau bằng tích cross, và tiếp tục training chung.

- Kết quả thu được sẽ lấy từ dự đoán của feature tốt. Chúng ta không thể chỉ lấy feature tốt để dự đoán là để tạo ra sự tương tác giữa 2 thành phần bước sóng trong quá trình training chung

Thông số như sau:

- Tất cả hidden layers đầu sử dụng hàm activation sigmoid, hidden layer cuối cùng sử dụng hàm activation Relu.
- Số epochs là 6000
- Tối ưu hóa Adam.
  Kết quả training được biểu diễn qua hình bên dưới:



Hình 14: Đường cong training của DTNN

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 2, 267)]          0
_________________________________________________________________________________________________
dense (Dense)                (None, 2, 32)             8576        input_1[0][0]
_________________________________________________________________________________________________
tf_op_layer_split (TensorFlowOp [(None, 1, 32), (Non 0           dense[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 1, 3)              99          tf_op_layer_split[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)              (None, 1, 3)              99          tf_op_layer_split[0][1]
_________________________________________________________________________________________________
tf_op_layer_Cross (TensorFlowOp [(None, 1, 3)]         0           dense_1[0][0]
                                                                   dense_2[0][0]
_________________________________________________________________________________________________
dense_3 (Dense)              (None, 1, 13)             52          tf_op_layer_Cross[0][0]
_________________________________________________________________________________________________
dense_4 (Dense)              (None, 1, 10)             140         dense_3[0][0]
_________________________________________________________________________________________________
dense_5 (Dense)              (None, 1, 1)              11          dense_4[0][0]
=================================================================================================
Total params: 8,977
