TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 2

Tìm Hiểu Mạng Nơ-Ron Và Ứng Dụng Cho Bài Toán Phân Tích Quan Điểm

Người hướng dẫn: **PGS. TS. Lê Anh Cường**

Người thực hiện: **Huỳnh Trung Lực – 51403182**

Lâm Phúc Nghi – 51403239

Lớp **: 14050302**

Khoá **: 18**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2017

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 2

Tìm Hiểu Mạng Nơ-Ron Và Ứng Dụng Cho Bài Toán Phân Tích Quan Điểm

Người hướng dẫn: **PGS. TS. Lê Anh Cường**

Người thực hiện:**Huỳnh Trung Lực – 51403182**

Lâm Phúc Nghi – 51403239

Lớp **: 14050302**

Khoá **: 18**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2017

LỜI CẢM ƠN

Qua nghiên cứu và thực nghiệm, nhóm em đã hoàn thành được đồ án phân tích luận điểm này.

Chúng em xin cảm ơn PGS.TS Lê Anh Cường đã hướng dẫn nhiệt tình những kiến thức cơ bản và hướng thực hiện đồ án này giúp chúng em có được nền tảng vững chắc để hoàn thành đề tài, thầy đã tận tình giải đáp những thắc mắc và hướng dẫn cách phát triển về đề tài của đồ án này trong việc nghiên cứu.

Nhóm em cũng xin chân thành cảm ơn anh Đ.B.Linh đã giúp đỡ, giải đáp thắc mắc và chỉ dẫn các phương pháp lập trình cho nhóm em. Đồng thời cũng cảm ơn các bạn trong phòng nghiên cứu đã hỗ trợ chúng em làm đồ án này.

Do kiến thức còn hạn hẹp, nên phần báo cáo của chúng em có thể còn nhiều sai sót. Chúng em mong thầy/cô thông cảm và góp ý, nhận xét để nhóm chúng em có thể tiếp tục khắc phục được những sai sót.

Nhóm em xin chân thành cảm ơn.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của PGS.TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Huỳnh Trung Lực

Lâm Phúc Nghi

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Từ khi ra đời, máy tính đã nhanh chóng phát triển và đóng một vai trò rất quan trọng trong nghiên cứu khoa học kỹ thuật cũng như trong đời sống. Nhưng một máy tính dù có mạnh đến đâu chăng nữa, cũng chỉ có thể làm việc theo một chương trình đã được hoạch định sẵn bởi con người. Nó vẫn không có khả năng liên tưởng, kết nối sự việc này với sự việc khác, và quan trọng hơn hết là khả năng sáng tạo như con người.

Vì lý do đó, mạng nơ-ron (Artificial neural networks) ra đời từ mục đích cố gắng mô phỏng hoạt động trí tuệ của con người. Mạng nơ-ron nhân tạo là một phương pháp mô phỏng xử lý thông tin, được nghiên cứu ra từ hệ thống thần kinh của sinh vật, giống như bộ não để xử lý thông tin. Nó bao gồm số lượng lớn các mối gắn kết cấp cao để xử lý các yếu tố làm việc trong mối liên hệ giải quyết vấn đề rõ ràng.

Giống như con người, được học bởi kinh nghiệm, lưu những kinh nghiệm hiểu biết và sử dụng trong những tình huống phù hợp. Từ khi ra đời, mạng nơ-ron đã nhanh chóng phát triển trong các lĩnh vực về nhận dạng, phân loại, giảm nhiễu, dự đoán…

Trong phạm vi đồ án này, chúng em chỉ xin trình bày về phương pháp "Tìm hiểu mạng nơ-ron và ứng dụng cho bài toán phân tích quan điểm ".

Nội dung đồ án gồm 3 chương :

- Chương I : Tìm hiểu chung về mạng nơ-ron
- Chương II : Thực Nghiệm trên mạng nơ-ron lan truyền ngược và phân tích, so sánh kết quả thực nghiệm
- Chương III : Tìm hiểu và sử dụng mạng nơ-ron tích chập để cải thiện khả năng phân loại của mạng nơ-ron lan truyền ngược.

Mục Lục

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

CHƯƠNG I. MẠNG NƠ-RON

Mạng nơ-ron trong một vài năm trở lại đây đã được quan tâm và đã áp dụng thành công trong nhiều lĩnh vực khác nhau, như tài chính, y tế, địa chất và vật lý. Mạng nơ-ron có thể được ứng dụng trong bất cứ các vấn đề về dự báo, phân loại và điều khiển. Ví dụ như khả năng nhận dạng mặt người trong các hệ thống quản lý thông tin liên quan đến con người (quản lý nhân sự ở các công sở, doanh nghiệp; quản lý học sinh, sinh viên trong các trường trung học, đại học và cao đẳng;…); các ngành khoa học hình sự, tội phạm; khoa học tướng số, tử vi,…

Mạng nơ-ron có khả năng mô phỏng bất kì hàm số hay quan hệ nào, là mô hình duy nhất có thể làm điều này với một số lượng tham số vừa phải mà máy tính có khả năng tính toán được. Kết hợp chặt chẽ với logic mờ, mạng nơ-ron nhân tạo đã tạo nên cuộc cách mạng thực sự trong việc thông minh hóa và vạn năng hóa các bộ điều khiển kỹ thuật cao cho cả hiện nay và trong tương lai. Ví dụ như ứng dụng tự động điều khiển hệ thống lái tàu, hệ thống dự báo sự cố,…

Dựa trên việc mô phỏng cấp thấp hệ thống nơ-ron sinh học. Trong tương lai với sự phát triển mô phỏng nơ-ron sinh học, chúng ta có thể có loại máy tính thông minh thật sự.

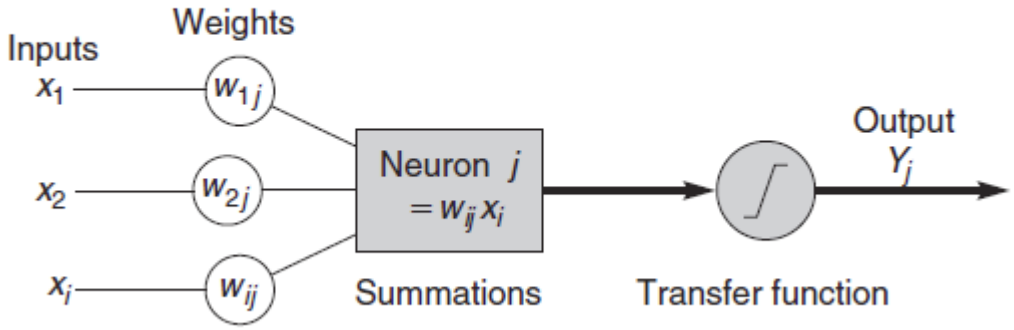
1.1 Giới Thiệu Về Mạng Nơ-Ron

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.

Ra đời xuất phát từ ý tưởng mô phỏng hoạt động của bộ não con người, là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người với vô số các nơ-ron được liên kết truyền thông với nhau qua mạng. Giống như con người, mạng nơ-ron nhân tạo được học bởi kinh nghiệm, lưu những kinh nghiệm đó và sử dụng trong những tình huống phù hợp.

Mạng nơ-ron 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. Sau khi 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, mạng nơ-ron sẽ cho mang lại kết quả cho vấn đề cần giải quyết trong ứng dụng đó. Về bản chất quá trình học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơ-ron.

- Nơ-ron nhân tạo là một đơn vị tính toán có nhiều đầu vào và một đầu ra.



Hình 1.1.1 : Cấu tạo một Nơ-Ron đơn giản

Các thành phần cơ bản của một nơ-ron nhân tạo bao gồm:

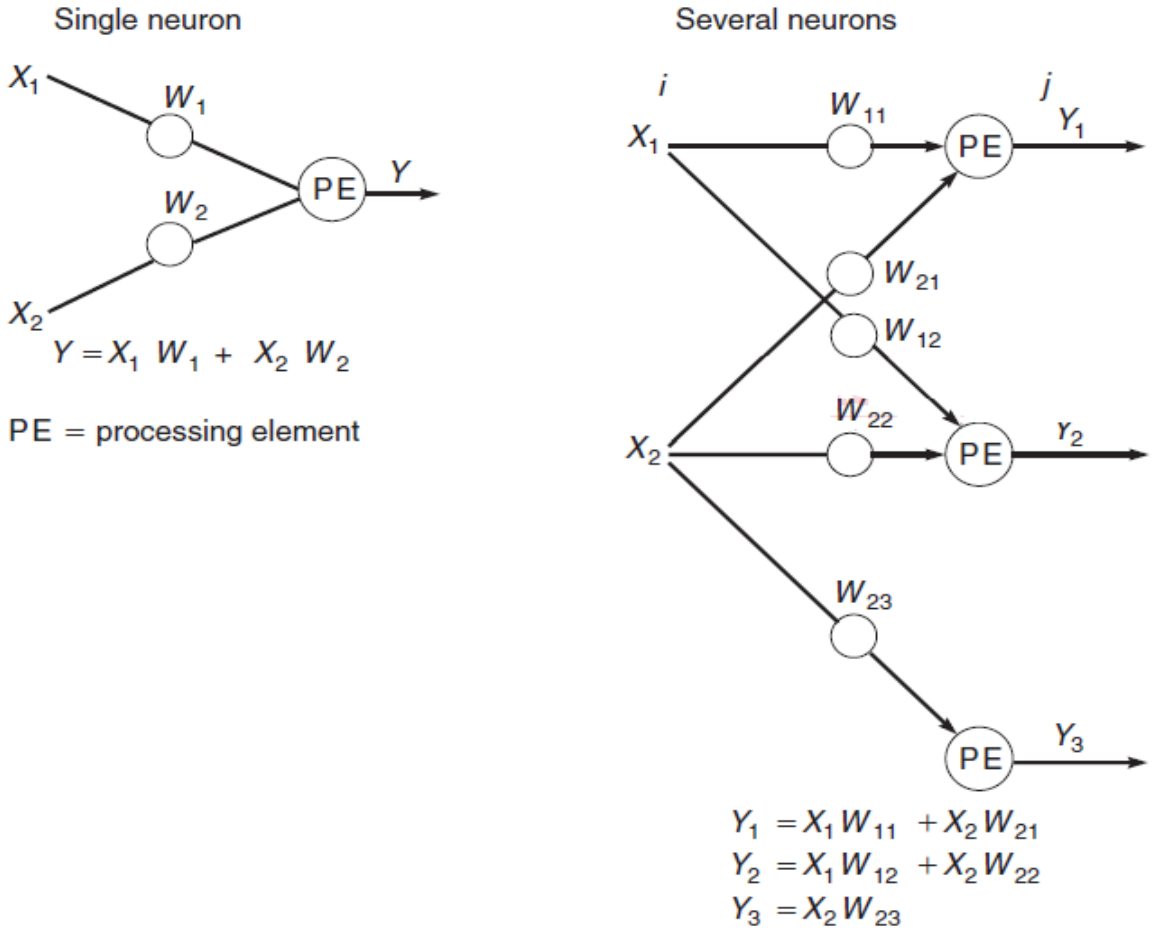
- Tập các đầu vào: Là các tín hiệu vào (input data) của nơ-ron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.
- Tập các liên kết: Mỗi liên kết được thể hiện bởi một trọng số liên kết – Synaptic weight.
- Đây là thành phần rất quan trọng của một mạng nơ-ron, nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu trong mạng). Trọng số liên kết giữa các tín hiệu vào từ layer này sang layer khác. Quá trình học thực ra là quá trình điều chỉnh các trọng số để có được kết quả mong muốn.
- 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.
- 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 Nơ-Ron, với mỗi Nơ-Ron sẽ có tối đa là một đầu ra.
- Hàm tổng (Summing function): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó. Hàm tổng của một Nơ-ron đối với n input được tính theo công thức cơ bản sau:

$$Y = \sum_{i=1}^{n} X_i W_i$$

Hàm tổng đối với nhiều Nơ-rons trong cùng một Layer :

$$Yj = \sum_{i=1}^{n} X_i W_{ij}$$

Ta có thể thể hiện qua hình sau :



Hình 1.1.2 : Cách tính hàm tổng trong một Nơ-Ron

- Hàm chuyển đổi (Transformation (Transfer) Function): Hàm tổng của một Nơ-ron cho biết khả năng kích hoạt của nơ-ron đó còn gọi là kích hoạt bên trong (internal activation).

  Mối quan hệ giữa Internal Activation và kết quả (output) được thể hiện bằng hàm chuyển đổi. Các Nơ-ron này có thể sinh ra một output hoặc không trong (output của 1 Nơ-ron có thể được chuyển đến layer tiếp theo hoặc không).

  Việc lựa chọn Transfer Function có tác động lớn đến kết quả của ANN. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong ANN là sigmoid (logical activation) function.

$$Y_T = 1/(1 + e^{-Y})$$

  Trong đó : Y_T: Hàm chuyển đổi
  Y: Hàm tổng

  Kết quả của Sigmoid Function thuộc khoảng [0,1] nên còn gọi là hàm chuẩn hóa (Normalized Function).

  Kết quả xử lý tại các Nơ-ron (Output) đôi khi rất lớn, vì vậy transfer function được sử dụng để xử lý output này trước khi chuyển đến layer tiếp theo.

  Đôi khi thay vì sử dụng Transfer Function người ta sử dụng giá trị ngưỡng (Threshold value) để kiểm soát các output của các nơ-ron tại một layer nào đó trước khi chuyển các output này đến các Layer tiếp theo. Nếu output của một nơ-ron nào đó nhỏ hơn ngưỡng thì nó sẽ không được chuyển đến Layer tiếp theo.

