

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

**SỬ DỤNG GIẢI THUẬT DI TRUYỀN ĐỂ XÁC ĐỊNH TRỌNG
SỐ CHO MẠNG NƠ RON TẾ BÀO BẠC HAI**

Sinh viên thực hiện: PHẠM DUY TUẤN

Mã sinh viên: B19DCCN618

Lớp: D19CNPM02

Khóa: 2019 - 2024

Hệ: Đại học chính quy

Giảng viên hướng dẫn: PGS. TS. NGUYỄN QUANG HOAN

Hà Nội, tháng 11/2023

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

**SỬ DỤNG GIẢI THUẬT DI TRUYỀN ĐỂ XÁC ĐỊNH TRỌNG
SỐ CHO MẠNG NƠ RON TẾ BÀO BẠC HAI**

Sinh viên thực hiện: PHẠM DUY TUẤN

Mã sinh viên: B19DCCN618

Lớp: D19CNPM02

Khóa: 2019 - 2024

Hệ: Đại học chính quy

Giảng viên hướng dẫn: PGS. TS. NGUYỄN QUANG HOAN

Hà Nội, tháng 11/2023

LỜI CẢM ƠN

Lời đầu tiên, em xin bày tỏ lòng biết ơn sâu sắc đến PGS. Nguyễn Quang Hoan – thầy hướng dẫn của em. Cảm ơn thầy vì đã luôn tận tình hướng dẫn chỉ dạy cho em trong suốt quá trình làm đồ án.

Đồng thời, em xin gửi lời cảm ơn chân thành đến NCS. Dương Đức Anh – người đã cùng em nghiên cứu về đề tài “Sử dụng giải thuật di truyền để xác định trọng số cho mạng nơ ron tế bào bậc hai”.

Để có thể hoàn thành được đồ án này, không thể không kể đến công sức của các thầy cô trong khoa Công nghệ thông tin khi đã tạo điều kiện thuận lợi cho em trong quá trình học tập và thực hiện đồ án.

Cuối cùng, em xin cảm ơn gia đình, bạn bè vì đã luôn bên cạnh động viên, hỗ trợ em trong quá trình hoàn thiện đồ án.

Tuy em đã có gắng hoàn thành đồ án trong khả năng cho phép nhưng không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, nhận xét cũng như góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2023

Sinh viên thực hiện

Phạm Duy Tuấn

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(Của giảng viên hướng dẫn)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:.....(Bằng chữ:.....)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm tốt nghiệp?

Hà Nội, ngày....tháng....năm 202....

GIẢNG VIÊN HƯỚNG DẪN

NGUYỄN CÔNG HOAN

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(Của giảng viên phản biện)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm:(Bằng chữ:)

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm tốt nghiệp?

Hà Nội, ngày....tháng....năm 202....

GIẢNG VIÊN PHẢN BIỆN

MỤC LỤC

LỜI CẢM ƠN	i
NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM (Của giảng viên hướng dẫn).....	ii
NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM (Của giảng viên phản biện)	iii
MỤC LỤC	iv
DANH MỤC TỪ VIẾT TẮT.....	vi
DANH MỤC CÁC BẢNG	vii
DANH MỤC HÌNH VẼ	viii
MỞ ĐẦU	1
CHƯƠNG I: TỔNG QUAN VỀ MẠNG NƠ RON TẾ BÀO.....	3
1.1. Tổng quan về mạng nơ ron	3
1.1.1. Định nghĩa và cấu trúc.....	3
1.1.2. Phân loại mạng nơ ron.....	3
1.1.3. Luật học trong mạng nơ ron.....	5
1.2. Mạng nơ ron tế bào chuẩn	6
1.2.1. Cấu trúc của mạng nơ ron tế bào chuẩn.....	6
1.2.2. Luật học trong mạng nơ ron tế bào chuẩn.....	8
1.3. Mạng nơ ron tế bào bậc hai.....	8
1.3.1. Cấu trúc của mạng nơ ron tế bào bậc hai.....	9
1.3.2. Luật học trong mạng nơ ron tế bào bậc hai.....	14
1.4. Kết chương.....	15
CHƯƠNG II: PHƯƠNG PHÁP XÁC ĐỊNH TRỌNG SỐ CỦA MẠNG NƠ RON TẾ BÀO BẬC HAI BẰNG GIẢI THUẬT DI TRUYỀN	16
2.1. Giải thuật di truyền	16
2.2. Thuật toán xác định trọng số của mạng nơ ron tế bào bậc hai bằng giải thuật di truyền.....	20

2.3. Phương pháp tính toán ma trận trạng thái từ phương trình trạng thái của SOCeNNs	24
2.3.1. Phương trình vi phân thường ODE	24
2.3.2. Phương pháp đa bước tuyến tính	25
2.3.3. Phương pháp Adams ngầm	26
2.3.4. Phương pháp Newton	27
2.4. Độ phức tạp của thuật toán GASOCeNNs	28
2.5. Kết chương.....	29
CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ.....	30
3.1. Bài toán tách biên ảnh và ứng dụng	30
3.2. Bộ dữ liệu thực nghiệm	31
3.3. Công cụ thực nghiệm.....	32
3.3.1. Ngôn ngữ lập trình.....	32
3.3.2. Công cụ lập trình	32
3.3.3. Thư viện sử dụng	33
3.4. Các tiêu chí đánh giá	34
3.5. Thực nghiệm và đánh giá.....	34
3.4.1. Bộ dữ liệu	35
3.4.2. Bộ trọng số ban đầu	37
3.4.3. Giá trị khởi đầu của phương trình vi phân.....	40
3.4.4. Time step	41
3.4.5. Số lần lai	43
3.4.6. Số lần chạy vòng lặp lớn	44
3.6. Kết chương.....	46
KẾT LUẬN	47
TÀI LIỆU THAM KHẢO	48

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu hoặc từ viết tắt	Diễn giải (tiếng Việt)
1	AI	Trí tuệ nhân tạo
2	ANN	Mạng nơ ron nhân tạo
3	CeNN	Mạng nơ ron tế bào
4	SOCeNNs	Mạng nơ ron tế bào bậc hai
5	GA	Giải thuật di truyền
6	FNN	Mạng nơ ron truyền thẳng
7	RNN	Mạng nơ ron phản hồi
8	ODE	Phương trình vi phân thường

DANH MỤC CÁC BẢNG

Bảng 1.1. Bảng phân bố tế bào lân cận của tế bào trung tâm $C(i,j)$, tương ứng $r=1$	9
Bảng 1.2. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $(i-1,j-1)$	11
Bảng 1.3. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i-1,j)$	11
Bảng 1.4. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i-1,j+1)$	11
Bảng 1.5. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j-1)$	12
Bảng 1.6. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j)$	12
Bảng 1.7. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j+1)$	12
Bảng 1.8. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j-1)$	12
Bảng 1.9. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j)$	13
Bảng 1.10. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j+1)$	13
Bảng 3.1. Kết quả chạy với các bộ dữ liệu khác nhau	35
Bảng 3.2. Kết quả tách biên với các bộ dữ liệu khác nhau.....	36
Bảng 3.3. Kết quả chạy với các bộ trọng số khác nhau.....	38
Bảng 3.4. Kết quả tách biên với các bộ trọng số khác nhau.....	38
Bảng 3.5. Kết quả chạy với các bộ giá trị khởi đầu khác nhau.....	40
Bảng 3.6. Kết quả tách biên với các bộ giá trị khởi đầu khác nhau.....	40
Bảng 3.7. Kết quả chạy với các bước nhảy khác nhau.....	41
Bảng 3.8. Kết quả tách biên với các bước nhảy khác nhau	42
Bảng 3.9. Kết quả chạy với số lần lai khác nhau	43
Bảng 3.10. Kết quả tách biên với số lần lai khác nhau	43
Bảng 3.11. Kết quả chạy với số lần chạy khác nhau.....	44
Bảng 3.12. Kết quả tách biên với số lần chạy khác nhau	45

DANH MỤC HÌNH VẼ

Hình 1.1. Mô tả cấu trúc của mạng nơ ron	3
Hình 1.2. Mô hình mạng nơ ron truyền thẳng	4
Hình 1.3. Mô hình mạng nơ ron phản hồi	5
Hình 1.4. Mô tả mạng nơ ron tế bào chuẩn kích thước $M \times N$	6
Hình 1.5. Sơ đồ khối mạng nơ ron tế bào chuẩn	7
Hình 1.6. Hàm tương tác đầu ra của nơ ron tế bào chuẩn	7
Hình 1.7. Các trọng số của mạng nơ ron tế bào chuẩn	8
Hình 1.8. Cấu trúc tổng quát mạng nơ ron tế bào bậc hai	10
Hình 1.9. Sơ đồ cấu trúc CNNs bậc hai quy đổi	14
Hình 2.1. Lưu đồ thuật toán giải thuật di truyền	19
Hình 2.2: Lưu đồ khái quát thuật toán GA cho SOCeNNs	23
Hình 3.1. Mô tả ảnh đầu vào	31
Hình 3.2. Mô tả đầu ra mong muốn	32

MỞ ĐẦU

Trí tuệ nhân tạo (AI) đang là lĩnh vực được quan tâm nhiều nhất trong thời đại ngày nay. Trong đó, mạng nơ ron – một kiến trúc mô phỏng bộ não con người – đang được chú trọng nhiều hơn cả. Dựa theo hướng của luồng tín hiệu, mạng nơ ron sẽ chia làm hai loại gồm mạng nơ ron truyền thẳng và mạng nơ ron phản hồi (hay còn gọi là mạng nơ ron truyền ngược hoặc mạng nơ ron hồi quy). Ngoài ra, mạng nơ ron còn được chia thành mạng nơ ron truyền thống (còn gọi là mạng nơ ron kinh điển) và mạng học sâu (được phát triển từ mạng nơ ron truyền thống, còn được gọi là mạng nơ ron hiện đại). Đồ án này sẽ tập trung vào một mô hình mạng nơ ron truyền thống phản hồi tên là mạng nơ ron tế bào (CeNN).

Mạng nơ ron tế bào được L. Chua và L. Yang đề xuất năm 1988. Từ đó cho tới nay, công bố của các nhà nghiên cứu cho thấy mạng nơ ron tế bào có thể được áp dụng tốt cho các bài toán nhận dạng và xử lý ảnh tốc độ cao.

Năm 2020, ở Việt Nam, thầy Nguyễn Tài Tuyên cùng với các cộng sự đã phát triển mô hình mạng nơ ron tế bào bậc hai (hay còn gọi là mạng nơ ron tế bào bậc cao) (viết tắt là SOCeNNs) từ mô hình mạng nơ ron tế bào của L. Chua. Trong công bố của mình, thầy Tuyên đã trình bày về tính ổn định và tính ứng dụng của mô hình SOCeNNs nhưng chưa đưa ra phương pháp cụ thể để xác định bộ trọng số phù hợp cho mô hình.

Đổi lại, trong khoa học máy tính, có một giải thuật thường được sử dụng để tối ưu kết quả cho các bài toán tìm kiếm. Đó là giải thuật di truyền (GA)

Mục tiêu của đồ án này là sử dụng giải thuật di truyền để xác định trọng số cho mạng nơ ron tế bào bậc hai rồi sử dụng mô hình đó cho bài toán tách biên ảnh.

Dựa theo mục tiêu đó thì đồ án sẽ có cấu trúc như sau:

- **Chương 1: Tổng quan về mạng nơ ron tế bào.**

Trong chương này, lần lượt trình bày chi tiết các khái niệm về mạng nơ ron, mạng nơ ron tế bào, mạng nơ ron tế bào bậc hai và các công bố liên quan.

- **Chương 2: Phương pháp xác định trọng số của mạng nơ ron tế bào bậc hai bằng giải thuật di truyền.**

Chương này tập trung chủ yếu vào việc trình bày chi tiết giải thuật di truyền và phương pháp xác định trọng số của mạng nơ ron tế bào bậc hai bằng giải thuật di truyền.

- **Chương 3: Thực nghiệm và đánh giá**

Trong chương này, xây dựng thuật toán và lập trình chương trình để kiểm nghiệm phương pháp đã trình bày ở chương hai. Rồi từ đó đánh giá kết quả thu được.

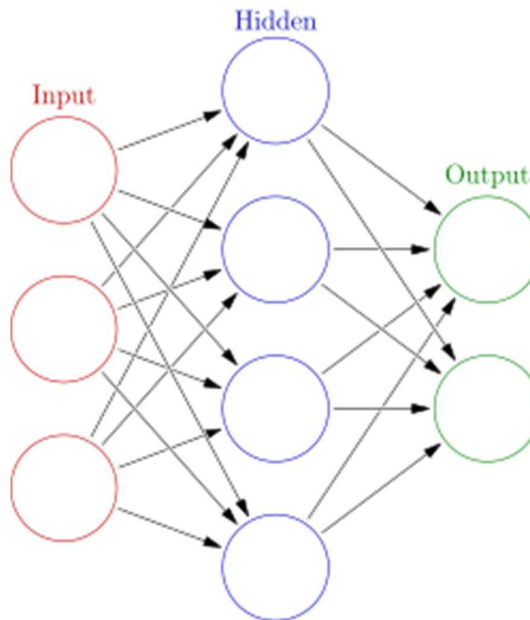
CHƯƠNG I: TỔNG QUAN VỀ MẠNG NƠ RON TẾ BÀO

Nội dung của chương này sẽ tập trung tìm hiểu về mạng nơ ron, cách phân loại cũng như cấu trúc của từng loại mạng nơ ron. Rồi từ đó đi sâu vào chi tiết về mạng nơ ron tế bào và cuối cùng là mạng nơ ron tế bào bậc hai. Ta sẽ thống nhất gọi tên mạng nơ ron tế bào do L. Chua tạo ra là mạng nơ ron tế bào bậc nhất (CeNN) còn mạng nơ ron tế bào do thầy Tuyên phát triển từ mạng của L. Chua là mạng nơ ron tế bào bậc hai (SOCeNNs).

1.1. Tổng quan về mạng nơ ron

1.1.1. Định nghĩa và cấu trúc

Mạng nơ ron nhân tạo (ANN) (còn được gọi là mạng nơ-ron) là một nhánh của các mô hình học máy được xây dựng bằng cách mô phỏng mạng lưới thần kinh của bộ não con người.



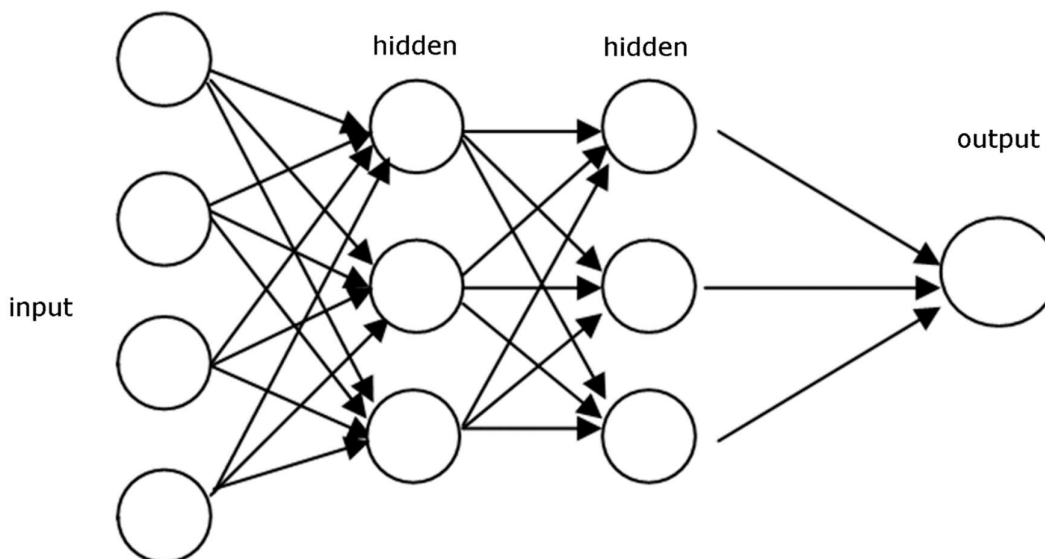
Hình 1.1. Mô tả cấu trúc của mạng nơ ron

Hình trên mô tả cấu trúc của mạng nơ ron. Cụ thể, mỗi nút tròn đại diện cho một tế bào nơ ron và một mũi tên đại diện cho một kết nối từ đầu ra của một tế bào nơ ron này đến đầu vào của một tế bào nơ ron khác.

