TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO HỌC MÁY

ĐỀ TÀI

PHÂN LOẠI PROTEIN DỰA TRÊN CHUỖI AMINO ACID

GIẢNG VIÊN: CAO VĂN CHUNG

Nhóm sinh viên thực hiện

Nguyễn Hoàng Việt	**22001658**
Nguyễn Thừa Tuân	**22001652**
Lương Vân Khoa	**22001602**

Hà Nội, 2024

Lời mở đầu

Trong thế giới sinh học, protein đóng vai trò rất quan trọng, với mỗi loại có chức năng, cấu trúc và hoạt động khác nhau. Để hiểu rõ về các đặc tính và chức năng của từng loại protein, việc phân loại chúng một cách chính xác là cần thiết. Phân loại protein không chỉ giúp chúng ta hiểu được sự đa dạng của các protein mà còn hỗ trợ trong các ứng dụng như phát hiện bệnh, chẩn đoán y tế, và thiết kế thuốc.

Amino axit là đơn vị cấu trúc cơ bản của protein. Một chuỗi amino axit dài có thể chứa từ hàng chục đến hàng nghìn amino axit, tạo nên sự phức tạp trong việc xác định cấu trúc và chức năng của protein. Mỗi chuỗi amino axit có một dãy các ký tự đặc trưng, biểu diễn mã di truyền mã hóa cho protein đó. Việc phân loại protein dựa vào chuỗi amino axit đòi hỏi phải sử dụng các phương pháp tiên tiến để xác định các tính chất đặc trưng như cấu trúc, chức năng và tương tác của chúng.

Trong nghiên cứu này, mục tiêu chính là áp dụng các phương pháp học máy và phân tích dữ liệu sinh học để phân loại các protein dựa trên chuỗi amino axit. Các đặc trưng từ chuỗi amino axit như chiều dài, tính acid-base, và các tính chất hóa học khác sẽ được trích xuất và sử dụng làm đầu vào cho các mô hình máy học. Quá trình này giúp tăng cường khả năng phân loại và dự đoán chính xác các loại protein, từ đó hỗ trợ cho việc nghiên cứu sâu hơn về sinh học phân tử.

Bằng cách áp dụng các phương pháp học máy, chúng ta có thể tận dụng sức mạnh của công nghệ để phân tích và tổ chức dữ liệu protein, từ đó mở rộng kiến thức của chúng ta về sinh học phân tử và thúc đẩy sự phát triển của các ứng dụng trong y học và công nghệ sinh học.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Chương 1 Giới thiệu đề tài

Phân loại protein dựa trên chuỗi amino acid là một bài toán quan trọng trong sinh học phân tử. Protein, với vai trò là thành phần cốt lõi trong cơ thể sống, đảm nhận nhiều chức năng quan trọng như xúc tác sinh học, vận chuyển chất, và cấu trúc tế bào. Chuỗi amino acid – các đơn vị cấu trúc cơ bản của protein – chứa thông tin về cấu trúc, chức năng và cơ chế hoạt động của chúng. Việc phân loại protein dựa trên đặc điểm này không chỉ giúp cải thiện hiểu biết về các quá trình sinh học mà còn mở rộng tiềm năng ứng dụng trong y học và công nghệ sinh học, đặc biệt là trong thiết kế thuốc và phát triển các liệu pháp điều trị mới.

Với sự phát triển của học máy (Machine Learning), các thuật toán hiện đại đã trở thành công cụ mạnh mẽ trong việc xử lý và phân tích dữ liệu sinh học. Học máy không chỉ giúp tự động hóa quá trình phân loại, mà còn tối ưu hóa độ chính xác khi làm việc với dữ liệu lớn và phức tạp. Trong nghiên cứu này, các phương pháp học máy được áp dụng để giải quyết bài toán phân loại protein. Nghiên cứu bao gồm việc chuẩn bị dữ liệu từ chuỗi amino acid, xây dựng các mô hình học máy cơ bản và nâng cao, đồng thời đánh giá hiệu quả của chúng thông qua các chỉ số đo lường hiệu suất.

Đề tài này không chỉ tập trung vào việc xây dựng các mô hình phân loại protein mà còn so sánh hiệu quả của chúng khi áp dụng trên các tập dữ liệu khác nhau. Qua đó, chúng tôi mong muốn cung cấp những đóng góp hữu ích trong lĩnh vực phân loại sinh học và ứng dụng học máy trong khoa học dữ liệu.

Chương 2 Nhắc lại về phần báo cáo giữa kỳ

2.1 Kết quả đạt được

Trong báo cáo giữa kỳ, nhóm đã cơ bản hoàn thành những yêu cầu mang tính nền tảng và quan trọng của đề tài. Cụ thể, nhóm đã tập trung xây dựng và triển khai các phần trọng yếu, bao gồm xử lý dữ liệu đồng thời thử nghiệm hai mô hình học máy trong việc phân loại Protein dựa trên dữ liệu là các chuỗi Amino axit.
Về xử lý dữ liệu, nhóm đã thu thập dữ liệu từ Kaggle và sử dụng nhiều bước để tiền xử lý dữ liệu, đó là:

- Tích hợp dữ liệu
- Loại bỏ dữ liệu trùng lặp và thiếu
- Lọc dữ liệu theo tần suất lớp
- Trích xuất đặc trưng từ chuỗi dữ liệu
- Chuẩn hóa dữ liệu
- Giảm chiều dữ liệu
- Chia dữ liệu thành tập train-test

Ngoài ra, nhóm cũng đã thực hiện phân cụm dữ liệu bằng phương pháp K-means với giá trị k tối ưu được tìm bằng thuật toán Elbow.
Bên cạnh đó, nhóm đã triển khai thử nghiệm các mô hình học máy Naive Bayes và Softmax Regression, qua đó đánh giá sơ bộ hiệu suất và khả năng áp dụng của các mô hình này vào bài toán. Cả hai mô hình đều cho ra kết quả phân loại chính xác khá cao trên cả dữ liệu gốc và dữ liệu đã giảm chiều. Những kết quả đạt được từ giai đoạn này đã giúp nhóm hiểu rõ hơn về môn học.
Đồng thời, ở báo cáo giữa kỳ, nhóm cũng đã nhận ra được những hạn chế cũng như định hướng được những điều cần cải thiện và những điều cần thử nghiệm thêm trong báo cáo cuối kỳ.

2.2 Những hạn chế

Việc sử dụng CountVectorizer với N-gram = 4 về mặt toán học có thể sinh ra 168420 cột dữ liệu mới, tạo thành một ma trận với kích thước N x 168420 (N là số lượng protein). Việc này làm tăng kích thước dữ liệu lên đáng kể. Bên cạnh đó do tính chất của dữ liệu, hầu hết các giá trị trong ma trận đều bằng 0. Do đó tạo thành một ma trận thưa (Sparse matrix). Điều này làm giảm đáng kể hiệu suất của mô hình, làm mô hình thiếu độ chính xác, thời gian huấn luyện kéo dài, cần nhiều tài nguyên để lưu trữ.