1.2 Kiến Trúc Của Mạng Nơ-Ron

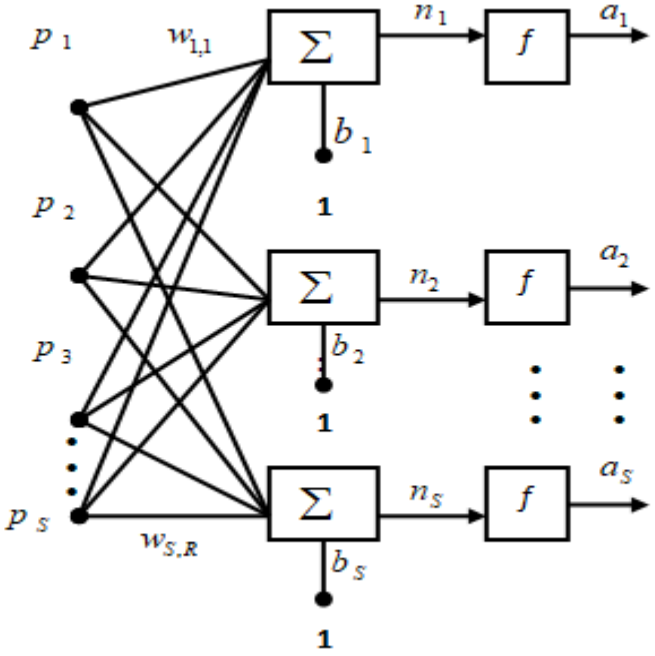
Là một hệ thống bao gồm nhiều phần tử xử lý đơn giản (hay còn gọi là nơ-ron) tựa như nơ-ron thần kinh của não người, hoạt động song song và được nối với nhau bởi các liên kết nơ-ron. Mỗi liên kết kèm theo một trọng số nào đó, đặc trưng cho tính kích hoạt hoặc ức chế giữa các nơ-ron.

Có thể xem các trọng số là phương tiện để lưu trữ thông tin dài hạn trong mạng nơ-ron và nhiệm vụ của quá trình huấn luyện của mạng là cập nhật các trọng số khi có thêm thông tin về mẫu học. Hay nói một cách khác, các trọng số đều được điều chỉnh sao cho dáng điệu vào ra của mạng sẽ mô phỏng hoàn toàn phù hợp với môi trường đang xem xét.

Mạng Một Tầng

Mỗi một đầu vào trong số R đầu vào sẽ được nối với từng nơ-ron và ma trận trọng số bây giờ sẽ có S hàng. Một tầng bao gồm ma trận trọng số, các bộ cộng, vector ngưỡng b, hàm chuyển và vector đầu ra a. Mỗi phần tử của vector đầu vào p được nối với từng nơ-ron thông qua ma trận trọng số W. Mỗi nơ-ron có một ngưỡng, một bộ cộng, một hàm chuyển *f* và một đầu ra.

Cùng với nhau, các đầu ra tạo thành một vector đầu ra a. Thông thường thì số lượng đầu vào của tầng khác với số lượng nơ-ron.



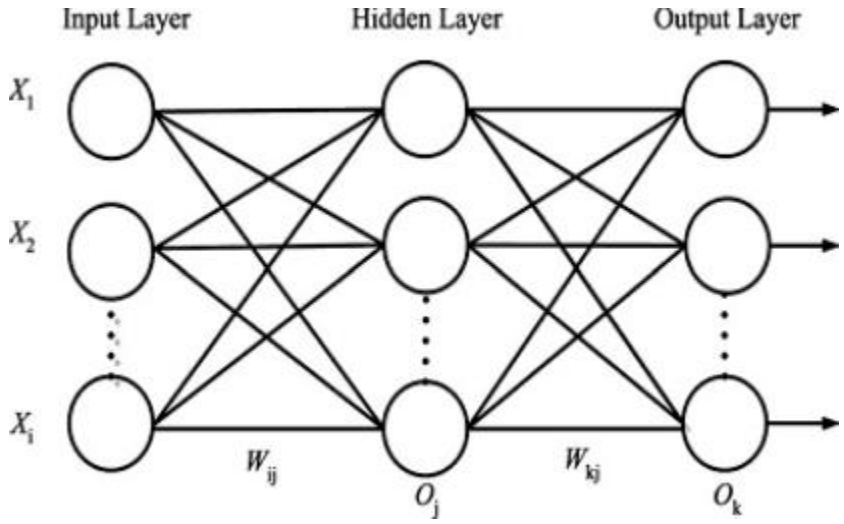
Hình 1.2.1 : Mô hình mạng một lớp

Mạng Đa Tầng

Mô hình mạng nơ-ron đa tầng gồm 3 lớp: lớp nhập (input), lớp ẩn (hidden) và lớp xuất (output). Mỗi nút trong lớp nhập nhận giá trị của một biến độc lập và chuyển vào mạng.

Dữ liệu từ tất cả các nút trong lớp nhập được tích hợp (gọi là tổng trọng số) và chuyển kết quả cho các nút trong lớp ẩn. Gọi là "ẩn" vì các nút trong lớp này chỉ liên lạc với các nút trong lớp nhập và lớp xuất, và chỉ có người thiết kế mạng mới biết lớp này (người sử dụng không biết lớp này).

Các nút trong lớp xuất nhận các tín hiệu tổng trọng hóa từ các nút trong lớp ẩn. Mỗi nút trong lớp xuất tương ứng với một biến phụ thuộc.

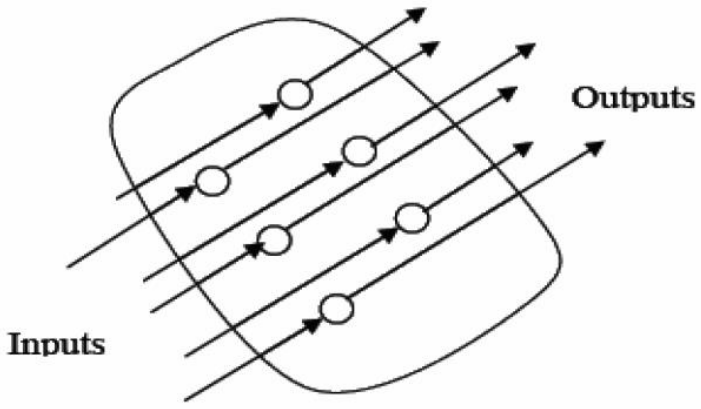


Hình 1.2.2 : Mô hình mạng nhiều lớp

1.3 Các Kiểu Mạng Của Mạng Nơ-Ron

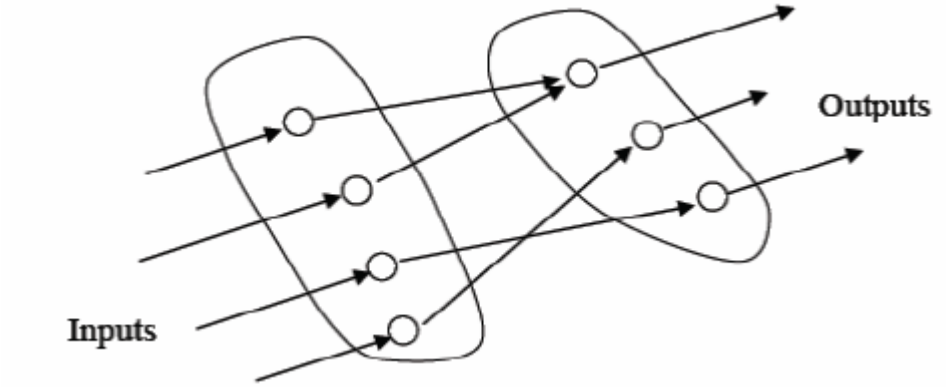
Cách thức kết nối các nơ-ron trong mạng xác định kiến trúc của mạng. Các nơ-ron trong mạng có thể kết nối đầy đủ (*fully connected*) tức là mỗi nơ-ron đều được kết nối với tất cả các nơ-ron khác, hoặc kết nối cục bộ (*partially connected*) chẳng hạn chỉ kết nối giữa các nơ-ron trong các tầng khác nhau. Người ta chia ra hai loại kiến trúc mạng chính:

Tự kết hợp (*autoassociative*): là mạng có các nơ-ron đầu vào cũng là các nơ-ron đầu ra. Mạng Hopfield là một kiểu mạng tự kết hợp.



Hình 1.3.1 : Mô hình mạng tự kết hợp

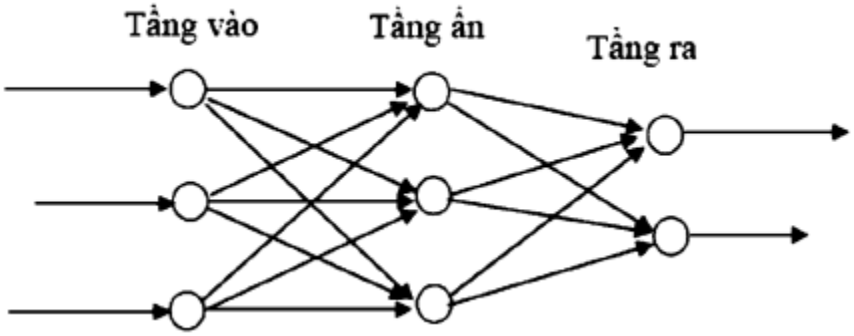
Kết hợp khác kiểu (*heteroassociative*): là mạng có tập nơ-ron đầu vào và đầu ra riêng biệt. Các mạng Perceptron nhiều tầng (MLP: MultiLayer Perceptron), mạng Kohonen, … thuộc loại này.



Hình 1.3.2 : Mô hình mạng kết hợp khác kiểu

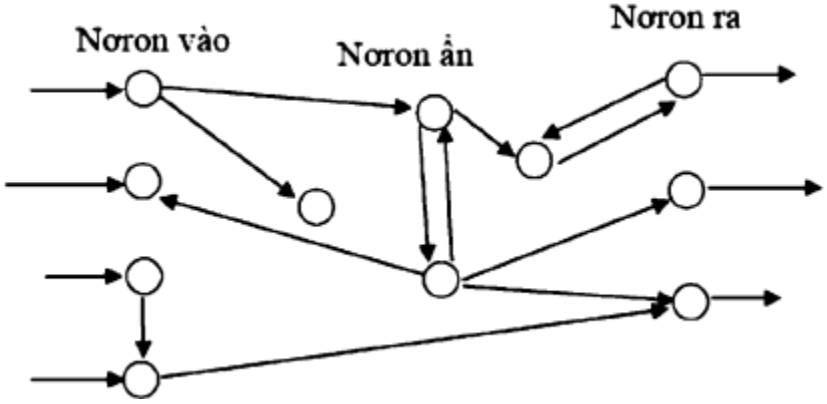
Ngoài ra tùy thuộc vào mạng có các kết nối ngược (*feedback connections*) từ các nơ-ron đầu ra tới các nơ-ron đầu vào hay không, người ta chia ra làm 2 loại kiến trúc mạng khác.

Kiến trúc truyền thẳng (*feedforward architechture*): là kiểu kiến trúc mạng không có các kết nối ngược trở lại từ các nơ-ron đầu ra về các nơ-ron đầu vào; mạng không lưu lại các giá trị output trước và các trạng thái kích hoạt của nơ-ron. Các mạng nơ-ron truyền thẳng cho phép tín hiệu di chuyển theo một đường duy nhất; từ đầu vào tới đầu ra, đầu ra của một tầng bất kì sẽ không ảnh hưởng tới tầng đó. Các mạng kiểu Perceptron là mạng truyền thẳng.



Hình 1.3.3 : Mô hình mạng truyền thẳng

Kiến trúc hồi quy (*Feedback architecture*): là kiểu kiến trúc mạng có các kết nối từ nơ-ron đầu ra tới nơ-ron đầu vào. Mạng lưu lại các trạng thái trước đó, và trạng thái tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng.



Hình 1.3.3 : Mô hình mạng hồi quy

1.4 Huấn Luyện Mạng Nơ-Ron

Chức năng của một mạng nơ-ron được quyết định bởi các nhân tố như: Kiến trúc của mạng (số lớp, số đơn vị trên mỗi tầng, và cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Kiến trúc của mạng thường là cố định, và các trọng số được quyết định bởi một thuật toán huấn luyện (training algorithm).

Tiến trình điều chỉnh các trọng số để mạng "nhận biết" được quan hệ giữa đầu vào và đích mong muốn được gọi là học (learning) hay huấn luyện (training). Trong trạng thái học thông tin được lan truyền theo hai chiều nhiều lần để học các trọng số. Rất nhiều thuật toán học đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán.

Các thuật toán đó gồm ba nhóm chính:

- Học có giám sát (supervised learning) : Quá trình Training được lặp lại cho đến kết quả (output) của ANN đạt được giá trị mong muốn (Desired value) đã biết. Điển hình cho kỹ thuật này là mạng Nơ-ron lan truyền ngược (Backpropagation).

- Học không giám sát (unsupervised learning): Không sử dụng tri thức bên ngoài trong quá trình học (Learning), nên còn gọi là tự tổ chức (Self – Organizing). Mạng Nơ-ron điển hình được huấn luyện theo kiểu Unsupervised là Sefl – Organizing Map (SOM).