1.1.2. Phân loại mạng nơ ron

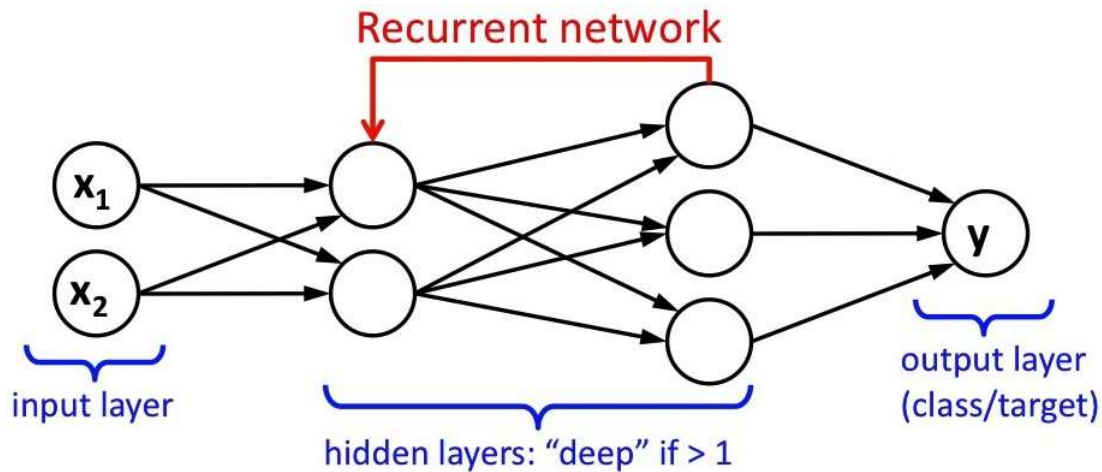
Dựa theo hướng của luồng tín hiệu, mạng nơ ron được phân làm hai loại là mạng nơ ron ruyền thẳng và mạng nơ ron phản hồi.

Trong mạng nơ ron truyền thẳng (FNN), luồng dữ liệu là một chiều từ lớp trước đến lớp sau mà không có kết nối nào tồn tại giữa các nơ ron trong cùng một lớp. Mỗi tế bào của FNN chỉ nhận đầu vào từ lớp ngay trước đó và truyền đầu ra cho lớp sau đó. Không có chu trình (chu kỳ) hoặc vòng lặp trong FNN. Về phương diện học tập hệ thống, FNN thuận tiện cho việc lập trình lại và có thể xử lý các câu hỏi phi tuyến tính.



Hình 1.2. Mô hình mạng nơ ron truyền thẳng

Trái lại với mạng nơ ron truyền thẳng, một nơ ron trong mạng nơ ron phản hồi (còn được gọi là mạng nơ ron truyền ngược hay mạng nơ ron hồi quy) có thể truyền dữ liệu đầu ra của nó cho các nơ ron khác trong cùng hay các lớp trước nó, hoặc cũng có thể truyền lại dữ liệu cho chính nơ ron đó. Các thuật toán sẽ được điều chỉnh đồng thời bởi các tín hiệu từ các nơ ron khác dựa trên tập dữ liệu tiền tri thức. Việc hiệu chuẩn lặp đi lặp lại góp phần vào độ mạnh và độ chính xác tuyệt vời của mạng nơ ron nhân tạo. Mạng nơ ron phản hồi thường được sử dụng để phân tích hình ảnh, chẩn đoán và dự đoán kết quả.



Hình 1.3. Mô hình mạng nơ ron phản hồi

1.1.3. Luật học trong mạng nơ ron

Luật học trong mạng nơ ron nhân tạo có thể chia làm hai loại:

- *Học cấu trúc*: Học cấu trúc trong mạng nơ ron là xác định số lớp (số tầng) kết nối và số các phần tử nơ ron trong mỗi lớp. Trong đồ án không đề cập đến vấn đề này.
- *Học tham số*: Học tham số có nghĩa là tìm giá trị của bộ trọng số trong mạng nơ ron gồm học giám sát, học không giám sát và học tăng cường.

Cụ thể như sau:

- *Học không giám sát*: học không giám sát là học từ các cặp mẫu vào/ra. Luật học Hebb là luật học không giám sát thường được áp dụng trong mạng Hopfield, BAM và nơ ron tế bào (sẽ đề cập ở phần sau của đồ án).
- *Học có giám sát*: học có giám sát là học từ các nhãn (tức là từ các đầu ra mong muốn). Khi đó nhãn đóng vai trò là một “giáo viên” nhằm giám sát quá trình học. Ví dụ như luật học Perceptron hồi quy là luật học có giám sát.
- *Học củng cố*: học củng cố là học sử dụng các thông tin trái ngược nhau được coi là “phần thưởng” hoặc “hình phạt” kết hợp với các phản hồi thực tế để củng cố các loại tín hiệu khác nhau. Hiện nay, phương pháp học củng cố ngày càng trở nên phổ biến trong việc giải các bài toán trí tuệ nhân tạo hiện đại. Trong chương hai của đồ án, sinh viên sử dụng giải thuật di truyền và giải thuật lai là ứng dụng của học củng cố.

1.2. Mạng nơ ron tế bào chuẩn

1.2.1. Cấu trúc của mạng nơ ron tế bào chuẩn

Mạng nơ ron tế bào thực chất là một mảng các tế bào (cell) với kích thước $M \times N$. Trong đó $C(i,j)$ là một tế bào của mạng nơ ron tế bào với $i=1,...,M$; $j=1,...,N$.

Láng giềng r của một tế bào $C(i, j)$ trong một mạng nơ ron tế bào được xác định bởi biểu thức sau:

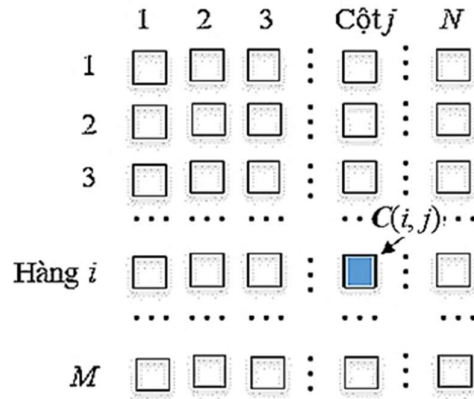
$$N_r(i, j) = \{C(k, l) \mid \max\{|k - i|, |l - j|\} \leq r\} \quad (1)$$

$$\text{với } 1 \leq k \leq M; 1 \leq l \leq N$$

Trong đó r là bán kính lân cận của một tế bào, r có giá trị nguyên dương

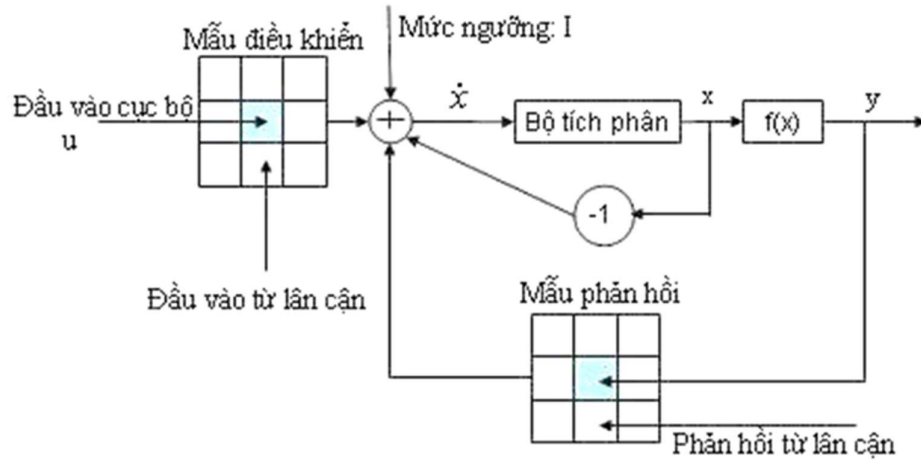
Ở đây ta chỉ xét $r=1$, tức là mỗi nơ ron $C(i,j)$ sẽ có 8 nơ ron láng giềng là $C(i-1,j)$, $C(i-1,j-1)$, $C(i-1,j+1)$, $C(i,j-1)$, $C(i,j+1)$, $C(i+1,j)$, $C(i+1,j-1)$, $C(i+1,j+1)$.

Một mạng nơ ron tế bào chuẩn kích thước $M \times N$ chiều được biểu diễn theo hình 1.4, với vị trí $C(i,j)$ đặt trong hệ tọa độ Đề các 2 chiều, thể hiện tế bào tại cột i và hàng j trong mạng.



Hình 1.4. Mô tả mạng nơ ron tế bào chuẩn kích thước $M \times N$

Theo như mô tả của L. Chua, mô hình luồng tín hiệu vào ra của từng nơ ron trong mạng được biểu diễn bằng sơ đồ dưới đây:



Hình 1.5. Sơ đồ khối mạng nơ ron tế bào chuẩn

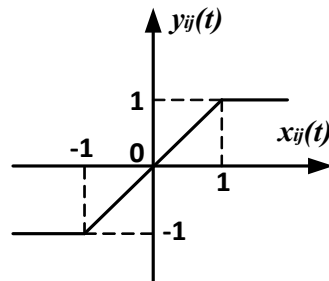
Trong đó, mẫu điều khiển và mẫu phản hồi lần lượt là ma trận điều khiển B và ma trận phản hồi A. Hai ma trận này đều là ma trận 3x3. Ma trận đầu vào u, ma trận đầu ra y, ma trận trạng thái x và mức ngưỡng I.

Phương trình trạng thái:

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{(k,l)} \mathbf{A}(i,j;k,l)y_{kl}(t) + \sum_{(k,l)} \mathbf{B}(i,j;k,l)u_{kl} + \mathbf{I} \quad (2)$$

Hàm đầu ra của mạng nơ ron tế bào như sau (hàm bão hòa):

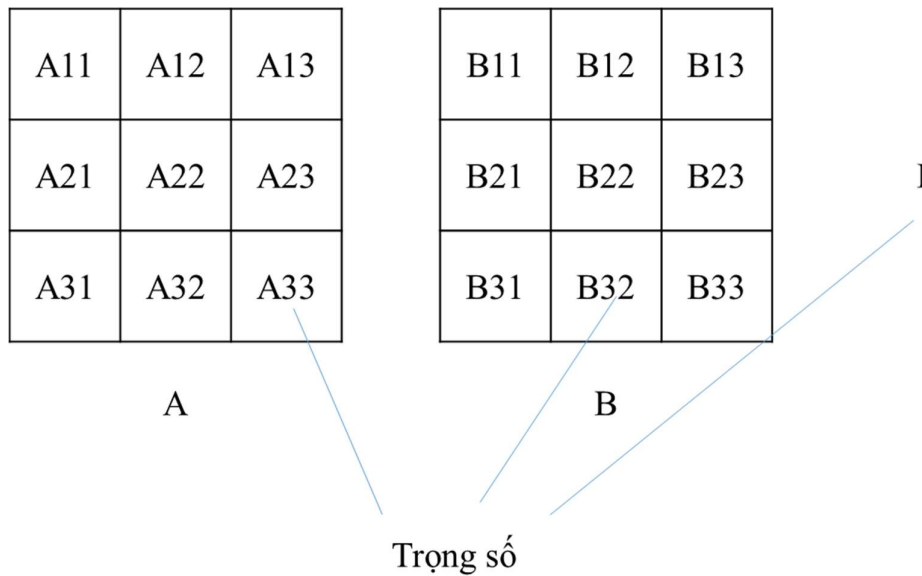
$$y_{ij}(t) = \frac{1}{2} \left(\left| x_{ij}(t) + 1 \right| - \left| x_{ij}(t) - 1 \right| \right) \quad (3)$$



Hình 1.6. Hàm tương tác đầu ra của nơ ron tế bào chuẩn

1.2.2. Luật học trong mạng nơ ron tế bào chuẩn

Dữ liệu đầu ra của mạng nơ ron tế bào chuẩn phụ thuộc vào dữ liệu đầu vào, ma trận điều khiển B, ma trận phản hồi A và mức ngưỡng I. Vì vậy, quá trình học của mạng nơ ron tế bào là quá trình xác định các giá trị của bộ ma trận A, B và mức ngưỡng I từ bộ dữ liệu đầu vào, đầu ra cho trước. Ta gọi các giá trị của ma trận A, B và mức ngưỡng I là các trọng số.



Hình 1.7. Các trọng số của mạng nơ ron tế bào chuẩn

Như vậy, bộ trọng số của CeNN sẽ là $[A_{11}, A_{12}, A_{13}, A_{21}, A_{22}, A_{23}, A_{31}, A_{32}, A_{33}, B_{11}, B_{12}, B_{13}, B_{21}, B_{22}, B_{23}, B_{31}, B_{32}, B_{33}, I]$ với 19 trọng số.

Tuy nhiên, theo L. Chua, để CeNN đạt được sự ổn định thì hai ma trận A, B phải đối xứng tâm. Tức là đối với ma trận A, ta có $A_{11} = A_{33}$, $A_{12} = A_{32}$, $A_{13} = A_{31}$, $A_{21} = A_{23}$. Tương tự với ma trận B, ta có $B_{11} = B_{33}$, $B_{12} = B_{32}$, $B_{13} = B_{31}$, $B_{21} = B_{23}$. Điều này có nghĩa là, thay vì phải tìm tất cả 19 giá trị của bộ trọng số thì ta chỉ cần tìm 5 giá trị của ma trận A là $A_{11}, A_{12}, A_{13}, A_{21}, A_{22}$ và 5 giá trị của ma trận B là $B_{11}, B_{12}, B_{13}, B_{21}, B_{22}$ cùng với mức ngưỡng I. Khi đó, bộ trọng số cần xác định sẽ chỉ còn lại là $[A_{11}, A_{12}, A_{13}, A_{21}, A_{22}, A_{31}, A_{32}, A_{33}, B_{11}, B_{12}, B_{13}, B_{21}, B_{22}, B_{23}, B_{31}, B_{32}, B_{33}, I]$ với 11 trọng số.

1.3. Mạng nơ ron tế bào bậc hai

Mạng nơ ron tế bào bậc hai (SOCeNNs) được đề xuất bởi TS. N. T. Tuyen từ mô hình của CeNN và được phát triển bởi NCS. D. D. Anh.

1.3.1. Cấu trúc của mạng nơ ron tế bào bậc hai

Một tế bào trong CeNNs được định nghĩa là tổng của phép nhân chập giữa các tín hiệu đầu ra và các tín hiệu điều khiển bất kỳ đối với các tế bào lân cận $C(k, l)$ và $C(m, n)$ của tế bào $C(i, j)$, trong đó lân cận của tế bào là tập các tế bào quanh tế bào trung tâm $C(i, j)$ với các bán kính r tương ứng. Bán kính ở đây được hiểu là số lớp kế cận với tế bào trung tâm $C(i, j)$. Khi $r=1$ tức là lớp kế cận gần nhất bao gồm 08 tế bào tương tác cộng với chính nó tạo thành một bộ 09 tế bào gọi là láng giềng hay lân cận của $C(i, j)$.

