# Chương 4 NHẬN DẠNG HOẠT ĐỘNG CỦA MẮT TỪ TÍN HIỆU EEG DÙNG MẠNG NƠ-RON TÍCH CHẬP

Nội dung chương này trình bày về cấu trúc của một mạng nơ-ron tích chập dùng để phân loại các tín hiệu EEG dựa trên cơ sở lý thuyết được trình bày ở chương 2. Các tín hiệu sau khi được lọc bởi các bộ lọc được tiến hành trong chương 3 được đưa vào huấn luyện bằng các cấu hình mạng nơ-ron tích chập được đề xuất bằng cách thay đổi hệ số của các lớp tích chập. Sau đó ứng dụng các phương pháp đánh giá, so sánh hiệu suất phân loại để tìm ra được cấu hình tốt nhất được trình bày và thảo luận.

## 4.1. Mạng nơ-ron tích chập

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

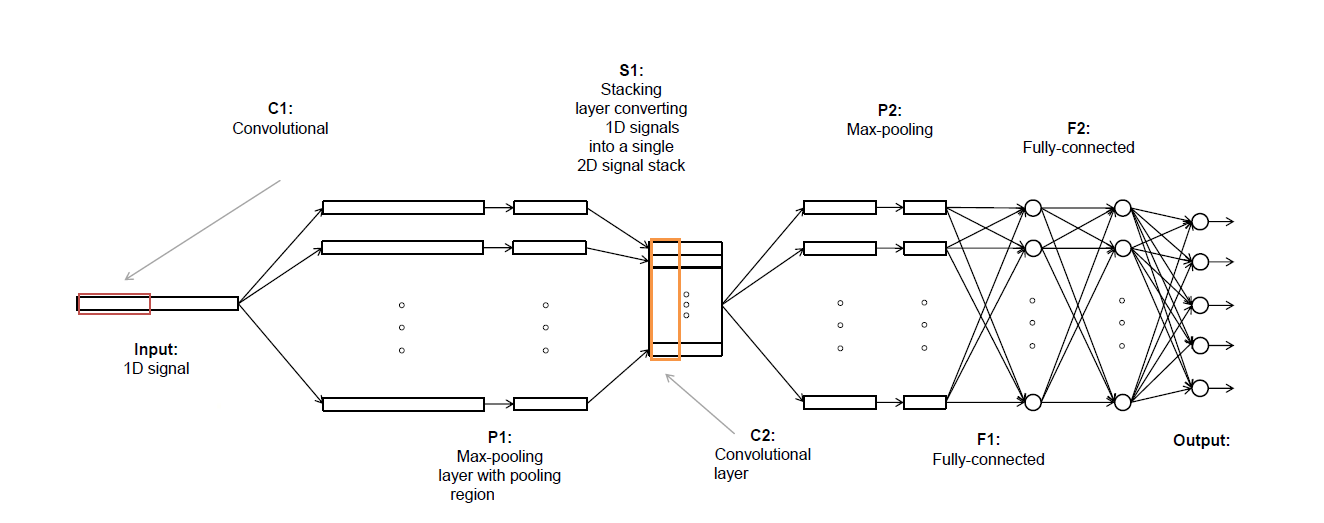
Một cải tiến của mạng nơ-ron nhân tạo là mạng nơ-ron tích chập (CNN). Cải thiện so với mạng nơ-ron nhân tạo về pha lẫn phép dịch chuyển bất biến.

CNN là một phần của kỹ thuật học sâu được ứng dụng mạnh trong xử lý và phân loại hình ảnh và tín hiệu.

Tương tự như mạng nơ-ron nhân tạo, kết quả ngõ ra của CNN dựa trên các trọng số và các bias của lớp trước đó trong cấu trúc mạng.

Các tham số dùng để huấn luyện mô hình CNN gồm lambda (regularization). Tốc độ học và momentum. Các tham số này có thể được điều chỉnh dựa trên bộ dữ liệu để có thể đạt được hiệu quả tốt nhất. Hệ số lambda dùng để kết quả không bị tản mát. Tốc độ học quy định tốc độ học của mô hình và momentum để hội tụ hóa dữ liệu [21].

Cấu trúc của mạng CNN cơ bản để giải quyết bài toán phân loại tín hiệu EEG [22] được giản đồ hóa thành các lớp được mô tả trong hình 4.1 của tác giả Rajedra Acharya và cộng sự.



Hình 4.1: Mô hình mạng nơ-ron tích chập của Rajedra Acharya

(1) **Input Layer**: Dữ liệu ngõ vào ở đây là các tín hiệu EEG.