- ❖ Học tăng cường (Reinforcement learning): đôi khi còn được gọi là học thưởng-phạt (reward-penalty learning), là sự tổ hợp của cả hai mô hình trên. Mạng nơ-ron nhân tạo thường được dùng trong học tăng cường như là một phần của thuật toán toàn cục. Các bài toán thường được giải quyết bằng học tăng cường là các bài toán điều khiển, trò chơi, và các nhiệm vụ quyết định tuần tự khác.

Phương pháp này cụ thể như sau: với vector đầu vào, quan sát vector đầu ra do mạng tính được. Nếu kết quả được xem là "tốt" thì mạng sẽ được thưởng theo nghĩa tăng các trọng số kết nối lên; ngược lại mạng sẽ bị phạt, các trọng số kết nối không thích hợp sẽ được giảm xuống.

Có nhiều thuật toán có thể dùng cho việc huấn luyện các mô hình mạng nơ-ron; hầu hết có thể được xem là áp dụng trực tiếp của lý thuyết tối ưu hóa và ước lượng thống kê.

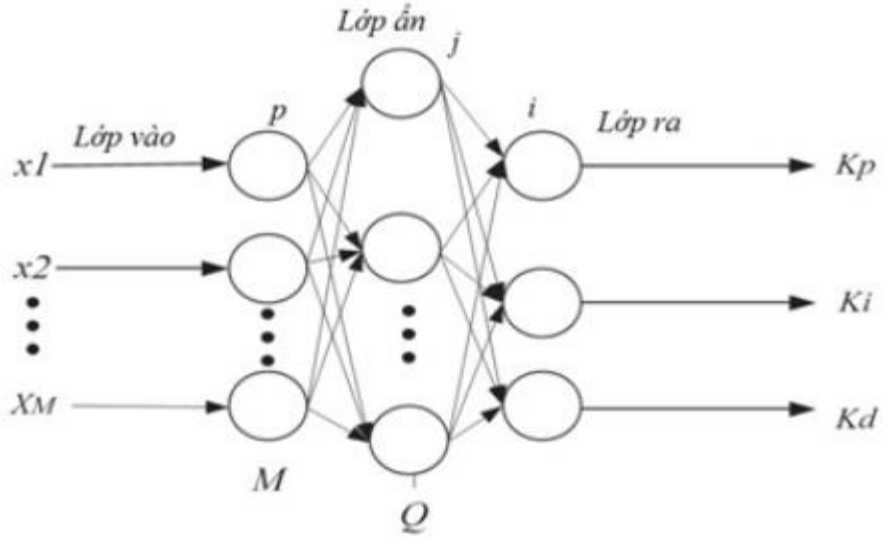
Phần lớn các thuật toán huấn luyện mạng nơ-ron sử dụng một kiểu xuống dốc (gradient descent - tiến dần tới cực tiểu địa phương) nào đó. Điều này được thực hiện bằng cách lấy đạo hàm của hàm chi phí theo các tham số của mạng và thay đổi các tham số đó theo một hướng được tính toán theo độ dốc (gradient-related direction) để tiến dần tới cực tiểu địa phương của hàm chi phí.

Các phương pháp thường dùng cho huấn luyện mạng nơ-ron là: phương pháp tiến hóa, giải thuật luyện kim (simulated annealing), expectation maximisation (cực đại hóa kỳ vọng) và các phương pháp không tham số (non-parametric methods).

1.5 Mạng Nơ-Ron Lan Truyền Ngược (Mạng Hồi Quy)

Mô hình mạng nơ-ron được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng (Multi Layer Perceptron).

Một mạng nhiều tầng tổng quát là mạng có n ($n \geq 2$) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.



Hình 1.5.1 : Mạng nơ-ron lan truyền ngược

Kiến trúc của một mạng nhiều tầng tổng quát có thể mô tả như sau:

- Đầu vào là các vector (x1, x2,..., xp) trong không gian p chiều, đầu ra là các vector (y1, y2,..., yq) trong không gian q chiều. Đối với các bài toán phân loại, p chính là kích thước của mẫu đầu vào, q chính là số lớp cần phân loại.
- Mỗi nơ-ron thuộc tầng sau liên kết với tất cả các nơ-ron thuộc tầng liền trước nó.
- Đầu ra của nơ-ron tầng trước là đầu vào của nơ-ron thuộc tầng liền sau nó.

Hoạt động của một mạng lan truyền ngược nhiều tầng:

- Tại tầng đầu vào các nơ-ron nhận dữ liệu đầu vào vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền)
- Kết quả này sẽ được truyền tới các nơ-ron thuộc tầng ẩn thứ nhất
- Các nơ-ron tại đây tiếp nhận như là dữ liệu đầu vào, xử lý và gửi kết quả đến tầng ẩn tiếp theo.
- So sánh đầu ra mong muốn và hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa dữ liệu đầu vào vào mạng, dữ liệu đầu ra sẽ giống như mong muốn hơn.
- Quá trình lặp lại tiếp tục cho đến khi các nơ-ron thuộc tầng ra cho kết quả như mong muốn hoặc lặp tới một giá trị mong muốn nhất định.

Hoạt động của mạng hồi quy nhiều tầng dựa theo phương pháp huấn luyện có giám sát dựa trên giải thuật Back – Propagation (giải thật lan truyền ngược)

Thuật toán Back – Propagation được sử dụng để điều chỉnh các trọng số kết nối sao cho tổng sai số E nhỏ nhất. Được tính bằng công thức sau :

$$\mathbf{E = \sum_{i=1}^{n} (t\,(x_i, w) - y\,(x_i))^2}$$

Trong đó:

E : Sai số giữa kết quả thực tế và kết quả mong muốn
t (x_i, w): giá trị của tập mẫu
y (x_i): giá trị tính được của mạng

Đây là phương pháp thông dụng nhất để huấn luyện cho các mạng nơ-ron truyền thẳng nhiều lớp. Có thể áp dụng cho các mạng truyền thẳng với các hàm chuyển và các hàm lỗi khả vi.

Tiêu chuẩn huấn luyện: Làm cho sai số đầu ra càng nhỏ càng tốt. Bằng cách dựa trên đầu ra để điều chỉnh trọng số của lớp ra, sau đó dựa trên tính toán của lớp ra để điều chỉnh trọng số của lớp ẩn.

Huấn luyện mạng nơ-ron nhiều lớp sử dụng thuật toán lan truyền ngược gồm hai quá trình: quá trình truyền tuyến tính và quá trình truyền ngược:

- Quá trình truyền tuyến tính: dữ liệu từ lớp nhập qua lớp ẩn và đến lớp xuất để thay đổi giá trị của trọng số liên kết của các nơ-ron trong mạng biểu diễn được dữ liệu học. Sau đó tìm ra sự khác nhau giữa giá trị thật hàm mẫu mà mạng tính được và kết quả dự đoán của mạng gọi là lỗi (học có giám sát).
- Quá trình truyền ngược: Giá trị lỗi sẽ được truyền ngược lại sao cho quá trình huấn luyện (học) sẽ tìm ra trọng số để lỗi nhỏ nhất.

Ngoài những thành công của giải thuật học lan truyền ngược, vẫn còn có một số khía cạnh làm cho giải thuật trở nên chưa được bảo đảm là mọi lúc đều tốt. Khó khăn chủ yếu là ở quá trình huấn luyện lâu. Có thể do nhịp độ học và động lực không tối ưu. Sự sai sót trong việc huấn luyện nói chung xuất hiện từ hai nguồn: mạng liệt và những cực tiểu địa phương.

- Mạng liệt: xảy ra khi những trọng số được điều chỉnh tới những giá trị rất lớn. Tổng đầu vào của một đơn vị ẩn hoặc đơn vị đầu ra có thể bởi vậy mà đạt giá trị rất cao (hoặc dương hoặc âm), và qua hàm kích hoạt sigmoid, đơn vị sẽ có một giá trị kích hoạt rất gần 0 hoặc rất gần 1. Giá trị hiệu chỉnh trọng số gần 0, và quá trình huấn luyện có thể đi đến một trạng thái dừng ảo.

- Cực tiểu địa phương: bề mặt sai số của mạng rất phức tạp đầy những điểm cực đại và cực tiểu. Trong quá trình điều chỉnh trọng số, mạng có thể bị k tại một cực tiểu địa phương khi có nhiều cực tiểu thấp hơn gần bên cạnh.

  Những phương pháp thống kê có thể giúp để tránh lỗi này, nhưng chúng làm chậm chương trình. Một phương án khác là tăng thêm số lượng Nơ-ron ẩn, tầng ẩn. Như vậy chương trình sẽ làm việc trong không gian sai số nhiều chiều, nên cơ hội gặp lỗi nhỏ hơn. Tuy nhiên việc tăng cũng có giới hạn trên, khi vượt qua giới hạn này, cơ hội mắc lỗi lại tăng lên.

1.6 Các Hàm Kích Hoạt Trong Mạng Nơ-Ron

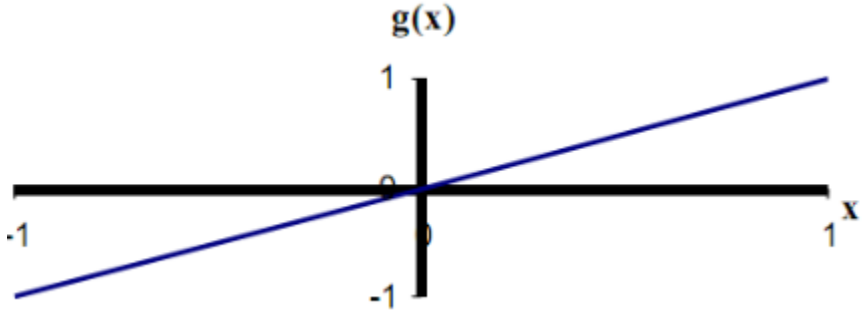
Phần lớn các đơn vị trong mạng nơ-ron chuyển dữ liệu bằng cách sử dụng một hàm vô hướng gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị. Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác.

Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Nhờ có các hàm kích hoạt này, dữ liệu đầu vào sẽ được chuẩn hóa theo một cách mà ta mong muốn, giúp dữ liệu Các hàm kích hoạt hay được sử dụng là:

❖ **Hàm đồng nhất (Linear function, Identity function)**

$$g(x) = x$$

còn gọi là quan hệ đồng nhất (identity relation), ánh xạ đồng nhất (identity map) hoặc phép biến đổi đồng nhất (identity transformation), là một hàm luôn luôn trả về giá trị bằng chính tham số của nó.



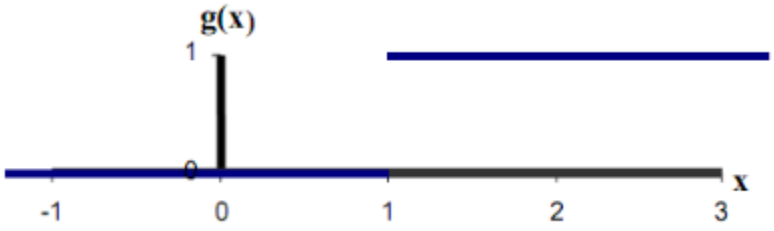
Hình 1.6.1 : Hàm đồng nhất

❖ **Hàm bước nhị phân (Binary step function, Hard limit function)**

Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function). Đầu ra của hàm này được giới hạn vào một trong hai giá trị (với θ là ngưỡng):

$$g(x) = \begin{cases} 1, & nếu\ (x \geq \theta) \\ 0, & nếu\ (x < \theta) \end{cases}$$

Dạng hàm này được sử dụng trong các mạng chỉ có một lớp. Trong hình vẽ sau, θ được chọn bằng 1.



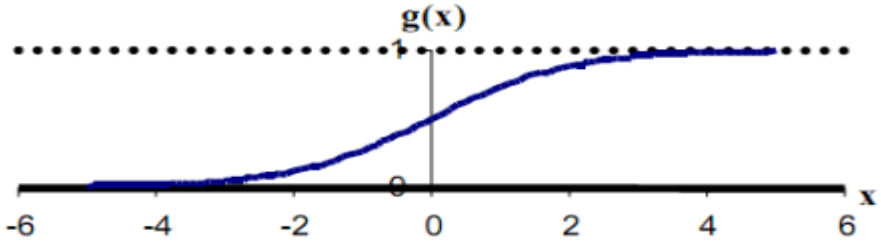
Hình 1.6.2 : Hàm bước nhị phân

❖ **Hàm Sigmoid (Sigmoid function)**

$$g(x) = \frac{1}{1+e^{-x}}$$

Hàm được gọi là hàm sigmoid hay đường cong sigmoid. Tên gọi của nó xuất phát từ hình dáng của nó. Hàm này còn gọi là hàm lôgit chuẩn. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng [0,1].

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện bởi thuật toán lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện.



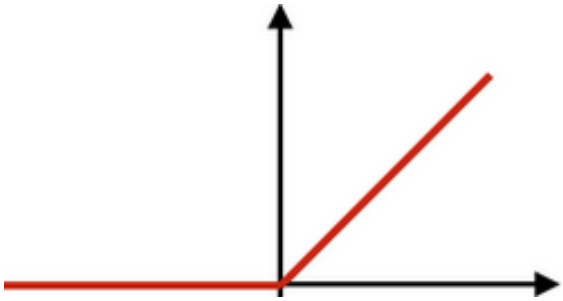
Hình 1.6.3 : Hàm Sigmoid

❖ **Hàm ReLU(Rectified Linear Units) (ReLU function)**

f(x) = max(x,0)

Thay vì hàm sigmoids, hầu hết các mạng gần đây sử dụng các đơn vị tuyến tính được chỉnh sửa (ReLUs) cho các lớp ẩn. Một đơn vị có đầu ra 0 nếu đầu vào nhỏ hơn 0. Tức là, nếu đầu vào lớn hơn 0, dữ liệu đầu ra bằng với đầu vào. Nếu đầu vào nhỏ hơn 0, đầu ra bằng 0.