Bảng 1.1. Bảng phân bố tế bào lân cận của tế bào trung tâm $C(i, j)$, tương ứng $r=1$

$C(i-1, j-1)$	$C(i-1, j)$	$C(i-1, j+1)$
$C(i, j-1)$	$C(i, j)$	$C(i, j+1)$
$C(i+1, j-1)$	$C(i, j+1)$	$C(i+1, j+1)$

Năm 2020, NCS. D.D.Anh cùng cộng sự đã phát triển cấu trúc mạng nơ ron tế bào bậc cao và sử dụng đại diện là mạng nơ ron tế bào bậc hai (SOCeNNs) tại bài báo [1]. Cấu trúc mạng SOCeNNs được xây dựng dựa trên cấu trúc mạng nơ ron tế bào chuẩn (bậc nhất), gồm bộ các trọng số điều khiển đầu vào bậc nhất B1, trọng số phản hồi đầu ra bậc nhất A1, trọng số đầu vào bậc hai B2 và trọng số phản hồi đầu ra bậc hai A2 và trọng số ngưỡng I (Theo hình 1.7). Khi đó, cấu trúc của SOCeNNs bao gồm:

Phương trình trạng thái:

$$\begin{aligned} \frac{dx_{ij}(t)}{dt} = & -x_{ij}(t) + \sum_{(k,l)} \mathbf{A1}(i, j; k, l) y_{kl}(t) + \sum_{(k,l)} \mathbf{B1}(i, j; k, l) u_{kl} + \mathbf{I} + \\ & + \sum_{(k,l),(m,n)} \mathbf{A2}(i, j; k, l; m, n) y_{kl}(t) y_{mn}(t) + \sum_{(k,l),(m,n)} \mathbf{B2}(i, j; k, l; m, n) u_{kl} u_{mn} \end{aligned} \quad (4)$$

Hàm đầu ra:

$$y_{ij}(t) = \frac{1}{2} \left(\left| x_{ij}(t) + 1 \right| - \left| x_{ij}(t) - 1 \right| \right) \quad (5)$$

Trong đó:

i, j : Thể hiện vị trí của tế bào $C(i, j)$ trong SOCeNNs; $i, j \in \mathbb{N}^*$

r : Bán kính lân cận của tế bào $C(i, j)$, chọn $r=1$;

k, l, m, n : Thể hiện vị trí các tế bào lân cận của $C(i, j)$ tương ứng với bán kính lân cận;

$$k, l, m, n \in \mathbb{N}^*$$

$x_{ij}(t)$: Tín hiệu trạng thái của tế bào $C(i, j)$;

$y_{ij}(t)$: Tín hiệu đầu ra của tế bào $C(i, j)$;

u_{ij} : Tín hiệu đầu vào của tế bào $C(i, j)$;

$y_{kl}(t)$: Tín hiệu đầu ra của SOCeNNs;

u_{kl} : Tín hiệu đầu vào của SOCeNNs;

$\mathbf{A1}(i, j; k, l)$: Ma trận trọng số phản hồi thành phần bậc nhất, kích thước (3×3) ;

$\mathbf{A2}(i, j; k, l; m, n)$: Ma trận trọng số phản hồi thành phần bậc hai, kích thước (9×9) ;

$\mathbf{B1}(i, j; k, l)$: Ma trận trọng số đầu vào của thành phần bậc nhất, kích thước (3×3) ;

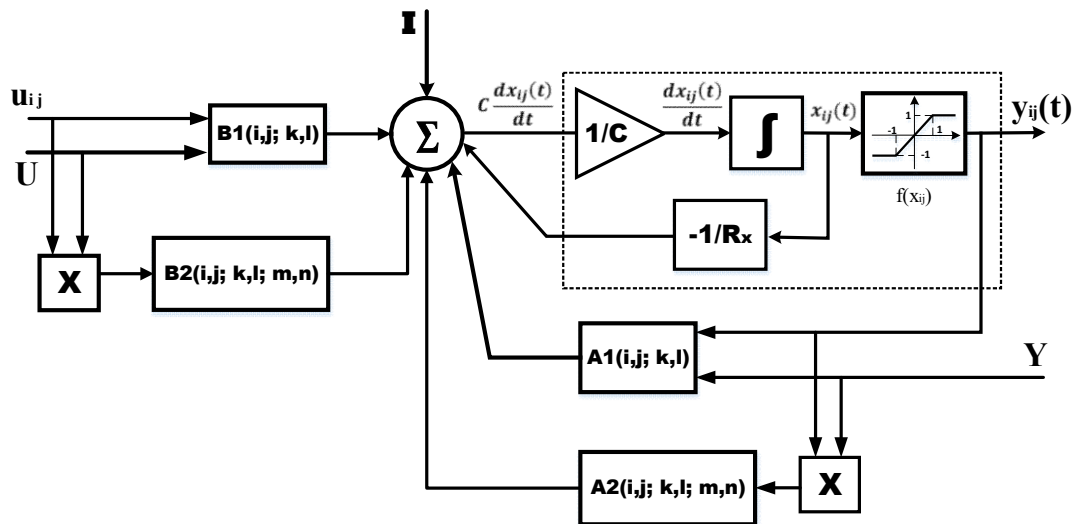
$\mathbf{B2}(i, j; k, l; m, n)$: Ma trận trọng số đầu vào của thành phần bậc hai, kích thước (9×9) ;

M, N : Kích thước của mạng SOCeNNs

\mathbf{I} : Ma trận trọng số ngưỡng SOCeNNs, kích thước (1×1) ;

\mathbf{U} : Ma trận tín hiệu đầu vào lân cận của tế bào $C(i, j)$;

\mathbf{Y} : Ma trận tín hiệu đầu ra lân cận của tế bào $C(i, j)$;



Hình 1.8. Cấu trúc tổng quát mạng nơ ron tế bào bậc hai

Cụ thể đối với tế bào $C(i,j)$ với bán kính lân cận $r=l$, các thành phần đầu vào, đầu ra bậc hai nó sẽ được thể hiện như sau:

Đầu vào, đầu ra bậc hai của $C(i,j)$ tại vị trí tế bào $C(i-l,j-l)$

Đầu vào:

$$u_{i-l,j-l} \times \begin{pmatrix} u_{i-l,j-l} & u_{i-l,j} & u_{i-l,j+l} \\ u_{ij-l} & u_{ij} & u_{ij+l} \\ u_{i+l,j-l} & u_{i+l,j} & u_{i+l,j+l} \end{pmatrix} \quad (6)$$

Đầu ra:

$$y_{i-l,j-l} \times \begin{pmatrix} y_{i-l,j-l} & y_{i-l,j} & y_{i-l,j+l} \\ y_{ij-l} & y_{ij} & y_{ij+l} \\ y_{i+l,j-l} & y_{i+l,j} & y_{i+l,j+l} \end{pmatrix} \quad (7)$$

Khi đó đầu vào, đầu ra bậc hai của tế bào có vị trí $C(i-l,j-l)$ có được trình bày theo dưới đây.

Bảng 1.2. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $(i-l,j-l)$

$u_{i-l,j-l} * u_{i-l,j-l}$	$u_{i-l,j} * u_{i-l,j-l}$	$u_{i-l,j+l} * u_{i-l,j-l}$	$y_{i-l,j-l} * y_{i-l,j-l}$	$y_{i-l,j} * y_{i-l,j-l}$	$y_{i-l,j+l} * y_{i-l,j-l}$
$u_{ij-l} * u_{i-l,j-l}$	$u_{ij} * u_{i-l,j-l}$	$u_{ij+l} * u_{i-l,j-l}$	$y_{ij-l} * y_{i-l,j-l}$	$y_{ij} * y_{i-l,j-l}$	$y_{ij+l} * y_{i-l,j-l}$
$u_{i+l,j-l} * u_{i-l,j-l}$	$u_{i+l,j} * u_{i-l,j-l}$	$u_{i+l,j+l} * u_{i-l,j-l}$	$y_{i+l,j-l} * y_{i-l,j-l}$	$y_{i+l,j} * y_{i-l,j-l}$	$y_{i+l,j+l} * y_{i-l,j-l}$

Đối với các tế bào lân cận khác còn lại của $C(i,j)$, sinh viên sử dụng tính chất tương tự, là tạo ra từng bảng giá trị bậc hai tương ứng với từng tế bào như dưới đây:

Bảng 1.3. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i-l,j)$

$u_{i-l,j-l} * u_{i-l,j}$	$u_{i-l,j} * u_{i-l,j}$	$u_{i-l,j+l} * u_{i-l,j}$	$y_{i-l,j-l} * y_{i-l,j}$	$y_{i-l,j} * y_{i-l,j}$	$y_{i-l,j+l} * y_{i-l,j}$
$u_{ij-l} * u_{i-l,j}$	$u_{ij} * u_{i-l,j}$	$u_{ij+l} * u_{i-l,j}$	$y_{ij-l} * y_{i-l,j}$	$y_{ij} * y_{i-l,j}$	$y_{ij+l} * y_{i-l,j}$
$u_{i+l,j-l} * u_{i-l,j}$	$u_{i+l,j} * u_{i-l,j}$	$u_{i+l,j+l} * u_{i-l,j}$	$y_{i+l,j-l} * y_{i-l,j}$	$y_{i+l,j} * y_{i-l,j}$	$y_{i+l,j+l} * y_{i-l,j}$

Bảng 1.4. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i-l,j+l)$

$u_{i-lj-l} * u_{i-lj+l}$	$u_{i-lj} * u_{i-lj+l}$	$u_{i-lj+l} * u_{i-lj+l}$
$u_{ij-l} * u_{i-lj+l}$	$u_{ij} * u_{i-lj+l}$	$u_{ij+l} * u_{i-lj+l}$
$u_{i+lj-l} * u_{i-lj+l}$	$u_{i+lj} * u_{i-lj+l}$	$u_{i+lj+l} * u_{i-lj+l}$

$y_{i-lj-l} * y_{i-lj+l}$	$y_{i-lj} * y_{i-lj+l}$	$y_{i-lj+l} * y_{i-lj+l}$
$y_{ij-l} * y_{i-lj+l}$	$y_{ij} * y_{i-lj+l}$	$y_{ij+l} * y_{i-lj+l}$
$y_{i+lj-l} * y_{i-lj+l}$	$y_{i+lj} * y_{i-lj+l}$	$y_{i+lj+l} * y_{i-lj+l}$

Bảng 1.5. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j-l)$

$u_{i-lj-l} * u_{ij-l}$	$u_{i-lj} * u_{ij-l}$	$u_{i-lj+l} * u_{ij-l}$
$u_{ij-l} * u_{ij-l}$	$u_{ij} * u_{ij-l}$	$u_{ij+l} * u_{ij-l}$
$u_{i+lj-l} * u_{ij-l}$	$u_{i+lj} * u_{ij-l}$	$u_{i+lj+l} * u_{ij-l}$

$y_{i-lj-l} * y_{ij-l}$	$y_{i-lj} * y_{ij-l}$	$y_{i-lj+l} * y_{ij-l}$
$y_{ij-l} * y_{ij-l}$	$y_{ij} * y_{ij-l}$	$y_{ij+l} * y_{ij-l}$
$y_{i+lj-l} * y_{ij-l}$	$y_{i+lj} * y_{ij-l}$	$y_{i+lj+l} * y_{ij-l}$

Bảng 1.6. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j)$

$u_{i-lj-l} * u_{ij}$	$u_{i-lj} * u_{ij}$	$u_{i-lj+l} * u_{ij}$
$u_{ij-l} * u_{ij}$	$u_{ij} * u_{ij}$	$u_{ij+l} * u_{ij}$
$u_{i+lj-l} * u_{ij}$	$u_{i+lj} * u_{ij}$	$u_{i+lj+l} * u_{ij}$

$y_{i-lj-l} * y_{ij}$	$y_{i-lj} * y_{ij}$	$y_{i-lj+l} * y_{ij}$
$y_{ij-l} * y_{ij}$	$y_{ij} * y_{ij}$	$y_{ij+l} * y_{ij}$
$y_{i+lj-l} * y_{ij}$	$y_{i+lj} * y_{ij}$	$y_{i+lj+l} * y_{ij}$

Bảng 1.7. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i,j+1)$

$u_{i-lj-l} * u_{ij+1}$	$u_{i-lj} * u_{ij+1}$	$u_{i-lj+l} * u_{ij+1}$
$u_{ij-l} * u_{ij+1}$	$u_{ij} * u_{ij+1}$	$u_{ij+l} * u_{ij+1}$
$u_{i+lj-l} * u_{ij+1}$	$u_{i+lj} * u_{ij+1}$	$u_{i+lj+l} * u_{ij+1}$

$y_{i-lj-l} * y_{ij+1}$	$y_{i-lj} * y_{ij+1}$	$y_{i-lj+l} * y_{ij+1}$
$y_{ij-l} * y_{ij+1}$	$y_{ij} * y_{ij+1}$	$y_{ij+l} * y_{ij+1}$
$y_{i+lj-l} * y_{ij+1}$	$y_{i+lj} * y_{ij+1}$	$y_{i+lj+l} * y_{ij+1}$

Bảng 1.8. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j-l)$

$u_{i-lj-l} * u_{i+lj-l}$	$u_{i-lj} * u_{i+lj-l}$	$u_{i-lj+l} * u_{i+lj-l}$
$u_{ij-l} * u_{i+lj-l}$	$u_{ij} * u_{i+lj-l}$	$u_{ij+l} * u_{i+lj-l}$
$u_{i+lj-l} * u_{i+lj-l}$	$u_{i+lj} * u_{i+lj-l}$	$u_{i+lj+l} * u_{i+lj-l}$

$y_{i-lj-l} * y_{i+lj-l}$	$y_{i-lj} * y_{i+lj-l}$	$y_{i-lj+l} * y_{i+lj-l}$
$y_{ij-l} * y_{i+lj-l}$	$y_{ij} * y_{i+lj-l}$	$y_{ij+l} * y_{i+lj-l}$
$y_{i+lj-l} * y_{i+lj-l}$	$y_{i+lj} * y_{i+lj-l}$	$y_{i+lj+l} * y_{i+lj-l}$

Bảng 1.9. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j)$

$u_{i-lj-l} * u_{i+l j}$	$u_{i-l j} * u_{i+l j}$	$u_{i-l j+l} * u_{i+l j}$	$y_{i-l j-l} * y_{i+l j}$	$y_{i-l j} * y_{i+l j}$	$y_{i-l j+l} * y_{i+l j-l}$
$u_{ij-l} * u_{i+l j}$	$u_{ij} * u_{i+l j}$	$u_{ij+l} * u_{i+l j}$	$y_{ij-l} * y_{i+l j}$	$y_{ij} * y_{i+l j}$	$y_{ij+l} * y_{i+l j}$
$u_{i+l j-l} * u_{i+l j}$	$u_{i+l j} * u_{i+l j}$	$u_{i+l j+l} * u_{i+l j}$	$y_{i+l j-l} * y_{i+l j}$	$y_{i+l j} * y_{i+l j}$	$y_{i+l j+l} * y_{i+l j}$

Bảng 1.10. Bảng giá trị bậc hai đối với đầu vào, ra tại vị trí lân cận $C(i+1,j+1)$

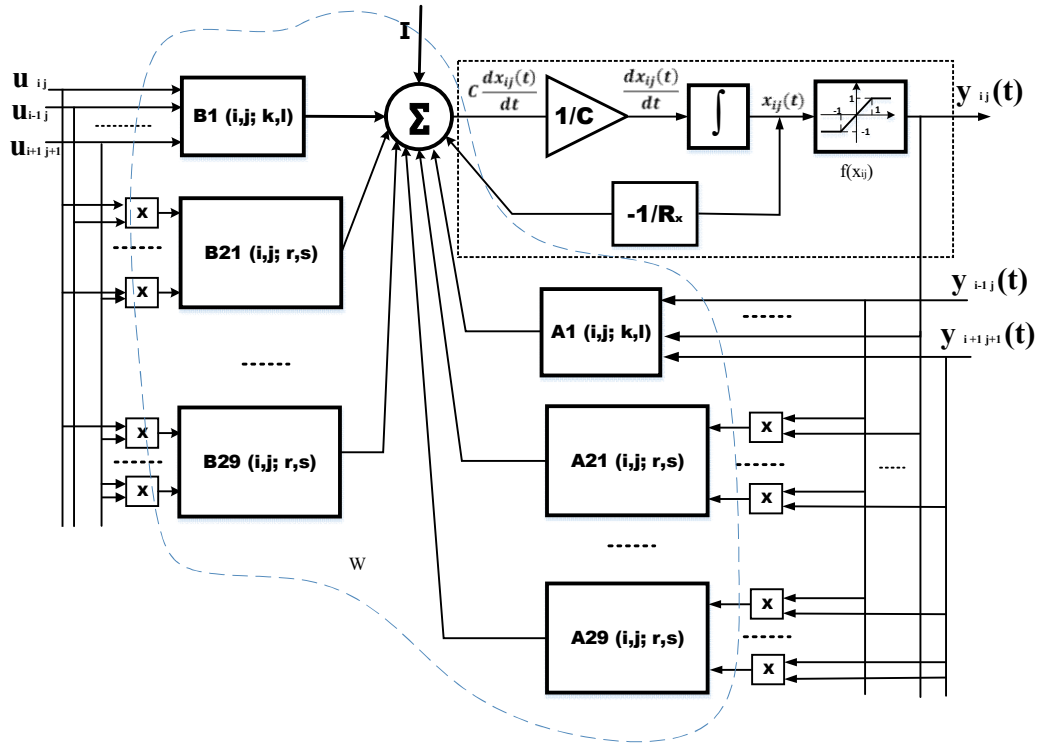
$u_{i-l j-l} * u_{i+l j+l}$	$u_{i-l j} * u_{i+l j+l}$	$u_{i-l j+l} * u_{i+l j+l}$	$y_{i-l j-l} * y_{i+l j+l}$	$y_{i-l j} * y_{i+l j+l}$	$y_{i-l j+l} * y_{i+l j+l}$
$u_{ij-l} * u_{i+l j+l}$	$u_{ij} * u_{i+l j+l}$	$u_{ij+l} * u_{i+l j+l}$	$y_{ij-l} * y_{i+l j+l}$	$y_{ij} * y_{i+l j+l}$	$y_{ij+l} * y_{i+l j+l}$
$u_{i+l j-l} * u_{i+l j+l}$	$u_{i+l j} * u_{i+l j+l}$	$u_{i+l j+l} * u_{i+l j+l}$	$y_{i+l j-l} * y_{i+l j+l}$	$y_{i+l j} * y_{i+l j+l}$	$y_{i+l j+l} * y_{i+l j+l}$