Trainable params: 8,977
Non-trainable params: 0
```

Hình 15: Những thông số của mô hình DTNN

Áp dụng vào bài toán ước lượng nồng độ rượu:

Theo như lý thuyết DTNN, thì chúng ta cần tối thiểu 2 features để phân chia ra thành hai tập dữ liệu con. Nhưng trong bài toán này, chỉ có một feature là mối liên hệ của bước sóng và tỷ lệ cường độ. Nên chúng ta sẽ chia dữ liệu này ra làm 2 phần:

- Phần 1: bao gồm những bước sóng tốt
- Phần 2: những bước sóng còn lại.

Dữ liệu đầu vào gồm 80 mẫu rượu đơn chất, 2 chiều và 534 điểm bước sóng (267 cho phần 1 và 267 cho phần 2).

Kích thước: Input shape (80, 2, 267)

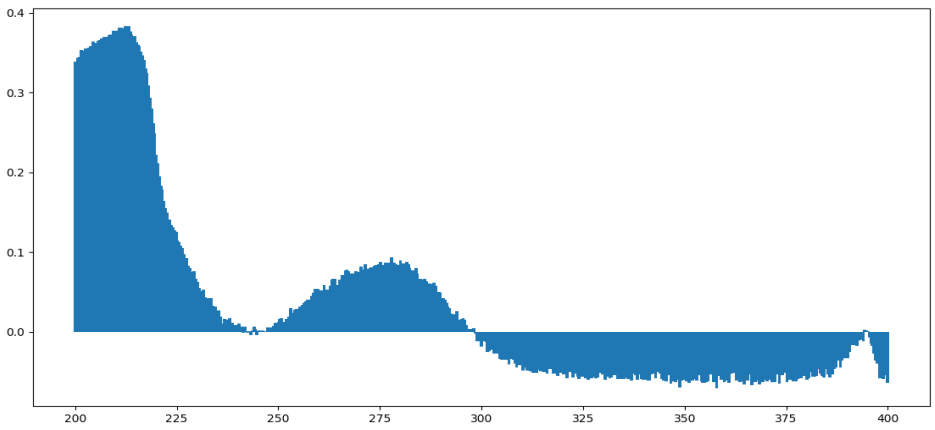
Hàm Loss: Mean absolute error

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i - \boldsymbol{x}|$$

Tìm feature importance sử dụng Logistic Regression:

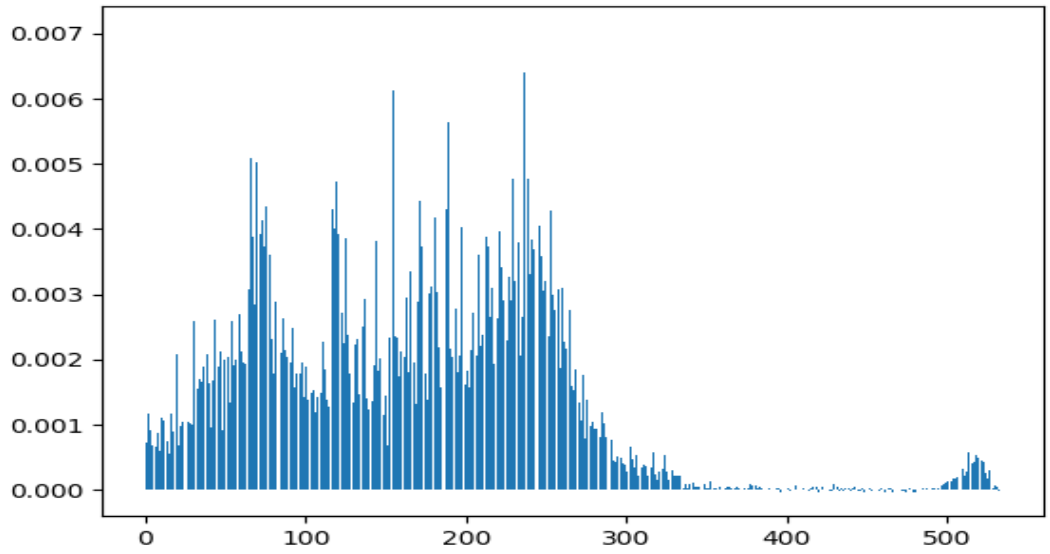
Trong bài toán này, logistic regression được chọn vì cho kết quả score tại tất cả các điểm bước sóng. Nó là bài toán phân loại với 2 lớp 0 và 1, giá trị dương là lớp 1, âm là lớp 0. Điều này phù hợp với yêu cầu bài toán là phân loại thành hai phần bước sóng.

Sau khi chạy thuật toán, chúng ta thu được đồ thị biểu diễn score tại các bước sóng như sau:



Hình 16: Logistic regression coefficient của các bước sóng

Kết quả sau khi kiểm định lại bằng Wald test. Giá trị z sẽ là tỷ số của coefficient và standard error.



Hình 17: Kết quả sau khi kiểm định lại Logistic regression coefficient dùng Wald test

Vì vậy, chúng ta có thể chia vùng bước sóng này thành 2 phần. Phần có bước sóng thấp hơn là phần 1, còn lại là phần 2.

Kết quả kiểm tra bằng DNN ở mỗi phần cho thấy:

- Phần 1 có thể ước lượng được nồng độ, tuy nhiên kết quả không tốt bằng dùng toàn bộ tập dữ liệu
- Phần 2: không thể dự đoán

Từ đó, rút ra kết luận là không thể chỉ dùng phần 1 để train, cần có thêm sự tương tác từ phần 2.

Kết quả dự đoán bao gồm 2 thành phần: kết quả do phần 1 dự đoán và kết quả do phần 2 dự đoán.

3.3 Kết quả:

3.3.1 Kết quả bằng cách sử dụng mô hình DNN

Vì các kết quả dự đoán các mẫu rất giống nhau nên chỉ lấy đại diện một kết quả cho một mẫu.