(2) **Convolution Layer**: lớp tích chập bao gồm các bộ lọc (kernel) kích thước nhỏ lần lượt quét qua các tín hiệu EEG. Một bộ lọc là ma trận được chập với tín hiệu EEG đầu vào, tại mỗi trước bộ lọc sẽ chập với vùng đó được biểu diễn bằng phương trình 4.1:

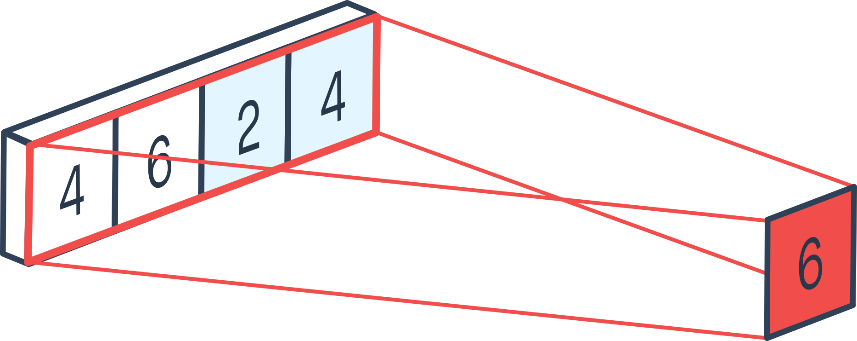
Trong đó x là tín hiệu, h là bộ lọc và N là số lượng phần tử của x. Vector ngõ ra là y

Ngõ ra của lớp chập được gọi là feature map.

(3) **rectified linear unit (ReLU)**: lớp hiệu chỉnh tuyến tính, ReLU là một hàm phi tuyến tính, chức năng của lớp này là chuyển toàn bộ giá trị âm từ kết quả của lớp tích chập (feature map) thành giá trị 0 mà vẫn giữ được sự tin cậy toán học của mạng.

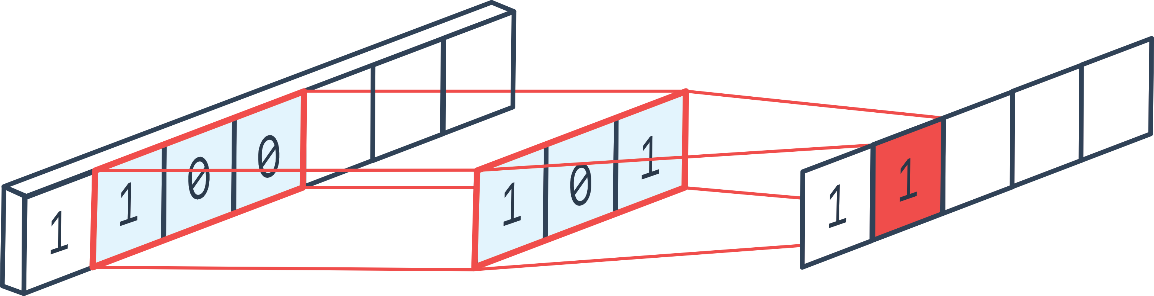
Nghĩa là . Đầu ra của lớp ReLU có kích thước giống với đầu vào, chỉ là tất cả các giá trị âm trong nó được loại bỏ.

(4) **Pooling layer**: lớp này giúp làm giảm kích thước của mẫu nhưng vẫn giữ được các đặc trưng quan trọng của mẫu. Khác với lớp tích chập, lớp này chỉ tiến hành lấy mẫu (subsampling) thay vì tích chập. Lớp Pooling lấy các mảng nhỏ từ lớp chập và tạo ra mẫu đặc trưng cho nó. Có 3 kiểu pooling là max pooling và mean pooling và min pooling được minh họa như trong hình 4.2.



Hình 4.2: Max pooling layer

Những điều này giúp làm giảm thiểu các bước tính toán và chống overfitting (tản mát dữ liệu) và lớp tích chập sẽ học được các thuộc tính lớn hơn. Max-pooling được sử dụng để chọn ra các giá trị lớn nhất của cửa sổ nhưng vẫn giữ được các đặc trưng quang trọng của chúng và giúp làm giảm số lượng nơ-ron ngõ ra được minh họa như trong hình 4.3.

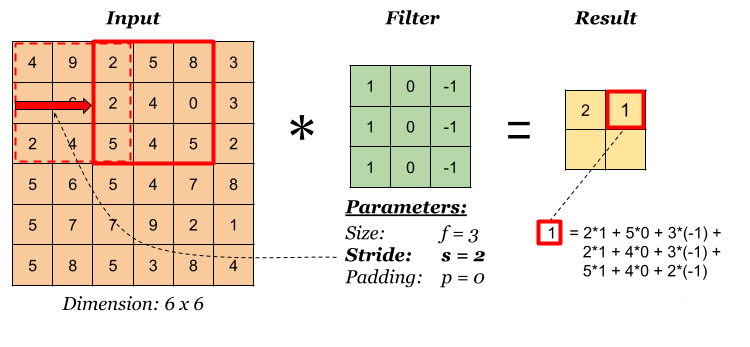


Hình 4.3: Cửa số trượt 1 chiều

CNN có thể tìm xem liệu một đặc trưng có nằm trong tín hiệu mà không cần quan tâm nó nằm ở đâu. Lớp pooling này có tính bất biến đối với kích thước của cửa sổ trượt. Kết quả được thể hiện qua biểu thức 4.2:

Trong đó là giá trị ngõ ra của kênh thứ , là chiều dài của chuỗi thời gian và là giá trị thứ của kênh thứ [13].