Từ các định nghĩa về tín hiệu đầu vào, đầu ra bậc hai, để xác định bộ ma trận trọng số cho SOCeNNs, với $r=l$, biến đổi các ma trận trọng số phản hồi bậc hai **A2** thành 09 bộ trọng số **A21, A22, A23, A24, A25, A26, A27, A28, A29** tương ứng với 09 bảng tín hiệu đầu ra bậc hai; ma trận **B2** thành 09 bộ **B21, B22, B23, B24, B25, B26, B27, B28, B29** tương ứng với 09 bảng giá trị tín hiệu đầu vào bậc hai. Khi đó, mỗi một bộ trọng số bậc hai này sẽ nhân tích chập với các tín hiệu đầu vào hoặc đầu ra bậc hai tương ứng theo hình (1.8). Như vậy, về mặt hình thức các ma trận trọng số bậc hai tương đồng với các ma trận trọng số bậc nhất và có kích thước 3×3 .

Với cách biến đổi đó, biểu thức (4) viết lại như sau:

$$\begin{aligned}
 \frac{dx_{ij}(t)}{dt} = & -x_{ij}(t) + \sum_{(k,l)} \mathbf{A1}(i,j;k,l) y_{kl}(t) + \sum_{(k,l)} \mathbf{B1}(i,j;k,l) u_{kl} + \mathbf{I} + \\
 & + \sum_{(r,s)} \mathbf{A21}(i,j;r,s) y_{i-1j-1}(t) y_{rs}(t) + \dots + \sum_{(r,s)} \mathbf{A29}(i,j;r,s) y_{i+1j+1}(t) y_{rs}(t) \\
 & + \sum_{(r,s)} \mathbf{B21}(i,j;r,s) u_{i-1j-1} u_{rs} + \dots + \sum_{(r,s)} \mathbf{B29}(i,j;r,s) u_{i-1j-1} u_{rs}
 \end{aligned} \quad (8)$$

Khi đó SOCeNNs có thể được mô tả bằng cấu trúc tương ứng như sau:



Hình 1.9. Sơ đồ cấu trúc CNNs bậc hai quy đổi

Trong đó:

r, s : Thứ tự của tế bào lân cận bậc hai tương ứng với các bộ trọng số đầu vào, đầu ra bậc hai quy đổi

$A21(i,j;r,s), \dots, A29(i,j;r,s)$: Ma trận trọng số phản hồi đầu ra bậc hai SOCeNNs, kích thước (3×3)

$B21(i,j;r,s), \dots, B29(i,j;r,s)$: Ma trận trọng số đầu vào bậc hai SOCeNNs, kích thước (3×3)

1.3.2. Luật học trong mạng nơ ron tế bào bậc hai

Tương tự như CeNN, quá trình học trong SOCeNNs là quá trình xác định giá trị của các bộ ma trận $A1, A21, \dots, A29, B1, B21, \dots, B29$ (20 ma trận) và mức ngưỡng I . Hơn nữa, cũng giống với CeNN, để mạng đạt được sự ổn định thì các ma

trên phải đối xứng tâm. Từ đó suy ra, ta chỉ cần xác định 5 trọng số cho mỗi ma trận. Vậy nên, số lượng trọng số trong bộ trọng số sẽ là $20 \times 5 + 1 = 101$ trọng số.

1.4. Kết chương

Vậy là ta đã đi hết chương I của đồ án: tổng quan về mạng nơ ron tế bào. Thông qua chương này, ta đã biết về định nghĩa cơ bản của mạng nơ ron, hệ thống phân loại mạng nơ ron, cấu trúc của một mạng nơ ron tế bào chuẩn và quan trọng nhất là cấu trúc, mô hình, sơ đồ luồng dữ liệu đầu vào đầu ra của mạng nơ ron tế bào bậc hai. Đây là những kiến thức cơ bản cần thiết để phát triển cho các chương tiếp theo.

CHƯƠNG II: PHƯƠNG PHÁP XÁC ĐỊNH TRỌNG SỐ CỦA MẠNG NƠ RON TẾ BÀO BẬC HAI BẰNG GIẢI THUẬT DI TRUYỀN

Chương này sẽ trình bày phương pháp để xác định bộ trọng số phù hợp cho SOCeNNs với đầu vào cụ thể thông qua giải thuật di truyền (GA). Vì vậy, trước đó, ta sẽ tìm hiểu qua về giải thuật di truyền và luật học trong SOCeNNs.

2.1. Giải thuật di truyền

Trong sinh học, tính trạng của một sinh vật là những đặc điểm về hình thái, cấu tạo, sinh lý, sinh hoá, di truyền... của sinh vật đó. Mà tính trạng của sinh vật được quyết định bởi bộ gen của sinh vật đó. Vì vậy, quá trình tiến hóa của sinh vật để giữ lại, phát triển tính trạng tốt và loại bỏ tính trạng xấu cũng chính là quá trình tối ưu hóa gen. Theo thuyết tiến hóa của Darwin, quá trình tiến hóa sẽ diễn ra thông qua 4 cơ chế sau đây:

- Di truyền: Thế hệ trước sẽ truyền lại bộ gen cho thế hệ sau. Như vậy thì con cái sẽ kế thừa một phần hoặc toàn bộ các tính trạng của bố mẹ.
- Lai tạo: Bộ gen của bố và mẹ được tách ra và tổ hợp lại theo nhiều cách khác nhau để tạo ra bộ gen của con cái. Góp phần tạo lên sự đa dạng sinh học.
- Đột biến: Đôi khi con cái có thể sở hữu một gen không đến từ bố cũng không đến từ mẹ. Điều này dẫn đến sự xuất hiện của những bộ gen hoàn toàn mới chưa từng có trong quần thể.
- Chọn lọc tự nhiên: Các cá thể sở hữu bộ gen tối ưu hơn (tính trạng tốt hơn), sẽ có khả năng sống sót cao hơn và tiếp tục di truyền bộ gen đó cho thế hệ sau.

Giải thuật di truyền (GA) là một thuật toán tối ưu hóa được lấy cảm ứng từ quá trình chọn lọc và tiến hóa tự nhiên. Ở đây, GA được sử dụng để tìm một giải pháp gần đúng cho các bài toán tìm kiếm hay bài toán tối ưu. Tức là từ tập các lời giải ban đầu thông qua nhiều bước tiến hóa hình thành các tập hợp mới với lời giải tốt hơn và cuối cùng sẽ tìm được lời giải gần tối ưu.

GA thuộc lớp các giải thuật xuất sắc nhưng lại rất khác các giải thuật ngẫu nhiên vì chúng kết hợp các phần tử tìm kiếm trực tiếp và ngẫu nhiên. Khác biệt quan trọng giữa tìm kiếm của GA và các phương pháp tìm kiếm khác là GA duy trì và xử lý một tập các lời giải, gọi là một quần thể. Trong GA, việc tìm kiếm giả thuyết thích hợp được bắt đầu với một quần thể, hay một tập hợp có chọn lọc ban đầu của các giả thuyết. Các cá thể của quần thể hiện tại khởi nguồn cho quần thể thế hệ kế tiếp bằng

các hoạt động lai ghép và đột biến ngẫu nhiên – được lấy mẫu sau các quá trình tiến hóa sinh học. Ở mỗi bước, các giả thuyết trong quần thể hiện tại được ước lượng liên hệ với đại lượng thích nghi, với các giả thuyết phù hợp nhất được chọn theo xác suất là các hạt giống cho việc sản sinh thế hệ kế tiếp, gọi là cá thể. Cá thể nào phát triển hơn, thích ứng hơn với môi trường sẽ tồn tại và ngược lại sẽ bị đào thải. GA có thể dò tìm thế hệ mới có độ thích nghi tốt hơn. GA giải quyết các bài toán quy hoạch toán học thông qua các quá trình cơ bản: lai tạo, đột biến và chọn lọc cho các cá thể trong quần thể. Dùng GA đòi hỏi phải xác định được: khởi tạo quần thể ban đầu, hàm đánh giá các lời giải theo mức độ thích nghi – hàm mục tiêu, các toán tử di truyền tạo hàm sinh sản.

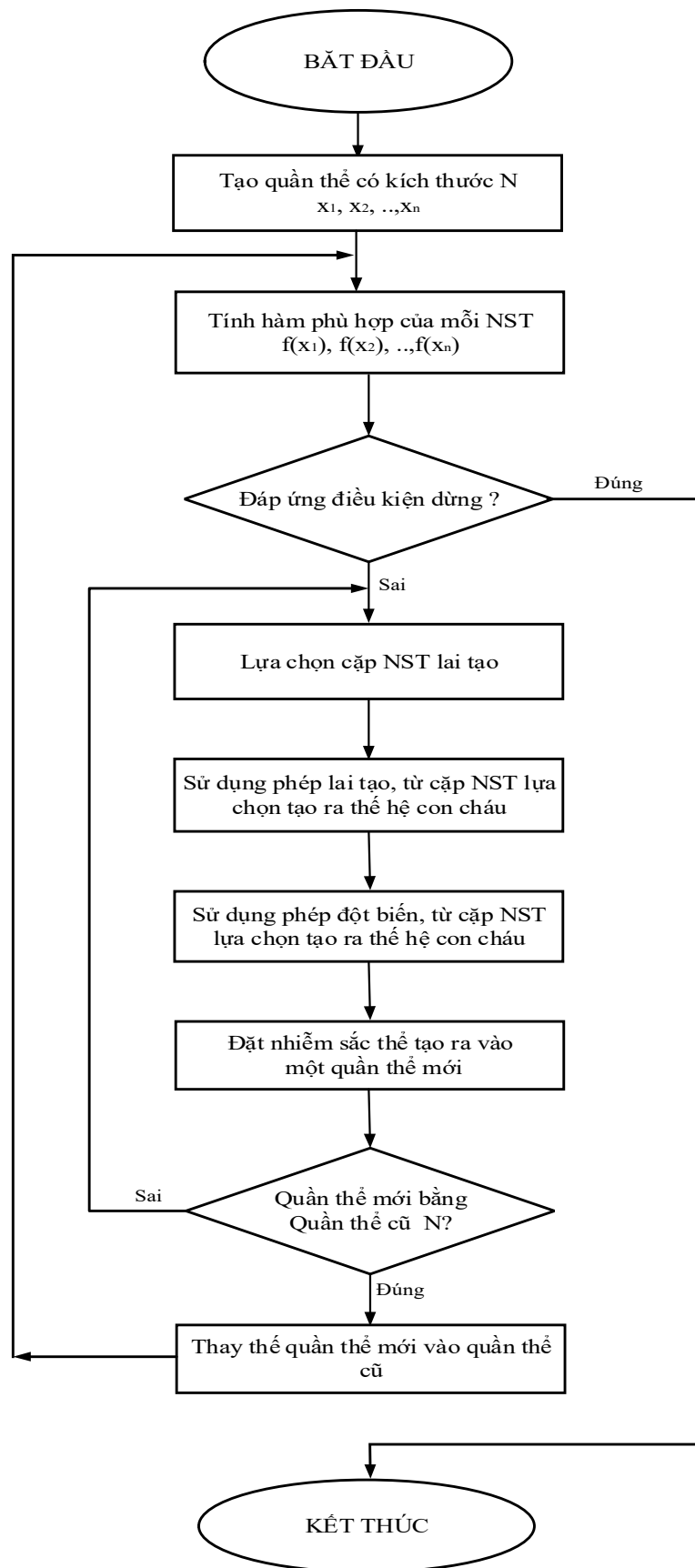
Ưu điểm của giải thuật di truyền [2]:

- Khả năng xử lý trong các bài toán phức tạp: GA rất thích hợp cho các bài toán phức tạp, không gian nhiều chiều hay bài toán phi tuyến. Nó có khả năng đưa ra nhiều phương án và xử lý các vấn đề có nhiều biến số và các điều kiện ràng buộc.
- Có khả năng giải quyết các bài toán ngẫu nhiên, hoặc nhiễu: Trong trường hợp giải quyết các bài toán có nhiễu hoặc biến thay đổi ngẫu nhiên. GA có khả năng đưa ra các giải pháp phù hợp với những thay đổi của các vấn đề.
- Tối ưu hóa: GA sử dụng cho các bài toán tìm cực trị của hàm toán học, đặc biệt trong các hàm phức tạp. GA tìm kiếm sự tối ưu toàn cục trong các vấn đề tối ưu hóa đa phương thức, có nhiều điểm tối ưu cục bộ. Giải thuật di truyền hạn chế sự tối ưu cục bộ bằng cách sử dụng một lượng lớn các giá trị dữ liệu của nhiều vùng khác nhau trong miền không gian giải pháp.
- Không cần mô hình cụ thể: Giải thuật di truyền không cần yêu cầu mô hình toán học chính xác của đối tượng, hay độ dốc của hàm đặc tính. Chính vì vậy, GA phù hợp cho các bài toán chưa rõ mô hình toán học hay khó xây dựng mô hình toán học.

Đối với các bài toán sử dụng giải thuật di truyền hầu như đều tuân theo một cấu trúc tương tự nhau và chia ra làm một số bước chính như sau [3]:

- Bước khởi tạo: Tạo ra một tập các cá thể mẫu ban đầu các giải pháp cho bài toán – nhiễm sắc thể. Mã hóa các nhiễm sắc thể bằng các Gen theo cách mà thuật toán có thể thực hiện. Tùy thuộc nhu cầu của bài toán mà dùng Gen cơ số 2 hay cơ số 10.

- Bước đánh giá hàm phù hợp: Sử dụng hàm phù hợp đánh giá mức độ tương quan của mỗi cá thể trong quần thể. Hàm phù hợp xác định cách giải quyết vấn đề của mỗi cá thể, nó phụ thuộc vào bài toán cụ thể cần giải quyết.
- Bước chọn lọc: Chọn các cá thể từ quần ban đầu để tạo thành nhóm cha mẹ. Quá trình chọn lọc thường ưu tiên các cá thể có khả năng thích ứng cao hơn, nó dựa trên khái niệm chọn lọc tự nhiên.
- Bước lai tạo: Ghép các cá thể từ nhóm cha mẹ ban đầu để tạo ra thế hệ tiếp theo. Áp dụng quy tắc lai ghép để trao đổi tính chất di truyền giữa bố mẹ, tạo ra một hoặc nhiều con cái. Mục tiêu của quá trình lai tạo là kết hợp các đặc điểm của cha mẹ để tạo ra một thế hệ tốt hơn.
- Bước đột biến: Đưa ra các thay đổi ngẫu nhiên không thường xuyên đối với thế hệ tiếp theo. Đột biến làm tăng thêm sự đa dạng cho quần thể và tạo ra những cá thể mới có tính đột phá đối với thế hệ tiếp theo. Tỷ lệ đột biến thường thấp để ngăn chặn sự gián đoạn quá mức (Trong một số trường hợp không cần dùng đến bước này trong quá trình thực hiện Giải thuật di truyền)
- Thay thế: Từ các thế hệ con cái mới được tạo ra, tiến hành thay thế quần thể chứa chứa cha mẹ ban đầu bằng quần thể chứa các thế hệ mới. Những cá thể có thành tích tốt nhất có thể được chuyển giao sang thế hệ tiếp theo.
- Kết thúc: Thực hiện quá trình lặp các bước chọn lọc, lai ghép, đột biến và thay thế cho một số thế hệ xác định hoặc cho đến khi đáp ứng điều kiện kết thúc của hàm phù hợp.