a) Đối với mẫu rượu nếp:

- Dự đoán nồng độ rượu nếp:

	Nồng độ thực (%)	Nồng độ dự đoán (%)	Sai số (%)	Mean absolute error
Mẫu 1	16.67	16.6709	0.005399	0.0009
Mẫu 2	33.33	33.3239	0.018305	0.0061

Mẫu 3	50	49.9748	0.050425	0.0252
Mẫu 4	66.67	66.5959	0.111268	0.0741
Mẫu 5	83.33	83.1733	0.188402	0.1567
Mẫu 6	100	99.9889	0.011101	0.0111

Bảng 1: Mẫu rượu nếp được dự đoán bằng mô hình nếp DNN

- Dự đoán nồng độ mẫu hỗn hợp:

	Nồng độ thực (%)	Nồng độ dự đoán (%)	Sai số (%)	Mean absolute error
Mẫu 1	30	16.6709	79.954	13.329
Mẫu 2	16.67	16.6708	0.0048	0.0008
Mẫu 4	86.67	82.1417	5.5128	4.5283

Bảng 2: Mẫu hợp chất được dự đoán bằng mô hình nếp DNN

b) Đối với mẫu Vodka:

- Dự đoán nồng độ rượu Vodka:

	Nồng độ thực (%)	Nồng độ dự đoán (%)	Sai số (%)	Mean absolute error
Mẫu 1	25	25.0871	0.3472	0.0871
Mẫu 2	33.33	33.5119	0.5428	0.1819
Mẫu 3	50	49.4077	1.1988	0.5923
Mẫu 4	66.67	66.0343	0.9627	0.6357
Mẫu 5	83.33	83.2924	0.0451	0.0376
Mẫu 6	100	99.9911	0.0089	0.0089

Bảng 3: Mẫu Vodka được dự đoán bằng mô hình Vodka DNN

- Dự đoán nồng độ mẫu hỗn hợp:

	Nồng độ thực (%)	Nồng độ dự đoán (%)	Sai số (%)	Mean absolute error
Mẫu 1	58.33	50.4818	15.547	7.8482
Mẫu 2	41.67	38.6808	7.7279	2.9892
Mẫu 3	75	67.1728	11.652	7.8272

Bảng 4: Mẫu hợp chất được dự đoán bằng mô hình Vodka DNN

Nhận xét: Nhìn chung DNN cho kết quả tốt mẫu đơn chất, đa số sai số đều dưới 1%. Tuy nhiên đối với mẫu hỗn hợp thì kết quả chưa tốt, sai số cao. Đặc biệt là mẫu hỗn hợp 1 khi dùng mô hình nếp thì không thể ước lượng được.

3.3.2 Kết quả bằng cách sử dụng mô hình DTNN

a) Đối với mẫu rượu nếp:

- Dự đoán nồng độ rượu nếp:

		Nồng độ thực (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
File 1	Phần 1	16.67	16.5253	0.8756	0.1447
		16.67	16.5121	0.9563	0.1579
	Phần 2	16.67	16.6322	0.2273	0.0378
		16.67	16.6617	0.0498	0.0083
File 2	Phần 1	33.33	33.1299	0.604	0.2001
		33.33	33.1286	0.6079	0.2014
	Phần 2	33.33	31.8912	4.5116	1.4388
		33.33	31.7401	5.0091	1.5899
File 3	Phần 1	50	49.7958	0.4101	0.2042
		50	49.7490	0.5045	0.251
	Phần 2	50	49.2102	1.605	0.7898
		50	48.6979	2.6738	1.3021
File 4	Phần 1	66.67	66.4276	0.3649	0.2424
		66.67	66.4192	0.3776	0.2508
	Phần 2	66.67	65.8631	1.2251	0.8069
		66.67	65.8888	1.1856	0.7812
File 5	Phần 1	83.33	83.0591	0.3262	0.2709
		83.33	83.1025	0.2738	0.2275
	Phần 2	83.33	83.3738	0.0525	0.0438
		83.33	83.2691	0.0731	0.0609
File 6	Phần 1	100	99.7391	0.2616	0.2609
		100	99.7408	0.2599	0.2592
	Phần 2	100	98.7445	1.2715	1.2555
		100	98.3914	1.6349	1.6086

Bảng 5: Mẫu rượu nếp được dự đoán bằng mô hình nếp DTNN

- Dự đoán nồng độ mẫu hỗn hợp:

		Nồng độ thực (%)	Nồng độ ước lượng	Sai số (%)	Mean absolute error
HH 1	Phần 1	30	28.3167	5.9445	1.6833