(5) **Stride**: quy định số pixel/phần tử phải dịch từ trái sang phải hoặc từ trên xuống dưới của mỗi kernel tới cuối ma trận ngõ vào. Stride thường được dùng để giảm kích thước của ma trận sau phép tính convolution. Phương pháp áp dụng stride trong phép tính tích chập ma trận được mô tả trong hình 4.4



Hình 4.4: Phép tính tích chập với stride = 2

(6) **Fully connected layer** (lớp kết nối đầy đủ): đây là lớp cuối cùng của CNN, là một mạng nơ-ron nhiều tầng sử dụng chức năng kích hoạt softmax trong lớp đầu ra. Mọi nơ-ron ở lớp trước được kết nối với tất cả các nơ-ron ở lớp kế tiếp, các nhãn sẽ được đánh giá ở đây dựa trên các thuộc tính đã được trích xuất. Mỗi một nơ-ron của lớp này sẽ liên kết tới mọi nơ-ron của lớp khác. Để đưa dữ liệu từ các lớp trước vào mạng này, buộc phải dàn phẳng dữ liệu ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại lớp cuối cùng sẽ sử dụng 1 hàm softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.

Biểu thức 4.3 tính toán phân phối xác suất của k lớp đầu ra. Do đó, Lớp sử dụng hàm softmax dựa vào các vector đầu vào đặc trưng đã được tính toán từ các lớp trước đó. Từ đó dự đoán tín hiệu EEG đầu vào thuộc lớp nào.

(7) **Output Layer** (lớp ngõ ra): là vector biểu diễn các lớp được định nghĩa ở các tín hiệu ngõ vào. Trong khuôn khổ đề tài, đây là một vector bao gồm dữ liệu đại diện cho các hoạt động của cơ thể [23].

## 4.2. Phương pháp đánh giá hiệu suất phân loại

Để xác định tính chính xác dữ liệu cần phân loại ta cần sử dụng các thuật toán machine learning để giải quyết yêu cầu này. Hiện tại có nhiều phương pháp để đánh giá độ chính xác của một bộ phân loại như confusion matrix, positive predictive value, null error rate, Cohen’s Kappa, F Score và ROC curve. Trong phạm vi và mục đích của để tài mà phương pháp confusion matrix sẽ được sử dụng để đánh giá tỉ lệ lỗi và từ đó đề xuất cấu hình huấn luyện tốt nhất. Confusion matrix là một mảng hiển thị các mối quan hệ giữa các lớp thực tế và các lớp dự đoán. Thông thường các biến số là một biến quan sát và một biến dự đoán. Mỗi hàng trong confusion matrix đại diện cho mốt lớp quan sát, mỗi cột đại diện cho một lớp dự đoán và mỗi ô là số các mẫu giao nhau giữa hai lớp. Bảng 4.1 là một ví dụ về bảng nhầm lẫn (confusion matrix) trong bộ phân loại số nhị phân:

**Bảng 4.1**: Bảng nhầm lẫn cho phân loại nhị phân

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Predicted Class | |
|  |  | Positive | Negative |
| Actual class | Positive | True Positive (TP) | False Negative (FN) |
| Negative | False Positive (FP) | True Negative (TN) |

Các định nghĩa trong confusion matrix:

* TP (true positive): mẫu mang nhãn dương được phân lớp đúng vào lớp dương.
* FN (false negative): mẫu mang nhãn dương bị phân lớp sai vào lớp âm.
* FP (false positive): mẫu mang nhãn âm bị phân lớp sai vào lớp dương.
* TN (true negative): mẫu mang nhãn âm được phân lớp đúng vào lớp âm. ịnh nghĩa cơ bản trong confusion matrix

Độ chính xác của mô hình được tính như sau:

Misclassification Rate: tỷ lệ phân loại sai, còn được gọi là “Tỷ lệ lỗi” (Error rate). Bằng 1 trừ đi độ chính xác (accuracy):

Precision: là tỷ lệ bộ phân loại dự đoán đúng là “nháy mắt trái/phải” và thực tế cũng là “nháy mắt trái/phải” so với tổng số lượng tín hiệu “nháy mắt trái/phải” dự đoán.

True positive rate/Sensitivity/Recall: là tỷ lệ bộ phân loại dự đoán là “nháy mắt trái/phải” thực tế cũng là “nháy mắt trái/phải” so với tổng số lượng “nháy mắt trái/phải” thực tế [24].

## 4.3. Cấu trúc mạng CNN cho phân loại tín hiệu EEG

Qua các công trình nghiên cứu đã tìm hiểu, tác giả chọn xây dựng cấu hình theo mô hình của Mô hình của Rajedra Acharya và cộng sự [10] vì có sự tương đồng về kiểu tín hiệu cần phân loại và hiệu suất phân loại mà cấu hình đạt được, sau đó có sự tùy chỉnh kích thước kernel và các lớp pooling cho thích hợp với kiểu dữ liệu đầu vào và theo yêu cầu giải quyết vấn đề của luận văn. Dựa vào hiệu suất phân loại để tìm ra được cấu hình tốt nhất cho bài toán phân loại của luận văn.