Ví dụ Chuỗi Protein

Giả sử chúng ta có 2 chuỗi protein sau:

`ACDEFGHIKLMNP và QRSTVWY`

Bước 1: Trích xuất N-Gram

Với n-gram = 4, chúng ta sẽ trích xuất tất cả các chuỗi 4 ký tự từ chuỗi protein trên. Các n-grams được tạo ra sẽ như sau:

`"ACDE", "CDEF", "DEFG", "EFGH", "FGHI", "GHIK", "HIKL", "IKLM", "LMNP","QRST","RSTV","STVW","T`

Bước 2: Tạo Ma Trận Xuất Hiện

Sau khi trích xuất, CountVectorizer sẽ tạo ra ma trận xuất hiện cho các n-grams. Mỗi hàng trong ma trận sẽ tương ứng với một n-gram, và mỗi cột sẽ đại diện cho một tài liệu (ở đây chỉ có một chuỗi duy nhất).

N-Gram	Chuỗi Protein 1	Chuỗi Protein 2
ACDE	1	0
CDEF	1	0
DEFG	1	0
EFGH	1	0
FGHI	1	0
GHIK	1	0
HIKL	1	0
IKLM	1	0
LMNP	1	0
QRST	0	1
RSTV	0	1
STVW	0	1
TVWY	0	1

Bảng 2.1: Ma trận xuất hiện cho n-grams của hai chuỗi protein.

Kết quả tạo thành một tập dữ liệu với kích thước rất lớn, nhiều ô chứa giá trị 0, làm tốn tài nguyên, giảm độ chính xác của các mô hình, gây phức tạp tính toán

2.3 Giải pháp

2.3.1 Chuẩn bị dữ liệu

Để đảm bảo dữ liệu đầu vào phù hợp với các mô hình học sâu, việc chuẩn bị dữ liệu cần được thực hiện một cách cẩn thận và nhất quán. Các bước dưới đây mô tả quy trình chuẩn bị dữ liệu cho bài toán.

- Sử dụng lớp Tokenizer trong Keras
- Chuẩn hóa chiều dài chuỗi protein
- Chuẩn hóa chiều dài chuỗi protein
- One-Hot Encoding

2.3.2 Sử dụng những mô hình phức tạp hơn

Để tăng tính đa dạng trong việc so sánh và đánh giá, các mô hình phức tạp hơn được đưa vào thử nghiệm, bao gồm:

- Support Vector Machine (SVM)
- Convolutional Neural Net-work (CNN)

Chương 3 Support Vector Machine

3.1 Giới thiệu

SVM là viết tắt của cụm từ support vector machine. Đây là một thuật toán khá hiệu quả trong lớp các bài toán phân loại nhị phân và dự báo của học có giám sát. Thuật toán này có ưu điểm là hoạt động tốt đối với những mẫu dữ liệu có kích thước lớn và thường mang lại kết quả vượt trội so với lớp các thuật toán khác trong học có giám sát.

3.2 Bài toán tối ưu

Bài toán tối ưu trong Support Vector Machine (SVM) chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là Maximum Margin Classifier.
Bài toán tối ưu trong SVM chính là bài toán tìm $\mathbf{w}$ và b sao cho margin đạt giá trị lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{\|\mathbf{w}\|_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}$$

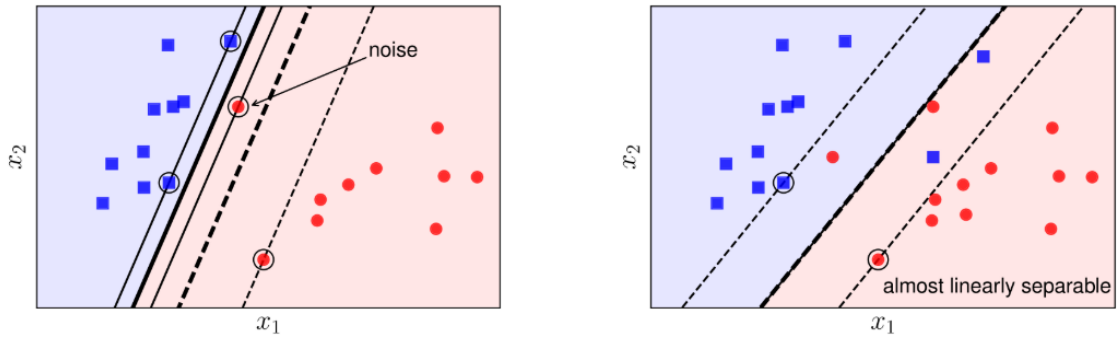
Nếu ta thay vector hệ số $\mathbf{w}$ bởi $k\mathbf{w}$ và b bởi kb, với k là một hằng số dương, thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi.
Dựa trên tính chất này, ta có thể giả sử:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

Qua một vài biến đổi đơn giản có thể đưa về bài toán sau:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{subject to:} \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \quad \forall n = 1, 2, \ldots, N$$



Hình 3.1: Hai trường hợp Hard Margin SVM làm việc không hiệu quả

Một biến thể của Hard Margin SVM có tên gọi là Soft Margin SVM. Với Soft Margin SVM, ta cộng thêm lượng slack variable nới trên tất cả các điểm trong vùng không an toàn và được hạn tổn thất mới:

$$J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n$$

trong đó $C > 0$ là hằng số điều chỉnh mức độ quan trọng giữa độ lớn margin và số điểm rơi vào vùng không an toàn (lượng hư sinh); còn

$$\xi = \{\xi_1, \xi_2, \ldots, \xi_N\}$$

Bài toán tối ưu cho Soft Margin:

$$\begin{aligned}(w,b,\xi_n) = \arg\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \\ \text{subject to:} \quad 1 - \xi_n - y_n(w^T x_n + b) \leq 0, \\ -\xi_n \leq 0, \quad \forall n = 1, 2, \ldots, N.\end{aligned}$$

Để giải quyết bài toán này, chúng ta áp dụng Kỹ thuật Lagrange Multiplier và KKT (Karush-Kuhn-Tucker) conditions để tìm ra nghiệm tối ưu.
Qua đó, ta có thể xác định được những điểm nào là điểm tựa (support vectors), cũng như tính toán giá trị của α_n.

3.3 Bài toán đối ngẫu

Hàm đối ngẫu của bài toán tối ưu là:

$$g(\lambda, \mu) = \min_{w,b} L(w, b, \lambda, \mu)$$

Bằng việc sử dụng đạo hàm Lagrangian, ta có được mối quan hệ giữa các tham số (λ, μ), và có thể sử dụng quy tắc Lagrangian để tìm ra:

$$C = \sum_{n=1}^{N}\lambda_n$$

Bài toán đối ngẫu được xác định bởi:

$$\begin{aligned}\lambda = \arg\max_{\lambda} g(\lambda) \\ \text{subject to:} \quad \sum_{n=1}^{N}\lambda_n y_n = 0, \\ 0 \leq \lambda_n \leq C, \ \forall n = 1, 2, \ldots, N.\end{aligned}$$