		30	28.9489	3.6309	1.0511
	Phần 2	30	19.8734	50.956	10.127
		30	19.9852	50.111	10.015
HH 2	Phần 1	16.67	137.7179	87.896	121.05
		16.67	137.8847	87.91	121.21
	Phần 2	16.67	16.3627	1.8781	0.3073
		16.67	16.4338	1.4373	0.2362
HH 4	Phần 1	86.67	79.7092	8.7327	6.9608
		86.67	79.7304	8.7038	6.9396
	Phần 2	86.67	83.6657	3.5908	3.0043
		86.67	83.6147	3.654	3.0553

Bảng 6: Mẫu hợp chất được dự đoán bằng mô hình nếp DTNN

b) Đối với mẫu Vodka:

- Dự đoán nồng độ rượu Vodka:

		Nồng độ thực (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
File 1	Phần 1	25	25.4218	1.6592	0.4218
		25	25.3348	1.3215	0.3348
	Phần 2	25	25.0651	0.2597	0.0651
		25	25.1052	0.419	0.1052
File 2	Phần 1	33.33	33.0545	0.8335	0.2755
		33.33	33.1323	0.5967	0.1977
	Phần 2	33.33	36.6885	9.1541	3.3585
		33.33	37.3602	10.787	4.0302
File 3	Phần 1	50	49.4915	1.0274	0.5085
		50	49.5733	0.8607	0.4267
	Phần 2	50	50.6802	1.3421	0.6802

		50	51.3352	2.6009	1.3352
File 4	Phần 1	66.67	66.1643	0.7643	0.5057
		66.67	66.0997	0.8628	0.5703
	Phần 2	66.67	70.6895	5.6861	4.0195
		66.67	71.1249	6.2635	4.4549
File 5	Phần 1	83.33	82.5874	0.8992	0.7426
		83.33	82.4929	1.0148	0.8371
	Phần 2	83.33	83.6734	0.4104	0.3434
		83.33	85.6145	2.6684	2.2845
File 6	Phần 1	100	99.6128	0.3887	0.3872
		100	99.6875	0.3135	0.3125
	Phần 2	100	103.7815	3.6437	3.7815
		100	103.8460	3.7036	3.846

Bảng 7: Mẫu Vodka được dự đoán bằng mô hình Vodka DTNN

- Dự đoán nồng độ mẫu hỗn hợp:

		Nồng độ thực (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
HH 1	Phần 1	58.33	51.3965	13.49	6.9335
		58.33	51.3729	13.542	6.9571
	Phần 2	58.33	57.7837	0.9454	0.5463
		58.33	57.8597	0.8128	0.4703
HH 2	Phần 1	41.67	40.5482	2.7666	1.1218
		41.67	40.6162	2.5945	1.0538
	Phần 2	41.67	39.8888	4.4654	1.7812
		41.67	39.9465	4.3145	1.7235
HH 3	Phần 1	75	72.2892	3.7499	2.7108
		75	72.2981	3.7372	2.7019
	Phần 2	75	74.5701	0.5765	0.4299

		75	74.6661	0.4472	0.3339

Bảng 8: Mẫu hợp chất được dự đoán bằng mô hình Vodka DTNN

Nhận xét:

- DTNN cho kết quả dự đoán mẫu đơn chất tốt, mẫu hỗn hợp khá tốt. Khắc phục được nhược điểm của DNN là sai số nhiều ở mẫu hỗn hợp.
- Kết qua từ 2 phần khác nhau khá rõ rệt.
- Khi ước lượng nồng độ mẫu hỗn hợp, thì cả hai phần bước sóng tốt và phần bước sóng còn lại đều cho kết quả tốt hơn ước lượng dùng DNN (ngoại trừ mẫu hỗn hợp 2, khi được ước lượng nồng độ dùng mô hình train bằng rượu nếp)
- Từ đây suy ra rằng, ta chỉ nên chọn lấy kết từ một trong hai phần.

4. TỔNG KẾT

4.1 Tóm tắt và kết luận:

Tóm tắt:

- Trong đồ án này, tôi đã xây dựng một phương pháp ước lượng nồng độ dựa trên phương pháp DTNN. Đồng thời so sánh với phương pháp cơ bản là DNN để thấy được sự hiệu quả.
- Tôi cũng đã xử lý dữ liệu gốc cho phù hợp với DTNN.