Hình 2.1. Lưu đồ thuật toán giải thuật di truyền

2.2. Thuật toán xác định trọng số của mạng nơ ron tế bào bậc hai bằng giải thuật di truyền

Đối với bài toán cụ thể xác định bộ trọng số cho mạng nơ ron tế bào bậc cao, ta lựa chọn các đặc tính của bộ trọng số và mạng nơ ron tế bào bậc cao để tạo ra các Gen, Nhiễm sắc thể, quần thể phù hợp. Nội dung chi tiết được trình bày như sau:

Gen: Việc lựa chọn Gen cho mạng nơ ron tế bào là bước quan trọng trong quá trình tối ưu hóa việc tính toán các trọng số phục vụ cho bài toán xác định biên ảnh. Đồ án này sử dụng gen cơ sở 10, $G = (0, 1, 2, \dots, 9)$. Mỗi gen sẽ là một trong các số từ $(0 - 9)$ hoặc là dấu $(+)$, $(-)$ để thể hiện số dương hoặc số âm. Do việc tính toán các bộ trọng số mạng nơ ron tế bào là quá trình tối ưu hóa liên tục, việc lựa chọn Gen cơ sở 10 là hoàn toàn thích hợp. Các gen thập phân này có khả năng dẫn đến sự đa dạng cao hơn trong quần thể. Các gen có giá trị thực có thể biểu thị trọng số và độ lệch của các nơ-ron trong mạng lưới thần kinh.

Nhiễm sắc thể: Mỗi một trọng số trong bộ 101 trọng số của SOCeNNs là một nhiễm sắc thể. Giá trị mỗi nhiễm sắc thể nằm trong đoạn $[-9.99, +9.99]$.

Quần thể: Là bộ trọng số với 101 trọng số hay 101 NST.

Hàm phù hợp: Sử dụng hàm phù hợp (hàm mục tiêu) cho bài toán xác định đường biên của ảnh. Theo từng yêu cầu của bài toán xác định biên ảnh cung cấp bộ số lượng mẫu phù hợp cho việc học của SOCeNNs. Phương trình

$$E(w) = \frac{1}{2} (d_{ij}^s - y_{ij}^s)^2 = e_{min}.$$

Xác định giá trị trạng thái $x_{ij}(t)$, đầu vào bậc hai và đầu ra bậc hai của SOCeNNs: Là quá trình chạy mạng SOCeNNs để từ ma trận đầu vào và bộ trọng số để tìm ra ma trận đầu ra và bộ trọng số để tính ma trận trạng thái x . Rồi từ đó tính toán ma trận đầu ra y . Giá trị trạng thái $x_{ij}(t)$ khi SOCeNNs đạt trạng thái ổn định được xác định theo biểu thức (8), còn giá trị đầu ra bậc nhất $y_{ij}(t)$ được xác định theo biểu thức (5). Từ đó xác định giá trị các thành phần đầu vào, đầu ra bậc hai cho SOCeNNs.

Thuật toán GASOCeNNs được trình bày cụ thể như sau:

Input: Đầu vào của thuật toán bao gồm:

- i) Cấu trúc mạng nơ ron tế bào bậc hai thỏa mãn các biểu thức

- ii) Giá trị sai lệch e_{min}
- iii) Chọn bộ ma trận tổng $\mathbf{W}[0]$ ban đầu

$$\mathbf{W}[0] = \begin{bmatrix} a_{11}[0] & \dots & a_{211}[0] & \dots & a_{295}[0] & b_{11}[0] & \dots & b_{211}[0] & \dots & b_{295}[0] & I[0] \end{bmatrix}$$

- iv) Chọn 101 quần thể cha mẹ ban đầu, với Gen cơ sở 10, mỗi cá thể cha mẹ có 5 Gen (bao gồm cả đầu)
- v) Chọn tỷ lệ lai tạo và đột biến

Output: Bộ trọng số W của SoCeNNs

Bước 1: Khởi tạo

Khởi tạo các điều kiện ban đầu của thuật toán theo các dữ liệu đã cung cấp tại Input.

Bước 2: Đánh giá hàm phù hợp:

$$E = \frac{1}{2} \left(d_{ij}^{(s)} - y_{ij}^{(s)} \right)^2 \rightarrow E_{min}$$

Bước 2.1.: Nếu $E > E_{min}$ chuyển sang bước 3

Bước 2.2: Nếu $E \leq E_{min}$ giá trị trọng số thỏa mãn bài toán. Chuyển sang bước 7.

Bước 3: Tính giá trị trạng thái $x_{ij}(t)$ đầu ra tính toán $y_{ij}(t)$ của SOCeNNs.

*Bước 4: Tính các giá trị đầu vào bậc hai $u_{kl} * u_{mn}$, đầu ra bậc hai $y_{kl}(t) * y_{mn}(t)$ của SOCeNNs.*

Bước 5: Lai tạo

Tiến hành lai tạo từng quần thể cha mẹ đối với mỗi trọng số của CeNNs, lựa chọn NST tối ưu tương ứng với mỗi NST trong ma trận trọng số bằng cách tính hàm phù hợp được tạo ra bởi mỗi NST mới. NST mới nào tạo ra hàm phù hợp nhỏ nhất sẽ được lựa chọn để thay thế vào NST ban đầu của SOCeNNs. Thực hiện phép lai tạo theo nguyên tắc từ trái qua phải của ma trận \mathbf{W} tức là từ a_{11} đến I.

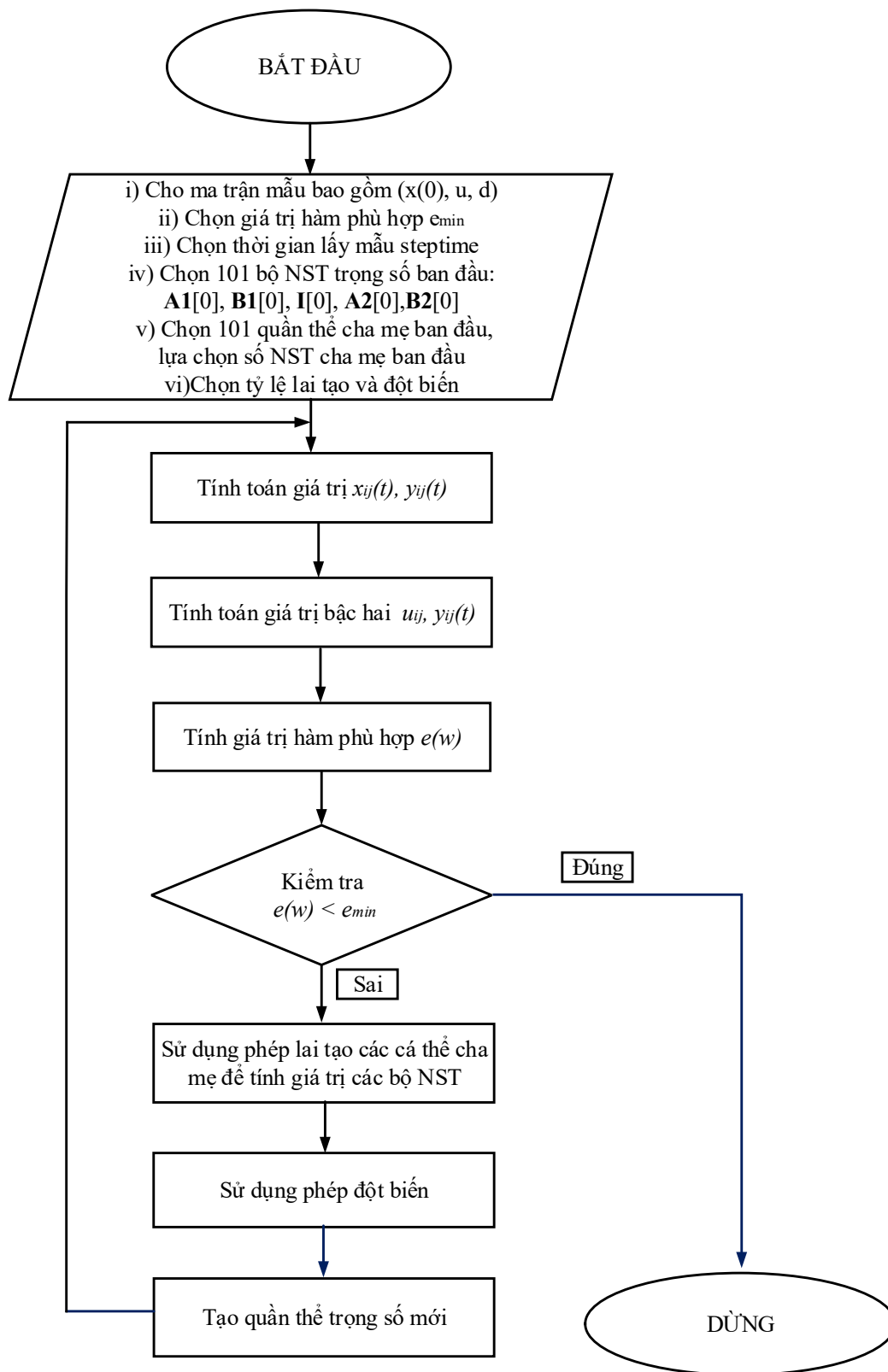
$$\text{Bước 5.1: Khi } E = \frac{1}{2} \left(d_{ij}^{(s)} - y_{ij}^{(s)} \right)^2 \leq E_{min} \text{ chuyển sang bước 7}$$

Bước 5.2: Khi $E = \frac{1}{2} \left(d_{ij}^{(s)} - y_{ij}^{(s)} \right)^2 > E_{min}$ chuyển sang bước 6

Bước 6: Thực hiện phép đột biến cho một số trọng số trong mạng. Quay lại bước 2 để xác định ma trận trọng số cho các vòng tiếp theo.

Bước 7: Dừng thuật toán.

Lưu đồ thuật toán của GA cho SOCeNNs:



Hình 2.2: Lưu đồ khái quát thuật toán GA cho SOCeNNs

2.3. Phương pháp tính toán ma trận trạng thái từ phương trình trạng thái của SOCeNNs

Do phương trình trạng thái (8) của SOCeNNs là một phương trình vi phân. Mà để có thể tính toán đầu ra của mạng SOCeNNs từ ma trận đầu vào và bộ ma trận trọng số, ta cần giải được phương trình vi phân này. Do vậy ta cần phải có phương pháp cũng như thuật toán tương ứng để có thể giải được phương trình vi phân này.

2.3.1. Phương trình vi phân thường ODE

Đầu tiên, hãy cùng xem lại phương trình trạng thái của SOCeNNs.

$$\begin{aligned} \frac{dx_{ij}(t)}{dt} = & -x_{ij}(t) + \sum_{(k,l)} \mathbf{A1}(i,j;k,l)y_{kl}(t) + \sum_{(k,l)} \mathbf{B1}(i,j;k,l)u_{kl} + \mathbf{I} + \\ & + \sum_{(r,s)} \mathbf{A21}(i,j;r,s)y_{i-1j-1}(t)y_{rs}(t) + \dots + \sum_{(r,s)} \mathbf{A29}(i,j;r,s)y_{i+1j+1}(t)y_{rs}(t) \quad (9) \\ & + \sum_{(r,s)} \mathbf{B21}(i,j;r,s)u_{i-1j-1}u_{rs} + \dots + \sum_{(r,s)} \mathbf{B29}(i,j;r,s)u_{i-1j-1}u_{rs} \end{aligned}$$

Hiển nhiên đây là một phương trình vi phân với biến t , hàm x là hàm biểu diễn theo t ($x = x(t)$) và đạo hàm của hàm x ($\frac{dx(t)}{dt}$). Các phần tử của bộ ma trận phản hồi $\mathbf{A1}$, ma trận điều khiển $\mathbf{B1}$, ma trận phản hồi bậc hai $\mathbf{A21} \dots \mathbf{A29}$, ma trận điều khiển bậc hai $\mathbf{B21} \dots \mathbf{B29}$, mức ngưỡng \mathbf{I} , các phần tử của ma trận đầu vào u , ... đều là các hệ số của phương trình vi phân.

Ngoài ra còn có hàm $y(t)$. Tuy nhiên, $y(t)$ lại phụ thuộc vào $x(t)$ thông qua hàm đầu ra:

$$y_{ij}(t) = \frac{1}{2} \left(|x_{ij}(t) + 1| - |x_{ij}(t) - 1| \right) \quad (10)$$

Nói các khác, ta có thể thay $y(t)$ bằng $x(t)$ và phương trình vi phân khi này sẽ có dạng:

$$x' = F(t, x) \quad (11)$$

Phương trình (11) được gọi là phương trình vi phân thông thường (còn được viết tắt là ODE). Trong toán học, phương trình vi phân thông thường là các phương trình bao gồm một hoặc nhiều hàm của một biến độc lập cùng với các đạo hàm của chúng. Các hàm ODE sẽ chỉ là hàm một biến và đạo hàm cũng sẽ chỉ là đạo hàm theo một biến.

Phân tích thêm một chút về việc giải phương trình trạng thái (9) của SOCeNNs. Ta thấy việc giải phương trình vi phân để tìm ra hàm số $x(t)$ là không cần thiết. Bởi thứ ta cần xác định là giá trị của các phần tử của ma trận trạng thái x_{ij} . Do vậy, phương pháp phù hợp để giải phương trình vi phân trong tình huống này là phương pháp số (Numerical Methods).

Phương pháp số cho phương trình vi phân thông thường là phương pháp được sử dụng để tìm xấp xỉ số cho các nghiệm của phương trình vi phân thông thường (ODE). Điều này có nghĩa là, thay vì tìm hàm $x(t)$ thỏa mãn phương trình vi phân thì ta chỉ cần một cặp giá trị khởi đầu t_0 và $x_0 = x(t_0)$. Từ đó, thông qua phương pháp số, ta có thể tìm được xấp xỉ giá trị của $x(t)$ với t bất kỳ mà không cần xác định chính xác hàm $x(t)$.

2.3.2. Phương pháp đa bước tuyến tính

Phương pháp đa bước tuyến tính [4] được sử dụng để giải phương trình vi phân thông thường theo nguyên lý của phương pháp số. Về mặt khái niệm, một phương pháp số sẽ bắt đầu từ một cặp giá trị ban đầu $(x(t_0); t_0)$ và sau đó thực hiện “một bước” ngắn về phía trước để tìm cặp giá trị $(x(t_1), t_1)$ tiếp theo. Rồi tiếp tục từ cặp giá trị (x_n, t_n) để tính cặp giá trị (x_{n+1}, t_{n+1}) . Theo đó, người ta đã sáng tạo ra phương pháp một bước, phương pháp hai nhiều bước.

Các phương pháp một bước (chẳng hạn như phương pháp Euler) chỉ đề cập đến một điểm trước đó và đạo hàm của nó để xác định giá trị hiện tại. Các phương pháp nhiều bước cố gắng đạt được hiệu quả bằng cách giữ và sử dụng thông tin từ các bước trước đó thay vì loại bỏ nó. Do đó, các phương pháp nhiều bước đề cập đến một số điểm và giá trị đạo hàm trước đó. Trong trường hợp phương pháp nhiều bước tuyến tính, một tổ hợp tuyến tính của các điểm trước đó và các giá trị đạo hàm được sử dụng.