3.4 Bài toán tối ưu không ràng buộc cho Soft Margin

Bài toán có tối ưu có thể được đưa về bài toán tối ưu không ràng buộc và có thể giải được bằng Gradient Descent.
Bằng cách kết hợp một số điều kiện ràng buộc lại với nhau, ta có thể viết bài toán tối ưu dưới dạng:

$$(w,b,\xi_n) = \arg\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n$$

subject to:

$$\xi_n \geq \max(0, 1 - y_n(w^T x + b)), \quad \forall n = 1, 2, \ldots, N$$

Bằng phương pháp chứng minh phản chứng có thể suy ra:

$$\xi_n = \max(0, 1 - y_n(w^T x_n + b)), \quad \forall n = 1, 2, \ldots, N$$

Từ đó, thay các giá trị ξ_n bởi vế phải. Rõ ràng ξ_n không còn vai trò nữa, nên ta có thể lược bỏ. Bài toán trở thành bài toán tối ưu không ràng buộc:

$$(w, b) = \arg\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{n=1}^{N} \max(0, 1 - y_n(w^T x_n + b))$$

$$\triangleq \arg\min_{(w,b)} J(w, b).$$

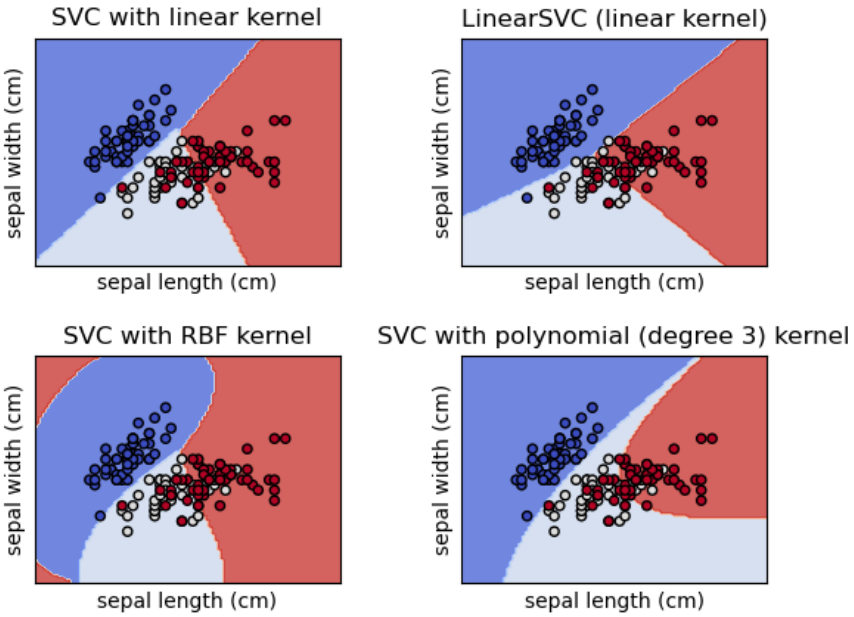
Cách xây dựng phần phạt trong hàm mất mát như trên còn gọi là hàm Hinge loss.

3.5 Bài toán Phân loại Protein

3.5.1 Cơ sở lựa chọn loại mô hình áp dụng

Đầu vào: Các chuỗi amino axit của một Protein
Đầu ra: Phân bố xác suất các lớp

SVM được chọn trong dự án phân loại các lớp protein vì khả năng xử lý dữ liệu lớn và nhiều đặc trưng phức tạp(CHAINID, structureMolecularWeight, phValue,...) như chuỗi amino axit. Dữ liệu này không phải lúc nào cũng rõ ràng và có thể phân tách dễ dàng, đặc biệt là khi các loại protein có sự tương đồng cao trong cấu trúc chuỗi. SVM giúp tạo ra một biên phân cách tốt nhất để phân loại các lớp protein này, ngay cả khi các dữ liệu không phân tách tuyến tính. Điều này rất quan trọng, vì chuỗi amino axit có thể có sự chồng lấn giữa các lớp.
Ví dụ: Hai chuỗi sau có cấu trúc gần giống nhau: ATGCGTACG và ATGCGUACG. Tuy nhiên một chuỗi thuộc lớp DNA, chuỗi còn lại thuộc lớp DNA/RNA Hybid.
Soft Margin SVM được sử dụng thay vì Hard Margin SVM vì dữ liệu thực tế không bao giờ hoàn hảo. Các chuỗi amino axit có thể có nhiễu, hoặc một số mẫu có thể không hoàn toàn thuộc về một lớp duy nhất, ví dụ như các protein có tính chất gần nhau. Soft Margin cho phép một chút "lỗi" trong việc phân tách các lớp, giúp mô hình học tốt hơn và không bị overfitting.
Các mô hình khác như One-Class SVM không phù hợp cho bài toán phân loại đa lớp của bạn. Bạn cần phân loại nhiều loại protein khác nhau, và Soft Margin SVM là lựa chọn tốt nhất vì nó hỗ trợ việc phân loại đa lớp hiệu quả hơn.
Với phương pháp này, dự án của bạn áp dụng Kernel SVM, cho phép mô hình xử lý dữ liệu không tuyến tính, và chính sự lựa chọn này giúp phân loại chính xác các loại protein từ dữ liệu chuỗi amino axit, tối ưu hóa hiệu suất phân loại.



Hình 3.2: Minh họa cách các kernel hoạt động

3.5.2 Bước xử lý và các thông số lựa chọn cho mô hình

Kernel

Việc lựa chọn loại Kernel rất quan trọng vì nó liên quan đến đặc điểm của dữ liệu và mục tiêu của bai toán.
Cụ thể, kernel được tỏ ra phù hợp nhất là "poly" bởi nó cung cấp sự cân bằng giữa độ phức tạp và khả năng tổng quát, đồng thời tối ưu về mặt tính toán, phù hợp với mục tiêu phân loại protein trong một không gian tính toán hạn chế.
Với hai kernel còn lại, "linear" tỏ ra kém hiệu quả nhất (đạt hiệu suất chỉ trên dưới 20%) vì dữ liệu là các chuỗi amino axit có thể có các đoạn tương tự nhưng cấu trúc lại hoàn toàn khác nhau dẫn đến việc không thể đơn giản hóa bằng bài toán tuyến tính. Trong khi đó "rbf" có khả năng học phi tuyến mạnh mẽ nhưng lại dễ xảy ra overfitting và đặc biệt là vì cấu trúc dữ liệu là các chuỗi rất dài nên tính toán rất phức tạp dẫn đến tốn rất nhiều thời gian và tài nguyên.

Các thông số khác

Degree = 3: Dữ liệu chuỗi thường rất dài và chứa nhiều đặc điểm phức tạp, do đó cần một độ phức tạp vừa phải
C = 1: giúp mô hình không quá nghiêm ngặt với các điểm phân loại sai (như trong Hard Margin SVM), đồng thời vẫn giữ được một ranh giới phân lớp có độ chính xác cao.
Gamma = "scale": Dữ liệu chứa những đặc trưng rất khác nhau về tỷ lệ amino axit, độ dài chuỗi, hoặc các tính chất sinh học.'scale' sẽ tự động điều chỉnh tỷ lệ giữa các điểm dữ liệu và giúp mô hình tránh được tình trạng quá phụ thuộc vào những điểm đặc biệt trong chuỗi amino axit.