Hàm này cũng được sử dụng nhiều khi sử dụng cho các mạng được huấn luyện bởi thuật toán lan truyền ngược bởi khả năng xử lý dữ liệu nhanh chóng, do đó có thể giảm thời gian giảm và quá trình tính toán trong huấn luyện. Các nghiên cứu đã chỉ ra rằng ReLUs dẫn đến quá trình huấn luyện nhanh hơn cho các mạng lớn.



Hình 1.6.4 : Hàm ReLU

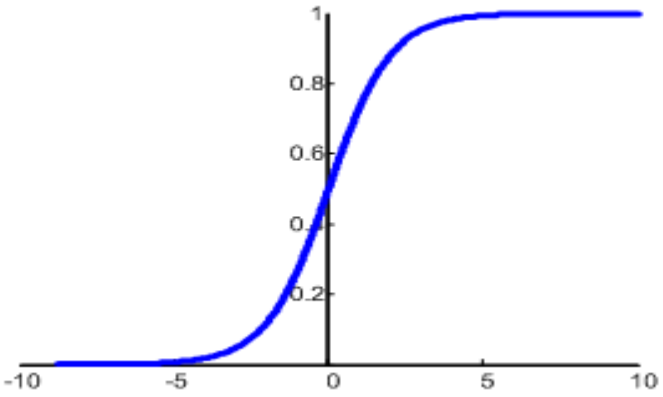
❖ **Hàm Softmax (Softmax function)**

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Sigmoidal hay còn gọi là chuẩn hóa softmax là một cách làm giảm sự ảnh hưởng của các giá trị cực và ngoại lai trong dữ liệu mà không phải loại bỏ chúng ra khỏi dải dữ liệu. Đó là những dữ liệu ngoại lai hữu ích mà ta muốn giữ chúng trong dải dữ liệu trong khi vẫn đảm bảo sự ý nghĩa của dữ liệu trong phạm vi sai lệch chuẩn của giá trị trung bình.

Hàm softmax làm kết quả đầu ra của mỗi đơn vị được giữa 0 và 1, giống như hàm sigmoid. Nhưng nó cũng phân chia mỗi đầu ra sao cho tổng số các kết quả đầu ra bằng 1.

Hình 1.6.5 Phương thức hoạt động của hàm Softmax



Hình 1.6.6 : Hàm Softmax

CHƯƠNG II: THỰC NGHIỆM TRÊN MẠNG NƠ-RON

2.1 Dữ Liệu Huấn Luyện

- Bộ dữ liệu huấn luyện :
  - Bộ Khen-Chê
    - Ngôn Ngữ : Tiếng Việt.
    - Số Lượng Nhãn : 2 Nhãn (Khen : 1, Chê : 0).
    - Số Lượng Câu : 1540 câu.
      - Khen (500 câu).
      - Chê (1040 câu).

  - Bộ Text Emotion
    - Ngôn Ngữ : Tiếng Anh.
    - Số Lượng Nhãn: 13 Nhãn (Anger : 0, Boredom : 1, Empty :2, Enthusiasm :3, Fun:4, Happiness: 5, Hate : 6, Love :7, Neutral :8, Relief :9, Sadness:10, Supprise :11, Worry :12)
    - Số Lượng Câu : ~ 40.000 câu. 39704
      - Anger (108 câu)
      - Boredom (179 câu)
      - Empty (800 câu)
      - Enthusiasm (755 câu)
      - Fun (1767 câu)
      - Happiness (5179 câu)
      - Hate (1319 câu)
      - Love (3829 câu)
      - Neutral (8486 câu)
      - Relief (1512 câu)
      - Sadness (5153 câu)
      - Supprise (2184 câu)
      - Worry (8433 câu)

    Nguồn : MohMehrnia [A]

- Vector hóa dữ liệu :

Đối với dữ liệu tiếng Việt, do tính chất phong phú và kết nối của các từ trong tiếng Việt, tiến hành các công đoạn xử lý nối các từ tương quan trong câu bằng công cụ UETsegmenter [B]. Đối với dữ liệu tiếng Anh không cần xử lý qua bước này.

Sau đó, sử dụng bộ dữ liệu vec-tơ của Glove [C] để chuyển đổi bộ huấn luyện từ dạng chữ sang một dãy vec-tơ trung vị có 300 chiều để có bộ huấn luyện phù hợp cho mạng nơ-ron tiến hành xử lý phân loại.

2.2 Thư Viện Và Các Hàm Sử Dụng

2.2.1 Thư Viện keras

Keras là một thư viện dễ sử dụng để phát triển và đánh giá mô hình học sâu (deep learning), viết bằng Python và có khả năng chạy trên TensorFlow, Theano.... Thư viện được phát triển với trọng tâm là cho phép thử nghiệm nhanh, cho phép bạn xác định và huấn luyện mô hình mạng nơ-ron trong một vài dòng code ngắn gọn.

Thư viện keras rất thân thiện với người dùng. Nó đặt trải nghiệm người dùng lên hàng đầu. Keras thực hiện các biện pháp tốt nhất để giảm tải quá trình tìm tòi rắc rối. Nó cung cấp các API nhất quán và đơn giản, giảm thiểu số lượng hành động của người dùng được yêu cầu cho các trường hợp sử dụng phổ biến, cung cấp phản hồi rõ ràng và có thể thực hiện được theo lỗi mắc phải của người dùng.

Một số lợi ích:

- Cho phép tạo mô hình huấn luyện dễ dàng và nhanh chóng (thông qua sự thân thiện của người dùng, tính mô-đun và tính mở rộng).
- Hỗ trợ cả mạng tích chập và mạng hồi quy, cũng như sự kết hợp của cả hai.
- Khả năng phát triển, mở rộng mô hình đơn giản và dễ dàng.
- Chạy liền mạch trên CPU và GPU

2.2.2 Các Hàm Hỗ Trợ Trong Thư Viện Keras Được Sử Dụng Trong Bài

- keras.layers.Dense (units, activation, input_shape,...) : Hàm tạo các mô hình kết nối với nhau, dùng để tạo ra mạng Nơ-ron huấn luyện dữ liệu.
  - Units : kích thước của không gian hoặc ma trận đầu ra, phải là một số nguyên dương.
  - Activation : kích hoạt các hàm kích hoạt để mô hình đạt hiệu quả cao hơn. Nếu không được chỉ định hàm kích hoạt nào, hàm kích hoạt sẽ được sử dụng là hàm "tuyến tính": a (x) = x
  - Input_shape : kích thước của không gian hoặc ma trận đầu vào

- keras.layers.Conv1D (filters, kernel_size, strides, padding, activation, input_shape...) :Hàm khai báo các hàm kích hoạt được sử dụng trong mạng.
  - Filters : kích thước của không gian hoặc ma trận đầu ra, phải là một số nguyên dương.
  - Kernel_size : Một số nguyên chỉ định độ dài của ma trận tích chập convolution 1D được sử dụng.

- Padding : Một trong số các giá trị "valid", " causal" hoặc "same" (không phân biệt chữ hoa chữ thường).
  "valid" nghĩa là không thay đổi,"same" kết quả đầu ra có chiều dài tương tự như đầu vào ban đầu, " causal" kết quả trong tích chập mở rộng, ví dụ. đầu ra [t] không phụ thuộc vào đầu vào [t + 1:]. Hữu ích khi mô hình hóa dữ liệu thời gian mà mô hình không vi phạm trật tự thời gian.
- Activation : kích hoạt các hàm kích hoạt để mô hình đạt hiệu quả cao hơn. Nếu không được chỉ định hàm kích hoạt nào, hàm kích hoạt sẽ được sử dụng là hàm "tuyến tính": a (x) = x
- Input_shape : kích thước của không gian hoặc ma trận đầu vào

- keras.layers.MaxPooling1D (pool_size, padding,...) : tìm giá trị lớn nhất trong 1 lần trượt để tối ưu hóa giá trị đặc trưng.
  - Pool_size : kích thước của các cửa sổ max pooling(chọn giá trị lớn nhất) tối đa.
  - Padding : Một trong số các giá trị "valid", " causal" hoặc "same" (không phân biệt chữ hoa chữ thường).

- keras.layers.Flatten() : chuyển đổi làm phẳng các ma trận đầu vào thành một dãy các giá trị.

- keras.layers.Reshape (target_shape) : chuyển định dạng các ma trận đầu vào (target_shape) thành định dạng phù hợp với mạng nơ-ron.

- keras.layers.Dropout (rate) thiết lập một tỷ lệ (rate) ngưỡng của các đơn vị đầu vào thành 0 tại mỗi lần cập nhật trong thời gian huấn luyện, giúp ngăn chặn overfitting.

- compile (loss, optimizer, metrics,...) Biên dịch dữ liệu cho mô hình huấn luyện.
  - Optimizer : phương thức tối ưu hóa mô hình huấn luyện.
  - Loss : Phương thức tính toán giá trị mất mát trobg mô hình huấn luyện, ngoài ra còn được sử dụng để chỉ định dạng mô hình huấn luyện gồm “categorical_crossentropy”(đánh giá mô hình có nhiều đầu ra) và“categorical_crossentropy”(đánh giá mô hình chỉ có 2 đầu ra) .

- Metrics : Danh sách chỉ số được đánh giá theo mô hình trong quá trình đào tạo và thử nghiệm. Thông thường sẽ sử dụng metrics = ['accuracy']. Để chỉ định các chỉ số khác nhau cho đầu ra khác nhau của mô hình nhiều đầu ra.

- fit (x, y, batch_size, epochs, validation_data) : Huấn luyện mô hình.
  - X : dữ liệu huấn luyện
  - Y : nhãn cũa dữ liệu huán luyện X.
  - Batch_size: số lượng mẫu mỗi lần cập nhật trọng số.
  - Epochs : số lần chạy của tập huấn luyện.
  - Validation_data : dữ liệu kiểm tra so sánh sự chuẩn xác của dữ liệu huấn luyện.

- predict(x,...) : dự đoán nhãn của dữ liệu bằng mô hình huấn luyện.
  - X : dữ liệu cần tìm nhãn

2.3 Thực Nghiệm trên mạng Nơ-ron Lan Truyền Ngược Cơ Bản

Mô hình thực nghiệm được tiến hành với sự hỗ trợ bằng các hàm xây dựng mô hình mạng nơ-ron của thư viện keras, cùng tập huấn luyện là 2 bộ dữ liệu tiếng Anh và tiếng Việt sử dụng các hàm kích hoạt khác nhau để tìm ra được cách phân loại câu tốt nhất.

2.3.1 Các yếu Tố Xây Dựng Mạng Nơ-ron

Dưới đây là các yếu tố khi tiến hành thiết kế và thực thi mạng nơ-ron nhân tạo cho bài toán phân loại quan điểm :

- Đầu vào và đầu ra mong muốn :
  - Đầu vào : dữ liệu đầu vào cho mô hình mạng nơ-ron là một tập các vec-tơ 300 chiều sau khi đã được chuẩn hóa từ các câu trong bộ huấn luyện và qua các bước vec-tơ hóa cùng trung vị câu.
  - Đầu ra : là một nhãn đã được phân loại dựa trên mô hình huấn luyện.
- Cấu trúc mạng : Mạng nơ-ron được xây dựng theo phương pháp học có giám sát. Bài toán lựa chọn mạng Feed-forward 3 lớp với cấu trúc như sau :
  - Số nơ-ron lớp đầu vào : 300 nơ-ron.