**Bảng 4.2**: Mô tả các lớp mạng trong mô hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 60 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 2 x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 50 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 40 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 40 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 4 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| output | Classoutput | - | - | - |

Theo bảng 4.2 ta có các lớp sau:

* Input: ngõ vào của tín hiệu cần nhận diện với kích thước được chọn là 7400x98
* conv0-1: lớp tích chập sử dụng 4 bộ lọc với kích thước mỗi bộ lọc là 60x1 phẩn tử, stride là 1.
* pool1-2: lớp max pooling có kích cỡ cửa sổ là 2 x 1 và stride là 2.
* conv2-3: lớp tích chập sử 4 bộ lọc có kích cỡ 50 x 1 phần tử, stride là 1.
* pool3-4: lớp max pooling có kích cỡ cửa sổ là 2 x 1 và stride là 2.
* conv4-5: lớp tích chập sử 10 bộ lọc có kích cỡ 40 x 1 phần tử, stride là 1.
* pool5-6: lớp max pooling có kích cỡ cửa sổ là 2 x 1 và stride là 2.
* conv6-7: lớp tích chập sử 10 bộ lọc có kích cỡ 40 x 1 phần tử, stride là 1.
* pool7-8: lớp max pooling có kích cỡ cửa sổ là 2 x 1 và stride là 2.
* conv8-9: lớp tích chập sử 15 bộ lọc có kích cỡ 4 x 1 phần tử, stride là 1.
* pool9-10: lớp max pooling có kích cỡ cửa sổ là 2 x 1 và stride là 2.
* fc10-11, fc11-12, fc12-13: lớp kết nối đầy đủ cuối cùng có số nơ-ron tương ứng với số danh tính cần nhận dạng lần lượt là 50, 20 và 2. Trong đó 2 chính là 2 dạng danh tính cần phân loại là nháy mắt trái và nháy mắt phải. Đi cùng với nó là hàm kích hoạt softmax.
* output: một lớp phân loại tính toán tổn thất Cross entropy cho các vấn đề phân loại đa lớp với các lớp loại trừ lẫn nhau [10].

## 4.4. Nhận dạng các hoạt động của mắt dùng CNN

### 4.4.1. Giới thiệu tập dữ liệu

Bộ dữ liệu được dùng trong luận văn là bộ dữ liệu tự thu thập gồm 2 thí nghiệm nháy mắt trái và nháy mắt phải. Protocol đã được giải thích trong chương 3. Tập dữ liệu bao gồm tổng cộng 140 tín hiệu được thu bởi 20 sinh viên thuộc bộ môn Kỹ Thuật Điện Tử - Y Sinh, thuộc Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh. Trong đó 80 tín hiệu thuộc về thí nghiệm nháy mắt trái và 60 tín hiệu của thí nghiệm nháy mắt phải. Trong số 140 tín hiệu, 30% được dùng để làm tập kiểm tra được chia đều tỉ lệ cho cả 2 thí nghiệm. Bảng 4.3 mô tả ngõ ra mong muốn của các tín hiệu.

Ban đầu, các tín hiệu sau khi thu bằng bồ thu Emotiv Epoc+ thì phần mềm sẽ xuất ra các file .csv chứa thông tin là giá trị theo thời gian của các lần thu bởi những người làm thí nghiệm. Dưới đây là tên các file dữ liệu:

* TN02\_L1\_DTL\_M\_09.07.20\_14.43.11.md.csv
* TN02\_L1\_KTDH\_F\_10.07.20\_10.40.43.md.csv
* TN02\_L1\_LHND\_M\_10.07.20\_12.30.12.md.csv
* TN02\_L1\_LTN\_M\_09.07.20\_13.22.04.md.csv
* TN02\_L1\_LTTL\_F\_10.07.20\_11.26.31.md.csv
* TN02\_L1\_NLYL\_F\_09.07.20\_15.09.54.md.csv
* TN02\_L1\_NNLT\_F\_10.07.20\_13.07.50.md.csv
* TN02\_L1\_NTH\_F\_10.07.20\_10.13.52.md.csv
* TN02\_L1\_TTH\_M\_10.07.20\_17.06.55.md.csv
* TN02\_L2\_DTL\_M\_09.07.20\_14.46.16.md.csv
* TN02\_L2\_KTDH\_F\_10.07.20\_10.42.42.md.csv
* TN02\_L2\_LHND\_M\_10.07.20\_12.31.38.md.csv
* TN02\_L2\_LTN\_M\_09.07.20\_13.19.24.md.csv
* TN02\_L2\_NLYL\_F\_09.07.20\_15.11.57.md.csv
* TN02\_L2\_NTH\_F\_10.07.20\_10.15.34.md.csv
* TN02\_L2\_TTH\_M\_10.07.20\_17.08.30.md.csv
* TN02\_L3\_DTL\_M\_09.07.20\_14.47.43.md.csv
* TN02\_L3\_LHND\_M\_10.07.20\_12.33.06.md.csv
* TN02\_L3\_LTN\_M\_09.07.20\_13.29.32.md.csv
* TN02\_L3\_TTH\_M\_10.07.20\_17.10.02.md.csv
* TN03\_L1\_DTL\_M\_09.07.20\_14.49.24.md.csv
* TN03\_L1\_LHND\_M\_10.07.20\_12.34.33.md.csv
* TN03\_L1\_LTN\_M\_09.07.20\_13.23.48.md.csv
* TN03\_L1\_NLYL\_F\_09.07.20\_15.13.48.md.csv
* TN03\_L1\_NTH\_F\_10.07.20\_10.21.15.md.csv
* TN03\_L1\_TTH\_M\_10.07.20\_17.12.37.md.csv
* TN03\_L2\_DTL\_M\_09.07.20\_14.50.52.md.csv
* TN03\_L2\_LHND\_M\_10.07.20\_12.37.03.md.csv
* TN03\_L2\_LTN\_M\_09.07.20\_13.26.13.md.csv
* TN03\_L2\_NLYL\_F\_09.07.20\_15.15.30.md.csv
* TN03\_L2\_NTH\_F\_10.07.20\_10.22.41.md.csv
* TN03\_L2\_TTH\_M\_10.07.20\_17.14.09.md.csv
* TN03\_L3\_DTL\_M\_09.07.20\_14.52.12.md.csv
* TN03\_L3\_LTN\_M\_09.07.20\_13.31.15.md.csv
* TN03\_L3\_TTH\_M\_10.07.20\_17.16.15.md.csv

Trong đó các file có tên bắt đầu bằng TN02 là các tín hiệu của thu được của thí nghiệm chớp mắt trái và TN03 là các tín hiệu thu được của thí nghiệm chớp mắt phải. F là người thí nghiệm mang giới tính là nữ và M là người thí nghiệm mang giới tính nam. Mỗi tín hiệu có độ lớn 7400 x 1 mẫu.