Ngoài ra, nhãn thường lưu dưới dạng one-hot coding. Tuy nhiên, thuật toán SVM yêu cầu đầu vào nhãn phải là các chỉ số lớp (index) thay vì vector one-hot. Sử dụng argmax() để chuyển nhãn thành 1D giúp mô hình có thể xử lý dữ liệu một cách hiệu quả. Cụ thể, mỗi phần tử trong y_train_1D sẽ đại diện cho lớp của từng mẫu trong tập huấn luyện, thay vì là một vector nhiều chiều.

Chương 4 Convolutional Neural Network

4.1 Khái niệm về mạng neural tích chập

Mạng nơ ron tích chập là một trong những mạng truyền thẳng đặc biệt. Mạng nơ ron tích chập là một mô hình học sâu phổ biến và tiên tiến nhất hiện nay. Hầu hết các hệ thống nhận diện và xử lý ảnh hiện nay đều sử dụng mạng nơ ron tích chập vì tốc độ xử lý nhanh và độ chính xác cao. Trong mạng nơ ron truyền thống, các tầng được coi là một chiều, thì trong mạng nơ ron tích chập, các tầng được coi là 3 chiều, gồm: chiều cao, chiều rộng và chiều sâu . Mạng nơ ron tích chập có hai khái niệm quan trọng: kết nối cục bộ và chia sẻ tham số. Những khái niệm này góp phần giảm số lượng trọng số cần được huấn luyện, do đó tăng nhanh được tốc độ tính toán.
Convolutional Neural Networks (CNN) là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng thị giác máy tính (Computer Vision). CNN được dùng trong trong nhiều bài toán như nhân dạng ảnh, phân tích video, ảnh MRI, hoặc cho bài các bài của lĩnh vự xử lý ngôn ngữ tự nhiên,và hầu hết đều giải quyết tốt các bài toán này

4.2 Mô hình mạng neural tích chập

Một kiến trúc CNN bao gồm các lớp: convolution layer, pooling layer và fully connected layer. Ở giữa các lớp convolution và pooling thường có các hàm kích hoạt phi tuyến. Dữ liệu khi đưa vào mạng sẽ được lan truyền qua tầng convolution layer, giá trị tính được từ các tầng convolution sẽ đi qua một hàm kích hoạt, sau đó giá trị này sẽ được lan truyền qua pooling layer. Cuối cùng dữ liệu sẽ được lan truyền đến tầng fully connected layer và đi qua hàm kích hoạt Softmax, thường thì cuối cùng sẽ thu được một vector chứa xác suất phần trăm thuộc về các lớp đối với các bài toán phân loại.

4.2.1 Convolution1D layer (tích chập một chiều)

Convolution layer là lớp quan trọng nhất và cũng là lớp đầu tiên của của mô hình CNN. Lớp này có chức năng chính là phát hiện các đặc trưng có tính không gian hiệu quả. Trong tầng này có 4 đối tượng chính là: ma trận đầu vào, bộ filters, và receptive field, feature map. Conv layer nhận đầu vào là một ma trận 3 chiều và một bộ filters cần phải học. Bộ filters này sẽ trượt qua từng vị trí trên bức ảnh để tính tích chập (convolution) giữa bộ filter và phần tương ứng trên bức ảnh. Phần tương ứng này trên bức ảnh gọi là receptive field, tức là vùng mà một neuron có thể nhìn thấy để đưa ra quyết định, và mà trận cho ra bởi quá trình này được gọi là feature map.
Tích chập là một khái niệm trong xử lý tín hiệu số nhằm biến đổi thông tin đầu vào thông qua một phép tích chập với bộ lọc để trả về đầu ra là một tín hiệu mới. Tín hiệu này sẽ làm giảm những đặc trưng mà bộ lọc không quan tâm và chỉ giữ những đặc trưng chính.
Tích chập thông dụng nhất là tích chập 2 chiều được áp dụng trên ma trận đầu vào và ma trận bộ lọc 2 chiều. Phép tích chập của một ma trận $X \in \mathbb{R}^{W_1 \times H_1}$ với một bộ lọc (receptive field) $F \in \mathbb{R}^{F \times F}$ là một ma trận $Y \in \mathbb{R}^{W_2 \times H_2}$ sẽ trải qua những bước sau:

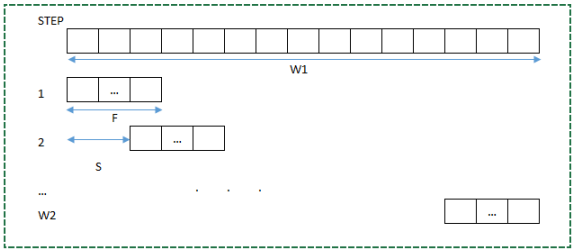
1. **Tính tích chập tại 1 điểm:** Tại vị trí đầu tiên trên cùng của ma trận đầu vào, ta sẽ lọc ra một ma trận con $X_{\text{sub}} \in \mathbb{R}^{F \times F}$ có kích thước bằng với kích thước của bộ lọc. Giá trị y_{11} tương

ứng trên Y là tích chập của X_{sub} với F được tính như sau:

$$y_{11} = \sum_{i=1}^{F} \sum_{j=1}^{F} x_{ij} f_{ij}$$

2. **Tiến hành trượt dọc theo ma trận:** Theo chiều từ trái qua phải, từ trên xuống dưới, với bước nhảy (stride) S, ta sẽ tính được các giá trị y_{ij} tiếp theo. Sau khi quá trình này kết thúc, ta thu được trọn vẹn ma trận Y.

Trong một mạng nơ-ron tích chập, các lớp liền sau lấy đầu vào từ lớp liền trước nó. Do đó, để hạn chế lỗi trong thiết kế mạng nơ-ron, chúng ta cần xác định kích thước đầu ra ở mỗi lớp. Điều đó có nghĩa là dựa vào kích thước ma trận đầu vào (W_1, H_1), kích thước bộ lọc (F, F) và bước nhảy S để xác định kích thước ma trận đầu ra (W_2, H_2).
Xét quá trình trượt trên chiều W_1 của ma trận đầu vào.



Hình 4.1: Quá trình trượt theo chiều W1.

Giả sử quá trình này sẽ dừng sau W_2 bước. Tại bước đầu tiên, quá trình đi được đến vị trí thứ F. Sau mỗi bước liền sau sẽ tăng so với vị trí liền trước là S. Như vậy đến bước thứ i, quá trình trượt sẽ đi đến vị trí $F + (i-1)S$. Suy ra tại bước cuối cùng W_2, ma trận sẽ đi đến vị trí $F + (W_2 - 1)S$. Đây là vị trí lớn nhất gần với vị trí cuối cùng là W_1. Trong trường hợp lý tưởng thì $F + (W_2 - 1)S = W_1$. Từ đó ta suy ra:

$$W_2 = \frac{W_1 - F}{S} + 1$$

Khi vị trí cuối cùng không trùng với W_1 thì số bước W_2 sẽ được lấy phần nguyên:

$$W_2 = \lfloor \frac{W_1 - F}{S} \rfloor + 1$$

Chúng ta luôn có thể tạo ra đẳng thức (1) nhờ thêm phần đường viền (padding) tại các cạnh của ảnh với độ rộng viền là P sao cho phép chia cho S là chia hết. Khi đó:

$$W_2 = \frac{W_1 + 2P - F}{S} + 1$$

Hoàn toàn tương tự ta cũng có công thức ứng với chiều cao:

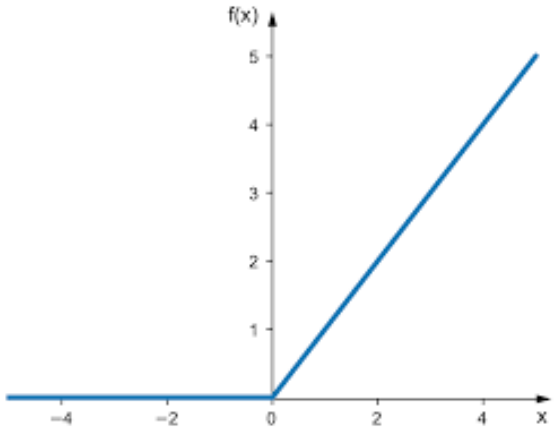
$$\frac{H_2 = H_1 + 2P - F}{S} + 1$$

4.2.2 Activation function

Hàm kích hoạt là một hàm số nhận vào một giá trị đầu vào và kết quả là một giá trị có miền giá trị nằm trên một khoảng (hay nửa khoảng) nào đó. Một số các hàm kích hoạt phổ biến có thể kể đến đó là Sigmod, Tanh, Relu. Hàm kích hoạt rất quan trọng bởi vì nó sẽ tăng khả năng dự đoán của mạng neural và giúp mô hình học được các quan hệ phi tuyến phức tạp tiềm ẩn trong dữ liệu. Thông thường hàm kích hoạt sử dụng ở giữa các tầng convolution và pooling là hàm Relu.

ReLU

Hàm Relu có công thức toán học là **f(x) = max(0,x)**. Hàm Relu được ưa chuộng vì tính toán đơn giản, giúp hạn chế tình trạng vanishing gradient, và cũng cho kết quả tốt hơn. Relu cũng như những hàm kích hoạt khác, được đặt ngay sau tầng convolution, Relu sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0.



Hình 4.2: Đồ thị hàm số hàm ReLU.

Softmax

Hàm softmax là một hàm kích hoạt (activation function) phổ biến trong mạng neural, đặc biệt ở các bài toán phân loại đa lớp (multi-class classification). Nó chuyển đổi một vector đầu vào (gồm các giá trị thực) thành một phân phối xác suất, trong đó tổng các xác suất bằng 1.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

- z_i: Giá trị đầu vào tại vị trí i.
- n: Số phần tử trong vector đầu vào.
- $\sigma(z_i)$: Giá trị xác suất tại vị trí i.

4.2.3 Pooling layer

Trong mạng nơ-ron tích chập (Convolutional Neural Network - CNN), **Pooling Layer** là một lớp dùng để giảm kích thước không gian của đầu vào, nhằm:

- Giảm số lượng tham số và tính toán.
- Kiểm soát hiện tượng quá khớp (*overfitting*).
- Tăng tính bất biến đối với các thay đổi nhỏ, như dịch chuyển hoặc nhiễu.

Các loại Pooling phổ biến

1. **Max Pooling (Gộp cực đại):**
   - Lấy giá trị lớn nhất trong một vùng cụ thể của ma trận đầu vào.
   - Công thức:

$$y = \max(x)$$

- Ví dụ: Với vùng
$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix},$$
giá trị gộp cực đại là 4.

2. **Average Pooling (Gộp trung bình):**
   - Tính giá trị trung bình của các giá trị trong một vùng cụ thể.
   - Công thức:
$$y = \frac{1}{N} \sum x,$$
với N là số lượng giá trị trong vùng.
   - Ví dụ: Với vùng
$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix},$$
giá trị gộp trung bình là
$$\frac{1+3+2+4}{4} = 2.5.$$

4.2.4 Fully connected layer

Fully Connected Layer (Lớp kết nối đầy đủ) là một lớp trong mạng nơ-ron, nơi mỗi nơ-ron trong lớp này được kết nối với *tất cả* các nơ-ron của lớp trước đó. Đây là một thành phần quan trọng trong mạng nơ-ron nhân tạo (Artificial Neural Network - ANN), thường được sử dụng ở cuối mạng để tạo ra đầu ra.

Đặc điểm của Fully Connected Layer

- **Kết nối đầy đủ:** Mỗi nơ-ron đầu vào được kết nối với tất cả các nơ-ron trong lớp này thông qua các trọng số.
- **Trọng số học được:** Các kết nối giữa các nơ-ron có trọng số (w_{ij}) được tối ưu hóa trong quá trình huấn luyện.
- **Tăng khả năng học:** Lớp này tổng hợp thông tin từ các lớp trước, giúp mô hình học được các mối quan hệ phức tạp giữa các đặc trưng.

Hoạt động của Fully Connected Layer

Đầu ra của mỗi nơ-ron được tính thông qua một phép nhân ma trận và cộng dồn, sau đó áp dụng hàm kích hoạt:

$$y_i = f\left(\sum_{j=1}^{n} w_{ij} x_j + b_i\right),$$

trong đó:

- x_j: Đầu vào từ lớp trước.
- w_{ij}: Trọng số giữa nơ-ron j của lớp trước và nơ-ron i của lớp hiện tại.
- b_i: Bias của nơ-ron i.
- f: Hàm kích hoạt, chẳng hạn như ReLU, sigmoid, hoặc softmax.

Hạn chế của Fully Connected Layer

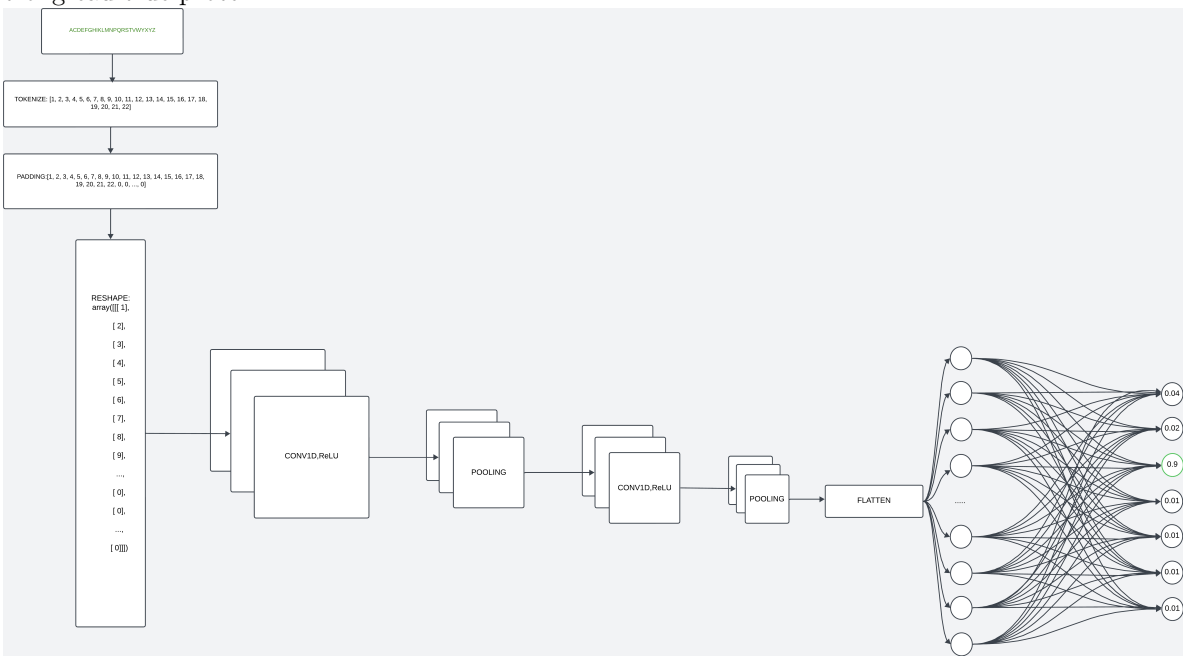
- **Nhiều tham số:** Lớp này có rất nhiều tham số, dễ dẫn đến hiện tượng quá khớp (*overfitting*) nếu không có đủ dữ liệu hoặc không sử dụng các kỹ thuật như dropout hoặc regularization.
- **Tốn tài nguyên:** Đòi hỏi nhiều bộ nhớ và thời gian tính toán, đặc biệt với các đầu vào có kích thước lớn.