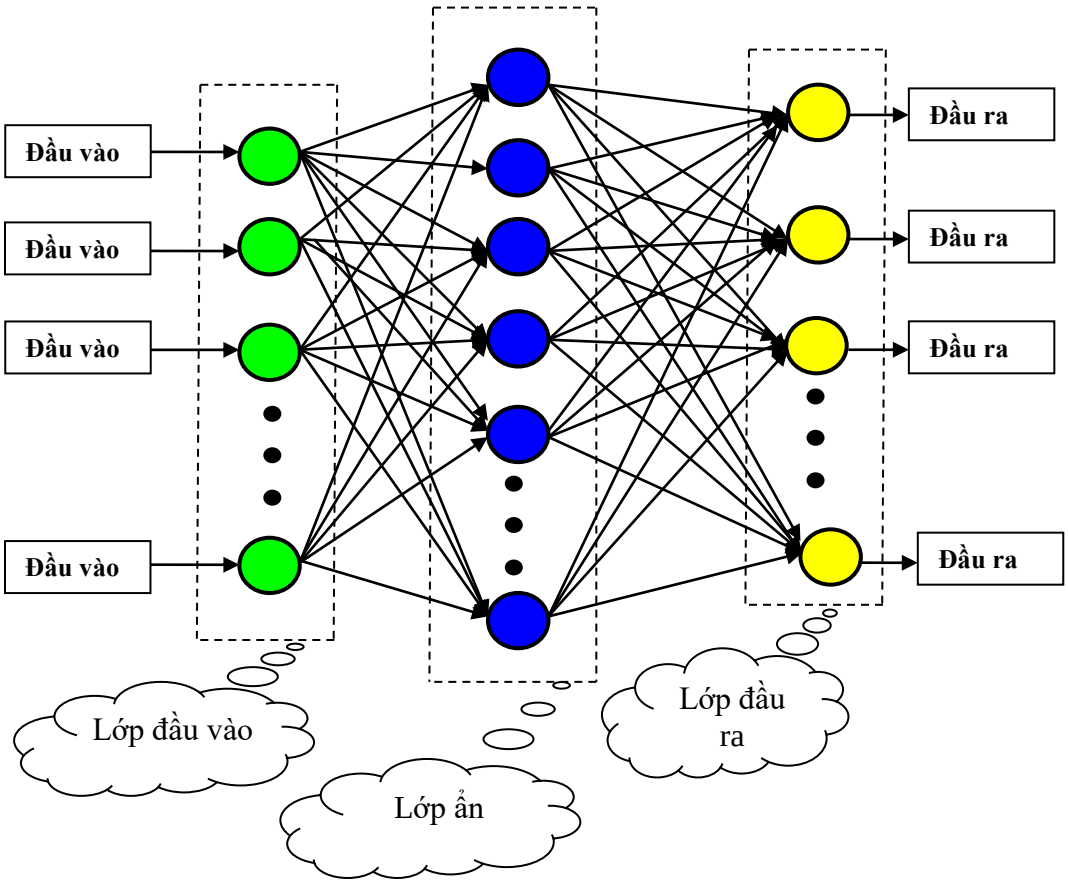
Việc huấn luyện cho mạng học là một vòng lặp duyệt qua lần lượt các câu giúp mạng nơ-ron nhớ và nhận dạng các vec-tơ này. Với mỗi vòng lặp, một chuỗi vec-tơ sẽ được đưa vào giảng dạy cho mạng nơ-ron học. Trong quá trình này, mỗi vec-tơ sẽ được phân tích và chuyển tỉ lệ vào một ma

trận. Ứng với mỗi điểm của ma trận sẽ được tuyến tính hóa tạo ra tương ứng với một nơ-ron đầu vào. Việc chọn số lượng nơ-ron đầu vào hay nói cách khác việc chọn tỉ lệ ma trận đầu vào cho mô hình là rất quan trọng.

- Nếu số lượng nơ-ron lớn quá sẽ dẫn đến việc : Chương trình viết ra phải duyệt qua rất nhiều vòng lặp dẫn đến chương trình bị chậm, hoặc gây ra tình trạng bị đứng máy khi chạy.
- Nếu số lượng nơ-ron quá nhỏ : Việc phân tích các câu thường sẽ dẫn đến sai số lớn, vì vậy, việc phân loại cũng sẽ thiếu tính chính xác.

➢ Số nơ-ron lớp ẩn : 100 nơ-ron.Việc lựa chọn giá trị này dựa trên độ lớn của tập huấn luyện. Số nơ-ron này được quyết định từ những kết quả tối ưu và giá trị lỗi cơ sở trong quá trình huấn luyện.

➢ Số nơ-ron đầu ra : 2 hoặc 13 nơ-ron phụ thuộc vào số lượng nhãn cần phân loại.



Hình 2.3.1.1 : Mô hình mạng nơ-ron

- Thuật toán huấn luyện mạng :

- Lan truyền xuôi đầu vào qua mạng:

Công thức chung tính đầu ra của một nơ-ron thứ i tại lớp thứ k:

$$y_{i=} \quad f\left(\sum_{j=0}^{n} w_{ij} x_j - b_i\right)$$

Áp dụng đối với mô hình mạng của chương trình :

Công thức cho đầu ra của một nơ-ron thứ i tại lớp ẩn :

$$a_i = f\left(\sum_{j=1}^{n} w_{ij} x_j - b_i\right)$$

Với w_{ij} : trọng số tại nơ-ron thứ i của lớp ẩn kết nối với đầu vào thứ j của lớp vào.

x_j : giá trị đầu vào của nơ-ron thứ j tại lớp vào.

b_i : giá trị ngưỡng hay độ lệch của nơ-ron thứ i của đầu vào.

Công thức cho đầu ra của một nơ-ron thứ k tại lớp output:

$$y_k = f\left(\sum_{j=1}^{n} w_{kj} a_j - b_k\right) = f\left[\sum_{j=1}^{n} w_{kj} \left(f\left(\sum_{j=1}^{n} w_{ij} x_j\right)\right)\right]$$

- Lan truyền ngược :

Tính toán sai lệch giữa đầu ra thực và đầu ra mong muốn của nơ-ron thứ k tại đầu ra.

$$e_{i=} t_i - y_i$$

Tổng bình phương sai số của mạng ứng với mẫu học (Xs, Ts):

$$E = \frac{1}{2} \sum_{k=1}^{p} (t_i - y_i)^2$$

P: số phần tử đầu ra

Nếu E< ε và l+1< số lần dạy (epochs)) thì :

Thông tin sai số sẽ được lan truyền ngược qua mạng để điều chỉnh lại trọng số tại vòng lặp l.

Công thức điều chỉnh trọng số với liên kết giữa nơ-ron thứ j trong lớp ẩn và nơ-ron thứ i trong lớp ra tại lần lặp l+1: (l+1<số lần dạy (epochs))

$$w_{ij}(l+1) = w_{ij}(l) + \eta . e_i(l) . y_j . f'(y_i(l))$$

Với η: hệ số học

$e_i(l)$: giá trị sai lệch của nơ-ron thứ i trong lớp ra, trong lần dạy (lặp) thứ l.

f': đạo hàm của hàm chuyển lưỡng cực, công thức $f'=\frac{1-x^2}{2}$

η : hệ số học

$y_i(l)$: giá trị đầu ra của nơ-ron thứ i trong lớp ra tại vòng lặp thứ l.

$y_j(l)$: giá trị đầu ra của nơ-ron thứ j trong lớp ẩn tại vòng lặp thứ l.

Công thức điều chỉnh trọng số với liên kết giữa nơ-ron vào thứ j và nơ-ron ẩn thứ i, tại lần lặp thứ l+1 (l+1 <epochs)

$$w_{ij}(l+1) = w_{ij}(l) + \eta.x_j.f'(y_i).\sum_{k=1}^{m} w_{ki}(l+1).e_k(l).f'(y_k(l))$$

Với :

η: hệ số học

x_j : giá trị đầu ra của nơ-ron thứ j trong lớp vào.

y_i : giá trị đầu ra của nơ-ron thứ i trong lớp ẩn

$w_{ki}(l+1)$: trọng số liên kết giữa nơ-ron thứ k trong lớp ra và nơ-ron thứ i trong lớp ẩn trong lần lặp thứ l+1.

$y_k(l)$: giá trị đầu ra của nơ-ron thứ k trong lớp ra.

2.3.3 Tạo Lập Và Huấn Luyện Mạng Nơ-ron Với Thư Viện Keras

Để tạo lập mô hình huấn luyện, đầu tiên cần phải khai báo các thư viện cần thiết như keras, numpy, os, scipy,... để phục vụ cho quá trình xây dựng và tính toán trong mô hình.

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation
import os
import numpy as np
from scipy import sparse
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

Hình 2.3.3.1 : Các thư viện cần thiết để huấn luyện mạng nơ-ron

Tiếp theo ta tiến hành lấy dữ liệu huấn luyện và tạo lập mô hình huấn luyện bằng các hàm hỗ trợ của phần models trong thư viện keras như sau.

```
# create model
model = Sequential()
model.add(Dense(100, input_dim=300 ))
model.add(Dropout(0.5))
model.add(Dense(13))
# Compile model
model.compile(optimizer='sgd',loss='categorical_crossentropy',metrics=['accuracy'])
```

Hình 2.3.3.2 : Tạo mô hình mạng Nơ-ron cơ bản bằng thu viện keras

Mô hình cơ bản được tạo bởi thư viện keras như trên có đầu vào là một chuỗi vec-tơ có 300 chiều, liên tiếp với một mạng nơ-ron một tầng ẩn gồm 100 nơ-ron và tầng cuối là số nhãn đầu ra là 13 nhãn. Sau đó mô hình được biên dịch để phù hợp với mạng huấn luyện.

Cuối cùng ta thực hiện việc huấn luyện với dữ liệu huấn luyện là X, Y và dữ liệu so sánh kết quả là validation_data rồi hiện độ chính xác của mô hình ra màn hình.

```
# Fit the model
history = model.fit(X, Y, epochs=40, batch_size=10, validation_data=(X_test, Y_test))
# evaluate the model
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

Hình 2.3.3.3 : Tạo mô hình mạng Nơ-ron cơ bản bằng thu viện keras