Tín hiệu được thu trong file .csv sẽ được trích lấy các tín hiệu từ các điện cực AF3, F7, AF4, F8 sau đó chuyển đổi sang file mat để thuận tiện trong việc xử lý trong matlab. Có tổng cộng 140 tín hiệu được chia ra 70%98 tín hiệu cho huấn luyện được đặt tên là Dx và 30%42 tín hiệu cho kiểm tra được đặt tên là Dy. Các file label (nhãn) được đánh dấu tùy theo loại thí nghiệm, được mô tả trong bảng 4.3. tên là Dx\_label và Dy\_label.

**Bảng 4.3**: Mô tả việc phân loại tín hiệu EEG và ngõ ra mong muốn

|  |  |
| --- | --- |
| Tín hiệu EEG | Ngõ ra mong muốn |
| Nháy mắt trái | A |
| Nháy mắt phải | B |

### 4.4.2. Mô tả phương pháp huấn luyện

Dưới đây mô tả các thông số khi cho huấn luyện mạng, được thực hiện với hàm “trainingOptions” trong Matlab cùng các đối số được thiết lập.

trainingOptions('sgdm','MaxEpochs',100, ...

'InitialLearnRate',0.0001,'Plots', 'training-progress');

* Tùy chọn huấn luyện cho quá trình tối ưu hóa, sgdm được sử dụng trong nghiên cứu này. Tùy chọn này giúp cập nhật các thông số mạng có thể học được trong một chu kỳ huấn luyện tùy chỉnh bằng cách sử dụng giảm độ dốc ngẫu nhiên để hoạt động tốt ngay cả với các bộ dữ liệu nhiễu hoặc phân tán.
* MaxEpochs là số lượng epoch (chu kỳ) tối đa được sử dụng cho huấn luyện. Iteration có thể hiểu là số lượng mini-batch cần để hoàn thành một epoch. Một epoch là một lần “duyệt” qua hết số lượng mẫu trong toàn bộ tập huấn luyện. Ở đây chọn MaxEpochs là 100.
* InitialLearnRate chỉ định tỷ lệ học ban đầu được sử dụng cho đào tạo là 0.0001. Nếu tỷ lệ học tập quá thấp, thì mất nhiều thời gian. Nếu tỷ lệ học tập quá cao, thì đào tạo có thể đạt được kết quả dưới mức tối ưu hoặc phân kỳ.
* Plots là biểu đồ để hiển thị trong quá trình đào tạo mạng và thiết lập là 'trainingprogress'. Nó hiển thị mini-batch loss và accuracy, validation loss và accuracy, và các thông tin khác trong quá trình đào tạo.

Ngoài ra còn nhiều thông số khác, tùy mục đích huấn luyện mà chỉnh sửa cho phù hợp với điều kiện thực tế. Việc huấn luyện nhằm mục đích phân loại đối tượng theo mong muốn, nghĩa là ngõ ra đã xác định trước. Nhiệm vụ của mạng là điều chỉnh các thông số sao cho ngõ ra gần đúng với mục tiêu nhất. Quá trình này gọi là huấn luyện có giám sát.

## 4.5. Các cấu hình mạng được sử dụng

Các cấu hình mạng được sử dụng trong luận văn được thể hiện trong bảng 4.4 tới 4.7:

**Bảng 4.4**: Cấu hình 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 60 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 2 x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 50 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 4 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 4 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 4 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| output | Classoutput | - | - | - |

Ở lớp đầu vào có 98 tín hiệu dùng để huấn luyện, mỗi tín hiệu lớp tích chập thứ nhất có kích cỡ 60 x 4, lớp tích chập thứ 2 có kích cỡ 50 x 4, lớp tích chập thứ 3 và 4 có kích cỡ 4 x 10, lớp tích chập thứ 5 có kích cỡ 4 x 15, các lớp max pooling lần lượt là 2 x 2, 2 x 1, 2 x 1, 2 x1, 2 x 1, có 3 lớp kết nối đầy đủ (fully-connected) có kích cỡ là 50, 20, và 2

**Bảng 4.5**: Cấu hình 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 60 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 2 x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 50 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 40 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 4 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 4 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| output | Classoutput | - | - | - |

Cấu hình 2 được mô tả ở bảng 4.5 có thay đổi so với cấu hình 1 tại lớp tích chập thứ 3 (conv 4-5), tăng kernel size từ 4 x 10 lên 40 x 10.