4.3 Bài toán phân loại protein

4.3.1 Cơ sở lý thuyết, lý do lựa chọn mô hình

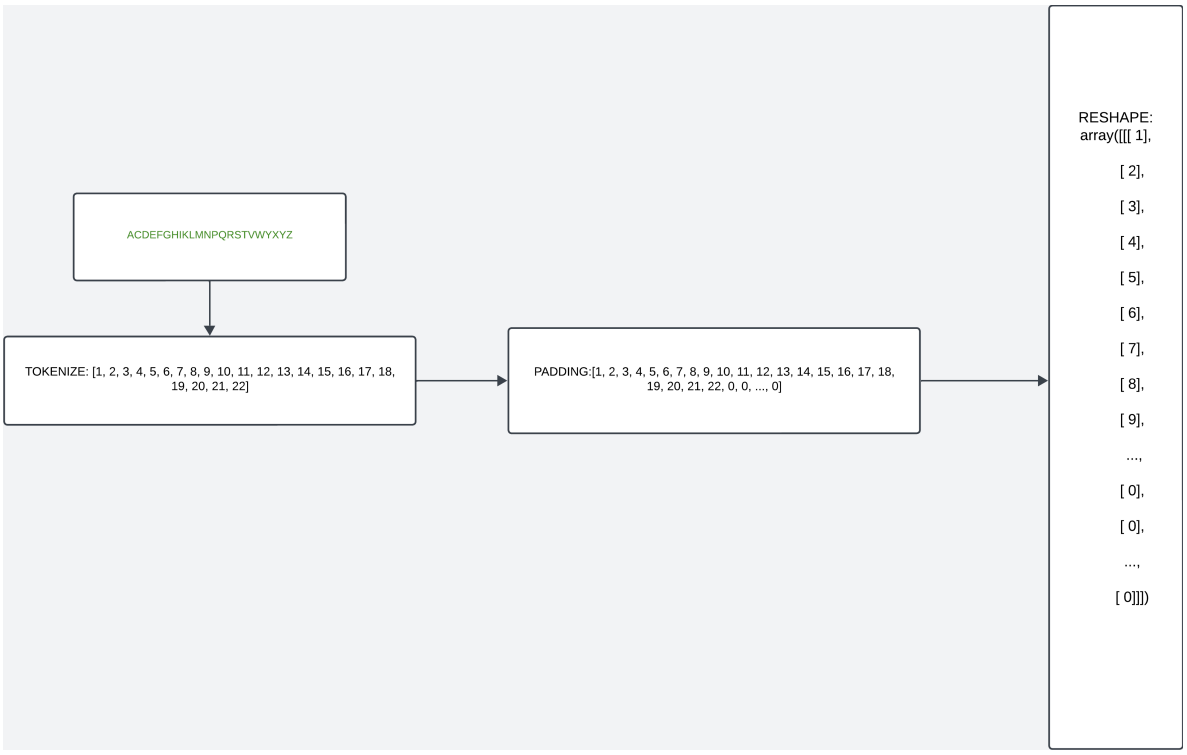
Đầu vào: chuỗi aa của một protein
Đầu ra: Phân bố xác suất của các lớp

Pha huấn luyện bao gồm các bước: mã hóa chuỗi aa bằng lớp Tokenizer trong Keras. Thêm khoảng đệm (padding) để chuẩn hóa kích thước chuỗi protein. Thay đổi kích thước để phù hợp với đầu vào của mô hình CNN. Sử dụng một kiến trúc mạng neural tích chập (Convolution neural network) để huấn luyện.
Việc sử dụng Tokenizer có thể biến các chuỗi aa thành các tensor tương tự như các bức ảnh, phù hợp với đầu vào của mạng neural tích chập. Mô hình CNN có thể học được các đặc trưng như sự lặp lại, các đoạn subsequence, và các cấu trúc tương quan lân cận trong chuỗi amino acid. Điều này giúp phát hiện các thông tin quan trọng như vùng liên kết, vùng chức năng, hay các yếu tố quan trọng khác trong cấu trúc protein



Hình 4.3: Sơ đồ tổng quan phương pháp đề xuất.