```
Epoch 50/50
1459/1459 [==============================] - 0s - loss: 0.2520 - acc: 0.9047
- val_acc: 0.9171
1280/1459 [=========================>....] - ETA: 0s
acc: 91.71%
```

Hình 2.3.3.4 :Kết quả của mô hình mạng Nơ-ron cơ bản bằng thu viện keras

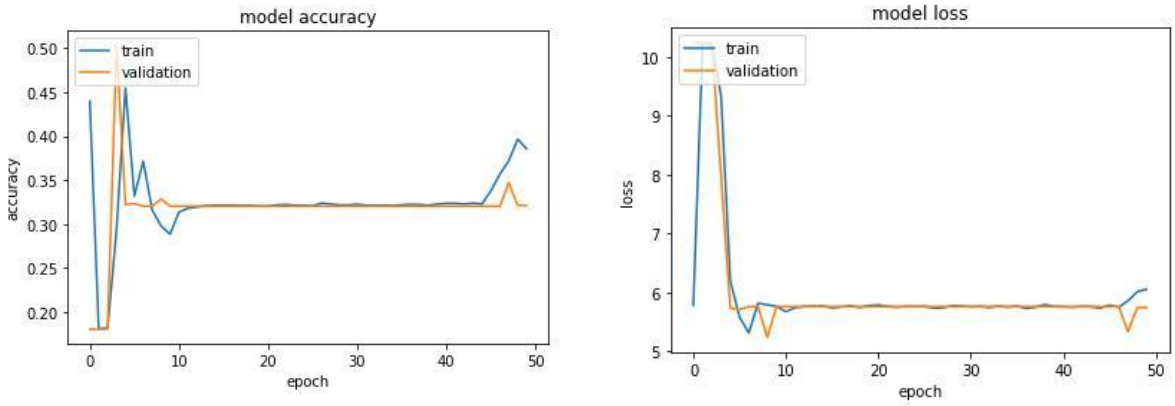
Sau khi huấn luyện mô hình sẽ phản hồi lại độ chính xác của kết quả huấn luyện, chỉ số này có thể dùng để xác định được mức độ huấn luyện của mô hình có thể sử dụng trong thực nghiệm là bao nhiêu.

2.3.4 Kết Quả Thực Nghiệm Mạng Nơ-ron Cơ Bản

Kết quả của quá trình huấn luyện mô hình với các hàm kích hoạt khác nhau để so sánh và tìm hiểu được phương thức chọn hàm kích hoạt phù hợp để phân tích quan điểm tốt nhất.

❖ Bộ dữ liệu Khen-Chê:

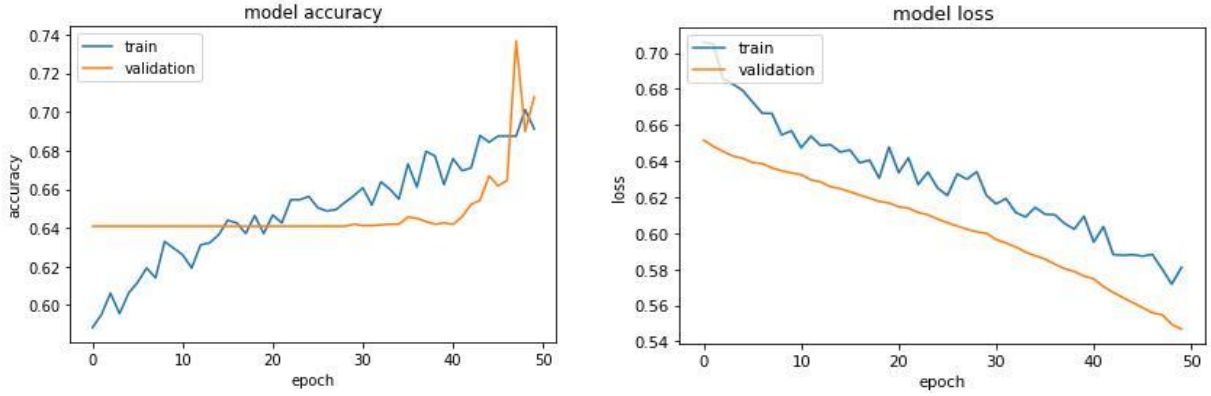
➢ Với hàm kích hoạt ReLU:



Hình 2.3.4.1-2 : Kết quả mô hình chỉ sử dụng hàm kích hoạt ReLU

Độ chính xác của mô hình đạt được là 32,11 % với 50 vòng lặp để huấn luyện.

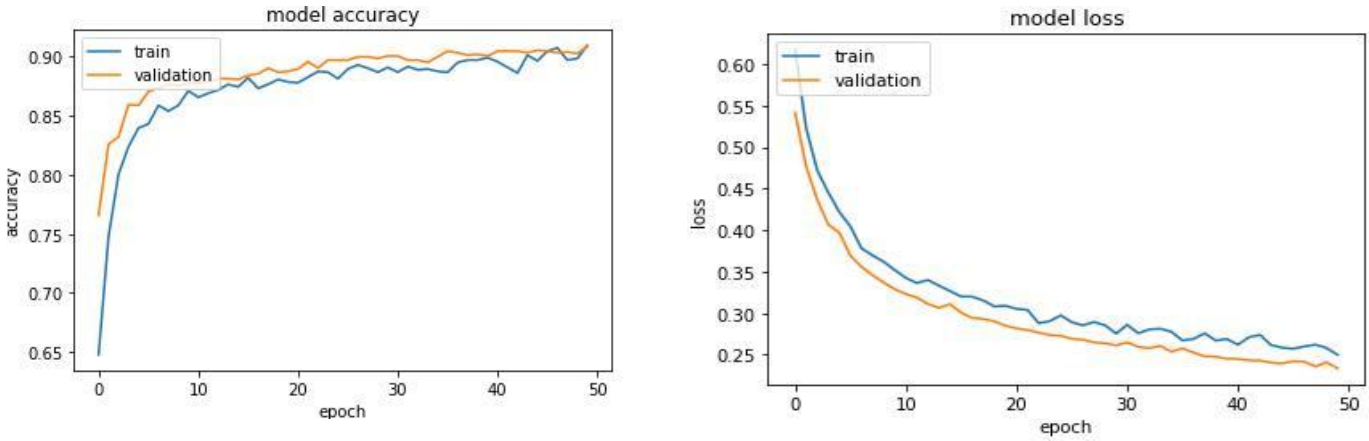
➢ Với hàm kích hoạt Sigmoid:



Hình 2.3.4.3-4 : Kết quả mô hình chỉ sử dụng hàm kích hoạt Sigmoid

Độ chính xác của mô hình đạt được là 70,77 % với 50 vòng lặp để huấn luyện.

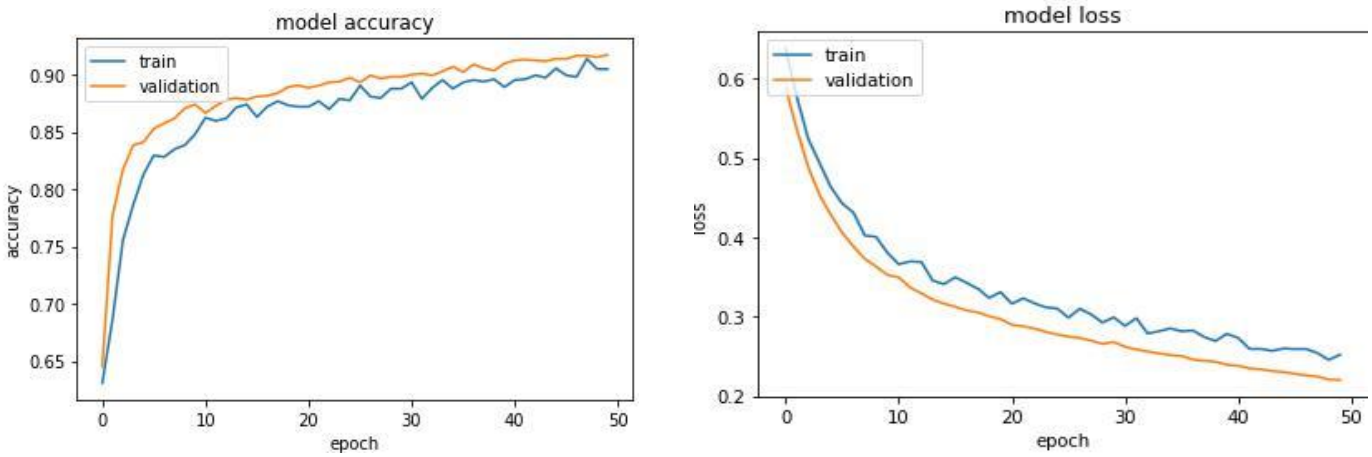
- Với hàm kích hoạt Softmax:



Hình 2.3.4.5-6 : Kết quả mô hình chỉ sử dụng hàm kích hoạt Softmax

Độ chính xác của mô hình đạt được là 90,88% với 50 vòng lặp để huấn luyện.

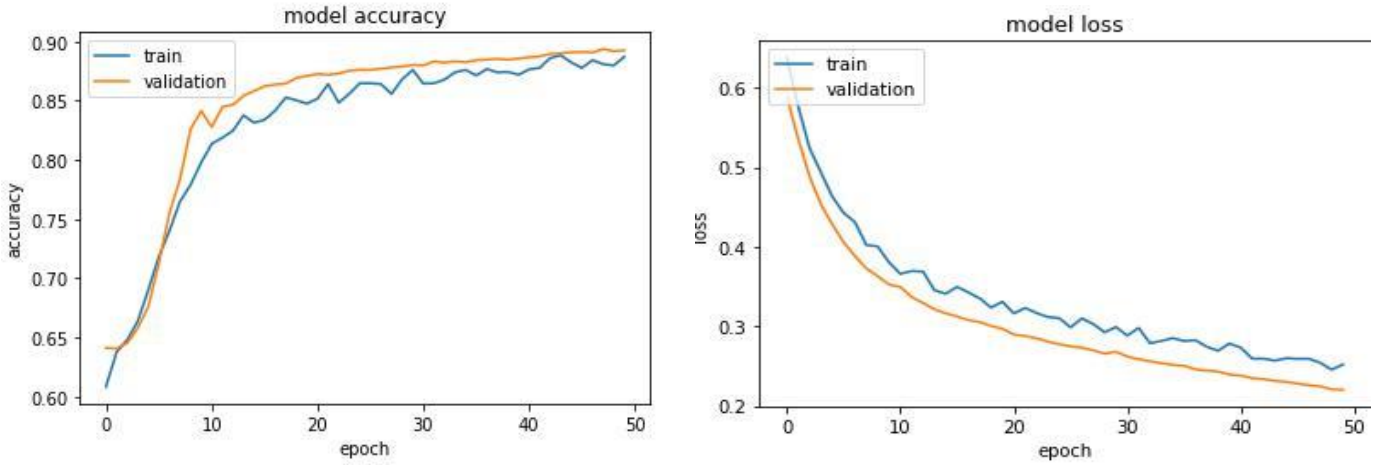
- Với hàm kích hoạt ReLU-Softmax:



Hình 2.3.4.7-8 : Kết quả mô hình sử dụng hàm kích hoạt ReLU kết hợp Softmax

Độ chính xác của mô hình đạt được là 91,71% với 50 vòng lặp để huấn luyện.

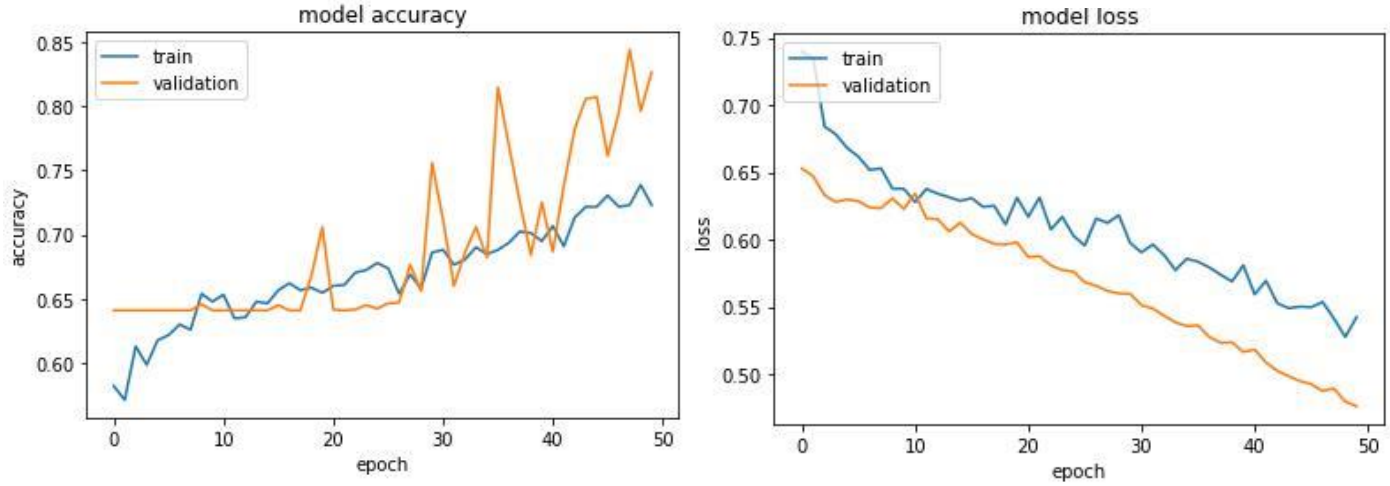
➢ Với hàm kích hoạt ReLU-Sigmoid:



Hình 2.3.4.9-10 : Kết quả mô hình sử dụng hàm kích hoạt ReLU kết hợp Sigmoid

Độ chính xác của mô hình đạt được là 89,24% với 50 vòng lặp để huấn luyện.

➢ Với hàm kích hoạt Sigmoid-Softmax:



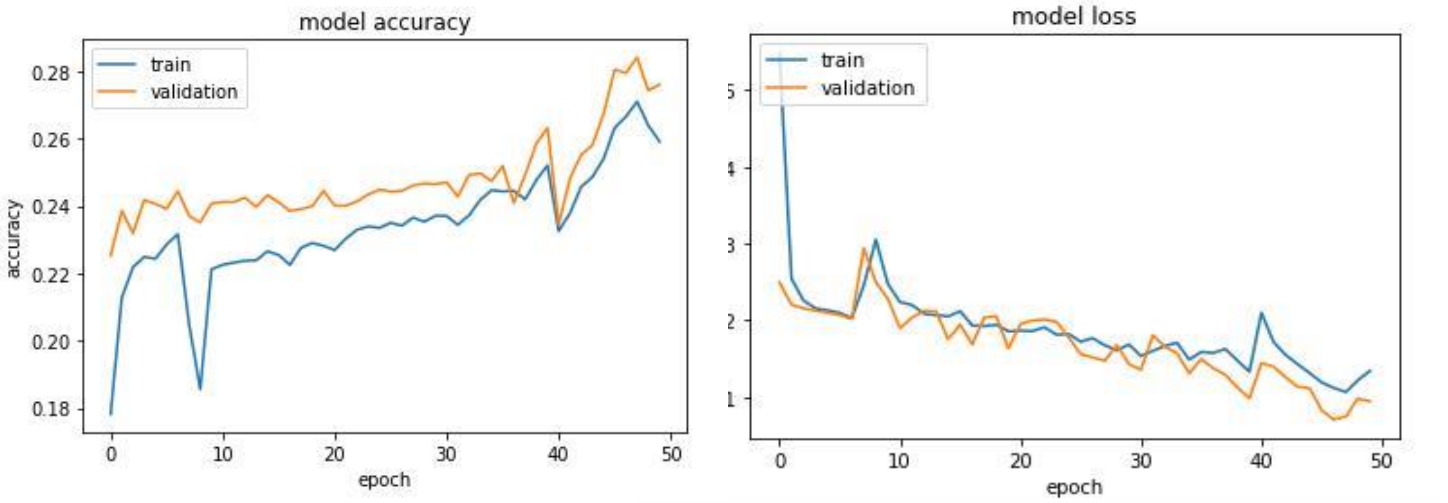
Hình 2.3.4.11-12 : Kết quả mô hình sử dụng hàm kích hoạt Sigmoid kết hợp Softmax

Độ chính xác của mô hình đạt được là 82,66% với 50 vòng lặp để huấn luyện.

Để tiếp tục xác định khả năng của mô hình với bộ dữ liệu lớn và có nhiều nhãn khác nhau ta sử dụng bộ dữ liệu Text Emotion với 13 nhãn.

❖ Bộ dữ liệu Text Emotion :

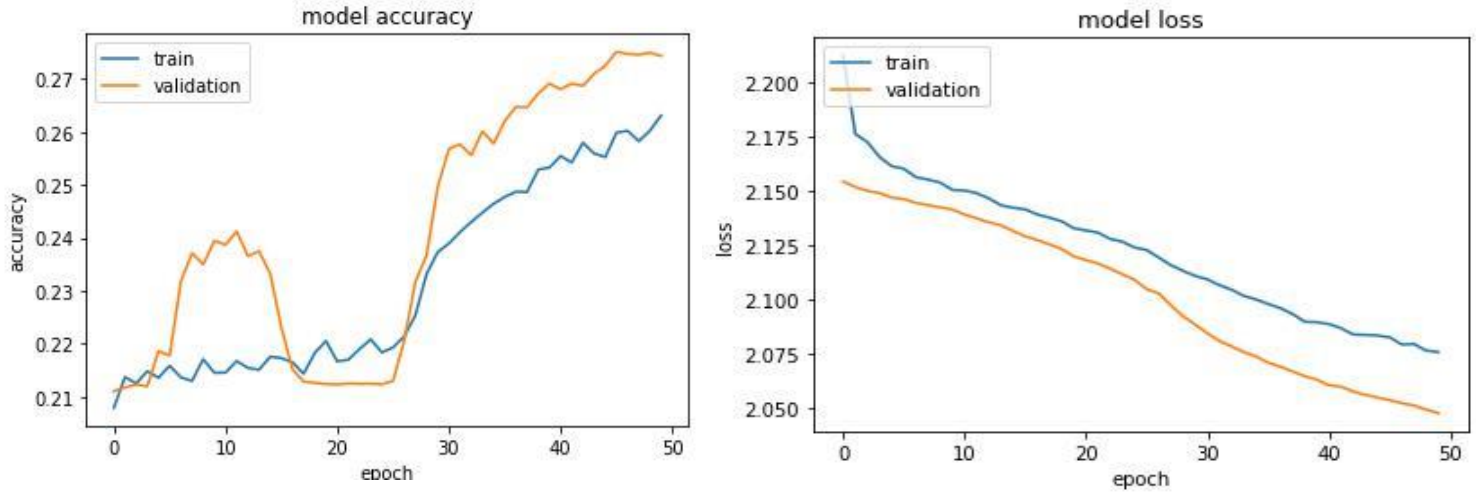
➢ Với hàm kích hoạt ReLu:



Hình 2.3.4.12-13 : Kết quả mô hình sử dụng hàm kích hoạt ReLU

Độ chính xác của mô hình đạt được là 27,60% với 50 vòng lặp để huấn luyện.

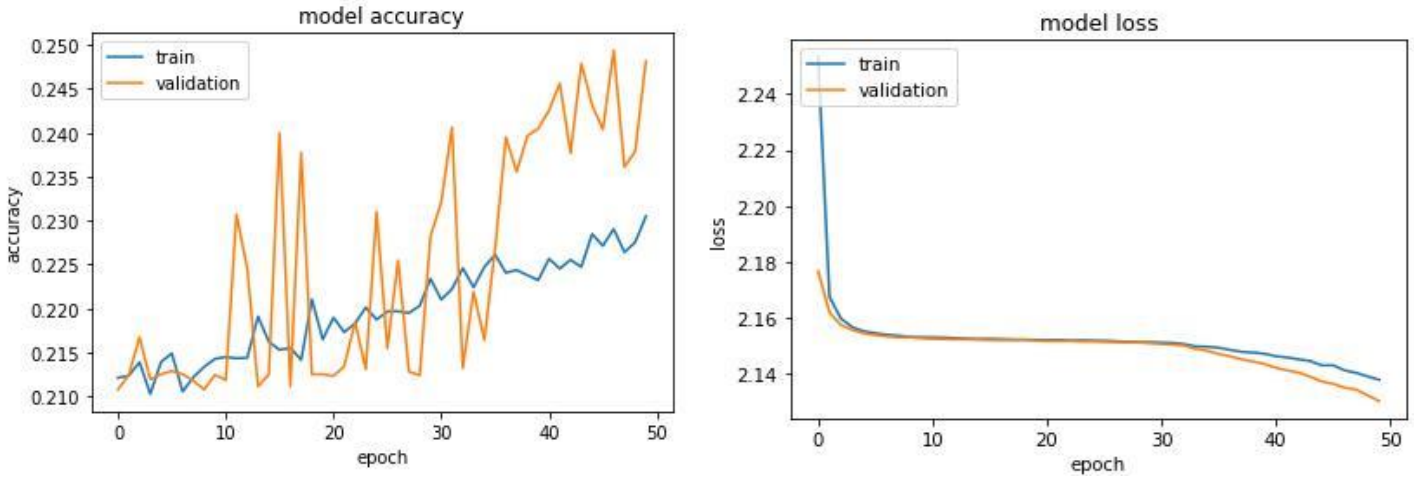
➢ Với hàm kích hoạt Sigmoid:



Hình 2.3.4.13-14 : Kết quả mô hình sử dụng hàm kích hoạt ReLU

Độ chính xác của mô hình đạt được là 27,43% với 50 vòng lặp để huấn luyện.

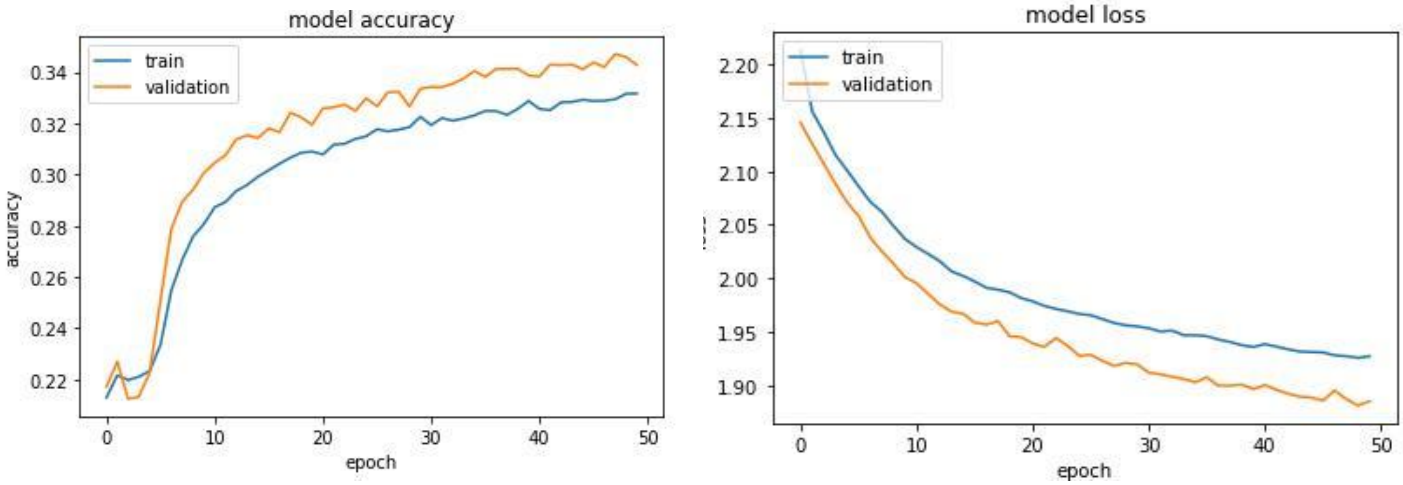
➢ Với hàm kích hoạt Softmax:



Hình 2.3.4.15-16 : Kết quả mô hình sử dụng hàm kích hoạt Softmax

Độ chính xác của mô hình đạt được là 24,82% với 50 vòng lặp để huấn luyện.

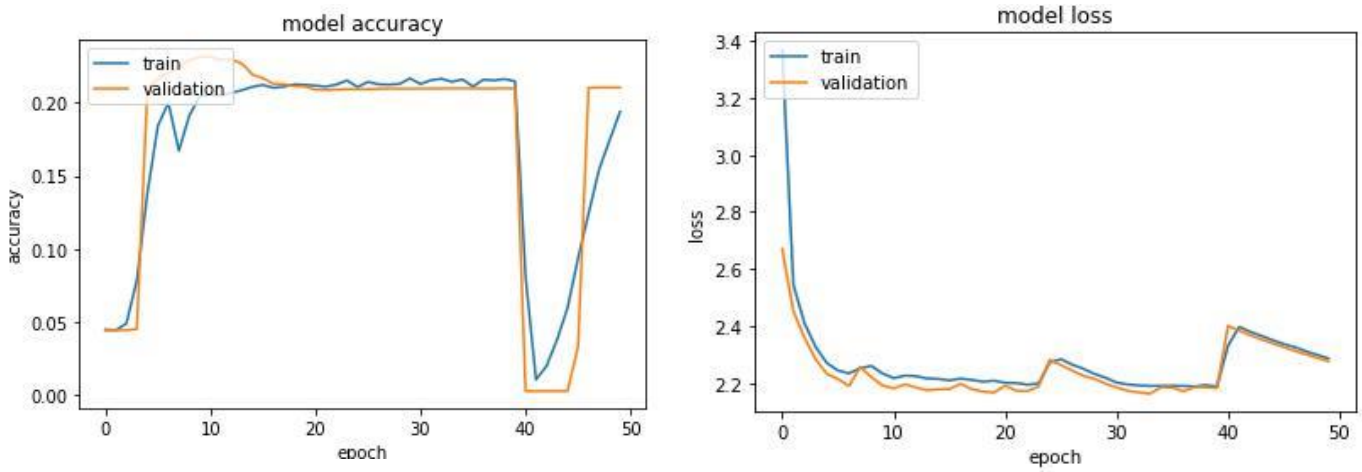
➢ Với hàm kích hoạt Softmax:



Hình 2.3.4.17-18 : Kết quả mô hình sử dụng hàm kích hoạt Softmax

Độ chính xác của mô hình đạt được là 34,28% với 50 vòng lặp để huấn luyện.

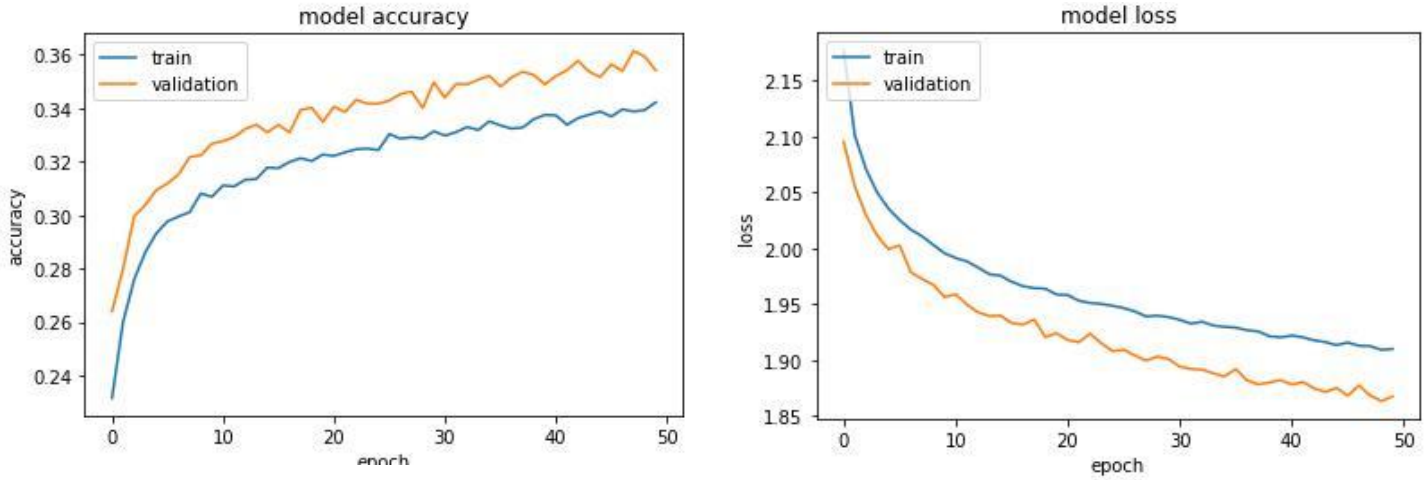
- Với hàm kích hoạt Sigmoid-Softmax:



Hình 2.3.4.19-20 : Kết quả mô hình sử dụng hàm kích hoạt Sigmoid kết hợp Softmax

Độ chính xác của mô hình đạt được là 21,04% với 50 vòng lặp để huấn luyện.

- Với hàm kích hoạt ReLU-Softmax:



Hình 2.3.4.21-22 : Kết quả mô hình sử dụng hàm kích hoạt ReLU kết hợp Softmax

Độ chính xác của mô hình đạt được là 35,41% với 50 vòng lặp để huấn luyện.

2.3.5 Tổng Kết Kết Quả Thực Nghiệm Mạng Nơ-ron Cơ Bản