**Bảng 4.6**: Cấu hình 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 60 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 2 x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 50 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 40 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 40 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 4 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| Output | Classoutput | - | - | - |

Cấu hình 3 được mô tả trong bảng 4.6 tăng kernel size của lớp tích chập thứ 4 (conv 6-7) lên thành 40 x 10 so với cấu hình 1 và 2.

**Bảng 4.7**: Cấu hình 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 70 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 2 x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 60 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 50 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 40 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 40 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| output | Classoutput | - | - | - |

Cấu hình 4 được mô tả trong 4.7 tăng kernel size của lớp tích chập thứ 5 (conv8-9) lên thành 40 x 15.

**Bảng 4.8**: Cấu hình 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layers | Type | Numbers of neurons (Output Layer) | Kernel size for each output feature map | Stride |
| Input | Data Input | 7400 x 98 | - | - |
| conv0-1 | Convolutional | 80 x 4 | 4 | 1 |
| pool1-2 | Max-pooling | 4x 2 | 2 | 2 |
| conv 2-3 | Convolutional | 70 x 4 | 4 | 1 |
| pool 3-4 | Max-pooling | 4 x 1 | 2 | 2 |
| conv 4-5 | Convolutional | 60 x 10 | 10 | 1 |
| pool 5-6 | Max-pooling | 4 x 1 | 2 | 2 |
| conv 6-7 | Convolutional | 50 x 10 | 10 | 1 |
| pool 7-8 | Max-pooling | 2 x 1 | 2 | 2 |
| conv 8-9 | Convolutional | 50 x 15 | 15 | 1 |
| pool 9-10 | Max-pooling | 2 x 1 | 2 | 2 |
| fc10-11 | Fully-connected | 50 | - | - |
| fc11-12 | Fully-connected | 20 | - | - |
| fc12-13 | Fully-connected | 2 | - | - |
| sm | softmax | - | - | - |
| output | Classoutput | - | - | - |

Cấu hình 5 được mô tả trong bảng 4.8 tăng kernel size của lớp tích chập thứ 1 (conv0-1) lên thành 80 x 4, tăng kích thước của lớp pool1-2 lên 4 x 2, tăng kernel size của lớp tích chập thứ 1 (conv2-3) lên thành 70 x 4, tăng kích thước của lớp pool3-4 lên 4 x 1, tăng kernel size của lớp tích chập thứ 3 (conv4-5) lên thành 60 x 10, tăng kích thước của lớp pool5-6 lên 4 x 1, tăng kernel size của lớp tích chập thứ4 (conv6-7) lên thành 50 x 10.

## 4.6. Kết quả nhận dạng và so sánh hiệu suất nhận dạng của các mạng và các bộ lọc khác nhau

Độ chính xác trung bình sau khoảng 10 lần cho huấn luyện và kiểm tra của các cấu hình mạng tương ứng với các bộ lọc có bậc và kích thước cửa sổ thay đổi được mô tả trong bảng 4.9.

**Bảng 4.9**: Hiệu suất nhận dạng tín hiệu của từng cấu hình mạng

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Cấu hình số 1 | Cấu hình số 2 | Cấu hình số 3 | Cấu hình số 4 | Cấu hình số 5 |
| Tín hiệu gốc | 80.47% | 83.3% | 82.8% | 78.5% | 80.9% |
| Tín hiệu loại 1 | 80.9% | 82.5% | 81.3% | 82.9% | 81.2% |
| Tín hiệu loại 2 | 85.3% | 83.3% | 80.47% | 80.9% | 80.0% |
| Tín hiệu loại 3 | 84.8% | 85.7% | 83.9% | 82.9% | 85.7% |
| Tín hiệu loại 4 | 82.8% | 86.7% | 86% | 83.8% | 81.5% |
| Tín hiệu loại 5 | 82.3% | 85.7% | 84.3% | 88.6% | 86.2% |
| Tín hiệu loại 6 | 87.9% | 88.9% | 86.4% | 91.5% | 85.5% |
| Tín hiệu loại 7 | 87.6% | 87.6% | 91.83% | 87.3% | 88.6% |
| Tín hiệu loại 8 | 89.0% | 89.2% | 87.1% | 88.6% | 86.6% |

Trong đó, các bộ lọc được sử dụng trong trong chương 3 được đưa vào để tiến hành huấn luyện tương ứng với mỗi cấu hình như sau:

* Tín hiệu loại 1: Bộ lọc Savitzky có bậc bằng 2 và kích thước cửa số bằng 7.
* Tín hiệu loại 2: Bộ lọc Savitzky có bậc bằng 2 và kích thước cửa số bằng 11.
* Tín hiệu loại 3: Bộ lọc Savitzky có bậc bằng 3 và kích thước cửa số bằng 7.
* Tín hiệu loại 4: Bộ lọc Savitzky có bậc bằng 3 và kích thước cửa số bằng 11.
* Tín hiệu loại 5: Bộ lọc Savitzky có bậc bằng 4 và kích thước cửa số bằng 7.
* Tín hiệu loại 6: Bộ lọc Savitzky có bậc bằng 4 và kích thước cửa số bằng 11.
* Tín hiệu loại 7: Bộ lọc Savitzky có bậc bằng 5 và kích thước cửa số bằng 7.
* Tín hiệu loại 8: Bộ lọc Savitzky có bậc bằng 5 và kích thước cửa số bằng 11.