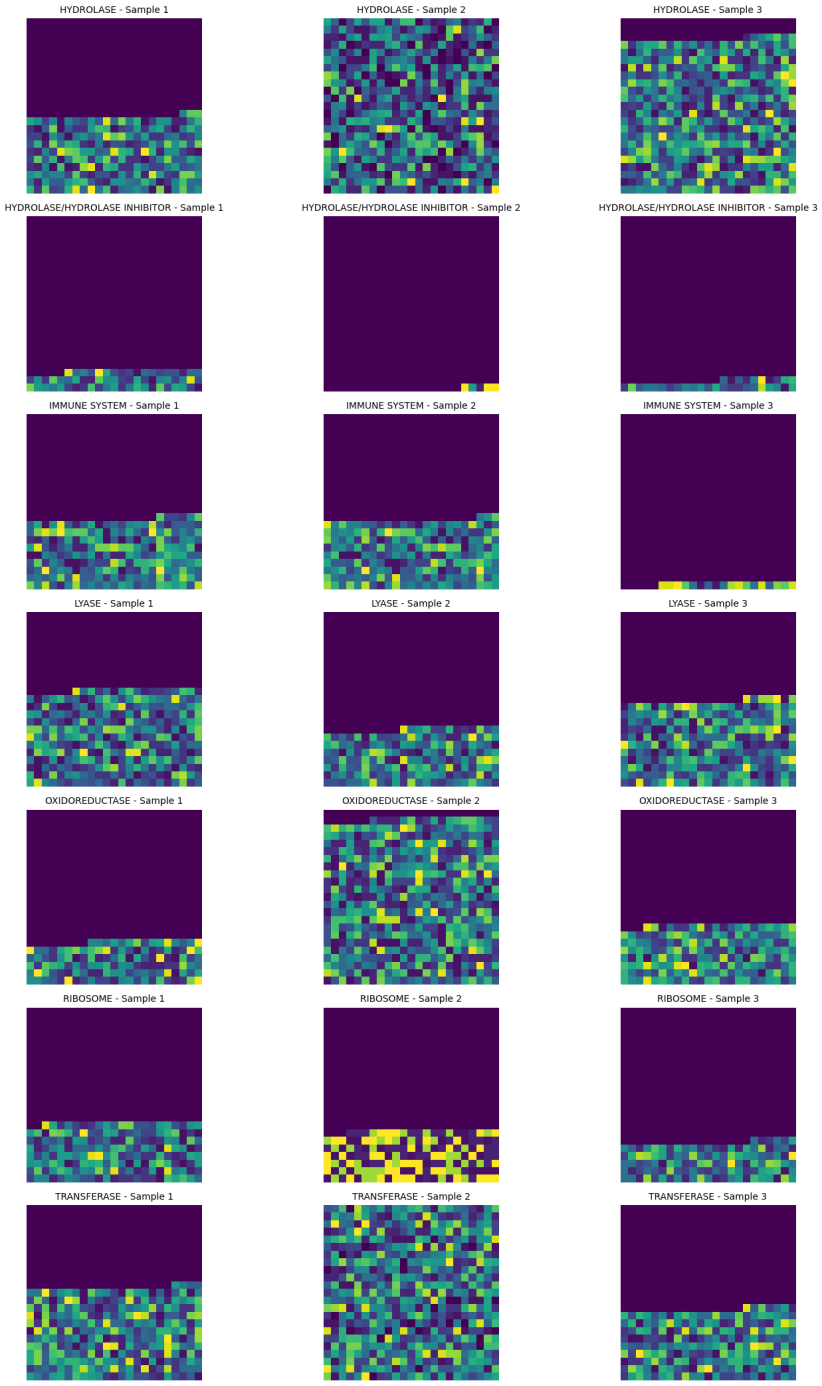
4.3.2 Tiền xử lý chuỗi protein



Hình 4.4: Sơ đồ tổng quan các bước tiền xử lý dữ liệu được áp dụng.

Tokenizer trong Keras là một công cụ mạnh mẽ để chuyển đổi văn bản thành các dạng mà máy tính có thể xử lý, cụ thể là các chuỗi số nguyên. Dưới đây là cách thức hoạt động chi tiết của Tokenizer:

- **Khởi tạo Tokenizer:**Có thể chỉ định một số tham số tùy chọn như num_words (số lượng từ tối đa sẽ được giữ lại) hoặc char_level (đặc biệt nếu bạn muốn phân tích từng ký tự thay vì từ). Ỏ bài toán này chúng ta sử dụng char_level
- **Fit tokenizer vào dữ liệu:**Phân tích dữ liệu đầu vào (các chuỗi văn bản) để xác định các từ hoặc ký tự (nếu là char_level=True). Tạo ra một từ điển, trong đó mỗi ký tự được gán một chỉ số duy nhất.
- **Tạo từ điển:**Từ điển được tạo ra chứa tất cả các từ hoặc ký tự duy nhất từ dữ liệu, được sắp xếp theo tần suất xuất hiện (từ thường xuyên nhất được gán chỉ số nhỏ nhất).



Hình 4.5: Minh họa chuỗi protein sau khi được xử lý bằng heatmap 23x23.

Lưu ý: trên đây chỉ là hình ảnh phục vụ cho việc minh họa, không phải là dữ liệu đầu vào của mạng neural tích chập. Dữ liệu đầu vào là một tensor có kích thước (529,1).

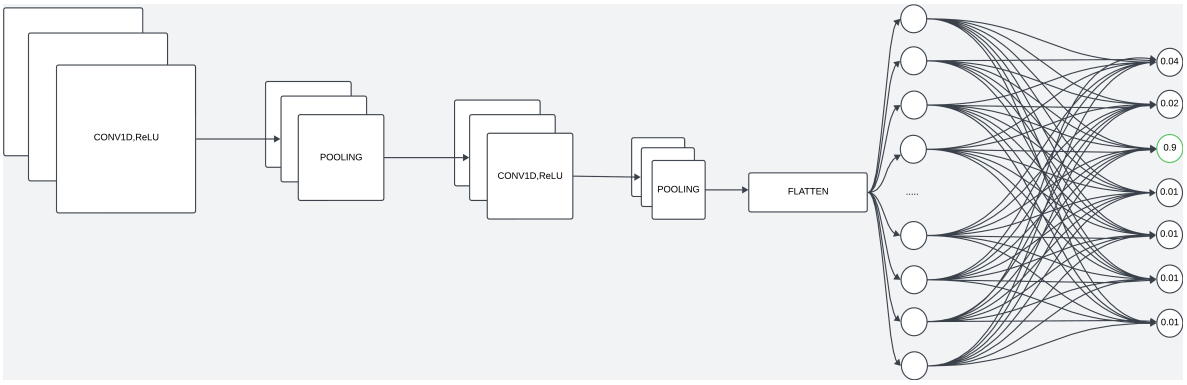
4.4 Mạng neural tích chập cho phân lớp protein

4.4.1 Kiến trúc mạng neural tích chập (Convolutional Neural Network)

Mô hình đề xuất có kiến trúc CNN như sau:

Layer	Output Shape	Param
Conv1D, ReLU	(None, 529, 64)	448
MaxPooling1D	(None, 264, 64)	0
Conv1D, ReLU	(None, 264, 32)	6176
MaxPooling1D	(None, 132, 32)	0
Flatten	(None, 4224)	0
Dense, ReLU	(None, 128)	540800
Dense, softmax	(None, 7)	903

Bảng 4.1: Neural Network Layers Summary



Hình 4.6: Sơ đồ tổng quan kiến trúc mạng neural tích chập được đề xuất

Ý nghĩa của convolution layer đầu tiên trong bài toán này:

- **Nhận diện đặc trưng địa phương:** Lớp Convolution 1D có khả năng nhận diện các đặc trưng như sự lặp lại, các đoạn subsequence, và các cấu trúc tương quan lân cận trong chuỗi amino acid. Điều này giúp phát hiện các thông tin quan trọng như vùng liên kết, vùng chức năng, hay các yếu tố quan trọng khác trong cấu trúc protein.
- **Giảm độ phức tạp:** Convolution 1D giảm số chiều của dữ liệu từ input (chuỗi dài) xuống những đại diện rút gọn hơn mà vẫn giữ được thông tin quan trọng. Điều này giúp tăng hiệu quả xử lý và rút ngắn thời gian tính toán mà vẫn giữ lại đặc trưng chính của dữ liệu.
- **Học các đặc trưng không gian:** Lớp Convolution 1D giúp học các đặc trưng có liên quan đến sự sắp xếp, thứ tự các amino acid trong chuỗi protein. Các đặc trưng này giúp phân biệt các mẫu protein khác nhau dựa trên cấu trúc.