Qua kết quả thực nghiệm với mạng Nơ-ron lan truyền ngược với cách kết hợp các hàm Sigmoid, reLU và Softmax để huấn luyện với nhau, ta có thể nhận thấy kết quả như sau :

	Sigmoild	ReLU	Softmax	ReLu -Sigmoid	ReLu- Softmax	Sigmoid- Softmax
Khen-Chê	32,11%	70,77%	90,88%	89,24%	91,71%	82,66%
Text Emotion	27,43%	27,60%	24,82%	34,28%	35,41%	21,04%

Bảng 2.3.5 : Kết quả thực nghiệm với mạng Nơ-ron cơ bản

Xét với sự chênh lệch giữa độ chính xác khi xử dụng các hàm kích hoạt khác nhau có ảnh hưởng lớn đến quá trình huấn luyện dữ liệu và phân loại nhãn. Đối với chỉ sử dụng một hàm kích hoạt duy nhất trong toàn bộ quá trình có thể làm giảm độ chính xác cuối cùng của mạng Nơ-ron.

Với sự kết hợp giữa các hàm kích hoạt để chuẩn hóa dữ liệu trong mạng, kết quả của việc huấn luyện có thể đạt được sẽ cao hơn so với chỉ dùng một hàm duy nhất. Tuy nhiên, nếu kết hợp không phù hợp cũng sẽ có thể làm giảm nghiêm trọng kết quả mà ta đạt tới.

Ta còn thấy được, đối với lượng dữ liệu câu nhỏ và ít nhãn cần phân loại như bộ Khen-Chê thì độ chính xác khi được phân loại cao nhất đạt tới hơn 90%. Còn đối với lượng dữ liệu câu lớn và có nhiều nhãn cần phân loại như bộ Text Emotion thì độ chính xác cao nhất chỉ đạt được khoảng 35%, chưa đạt được tới độ chính xác mà ta mong muốn.

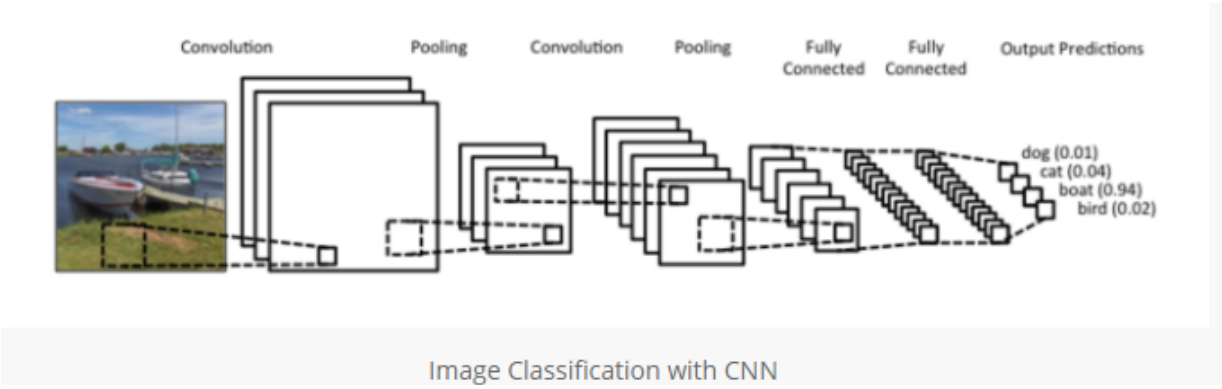
Vì vậy, ta cần tìm một phương thức để cải thiện được độ chính xác của mô hình huấn luyện.

CHƯƠNG III: MẠNG NƠ-RON TÍCH CHẬP

3.1 Giới Thiệu Về Mạng Nơ-ron Tích Chập (Convolutional Neural Network - CNNs)

Là một trong những mô hình Deep Learning tiên tiến có kiến trúc bao gồm nhiều tầng có chức năng khác nhau trong đó tầng chính hoạt động thông qua cơ chế Convolutional (Tích chập). Trong suốt quá trình huấn luyện, mạng nơ-ron tích chập sẽ tự động học được các thông số cho các bộ lọc, tương ứng là các đặc trưng theo từng cấp độ khác nhau.

Mạng nơ-ron tích chập có rất nhiều ứng dụng khác nhau, nhưng tiêu biểu nhất vẫn là được sử dụng trong xử lý ảnh để làm mờ, làm nhiễu và phân tích các đặc trưng của ảnh xử dụng cho quá trình huấn luyện.Ngoài ra, ta cũng có thể sử dụng nó cho việc phân tích các đặc trưng của dữ liệu khác.

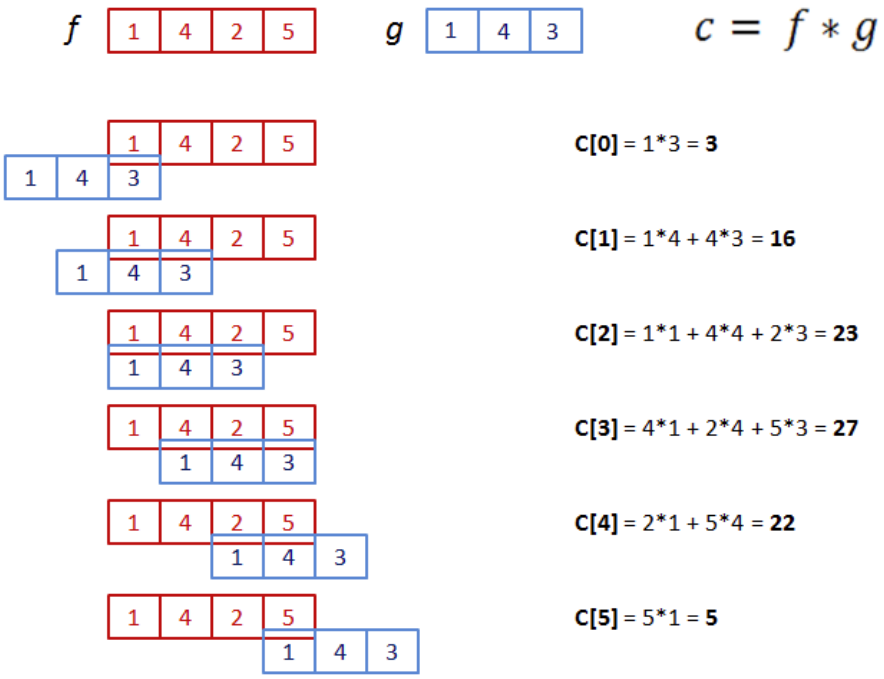


Hình 3.1.1 :Minh họa phân loại ảnh với mạng nơ-ron tích chập

3.2 Cơ Chế Của Mạng Nơ-ron Tích Chập

Sự đặc biệt của mạng nơ-ron tích chập là nằm ở quá trình tích chập.Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số. Nhờ vào nguyên lý biến đổi và phân tích thông tin của nó, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số cùng nhiều ứng dụng khác.

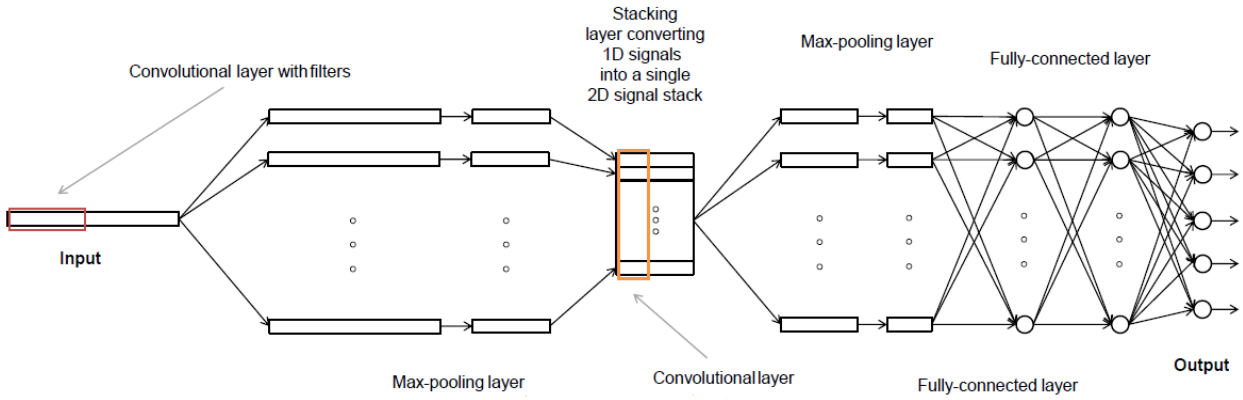
Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận.



Hình 3.2.1 : Cơ chế nhân chập của mạng nơ-ron tích chập

Như hình trên với ma trận f ta thực hiện nhân lần lượt tương ứng với của sổ trượt g, trải qua quá trình trượt ta sẽ được một ma trận mới.Với các cửa sổ trượt khác nhau, ta có thể có được các đặc điểm khác nhau của dữ liệu để tiến hành huấn luyện.

Quá trình thực hiện của mạng nơ-ron tích chập cũng giống như mạng nơ-ron cơ bản nhưng được thêm vào các lớp tích chập và các lớp maxpooling(phương pháp chọn lọc lấy giá trị lớn nhất trong 1 lần trượt để tối ưu hóa giá trị đặc trưng) để phân tích dữ liệu được tối ưu.Từ đó cải thiện hiệu suất huấn luyện được tốt hơn.



Hình 3.2.2 : Quá trình huấn luyện của mạng nơ-ron tích chập

3.3 Thực Nghiệm Với Mạng Nơ-ron Tích Chập

Đối với mạng Nơ-ron tích chập ta vẫn có thể sử dụng một cách đơn giản dựa vào thư viện keras bằng cách thêm vào một số phần mở rộng như conv1d, maxpooling1D,...

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation
from keras.layers import Conv1D, GlobalAveragePooling1D, MaxPooling1D
```