Cụ thể, cho phương trình vi phân ODE và cặp giá trị ban đầu (t_0, y_0) như sau:

$$y' = f(t, y), \quad y(t_0) = y_0.$$

Kết quả xấp xỉ cho giá trị của $y(t)$ vào những thời điểm rời rạc t_i như sau:

$$y_i \approx y(t_i) \quad \text{where} \quad t_i = t_0 + ih$$

Trong đó, h là time step, còn i là một số nguyên.

Phương pháp đa bước tuyến tính sử dụng s cặp giá trị (t, y) từ các bước trước đó để tính cặp giá trị (y, t) tiếp theo theo như công thức dưới đây:

$$\begin{aligned} y_{n+s} &+ a_{s-1} \cdot y_{n+s-1} + a_{s-2} \cdot y_{n+s-2} + \cdots + a_0 \cdot y_n \\ &= h \cdot (b_s \cdot f(t_{n+s}, y_{n+s}) + b_{s-1} \cdot f(t_{n+s-1}, y_{n+s-1}) + \cdots + b_0 \cdot f(t_n, y_n)) \\ \Leftrightarrow \sum_{j=0}^s a_j y_{n+j} &= h \sum_{j=0}^s b_j f(t_{n+j}, y_{n+j}), \end{aligned}$$

Trong đó, hệ số $a_s = 1$, giá trị của các hệ số còn lại sẽ phụ thuộc tùy theo phương pháp cụ thể. Thường thì các giá trị sẽ được chọn để phù hợp cho việc toán.

Tùy theo giá trị của hệ số $b_s = 0$ hoặc $b_s \neq 0$. Phương pháp đa bước tuyến tính sẽ chia ra làm phương pháp rõ ràng (explicit methods) và phương pháp ngầm (implicit methods).

Với phương pháp rõ ràng, với $b_s = 0$, do đã biết trước giá trị của các cặp giá trị của $(t_{n+s-1}, y_{n+s-1}), (t_{n+s-2}, y_{n+s-2}), \dots, (t_n, y_n)$ nên ta có thể tính được trực tiếp giá trị của y_{n+s} .

Nhưng khi $b_s \neq 0$, dựa theo công thức của phương pháp đa bước tuyến tính, để tính được giá trị của y_{n+s} , ta cần tính được giá trị của $f(t_{n+s}, y_{n+s})$. Mà bên trong $f(t_{n+s}, y_{n+s})$ lại có chứa y_{n+s} . Do vậy, không thể tính toán trước tiếp được giá trị của y_{n+s} . Cũng chính vì vậy mà người ta gọi đây là phương pháp ngầm.

2.3.3. Phương pháp Adams ngầm

Phương pháp Adams [4] là một phương pháp đa bước tuyến tính được dùng để giải phương trình vi phân theo phương pháp số.

Phương pháp Adams có hai phiên bản. Phiên bản Adams–Bashforth thuộc loại phương pháp rõ ràng và phiên bản Adams–Moulton thuộc loại phương pháp ngầm.

Trong đồ án, sinh viên sử dụng phương pháp Adams ngầm (phương pháp Adams–Moulton) để tính xấp xỉ giá trị của ma trận trạng thái thông qua phương trình trạng thái trong mạng nơ ron tế bào bậc hai.

Trong phương pháp Adams ngầm, $a_{s-1} = -1, a_{s-2} = \dots = a_0 = 0$, các hệ số b được chọn theo công thức sau đây:

$$b_{s-j} = \frac{(-1)^j}{j!(s-j)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^s (u+i-1) du, \quad \text{for } j = 0, \dots, s.$$

Kết quả là ta có công thức triển khai \mathcal{Y}_{n+s} theo phương pháp Adams ngầm với $s = 0, 1, 2, 3, 4$ như sau:

$$\begin{aligned} y_n &= y_{n-1} + hf(t_n, y_n), \\ y_{n+1} &= y_n + \frac{1}{2}h(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)), \\ y_{n+2} &= y_{n+1} + h\left(\frac{5}{12}f(t_{n+2}, y_{n+2}) + \frac{8}{12}f(t_{n+1}, y_{n+1}) - \frac{1}{12}f(t_n, y_n)\right), \\ y_{n+3} &= y_{n+2} + h\left(\frac{9}{24}f(t_{n+3}, y_{n+3}) + \frac{19}{24}f(t_{n+2}, y_{n+2}) - \frac{5}{24}f(t_{n+1}, y_{n+1}) + \frac{1}{24}f(t_n, y_n)\right), \\ y_{n+4} &= y_{n+3} + h\left(\frac{251}{720}f(t_{n+4}, y_{n+4}) + \frac{646}{720}f(t_{n+3}, y_{n+3}) - \frac{264}{720}f(t_{n+2}, y_{n+2}) + \frac{106}{720}f(t_{n+1}, y_{n+1}) - \frac{19}{720}f(t_n, y_n)\right). \end{aligned}$$

Vì là một phương pháp ngầm, không thể tính trực tiếp \mathcal{Y}_{n+s} do giá trị của \mathcal{Y}_{n+s} phụ thuộc vào giá trị của $f(t_{n+s}, \mathcal{Y}_{n+s})$. Do đó, Adams đã phải tính toán một cách gián tiếp giá trị của \mathcal{Y}_{n+s} thông qua một phương pháp gọi là phương pháp của Newton.

2.3.4. Phương pháp Newton

Phương pháp Newton [5], còn được gọi là phương pháp Newton-Raphson, được đặt theo tên của Isaac Newton và Joseph Raphson, là một thuật toán tìm nghiệm của một phương trình bằng cách liên tục tính toán các giá trị xấp xỉ nghiệm và hội tụ về nghiệm của một hàm có giá trị thực. Phiên bản cơ bản nhất của phương pháp Newton

bắt đầu với một hàm có giá trị thực $f(x)$, đạo hàm $f'(x)$ của nó và một dự đoán về giá trị nghiệm x_0 ban đầu của phương trình $f(x) = 0$. Khi đó, ta có:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Newton đã chứng minh được rằng x_1 sẽ có giá trị gần với nghiệm thực sự của phương trình $f(x) = 0$ hơn là x_0 . Do vậy, khi lặp lại quá trình tính toán x_{n+1} từ x_n theo công thức:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Ta sẽ tìm được giá trị xấp xỉ nghiệm của phương trình với độ chính xác ngày càng cao.

2.4. Độ phức tạp của thuật toán GASOCeNNs

Thuật toán xác định trọng số bằng giải thuật di truyền GASOCeNNs có chứa thuật toán tìm ma trận đầu ra của mạng nơ ron SOCeNNs. Để tìm đầu ra của mạng SOCeNNs, cần giải phương trình vi phân là phương trình trạng thái của mạng. Để giải phương trình vi phân, ta dùng phương pháp Adams ngầm và phương pháp Newton.

Chính vì vậy việc tính toán độ phức tạp cho toàn bộ thuật toán GASOCeNNs là một vấn đề cực kỳ phức tạp. Hơn nữa, giải thuật di truyền là một giải thuật mang tính ngẫu nhiên. Nghĩa là tốc độ hội tụ của kết quả hay số bước chạy của thuật toán sẽ ngẫu nhiên, không ổn định và khác nhau tùy theo mỗi lần chạy. Vì vậy, việc tính toán số bước trung bình mỗi lần chạy lại là một vấn đề nan giải khác.

Sinh viên chưa tính toán được chi tiết độ phức tạp của thuật toán GASOCeNNs nên chỉ có thể sử dụng một công thức tương đối cho độ phức tạp. Cụ thể như sau:

- Thuật toán GASOCeNNs chia ra làm vòng lặp lớn và vòng lặp nhỏ, mỗi lần chạy một vòng của vòng lặp lớn sẽ cập nhật lại toàn bộ trọng số. Số lần chạy vòng lặp lớn sẽ phụ thuộc vào tốc độ hội tụ của thuật toán (hiện tại chưa xác định công thức chính xác) nhưng không lớn hơn số lần chạy tối đa cho phép.
- Vòng lặp nhỏ sẽ cập nhật từng trọng số trong bộ trọng số.

- Mỗi lần cập nhật trọng số, sẽ tiến hành lai tạo với số lần nhất định.
- Mỗi lần lai tạo sẽ chạy mạng nơ ron SOCeNNs với toàn bộ các mẫu đầu vào, rồi từ đó xác định đầu ra cho từng mẫu và tính toán lại sai số.
- Mỗi lần chạy mạng nơ ron cho từng mẫu sẽ thực hiện tính toán giá trị ma trận trạng thái bằng cách giải phương trình vi phân.

Tổng hợp lại, ta có độ phức tạp của thuật toán GASOCeNNs:

$$O(GA \times slts \times lai \times slm \times n \times m \times ode) \quad (12)$$

Trong đó:

GA: Trung bình số lần cập nhật bộ trọng số.

Slts: Số lượng trọng số trong bộ trọng số của SOCeNNs (mạng SOCeNNs được sử dụng trong đồ án có 101 trọng số).

lai: Số lần lai mỗi khi xét một trọng số trong vòng lặp nhỏ.

slm: Số lượng mẫu được sử dụng.

n, m: Trung bình kích thước mỗi mẫu.

ode: Trung bình độ phức tạp của thuật toán giải phương trình vi phân.

2.5. Kết chương

Tại chương 2, sinh viên đã thực hiện được một số nội dung sau đây:

- Khái quát chung về giải thuật di truyền, nguyên lý cũng như ý nghĩa của nó.
- Xây dựng được chi tiết thuật toán GASOCeNNs. Sinh viên sẽ thực nghiệm và đánh giá thuật toán này trong chương tiếp theo.
- Trình bày phương pháp tính toán phương trình trạng thái của SOCeNNs và các phương pháp trung gian.
- Cuối cùng, sinh viên đưa ra công thức cho độ phức tạp của thuật toán GASOCeNNs.

CHƯƠNG III: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong chương này, sinh viên tiến hành thực nghiệm thuật toán GASOCeNNs cho bài toán tách biên ảnh. Quá trình cụ thể sẽ bao gồm các bước sau đây:

- Thực nghiệm xác định bộ trọng số tối ưu cho việc tách biên bằng thuật toán GASOCeNNs: Chạy thuật toán GASOCeNNs với bộ dữ liệu cho trước để tìm ra bộ trọng số phù hợp với bài toán tách biên ảnh cho mạng SOCeNNs.
- Thực nghiệm khả năng tách biên của bộ trọng số sau khi được tối ưu: Sử dụng bộ trọng số tìm được để tách biên cho một số ảnh cụ thể. Rồi từ đó, đánh giá khả năng tách biên của bộ trọng số tìm được thông qua phương pháp định tính và định lượng.

3.1. Bài toán tách biên ảnh và ứng dụng

Điểm biên là điểm có sự thay đổi đột ngột của các giá trị điểm ảnh (mức xám, màu sắc). Do đó, tách biên ảnh là xác định tập hợp các điểm biên của ảnh. Việc tách biên ảnh sẽ làm nổi bật hình dạng đường nét của vật thể.

Tách biên ảnh ảnh là một bài toán trung gian, cho phép xác định vùng chứa đối tượng (bao bởi đường biên) hoặc tính toán các đặc điểm về hình dáng của đối tượng dựa trên đường biên. Tách biên ảnh có nhiều ứng dụng khác nhau như:

- Phân vùng ảnh
- Tìm kiếm và theo vết đối tượng ảnh
- Nhận dạng đặc điểm sinh trắc học (khuôn mặt, vân tay)
- Phân tích ảnh y tế

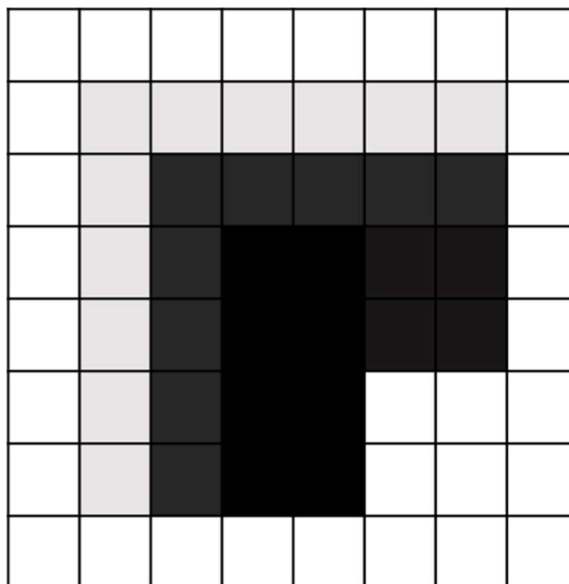
Bài toán tách biên ảnh đã được nghiên cứu từ những năm 70 của thập kỷ trước. Cho đến nay, có nhiều thuật toán tách biên ảnh đã được tạo ra như bộ lọc Canny, bộ lọc Gauss, ... Thậm chí có một số phương pháp sử dụng mô hình mạng học sâu để tách biên ảnh.

Sinh viên sẽ dùng mạng SOCeNNs để tách biên ảnh. Để làm được điều đó, sinh viên cần cho mạng SOCeNNs học được cách tách biên ảnh hay cụ thể hơn là tìm được bộ trọng số phù hợp mà khi cho ảnh đầu vào chạy qua mạng SOCeNNs với bộ trọng số đó sẽ cho ảnh đầu ra là ảnh đầu vào đã được tách biên. Còn về việc xác định bộ trọng số phù hợp thì ta sử dụng thuật toán GASOCeNNs.

3.2. Bộ dữ liệu thực nghiệm

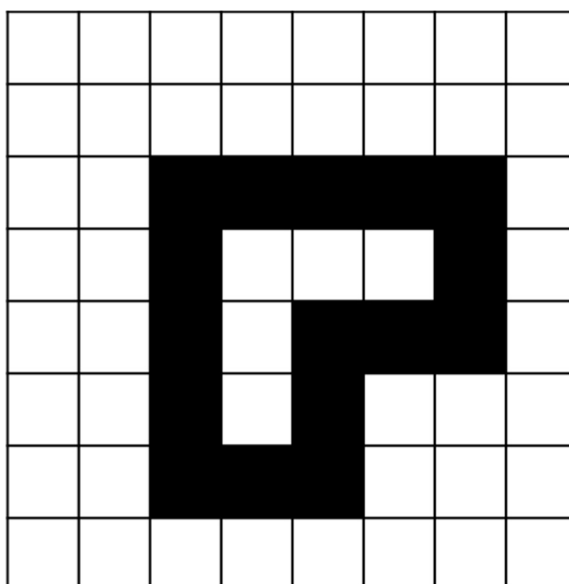
Bộ dữ liệu sẽ gồm 8 mẫu. Mỗi mẫu là một ảnh đầu vào và một ảnh đầu ra là kết quả tách biên của ảnh đầu vào. Ảnh đầu vào và đầu ra có cùng kích thước. Kích thước các ảnh nằm trong khoảng 8x8 đến 32x32. Cụ thể như sau:

Ảnh đầu vào là ảnh xám với các điểm ảnh có giá trị trong khoảng từ 0 đến 255. Trong đó, 255 là màu trắng tuyệt đối, 0 là màu đen tuyệt đối.



Hình 3.1. Mô tả ảnh đầu vào

Ảnh đầu ra mong muốn là ảnh đầu vào sau khi được lọc biên. Ảnh này là ảnh đen trắng và điểm ảnh chỉ có hai giá trị 0 và 255 tương ứng với hai màu đen và trắng.



Hình 3.2. Mô tả đầu ra mong muốn

Lưu ý là không phải mọi sự thay đổi giá trị điểm ảnh đều sẽ tạo ra biên ảnh. Chỉ những sự biến đổi đột ngột khiến cho vùng điểm ảnh đó trông khác hẳn so với các vùng ảnh lân cận thì mới có thể tách biên. Điều này không được mô tả trong thuật toán nhưng các mẫu đều được tạo ra theo cách tương tự. Mạng nơ ron sẽ tự “học” được các đặc trưng này trong quá trình chạy thuật toán.

3.3. Công cụ thực nghiệm

3.3.1. Ngôn ngữ lập trình

Ngôn ngữ lập trình được sử dụng là ngôn ngữ Python.