Kết quả thu được sau khi chỉ lấy phần bước sóng tốt cho việc dự đoán:

Mô hình nếp:

Mẫu đơn chất:

		Nồng độ thực (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
File 1	Phần 2	16.67	16.6322	0.2273	0.0378
		16.67	16.6617	0.0498	0.0083
File 2	Phần 1	33.33	33.1299	0.604	0.2001
		33.33	33.1286	0.6079	0.2014
File 3	Phần 1	50	49.7958	0.4101	0.2042
		50	49.7490	0.5045	0.251
File 4	Phần 1	66.67	66.4276	0.3649	0.2424
		66.67	66.4192	0.3776	0.2508
File 5	Phần 2	83.33	83.3738	0.0525	0.0438
		83.33	83.2691	0.0731	0.0609

File 6	Phần 1	100	99.7391	0.2616	0.2609
		100	99.7408	0.2599	0.2592

Bảng 9: Tổng hợp kết quả ước lượng nồng độ rượu nếp bằng mô hình nếp DTNN

Mẫu hợp chất:

		Nồng độ thưc (%)	Nồng độ ước lượng	Sai số (%)	Mean absolute error
HH 1	Phần 1	30	28.3167	5.9445	1.6833
		30	28.9489	3.6309	1.0511
HH 2	Phần 2	16.67	16.3627	1.8781	0.3073
		16.67	16.4338	1.4373	0.2362
HH 4	Phần 2	86.67	83.6657	3.5908	3.0043
		86.67	83.6147	3.654	3.0553

Bảng 10: Tổng hợp kết quả ước lượng nồng độ rượu hỗn hợp bằng mô hình nếp DTNN

Mô hình nếp:

Mẫu đơn chất:

		Nồng độ thưc (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
File 1	Phần 2	25	25.0651	0.2597	0.0651
		25	25.1052	0.419	0.1052
File 2	Phần 1	33.33	33.0545	0.8335	0.2755
		33.33	33.1323	0.5967	0.1977
File 3	Phần 1	50	49.4915	1.0274	0.5085
		50	49.5733	0.8607	0.4267
File 4	Phần 1	66.67	66.1643	0.7643	0.5057
		66.67	66.0997	0.8628	0.5703
File 5	Phần 1	83.33	82.5874	0.8992	0.7426
		83.33	82.4929	1.0148	0.8371
File 6	Phần 1	100	99.6128	0.3887	0.3872
		100	99.6875	0.3135	0.3125

Bảng 11: Tổng hợp kết quả ước lượng nồng độ Vodka bằng mô hình Vodka DTNN

Mẫu hợp chất:

		Nồng độ thực (%)	Nồng độ ước lượng (%)	Sai số (%)	Mean absolute error
HH 1	Phần 2	58.33	57.7837	0.9454	0.5463
		58.33	57.8597	0.8128	0.4703
HH 2	Phần 1	41.67	40.5482	2.7666	1.1218
		41.67	40.6162	2.5945	1.0538
HH 3	Phần 2	75	74.5701	0.5765	0.4299
		75	74.6661	0.4472	0.3339

Bảng 12: Tổng hợp kết quả ước lượng nồng độ rượu hỗn hợp bằng mô hình Vodka DTNN

Kết luận:

- Mô hình DTNN cho khả năng dự đoán mẫu đơn chất tương đương với DNN
- Nhưng đối với mẫu hợp chất, phức tạp và nhiều nhiễu, thì DTNN lại cho kết quả tốt hơn rõ rệt.
- Tốt hơn mô hình DNN khoảng 4-5 %, đúng với lý thuyết đã nêu đầu bài.
- Ngoài ra DTNN còn cải thiện được tốc độ training so với DNN.

Những điểm đạt và hạn chế:

Điểm đạt:

- Xây dựng thành công mô hình DTNN cho độ chính xác khá cao.
- Phát triển cách tách dữ liệu quan trọng phù hợp với mô hình DTNN.

Hạn chế:

- Số mẫu rượu đơn chất và đa chất khá ít.
- Vì là sử dụng mẫu đo có sẵn nên không kiểm soát được chất lượng mẫu đo.

4.2 Hướng phát triển của đề tài

- Có thể xây dựng đề tài với nhiều mẫu đơn chất hơn, áp dụng cho nhiều loại chất khác nhau, nhiều hỗn hợp khác nhau.