Hình 3.3.1 : Thư viện của mạng nơ ron tích chập

Sau đó, ở phần mô hình ta thêm vào một số lớp tích chập để phân tích dữ liệu đầu vào và hàm maxpooling để tối ưu hóa dữ liệu để có thể chuyển tiếp tới mạng nơ-ron thực hiện quá trình phân loại.

```
# create model
model = Sequential()
model.add(Conv1D(32, 3, activation='relu', input_shape=(300, 1) ))
model.add(Conv1D(32, 3, activation='relu') )
model.add(MaxPooling1D())

model.add(Conv1D(64, 3 activation='relu', strides=1) )
model.add(Conv1D(64, 3, activation='relu', strides=1) )
model.add(MaxPooling1D())


model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(13, activation='sigmoid'))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

Hình 3.3.2 : Mô hình của mạng nơ ron tích chập với thư viện keras

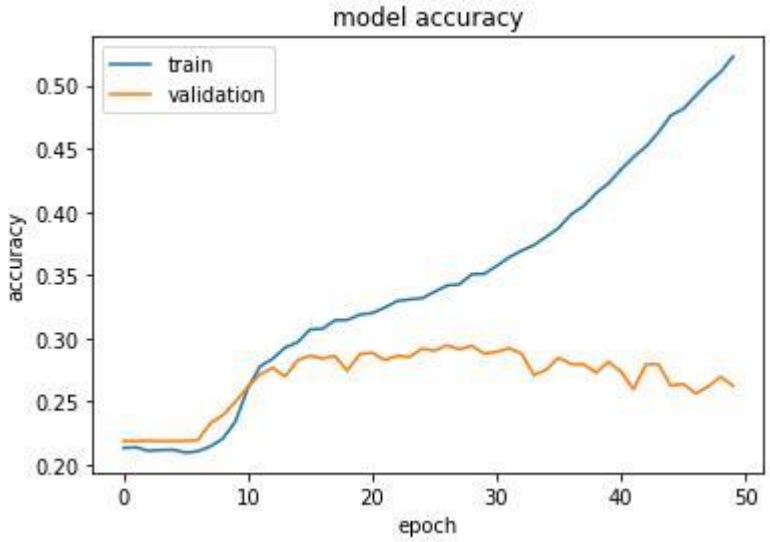
Với cách phân tích dữ liệu này ta có thể làm cho các đặc trưng của dữ liệu được thể hiện ra trước khi chuyển đến mạng nơ-ron.

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 298, 32)	128
conv1d_2 (Conv1D)	(None, 296, 32)	3104
max_pooling1d_1 (MaxPooling1	(None, 148, 32)	0
conv1d_3 (Conv1D)	(None, 146, 64)	6208
conv1d_4 (Conv1D)	(None, 144, 64)	12352
max_pooling1d_2 (MaxPooling1	(None, 72, 64)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 200)	921800
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 13)	2613

Total params: 946,205
Trainable params: 946,205
Non-trainable params: 0

Hình 3.3.3 : Đặc trưng phân tích theo từng giai đoạn của mạng nơ ron tích chập

Thử nghiệm với bộ dữ liệu Text Emotion, ta có thể thấy được kết quả khả quan của việc phân loại dữ liệu so với chỉ dùng mạng nơ ron cơ bản.



Hình 3.3.3 : Kết quả thực nghiệm với mạng Nơ-ron tích chập với hàm kích hoạt ReLu kết hợp Sigmoid

Thực nghiệm cho thấy sau khi sử dụng thêm cách lớp tích chập khả năng phân loại của mạng nơ-ron đã tăng đến 63,87%, gấp đôi so với chỉ dùng,mạng nơ-ron cơ bản.

3.4 Tổng Kết

- **Kết Luận :** Với sự hỗ trợ từ các lớp tích chập của mô hình mạng nơ-ron tích chập, ta đã có thể cái thiện được hiệu suất huấn luyện và tăng cường độ chính xác khi phân loại của mạng nơ-ron lan truyền ngược cơ bản. Độ chính xác khi phân loại các bộ dữ liệu lớn và có nhiều nhãn đã đạt tới được ngưỡng mong muốn của quá trình thực nghiệm.

Tuy vậy, vẫn còn nhiều lỗi trong quá trình sử dụng mạng nơ-ron tích chập mà ta cần phải nghiên cứu thêm. Như số lượng tích chập cần thiết cho những bộ dữ liệu để không làm chương trình chạy chậm gây lãng phí thời gian và tài nguyên, hay mặt nạ trượt cần thiết để có thể phân tích đặc điểm của dữ liệu một cách tối ưu nhất, cơ chế điều chỉnh trọng số cho mô hình,...

Cuối cùng quan trọng nhất là cách chọn lớp tích chập phù hợp cho mô hình để tránh hiện tượng Overfitting - hiện tượng mô hình tìm được quá khớp với dữ liệu huấn luyện, có thể dẫn đến việc dự đoán nhầm nhiễu làm chất lượng mô hình và phân loại không còn tốt trên dữ liệu kiểm thử nữa.

- **Hướng Phát Triển :**

  - Tìm hiểu thêm các phương thức cải thiện khả năng phân loại của mạng nơ-ron lan truyền ngược.
  - Tìm hiểu cách xây dựng tích chập hỗ trợ cho mô hình trong mạng nơ-ron tích chập.
  - Xây dựng các bộ dữ liệu phù hợp cho phân loại.

TÀI LIỆU THAM KHẢO

1. Lê Thị Thu Hằng, Luận Văn "Nghiên cứu về mạng neural tích chập và ứng dụng ", 2016
2. Nguyễn Văn Chức, Bài Viết "Tổng quan về Mạng Nơ-ron (Neural Network)", 2008
3. Bài Viết "Lý thuyết mạng Nơ-ron"
4. David Kriesel, Sách "Neural Networks"
5. Stanford, bài Viết "Convolutional Neural Network"

Dữ Liệu Huấn Luyện

A. Text Emotion
https://github.com/MohMehrnia/TextBaseEmotionDetectionWithEnsembleMethod/tree/master/Dataset

Chuyển Đổi Dữ Liệu

B. https://github.com/phongnt570/UETsegmenter
C. https://github.com/stanfordnlp/GloVe