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và là ngôn ngữ lập trình dễ học. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Quan trọng nhất là Python được sử dụng cực kì phổ biến trong việc phát triển trí tuệ nhân tạo và có rất nhiều thư viện hỗ trợ cho việc tính toán khoa học và giải các bài toán về mạng nơ ron. Đó là lí do tại sao sinh viên lại chọn Python làm ngôn ngữ lập trình trong đồ án này.

3.3.2. Công cụ lập trình

Công cụ được sử dụng để lập trình là Visual studio code.

Visual Studio Code là một trong những trình soạn thảo mã nguồn rất phổ biến được các lập trình viên sử dụng. Với các ưu điểm nổi bật là sự nhanh chóng, nhẹ, hỗ trợ đa nền tảng cùng nhiều tính năng và là mã nguồn mở chính. Visual Studio Code ngày càng được ưa chuộng sử dụng, là lựa chọn hàng đầu của các lập trình viên.

Hơn nữa, Visual studio code còn hỗ trợ lập trình và thực thi mã cho tệp đuôi ipynb. Ipynb là một loại tệp được liên kết với Jupyter Notebook, một ứng dụng web nguồn mở phổ biến cho phép người dùng tạo và chia sẻ tài liệu có chứa mã trực tiếp, phương trình, trực quan hóa và văn bản giải thích. Ưu điểm của tệp ipynb là không chỉ chia code ra từng vùng khác nhau để chạy một cách độc lập; có thể lưu lại output

thời gian chạy chương trình; hỗ trợ hiển thị hình ảnh, biểu đồ, đồ thị, ... Trong đồ án, sinh viên sử dụng tập ipynb để soạn thảo code.

3.3.3. Thư viện sử dụng

Thư viện là tập hợp các mô-đun code viết sẵn chứa các hàm, lớp và cấu trúc dữ liệu có thể tái sử dụng, được thiết kế để thực hiện các tác vụ thông thường. Nói một cách đơn giản, nó giống như một bộ sách tham khảo hỗ trợ ta trong việc lập trình. Dưới đây là danh sách các thư viện được sinh viên sử dụng trong đồ án:

- *Numpy*: Một thư viện toán học phổ biến và mạnh mẽ của Python. Hỗ trợ rất nhiều phép tính khi làm việc với ma trận và mảng như cộng, trừ, nhân ma trận với một số; cộng, trừ, nhân hai ma trận; phép tích chập; các phép biến đổi ma trận. Hơn nữa, một số thư viện khác mà sinh viên dùng cho đồ án cũng sử dụng kiểu dữ liệu ma trận của Numpy.
- *PIL*: Một thư viện chuyên dùng để xử lý hình ảnh của Python. Cho phép đọc hầu hết các loại tệp hình ảnh như JPEG, JEG, PNG, BMP, GIF, ... Hơn nữa còn hỗ trợ tạo ảnh, xuất ảnh, hiển thị ảnh, biến đổi ảnh. Ảnh sau khi đọc từ tệp sẽ được lưu dưới dạng ma trận Numpy.
- *Scipy*: Là thư viện mã nguồn mở của Python chuyên dùng cho toán học, khoa học và kỹ thuật. Thư viện Scipy được xây dựng dựa trên thư viện NumPy, cung cấp các công cụ để thao tác, tính toán trên mảng một chiều, mảng hai chiều hay mảng N chiều một cách thuận tiện và nhanh chóng. Các thao tác mà Scipy cung cấp bao gồm tích chập, đạo hàm, vi phân, giải phương trình vi phân, ... Trong đồ án, sinh viên sử dụng thư viện Scipy để tính tích chập và giải phương trình vi phân. Để giải phương trình vi phân và tính toán giá trị của ma trận trạng thái, sinh viên sử dụng lớp ODE [6] của thư viện Scipy. Lớp ODE cung cấp các hàm giải phương trình vi phân thường theo phương pháp số. ODE có hỗ trợ giải phương trình bằng phương pháp Adams ngầm (phương pháp này được sinh viên sử dụng để chạy thuật toán).
- *Numba*: Thư viện này được dùng để cải thiện tốc độ chạy cho Python. Trong Python, source code Python sẽ được chuyển đổi sang thành định dạng pyc file (hay còn được gọi là pycache) sau đó mới thực hiện biên dịch pyc file sang mã máy. Do qua các bước trung gian nên việc thực thi Python source code bị chậm đi. Numba là một trình dịch JIT mã nguồn mở dùng cho Python và được tài trợ phát triển bởi Anaconda. Numba sẽ thực hiện dịch source code python sang trực tiếp mã máy để tăng tốc độ thực thi. Hiện nay, thư viện này được

ứng dụng nhiều trong khoa học dữ liệu giúp tăng tốc độ biên dịch code với Numpy Array.

3.4. Các tiêu chí đánh giá

Ta đánh giá thuật toán GASOCeNNs theo nhiều tiêu chí sau đây:

- Sai số: là tỉ lệ số điểm ảnh sai lệch trên toàn bộ điểm ảnh của ảnh đầu ra sau khi chạy mạng SOCeNNs so với ảnh đầu ra mong muốn. Theo công thức sau:

$$E = \frac{\sum_{i,j} (y_{ij} - d_{ij})^2}{4 \times n \times m} \quad (13)$$

Sai số sẽ nằm trong khoảng $[0,1]$ và thể hiện mức độ tối ưu của bộ trọng số (bộ trọng số càng tốt thì ảnh đầu ra càng chính xác so với đầu ra mong muốn). Khi chạy bộ dữ liệu với nhiều mẫu, sai số sẽ là sai số trung bình của tất cả các mẫu. Sai số là tiêu chí quan trọng nhất khi đánh giá thuật toán GASOCeNNs. Đặc biệt, do bộ dữ liệu chia làm bộ dữ liệu huấn luyện và bộ dữ liệu đánh giá, nên ta cũng xét cả sai số với bộ dữ liệu huấn luyện và sai số với bộ dữ liệu đánh giá. Ngoài ra còn phải chú ý đến một số tham số như độ chênh lệch giữa sai số của đầu ra so với bộ dữ liệu huấn luyện và sai số của đầu ra so với bộ dữ liệu đánh giá, độ hội tụ của sai số, ...

- Thời gian chạy: là thời gian tối ưu bộ trọng số đến khi đạt được sai số cho phép hoặc cho đến khi thuật toán chạy đến số vòng lặp nhất định.

Ngoài các tiêu chí đánh giá bằng số liệu (hay còn gọi là đánh giá định lượng), thì còn phương pháp đánh giá định tính. Đó là dựa vào ảnh cụ thể sau khi tách biên với bộ trọng số sau khi được tối ưu để từ đó, đưa ra các nhận xét về khả năng lọc biên của mạng SOCeNNs như là biên có mịn, liên tục hay là biên đứt quãng, có các vùng màu còn tồn tại hay không (bởi vì sau khi lọc biên, ảnh chỉ còn lại đường biên, các vùng màu đều sẽ thay bằng nền trắng), ...

3.5. Thực nghiệm và đánh giá

Có nhiều tham số ảnh hưởng tới độ tối ưu của bộ trọng số và thời gian chạy của thuật toán. Sinh viên tiến hành chạy thuật toán nhiều lần với các giá trị khác nhau của từng tham số, từ đó đánh giá sự ảnh hưởng của tham số đó đến quá trình chạy thuật toán GASOCeNNs và chọn ra giá trị phù hợp cho tham số:

3.4.1. Bộ dữ liệu

Bộ dữ liệu thực nghiệm được chia làm hai phần gồm:

- Bộ dữ liệu huấn luyện: Dùng để huấn luyện cho mạng SOCeNNs và tối ưu trọng số.
- Bộ dữ liệu đánh giá: Dùng để đánh giá, kiểm tra độ tối ưu của bộ trọng số sau khi chạy thuật toán.

Ngoài ra, các mẫu ảnh trong bộ dữ liệu có kích thước khác nhau (8x8) đến (32x32). Lý do chọn kích thước này là vì ảnh nhỏ sẽ chạy thuật toán nhanh hơn nhưng mẫu sẽ không có đủ đặc trưng để cho mạng SOCeNNs học, ảnh kích thước lớn sẽ mang nhiều đặc trưng hơn về việc tách biên ảnh, nhưng tốc độ chạy lâu hơn và có khả năng dẫn tới lỗi phương trình vi phân dẫn đến dừng chương trình (hàm giải phương trình vi phân ODE của thư viện Scipy sẽ có khả năng gặp lỗi khi phải giải phương trình vi phân với kích thước đầu vào quá lớn, lý do lỗi có thể là do tràn biến hoặc tràn dấu phẩy động). Tóm lại là kích thước mẫu đầu vào có thể ảnh hưởng tới quá trình chạy của thuật toán GASOCeNNs.

Ta sẽ tiến hành chạy thuật toán GASOCeNNs với ba cách phân chia bộ dữ liệu huấn luyện và bộ dữ liệu đánh giá như sau:

- Bộ 1: 87% bộ dữ liệu dùng để huấn luyện (các bộ dữ liệu dùng để huấn luyện sẽ là các bộ có kích thước lớn), 13% còn lại dùng để đánh giá (chứa các bộ dữ liệu kích thước nhỏ)
- Bộ 2: 87% bộ dữ liệu dùng để huấn luyện (các bộ dữ liệu dùng để huấn luyện sẽ là các bộ có kích thước nhỏ), 13% còn lại dùng để đánh giá (chứa các bộ dữ liệu kích thước lớn)
- Bộ 3: 50% bộ dữ liệu dùng để huấn luyện (các bộ dữ liệu dùng để huấn luyện gồm các bộ có kích thước lớn và nhỏ), 50% còn lại dùng để đánh giá (chứa cả các bộ dữ liệu kích thước và nhỏ)

Kết quả được ghi trong bảng dưới đây:

Bảng 3.1. Kết quả chạy với các bộ dữ liệu khác nhau

STT	Bộ dữ liệu	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	Bộ 1	8 phút 11.1 giây	0.035	0.172

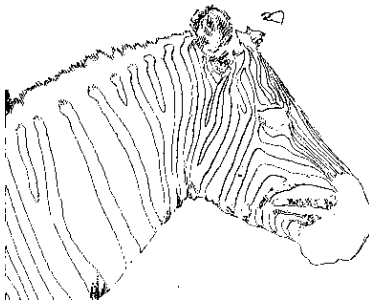

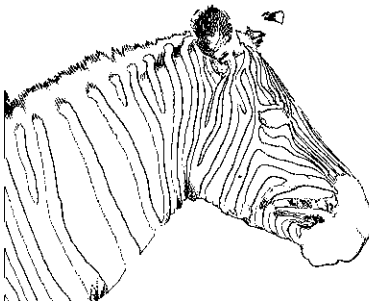

2	Bộ 2	2 phút 17.6 giây	0.036	0.148
3	Bộ 3	4 phút 48 giây	0.078	0.08

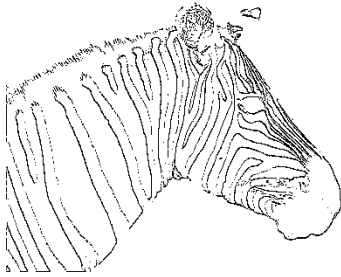
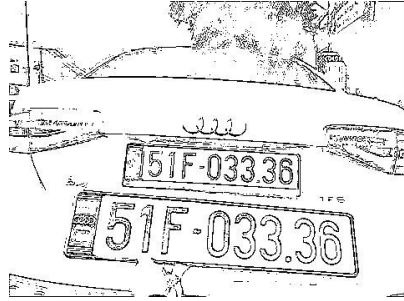


Nhận xét:

- Bộ 1 và bộ 2 đều cho sai số thấp với bộ huấn luyện nhưng lại cao với bộ đánh giá. Sự chênh lệch này dẫn đến hiện tượng overfitting (hiện tượng khi mạng nơ ron chỉ chạy tốt với bộ dữ liệu huấn luyện nhưng k tốt với các bộ dữ liệu mới). Điều này có nghĩa là bộ trọng số ở bộ 1 và bộ 2 không chuẩn.
- Bộ 1 có tốc độ chạy lâu nhất do chạy các mẫu kích thước lớn.
- Bộ 2 có tốc độ chạy nhỏ nhất do chạy các mẫu kích thước nhỏ.
- Bộ 3 không bị hiện tượng overfitting. Bộ trọng số của bộ 3 sẽ tối ưu hơn hai bộ còn lại.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.2. Kết quả tách biên với các bộ dữ liệu khác nhau

STT	Bộ dữ liệu	Tách biên ảnh xám	Tách biên ảnh màu
1	Bộ 1		
2	Bộ 2		

3	Bộ 3		
Ảnh gốc			

Nhận xét:

- Cả 3 bộ đều lọc biên tốt ở ảnh xám.
- Bộ 1 và bộ 2 không lọc hết biên ở ảnh màu, vẫn còn vùng xám.
- Bộ 3 lọc ảnh màu tốt.

Đánh giá về bộ dữ liệu: Bộ dữ liệu số 3 là bộ dữ liệu tốt nhất.

3.4.2. Bộ trọng số ban đầu

Thuật toán GASOCeNNs yêu cầu một bộ trọng số ban đầu, thứ sẽ liên tục được cập nhật, tối ưu trong quá trình chạy thuật toán. Bộ trọng số ban đầu có thể ảnh hưởng đến tốc độ học, độ hội tụ của kết quả. Trong đồ án, sinh viên sử dụng 3 bộ trọng số ban đầu khác nhau. Cụ thể là:

- Bộ 1: Tất cả các trọng số đều bằng 0.
- Bộ 2: Các trọng số sẽ có giá trị thực ngẫu nhiên trong khoảng $(-10;10)$
- Bộ 3: Bộ trọng số cố định với các ma trận trọng số đều bằng nhau và bằng

-1.23	4.56	-7.89
0	0	0
-7.89	4.56	-1.23

Mức ngưỡng có giá trị $I = 1$. Lý do chọn bộ trọng số này là vì các số khác 0 sẽ tăng độ đa dạng cho bộ trọng số thông qua phép lai, còn các trọng số bằng 0 sẽ tăng tốc độ tính toán.

Kết quả được ghi trong bảng dưới đây:

Bảng 3.3. Kết quả chạy với các bộ trọng số khác nhau

STT	Bộ trọng số ban đầu	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	Bộ 1	5 phút 17.7 giây	0.068	0.0805
2	Bộ 2	11 phút 46.6 giây	0.102	0.23
3	Bộ 3	6 phút 19.5 giây	0.066	0.102

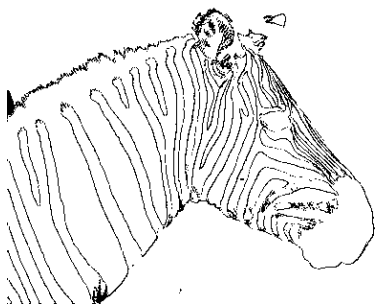

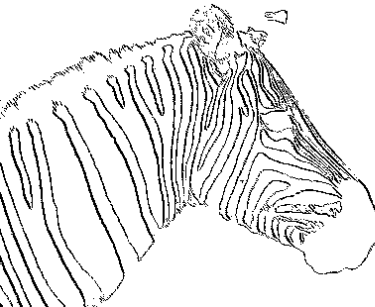

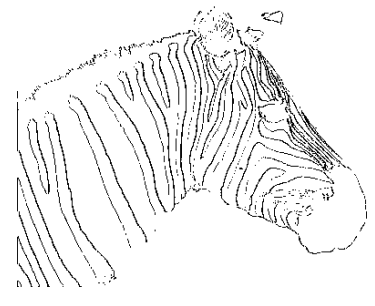



Nhận xét:

- Bộ 1 có tốc độ tính toán nhanh nhất do tốc độ thực hiện các phép tính với số 0 của máy tính rất nhanh. Đổi lại, do các trọng số đều bằng 0 lên kết quả phép lai tạo ban đầu sẽ không tạo ra sự đa dạng giá trị cho bộ trọng số. Phải sau vài lần chạy cho quá trình đột biến xảy ra. Thì phép lai mới có ý nghĩa. Bộ 1 có độ lệch giữa sai số với bộ dữ liệu huấn luyện và sai số với bộ dữ liệu đánh giá nhỏ nhất.
- Bộ 2 có tốc độ chạy lâu do các trọng số đều khác 0. Và có sai số và độ overfitting cao nhất.
- Bộ 3 có tốc độ chạy, sai số và độ overfitting ở mức vừa phải.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.4. Kết quả tách biên với các bộ trọng số khác nhau

STT	Bộ trọng số ban đầu	Tách biên ảnh xám	Tách biên ảnh màu
-----	---------------------	-------------------	-------------------

1	Bộ 1		
2	Bộ 2		
3	Bộ 3		
Ảnh gốc			

Nhận xét:

- Cả 3 bộ đều tác biên tốt cho ảnh xám.
- Bộ 1 có sai số và độ overfitting nhỏ nhưng lại tách biên cho ảnh màu kém hơn so với hai bộ còn lại.
- Bộ 2 tách biên tốt nhất.