- Tăng thêm số lượng DP layer, số chiều dữ liệu, …
- Áp dụng mô hình DTNN đã phát triển ở trên cho những loại bài toán khác, phức tạp hơn, dữ liệu nhiều chiều hơn như: nhận dạng khuôn mặt, giọng nói, xử lý ngôn ngữ tự nhiên, …

9. TÀI LIỆU THAM KHẢO:

1. The Deep Tensor Neural Network with Applications to Large Vocabulary Speech Recognition (Dong Yu, Senior Member, IEEE, Li Deng, Fellow, IEEE, and Frank Seide, Member, IEEE)
2. Sách Hands-on Machine learning with Scikit-Learn & TensorFlow (Aurélien Géron)
3. Sách Machine Learning cơ bản (Vũ Hữu Tiệp)
4. Luận văn kỹ sư: ước lượng nồng độ rượu pha tạp sử dụng phân tích quang phổ và thuật toán lấy mẫu ngẫu nhiên (thực hiện: Thái Văn Quang, hướng dẫn: TS. Phạm Quang Thái)
5. Large Vocabulary Speech Recognition Using Deep Tensor Neural Networks (Dong Yu, Li Deng, Frank Seide. Microsoft Research, Redmond, WA, USA and Microsoft Research Asia, Beijing, China)
6. Computer-implemented deep tensor neural network (Dong Yu, Bothell, WA (US): Li Deng, Redmond, WA (US); Frank Seide, Beijing (CN))
7. Deep Learning Spectroscopy: Neural Networks for Molecular Excitation Spectra (Kunal Ghosh, Annika Stuke, Milica Todorović, Peter Bjørn Jørgensen, Mikkel N. Schmidt, Aki Vehtari, and Patrick Rinke)
8. CS 20: Tensorflow for Deep Learning Research (Stanford online course)
9. Wikipedia: Deep learning
10. Wikipedia: Logistic Regression
11. How to calculate feature importance with Python (Jason Brownlee, PhD)
12. Tensorflow tutorial (https://www.tensorflow.org)

10. PHỤ LỤC:

Phần mềm, ngôn ngữ lập trình và thư viện hỗ trợ:

Ngôn ngữ lập trình: Python (version 3)

Phần mềm và công cụ hỗ trợ:

- Visual studio code
- Google Colab
- Tensorboard

Thư viện hỗ trợ:

- Tensorflow (version 2.0)
- Scikit learn (0.23.1)
- Matplotlib (3.1.1)
- Pandas (0.24.2)
- Numpy (1.18.2)

- Statsmodels

Code:

Link Github: https://github.com/hieunguyen810/Deep-tensor-neural-networks

Link Google Drive (cần tài khoản hcmut.edu.vn, bao gồm database): https://drive.google.com/drive/folders/1hdYPGWIv3pZ9mG4dd5t5xXcAHTG6SCX?usp=sharing

Một số file tiêu biểu trong bài

Get_data.py

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
def get_data(filename, label):
    data = pd.read_csv(filename)
    a = np.arange(54, 321)
    b = np.arange(321, 588)
    X1 = data.iloc[5:85, a].values
    Z = data.iloc[200, a].values
    X2 = data.iloc[5:85, b].values
    K = data.iloc[200, b].values
    y = np.full((1, 80), label)
    X_1 = Z/X1
    X_2 = K/X2
    X = np.vstack([X_1, X_2])
    X = np.reshape(X, [80, 2, 267])
    return X, y
def get_mix_data(filename):
    a = np.arange(54, 321)
    b = np.arange(321, 588)
    data = pd.read_csv(filename)
    X1 = data.iloc[11:21, a].values
    Z = data.iloc[300, a].values
    X2 = data.iloc[11:21, b].values
    K = data.iloc[300, b].values
    X_1 = Z/X1
    X_2 = K/X2
```

```
    X = np.vstack([X_1, X_2])
    X = np.reshape(X, [10, 2, 267])
    return X
def get_data_Vodka (filename, label):
    data = pd.read_csv(filename)
    a, b = feature_selection.feature_importance()
    X_1 = data.iloc[1:81,a].values
    Z = data.iloc[260,a].values
    X_2 = data.iloc[1:81, b].values
    K = data.iloc[260, b].values
    y = np.full((1, 80), label)
    X_1 = Z/X_1
    X_2 = K/X_2
    X = np.vstack([X_1, X_2])
    X = np.reshape(X, [80, 2, 267])
    return X, y
```

Vodka.py

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import Get_data
import Show_result
import Train_tf2
def main (argv = None):
    X_1, y_1=Get_data.get_data ('Dataset/Vodka15.csv', 25)
    X_2, y_2=Get_data.get_data_Vodka ('Dataset/Vodka20.csv', 33.33)
    X_3, y_3=Get_data.get_data_Vodka ('Dataset/Vodka30.csv', 50)
    X_4, y_4=Get_data.get_data_Vodka ('Dataset/Vodka40.csv', 66.67)