Hình 4.5: Biểu đồ so sánh hiệu suất phân loại của các bộ lọc

Từ bảng 4.9 và hình 4.4 thì các tín hiệu gốc và tín hiệu được lọc thì tín hiệu gốc có hiệu suất trung bình của các cấu hình thấp nhất. Tín hiệu loại 7 là bộ lọc có hiệu suất trung bình cao nhất so với các cấu hình khác.

Dữ liệu được huấn luyện với Tín hiệu loại 7 và cấu hình số 3 có hiệu suất phân loại trung bình cao nhất với tỉ lệ phân loại chính xác tới 91.83%.

Với định nghĩa của tỉ lệ phân loại sai được trình bày trong mục 4.2. Ta có bảng liệt kê tỉ lệ phân loại sai của các cấu hình được mô tả trong bảng 4.10.

**Bảng 4.10**: Tỷ lệ phân loại sai của các cấu hình

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Cấu hình số 1 | Cấu hình số 2 | Cấu hình số 3 | Cấu hình số 4 | Cấu hình số 5 |
| Tín hiệu gốc | 19.53% | 16.70% | 17.20% | 21.50% | 19.1% |
| Tín hiệu loại 1 | 19.10% | 17.50% | 18.70% | 17.10% | 18.8% |
| Tín hiệu loại 2 | 14.70% | 16.70% | 19.53% | 19.10% | 20% |
| Tín hiệu loại 3 | 15.20% | 14.30% | 16.10% | 17.10% | 14.3% |
| Tín hiệu loại 4 | 17.20% | 13.30% | 14% | 16.20% | 18.5% |
| Tín hiệu loại 5 | 17.70% | 14.30% | 15.70% | 11.40% | 13.8% |
| Tín hiệu loại 6 | 12.10% | 11.10% | 13.60% | 8.50% | 14.5% |
| Tín hiệu loại 7 | 12.40% | 12.40% | 8.17% | 13.70% | 11.4% |
| Tín hiệu loại 8 | 11% | 10.80% | 12.90% | 11.40% | 13.4% |

Trong hình 4.5, các hàng tương ứng với lớp dự đoán (Output Class) và các cột tương ứng với lớp thực (Target Class). Các ô chéo màu xanh tương ứng với các phân loại chính xác, ngoài đường chéo tương ứng với các phân loại không chính xác. Cột ở phía bên phải của ma trận là Precision - tỷ lệ phần trăm của tất cả các mẫu được dự đoán của mỗi lớp được phân loại chính xác (chữ xanh) và không chính xác (chữ đỏ). Hàng ở dưới cùng của ma trận là Recall - tỷ lệ phần trăm của tất cả các mẫu của mỗi lớp được phân loại chính xác và không chính xác. Ô ở dưới cùng bên phải của ma trận là độ chính xác tổng thể (Accuracy) và nó được dùng để đánh giá trong luận văn này. Cụ thể, độ chính xácnhận dạng của Tín hiệu loại 7 và cấu hình mạng số 3 thì hiệu suẩ phân loại chính xác đạt được là 92.9%.