Đánh giá về bộ trọng số ban đầu: Nếu xét vấn đề tối ưu giữa thời gian chạy và khả năng lọc biên thì bộ số 3 sẽ là tốt nhất.

3.4.3. Giá trị khởi đầu của phương trình vi phân

Giá trị này là cần thiết để giải phương trình vi phân và ảnh hưởng lớn tới kết quả đầu ra.

Ta sẽ chạy thuật GASOCeNNs với 2 bộ giá khởi đầu khác nhau:

- Bộ 1: Giá trị khởi đầu là ma trận 0 (ma trận chứa toàn số 0).
- Bộ 2: Giá trị khởi đầu là ma trận ảnh đầu vào.

Kết quả được ghi trong bảng dưới đây:

Bảng 3.5. Kết quả chạy với các bộ giá trị khởi đầu khác nhau

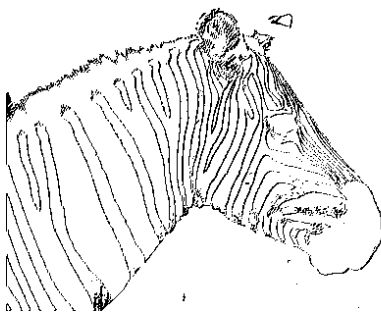

STT	Giá trị khởi đầu	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	Bộ 1	5 phút 4.5 giây	0.076	0.068
2	Bộ 2	6 phút 33.3 giây	0.058	0.103

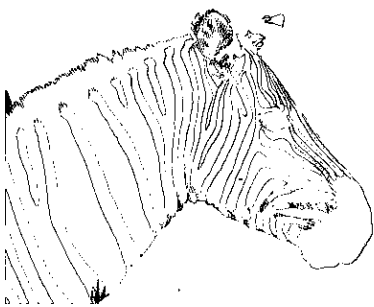



Nhận xét:

- Bộ 1 có thời gian chạy ngắn hơn và độ overfitting thấp hơn bộ 2. Vậy nên bộ một tối ưu hơn.
- Bộ 2 tuy có sai số với bộ dữ liệu huấn luyện thấp hơn nhưng, độ overfitting cao nên sai số với bộ dữ liệu huấn luyện sẽ bị mất đi ý nghĩa.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.6. Kết quả tách biên với các bộ giá trị khởi đầu khác nhau

STT	Giá trị khởi đầu	Tách biên ảnh xám	Tách biên ảnh màu
1	Bộ 1		

2	Bộ 2		
Ảnh gốc			

Nhận xét:

- Với ảnh xám, bộ 1 tách biên tốt, bộ 2 biên mờ và bị đứt quãng.
- Với ảnh màu, cả hai bộ đều bị sót lại vùng xám chưa loại bỏ.

Đánh giá về giá trị khởi đầu của phương trình vi phân: Dựa theo dữ liệu tính toán trên thì việc lấy ma trận 0 làm giá trị khởi đầu sẽ cho kết quả tối ưu hơn.

3.4.4. Time step

Bước nhảy (time step) (trong thuật toán giải phương trình vi phân, bước nhảy được ký hiệu là dt) là một thuộc tính quan trọng của hàm giải phương trình vi phân ODE. Giá trị của bước nhảy ảnh hưởng tới giá trị đầu ra của mạng SOCeNNs.

Ta sẽ chạy thuật toán với lần lượt 2 bước nhảy: $dt = 0.05$ và $dt = 1$.

Kết quả được ghi trong bảng dưới đây:

Bảng 3.7. Kết quả chạy với các bước nhảy khác nhau

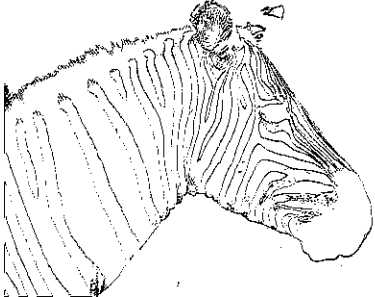
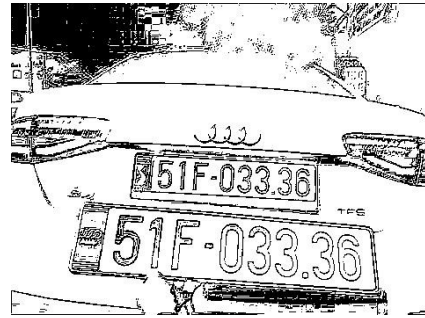
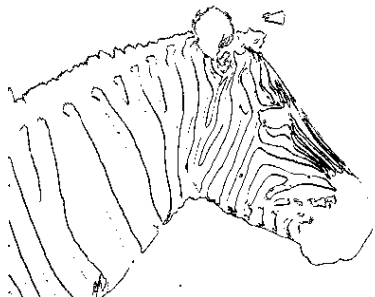
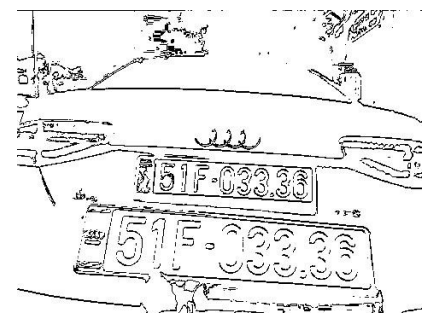


STT	Bước nhảy	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	0.05	3 phút 15.2 giây	0.077	0.126
2	1	23 phút 24 giây	0.178	0.376

Nhận xét:

- Với bước nhảy $dt = 1$, thuật toán chạy mất thời gian hơn nhiều lần so với bước nhảy $dt = 0.05$. Hơn nữa, cả sai số và độ overfitting cũng lớn hơn.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.8. Kết quả tách biên với các bước nhảy khác nhau

STT	Bước nhảy	Tách biên ảnh xám	Tách biên ảnh màu
1	0.05		
2	1		
Ảnh gốc			

Nhận xét:

- Với $dt = 1$, ảnh kết quả bị mất đường biên.
- Với $dt = 0.05$, không bị mất đường biên nhưng còn vùng xám.

Đánh giá về bước nhảy: Bước nhảy càng nhỏ sự hội tụ càng ổn định. Bước nhảy lớn sẽ cho độ hội tụ kém, dễ gây ra lỗi phương trình vi phân do phải tính toán nhiều và tăng thời gian chạy thuật toán.

3.4.5. Số lần lai

Mỗi khi xét một trọng số trong vòng lặp nhỏ, ta bắt đầu tiến hành lai để tạo ra trọng số mới. Tùy theo số lần lai mà ta có thể tăng tỉ lệ tìm được giá trị tối ưu cho trọng số đang xét.

Ta sẽ chạy thuật toán GASOCeNNs với số lần lai bằng 1 và số lần lai bằng 5.

Kết quả được ghi trong bảng dưới đây:

Bảng 3.9. Kết quả chạy với số lần lai khác nhau

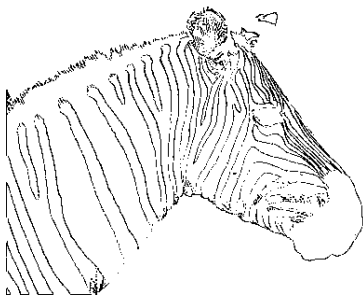

STT	Số lần lai	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	1	3 phút 44.2 giây	0.064	0.054
2	5	12 phút 17.8 giây	0.064	0.046





Nhận xét:

- Thời gian chạy thuật toán với số lần lai bằng 5 lớn hơn thời gian chạy với số lần lai bằng 1.
- Sai số của hai lần chạy với số lần lai bằng 5 và bằng 1 gần như tương đương nhau. Trong đó, chạy với số lần lai bằng 5 có sai số nhỏ hơn.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.10. Kết quả tách biên với số lần lai khác nhau

STT	Số lần lai	Tách biên ảnh xám	Tách biên ảnh màu
1	1		

2	5		
Ảnh gốc			

Nhận xét:

- Cả số lần lai bằng 1 và số lần lai bằng 5 đều chạy ra kết quả đều tách biên tốt với ảnh xám.
- Cả hai kết quả tách biên với ảnh màu đều bị sót vùng xám chưa xóa. Nhưng kết quả khi chạy với số lần lai bằng 5 có nhiều vùng xám hơn

Đánh giá về số lần lai: Việc tăng số lần lai sẽ tăng thời gian chạy thuật toán nhưng hầu như không cải thiện được kết quả đầu ra. Vì vậy, ta nên ưu tiên chạy với số lần lai nhỏ.

3.4.6. Số lần chạy vòng lặp lớn

Mỗi lần chạy một vòng của vòng lặp lớn sẽ tối ưu một lần cho toàn bộ bộ trọng số của mạng SOCeNNs. Để bộ trọng số đạt được giá trị tối ưu, ta cần chạy đi chạy lại vòng lặp lớn nhiều lần. Vì thế, số lần chạy vòng lặp lớn ảnh hưởng trực tiếp tới độ chính xác của mạng SOCeNNs. Càng chạy vòng lặp nhiều lần thì sai số càng nhỏ, nhưng sai số khi gần như không đổi khi gần đạt tới giới hạn.

Ta tiến hành chạy thuật toán GASOCeNNs với số lần chạy lần lượt bằng 5, 25, 50.

Kết quả được ghi trong bảng dưới đây:

Bảng 3.11. Kết quả chạy với số lần chạy khác nhau

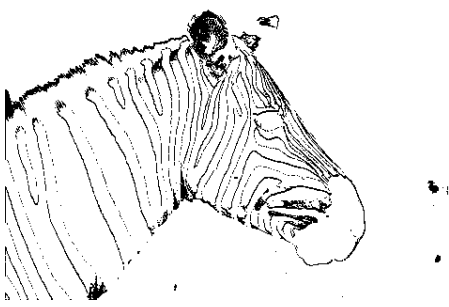

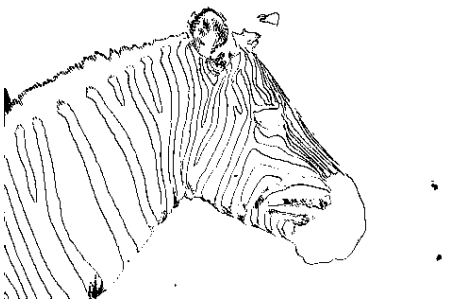

STT	Số lần chạy	Thời gian chạy	Sai số (với bộ dữ liệu huấn luyện)	Sai số (với bộ dữ liệu đánh giá)
1	5	14.7 giây	0.084	0.055
2	25	1 phút 31.1 giây	0.065	0.064
3	50	5 phút 39.3 giây	0.065	0.065





Nhận xét:

- Số lần chạy bằng 5 cho tốc độ chạy nhanh nhất nhưng kết quả kém chính xác nhất.
- Số lần chạy bằng 50 có thời gian chạy lâu hơn nhiều số lần chạy bằng 25 nhưng kết quả chỉ tốt hơn một chút.

Kết quả khi chạy với ảnh xám và ảnh màu:

Bảng 3.12. Kết quả tách biên với số lần chạy khác nhau

STT	Số lần chạy	Tách biên ảnh xám	Tách biên ảnh màu
1	5		
2	25		

3	50		
Ảnh gốc			

Nhận xét:

- Với ảnh xám, cả ba lần chạy đều lọc biên tốt.
- Với ảnh màu, chạy 25 lần và chạy 50 lần đều ra kết quả tách biên gần như nhau và đều tốt hơn kết quả chạy 5 lần.

Đánh giá về số lần chạy: Tăng số lần chạy sẽ tăng thời gian chạy. Nhưng cùng với số lần chạy tăng lên, sai số dần dần giảm đi và hội tụ về một ngưỡng nào đó. Đến một mức độ nhất định, việc tăng số lần chạy sẽ không còn cải thiện kết quả.

3.6. Kết chương

Chương 3 đã mô tả chi tiết bài toán tách biên ảnh, bộ dữ liệu được sử dụng, các công cụ, các tiêu chí đánh giá thuật toán cũng như toàn bộ quá trình thực nghiệm và đánh giá thuật toán GASOCeNNs. Qua đó, ta thấy được giải thuật di truyền cho mạng SOCeNNs có khả năng tối ưu trọng số tốt. Các bộ trọng số tìm được trong quá trình chạy đều có khả năng tách biên. Tuy rằng phần lớn ảnh tách biên vẫn còn chưa xóa hết các vùng xám. Nhưng cơ bản đã làm nổi bật được biên của ảnh.

KẾT LUẬN

1. Kết luận

Trong đồ án này, sinh viên đã trình bày mô hình và luật học của mạng nơ ron tế bào bậc hai được đề xuất bởi TS. N. T. Tuyên và được cải tiến bởi NCS. D. D. Anh. Đồng thời, cũng đã triển khai thực nghiệm thuật toán xác định bộ trọng số của mạng nơ ron tế bào bậc hai bằng giải thuật di truyền. Thông qua kết quả thực nghiệm, thuật toán đã đạt được một số thành tựu sau đây:

- Xây dựng thuật toán chi tiết cho giải thuật GASOCeNNs và thực nghiệm thành công. Từ giờ, ta sẽ có thêm một phương pháp nữa để tiến hành học cho mạng SOCeNNs là giải thuật GA.
- Tách biên ảnh thành công với mạng SOCeNNs. Chứng minh mạng SOCeNNs phù hợp với các bài toán xử lý ảnh.

Đồng thời, các kết quả của quá trình thực nghiệm giải thuật GASOCeNNs đã được sử dụng làm một phần nội dung trong bài báo [7] được trình bày ở Hội nghị quốc tế lần thứ 15 của IEEE về Mạng truyền thông và trí tuệ tính toán (CICN) năm 2023 ở Thái Lan.

2. Hướng phát triển trong tương lai

Tuy không được ưa chuộng như các mô hình mạng nơ ron học sâu, nhưng mạng nơ ron tế bào cũng có các ưu điểm của riêng nó. Kết quả thử nghiệm trong đồ án này cho thấy, mạng SOCeNNs thích hợp với các bài toán xử lý ảnh. Trong tương lai, từ giải thuật GASOCeNNs và mạng SOCeNNs, có thể phát triển theo các hướng nghiên cứu sau:

- Phát triển giải thuật GASOCeNNs bằng cách kết hợp với các giải thuật khác.
- Sử dụng giải thuật GA (hoặc một số giải thuật khác) cho mạng SOCeNNs để giải quyết các bài toán xử lý ảnh cao cấp hơn như bài toán phân vùng ảnh, bài toán nhận diện ảnh, ...

TÀI LIỆU THAM KHẢO

- [1] N. Q. Hoan, N. T. Tuyen and D. D. Anh, "Kiến trúc và Ổn định của mạng nơ-ron tế bào bậc hai," *Tạp chí Khoa học và Công nghệ Đại học Sư phạm Kỹ thuật Hưng Yên*, vol. 27, no. 9, pp. 91-97, 2020.
- [2] A. Vié, *Qualities, Challenges and Future of Genetic Algorithms: a Literature Review*, Oxford, United Kingdom: University of Oxford, 2021.
- [3] M. Negnevitsky, *Artificial Intelligence*, London, England: Addison-Wesley, 2005.
- [4] [Trực tuyến]. Available:
https://en.wikipedia.org/wiki/Linear_multistep_method.
- [5] [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Euler_method.
- [6] [Trực tuyến]. Available:
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html>.
- [7] Dương Đức Anh, Phạm Duy Tuấn and Nguyễn Quang Hoan, "Using Genetic Algorithms for Second-Order Cellular Neural Networks," in *IEEE Computer Society (CPS)*, Bangkok, Thailand, 2023.