    X_5, y_5=Get_data.get_data_Vodka ('Dataset/Vodka50.csv', 83.33)
    X_6, y_6=Get_data.get_data_Vodka ('Dataset/Vodka60.csv', 100)
    X = np.vstack([X_1, X_2, X_3, X_4, X_5, X_6])
    y = np.append(y_1, y_2)
    y = np.append(y, y_3)
    y = np.append(y, y_4)
    y = np.append(y, y_5)
```

```
    y = np.append(y, y_6)
    X = np.asarray(X).astype(float)
    y = np.asarray(y).astype(float)
    test, label = Get_data.get_data_Vodka('Dataset/Vodka60.csv', 1)
    test = np.asarray(test).astype(float)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=42)
    loss, val_loss = Train_tf2.train(X_train, X_test, y_train, y_test, X_val, y_val, test)
    num_epochs = 6000
    Show_result.show_result_tf2(loss, val_loss, num_epochs)
if __name__ == '__main__':
    tf.compat.v1.app.run ()
```

Nep.py

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import Get_data
import Show_result
import Train_tf2
def main (argv = None):
    X_1, y_1=Get_data.get_data ('Dataset/nep10.csv', 16.67)
    X_2, y_2=Get_data.get_data ('Dataset/nep20.csv', 33.33)
    X_3, y_3=Get_data.get_data ('Dataset/nep30.csv', 50)
    X_4, y_4=Get_data.get_data ('Dataset/nep40.csv', 66.67)
    X_5, y_5=Get_data.get_data ('Dataset/nep50.csv', 83.33)
    X_6, y_6=Get_data.get_data ('Dataset/nep60.csv', 100)
    X = np.vstack([X_1, X_2, X_3, X_4, X_5, X_6])
    y = np.append(y_1, y_2)
    y = np.append(y, y_3)
    y = np.append(y, y_4)
    y = np.append(y, y_5)
    y = np.append(y, y_6)
    X = np.asarray(X).astype(float)
    y = np.asarray(y).astype(float)
```

```
    test, label = Get_data.get_data('Dataset/nep10.csv', 1)

    test = np.asarray(test).astype(float)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=42)

    loss, val_loss = Train_tf2.train(X_train, X_test, y_train, y_test, X_val, y_val, test)

    num_epochs = 6000

    Show_result.show_result_tf2(loss, val_loss, num_epochs)

if __name__ == '__main__':

    tf.compat.v1.app.run ()
```

Train_tf2.py

```
import tensorflow as tf

import os.path

from os import path

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from datetime import datetime

def train(X_train, X_test, y_train, y_test, X_val, y_val, test):

    num_epochs = 6000

    inputs = tf.keras.Input(shape = (2, 267))

    L1 = tf.keras.layers.Dense(32, activation = 'sigmoid')(inputs)

    split0, split1 = tf.split(L1, num_or_size_splits=2, axis = 1)

    h1 = tf.keras.layers.Dense(3, activation='sigmoid')(split0)

    h2 = tf.keras.layers.Dense(3, activation='sigmoid')(split1)

    v = tf.linalg.cross(h1, h2)

    L2 = tf.keras.layers.Dense(13, activation = 'sigmoid')(v)

    L3 = tf.keras.layers.Dense(10, activation = 'relu')(L2)

    outputs = tf.keras.layers.Dense(1)(L3)

    model = tf.keras.Model(inputs = inputs, outputs = outputs)

    print(model.summary())

    callback = tf.keras.callbacks.TensorBoard(log_dir=".\logs")

    model.compile(loss = "mean_absolute_error", optimizer = "Adam", metrics = ['mae',
'mse'])

    if path.exists("vodka.h5"):

        h = model.load_weights("vodka.h5")

    else:
```

```
        h = model.fit(X_train, y_train, validation_data =(X_val, y_val), batch_size=32,
epochs=num_epochs, verbose=1, callbacks = [callback])

        model.save("vodka.h5")

    eval = model.evaluate(X_test, y_test, verbose = 1)

    print("\nEvaluation on test data: \nloss = %0.4f " % (eval[0]))

    pre = model.predict(test)

    print(pre)

    return h.history['loss'], h.history['val_loss']
```

Show_result.py

```
import matplotlib.pyplot as plt

def show_result_tf2(loss, val_loss, num_epochs):

    epochs_range = range(num_epochs)

    plt.plot(epochs_range, loss, label='Test Loss')

    plt.plot(epochs_range, val_loss, label='Validation loss')

    plt.legend(loc='upper right')

    plt.title('Loss')

    plt.show()
```