Hình 4.6: Đánh giá độ chính xác trên bảng nhầm lẫn

Với mạng nơ-ron tích chập được dùng để phân loại tín hiệu sau khi đã thực hiện bước tiền xử lý bằng bộ lọc Savitzky với các cấu hình của bộ lọc khác nhau. Kết quả nhận dạng trung bình sau nhiều lần huấn luyện và kiểm tra các cấu hình mạng được thể hiện trong bảng 4.8 thì cấu hình mạng số 3 và tín hiệu loại 7 (Bộ lọc Savitzky có bậc bằng 5 và kích thước cửa số bằng 7) được dùng để xác định các hoạt động nháy mắt trái và nháy mắt phải của mắt. Cấu hình của bộ lọc cũng phù hợp với nhận xét về kết quả tín hiệu sau khi lọc đã được trình bày ở chương 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TÀI LIỆU THAM KHẢO  |  |  | | --- | --- | | [1] | Paul L. Nunez, Ramesh Srinivasan, "Electric Fields of the Brain: The Neurophysis of EEG," Proceeding IEEE, vol. 2, pp. 7-8, 2006. | | [2] | Yunyong Punsawad, Yodchanan Wongsawat, and Manukid Parnichkun, "Hybrid EEG-EOG Brain-Computer Interface System for Practical Machine Control," 32nd Annual International Conference of the IEEE EMBS Buenos Aires, pp. 2-5, August 31 - September 4, 2010. | | [3] | Chiron C, Jambaque I, Nabbout R, Lounces R, Sytora A and Dulac O, "The right brain hemisphere is dominant in human infants," IEEE, transactions on neural systems and rehabilitation, vol. 12, p. 120, 1997. | | [4] | Yeongjoon Gil, Ssanghee seo and Jungtae Lee, "EEG Analysis of Frontal Lobe Activities by decision stimuli," IEEE Second Internaltion Conference on Future Generation Communication and Networking, 2008. | | [5] | Nguyễn Thanh Nghĩa, "Ứng Dụng Wavelet Trong Xử Lý Tín Hiệu Não," 2012. | | [6] | M. Teplan, "Fundamentals of EEG Measurement," Measurement Science Review, vol. 2, 2002. | | [7] | Avsar Yerleskesi, "EEG signal classification using wavelet feature extraction and a mixture of expert model," Mathematical and Computational Applications, pp. 57-70, 2006. | | [8] | Olof Persson, Gilbert Strang, "Smoothing by Savitzky-Golay," pp. 3-5, 2005. | | [9] | R. Rojas, "Neural Networks," Springer-Verlag, Berlin, pp. 50-75, 1996. | | [10] | U. Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, Hojjat Adeli, "Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signal," Computers in Biology and Medicine, pp. 3-9, 2017. | | [11] | K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biological Cybernetics, vol. 36, pp. 193-202, 1980. | | [12] | Soumya Sen Gupta, Sumit Soman, P. Govind Raj, Rishi Prakash, S. Sailaja, Rupam Borgohain, "Detecting Eye Movements in EEG for Controlling Devices," IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom), Bali, Indonesia,, February 14, 2013. | | [13] | Jianhua Wang, Gaojie yu, Liu Zhong, Weihai Chen, Yu Sun, "Classification of EEG signal using convolution neural network," 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 2-7, 2019. | | [14] | Emotiv, "Emotiv Epoc+ User Manual," pp. 5-7, 2018. | | [15] | Jiang-Jian Guo, Rong Zhou, Li-Ming Zhao and Bao-Liang Lu, "Multimodal Emotion Recognition from Eye Image, Eye Movement and EEG Using Deep Neural Networks," 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, p. 6, 2019. | | [16] | J Satheesh Kumar, P Bhuvaneswari, "Analysis of Electroencephalography (EEG) Signals and Its Categorization - A Study," International Conference on Modeling, Optimization and Computing (ICMOC 2012), 2012. | | [17] | Carlos Escolano, Ander Ramos Murguialday, Tamara Matuz, Niels Birbaumer, and Javier Minguez, "A Telepresence Robotic System operated with a P300-based Brain-Computer Interface: Initial Tests with ALS patients," 32nd Annual International Conference of the IEEE EMBS Buenos Aires, Argentina, pp. 3-5, August 31 - September 4, 2010. | | [18] | H. V. Đạo, "Ngưỡng wavelet để xác định vùng vận động dựa vào fnirs," pp. 69-79, 2016. | | [19] | Abdulhamit Subasi, Ergun Ercelebi, "Classification of EEG signals using neural network and logistic regression," Applied Psychophysiology and Biofeedback, pp. 20-25, 2005. | | [20] | Bengio, Y. LeCun and Y., "Convolutional networks for images, speech, and time-series," MIT Press The Handbook of Brain Theory and Neural, pp. 7-11, 1995. | | [21] | W. S. Krumholz. A., "Quality Standards Subcommittee of the American Academy of Neurology," American Epilepsy Society, pp. 69-72, 2007. | | [22] | Tsinalis, O., P. M. Matthews, and Y. Guo, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," Annals of Biomedical Engineering, pp. 1-15, 2015. | | [23] | Zhiguang Wang, Weizhong Yan, and Tim Oates, "Time series classification from scratch with deep neural networks: A strong baseline," IEEE international joint conference on neural networks, pp. 157-1585, 2017. | | [24] | Sofia Visa, B. Ramsay, A. Ralescu, and E. VanDerKnaap, "Confusion Matrix-Based Feature Selection," Proceedings of The 22nd Midwest Artificial Intelligence and Cognitive Science Conference, 2011. | |

# PHỤ LỤC

**Code phần huấn luyện và tính accuracy**

%% Load data

clear ; clc;

load('Dx\_Sgolay\_5\_71.mat');

load('Dy\_Sgolay\_5\_71.mat');

load('DxLabel.mat');

load('DyLabel.mat');

Dx=Dx\_Sgolay';

Dy=Dy\_Sgolay';

Dx=reshape(Dx, [size(Dx,1),1,1,size(Dx,2)]);

Dy=reshape(Dy, [size(Dy,1),1,1,size(Dy,2)]);

DxLabel=categorical(DxLabel);

DyLabel=categorical(DyLabel);

convnet = [imageInputLayer([7400 1])

convolution2dLayer([60 1],4,'stride',1)

maxPooling2dLayer([2 1],'stride',2)

convolution2dLayer([50 1],4,'stride',1)

maxPooling2dLayer([2 1],'stride',2)

convolution2dLayer([40 1],10,'stride',1)

maxPooling2dLayer([2 1],'stride',2)

convolution2dLayer([40 1],10,'stride',1)

maxPooling2dLayer([2 1],'stride',2)

convolution2dLayer([4 1],15,'stride',1)

maxPooling2dLayer([2 1],'stride',2)

fullyConnectedLayer(50)

fullyConnectedLayer(20)

fullyConnectedLayer(2)

softmaxLayer

classificationLayer];

opts = trainingOptions('sgdm','MaxEpochs',100, ...

'InitialLearnRate',0.0001,'Plots', 'training-progress');

net = trainNetwork(Dx,DxLabel,convnet,opts);

% Test the Network

YTest = classify(net, Dy);

accuracy = sum(YTest == DyLabel)/numel(DyLabel)\*100

**Mã nguồn lọc dữ liệu**

%% Load data

clear ; clc; close all;

load('Dx.mat');

Dx=Dx';

for i = 1:98

Dx\_Sgolay(:,i) = sgolayfilt(Dx(:,i),3,11);

figure

plot(Dx(:,i)); hold on;

plot(Dx\_Sgolay(:,i));

end

Dx\_Sgolay = Dx\_Sgolay';