Ý nghĩa của convolution layer thứ hai trong bài toán này:

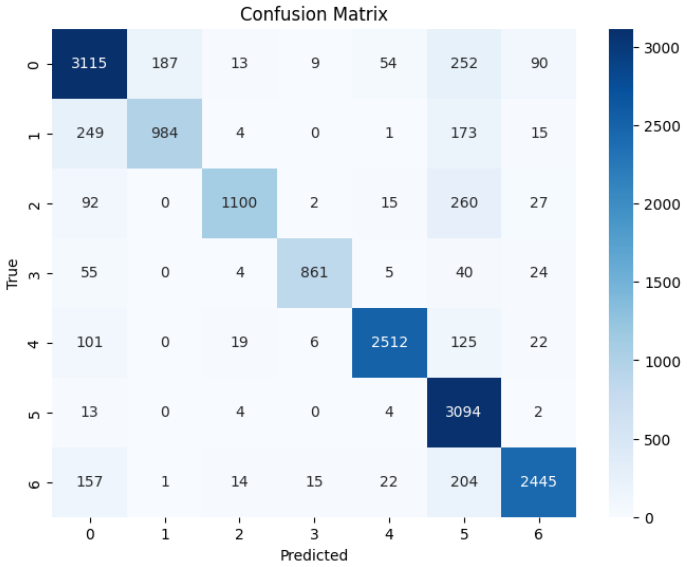
Chương 5 THỬ NGHIỆM VÀ THẢO LUẬN

5.1 Support vector machine

Class	Precision	Recall	F1-score	Support
HYDROLASE	0.82	0.84	0.83	3720
HYDROLASE/HYDROLASE INHIBITOR	0.84	0.69	0.76	1426
IMMUNE SYSTEM	0.95	0.74	0.83	1496
LYASE	0.96	0.87	0.91	989
OXIDOREDUCTASE	0.96	0.90	0.93	2785
RIBOSOME	0.75	0.99	0.85	3117
TRANSFERASE	0.93	0.86	0.89	2858
Accuracy			0.86	16391
Macro avg	0.99	0.84	0.86	16391
Weighted avg	0.87	0.86	0.86	16391

Bảng 5.1: SVM Classification report

Kết quả chạy cho thấy mô hình có độ chính xác đạt khá cao 86%. OXIDOREDUCTASE và LYASE là những lớp có độ chính xác cao nhất. Lớp HYDROLASE/HYDROLASE INHIBITOR và HYDROLASE có độ chính xác thấp hơn các lớp khác nhưng nhìn chung vẫn tương đối ổn. Giải thích cho việc này là do chúng có cấu trúc và chức năng tương tự, đặc biệt Lớp HYDROLASE/HYDROLASE INHIBITOR có thể bao gồm các protein vừa có chức năng của Hydrolase, vừa có vai trò ức chế hoạt động của Hydrolase.Điều này khiến mô hình khó phân biệt rõ ràng giữa chúng. Nhìn chung, mô hình hoạt động khá tốt và không bị overfitting.



Hình 5.1: SVM Confusion matrix

Kết quả của ma trận nhầm lẫn phản ánh đúng với classification report. Đặc biệt là phản ánh rõ sự nhầm lẫn giữa hai lớp HYDROLASE/HYDROLASE INHIBITOR và HYDROLASE. Tỉ lệ nhầm lẫn từ lớp HYDROLASE sang lớp HYDROLASE/HYDROLASE INHIBITOR khoảng 7.4% trong khi tỷ lệ nhầm lẫn từ lớp HYDROLASE/HYDROLASE INHIBITOR sang lớp HYDROLASE khoảng 16.2%, một tỉ lệ khá cao. Điều này cho thấy trong phân loại các protein có cấu tạo, chức năng tương đồng mô hình còn có sự hạn chế nhất định.

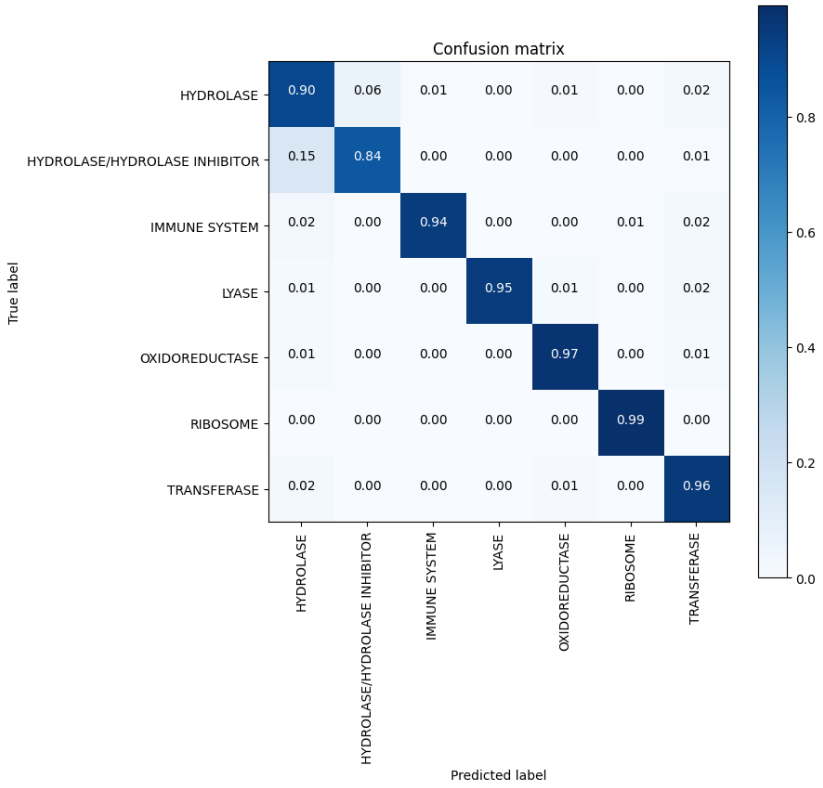
5.2 Convolution neural network

Kịch bản 1: Đánh giá mô hình trên tập dữ liệu kiểm tra

Class	Precision	Recall	F1-Score	Support
HYDROLASE	0.93	0.93	0.93	4592
HYDROLASE/HYDROLASE INHIBITOR	0.84	0.86	0.85	1794
IMMUNE SYSTEM	0.99	0.99	0.99	1964
LYASE	0.98	0.99	0.99	1215
OXIDOREDUCTASE	0.99	0.99	0.99	3395
RIBOSOME	0.99	1.00	1.00	3832
TRANSFERASE	0.99	0.98	0.98	3697
Accuracy			0.96	20489
Macro Avg	0.96	0.96	0.96	20489
Weighted Avg	0.96	0.96	0.96	20489

Bảng 5.2: CNN Classification Report

Kết quả chạy cho thấy mô hình có độ chính xác đạt 94%. Các lớp RIBOSOME, LYASE, IMMUNE SYSTEM có độ chính xác gần như tuyệt đối. 2 lớp HYDROLASE và HYDROLASE/HYDROLASE INHIBITOR có độ chính xác thấp hơn các lớp khác nhưng nhìn chung vẫn tương đối cao. Điều này cho thấy mô hình hoạt động rất tốt. Mô hình không gặp tình trạng overfitting. Có thể mở rộng mô hình để phân loại nhiều lớp protein hơn.

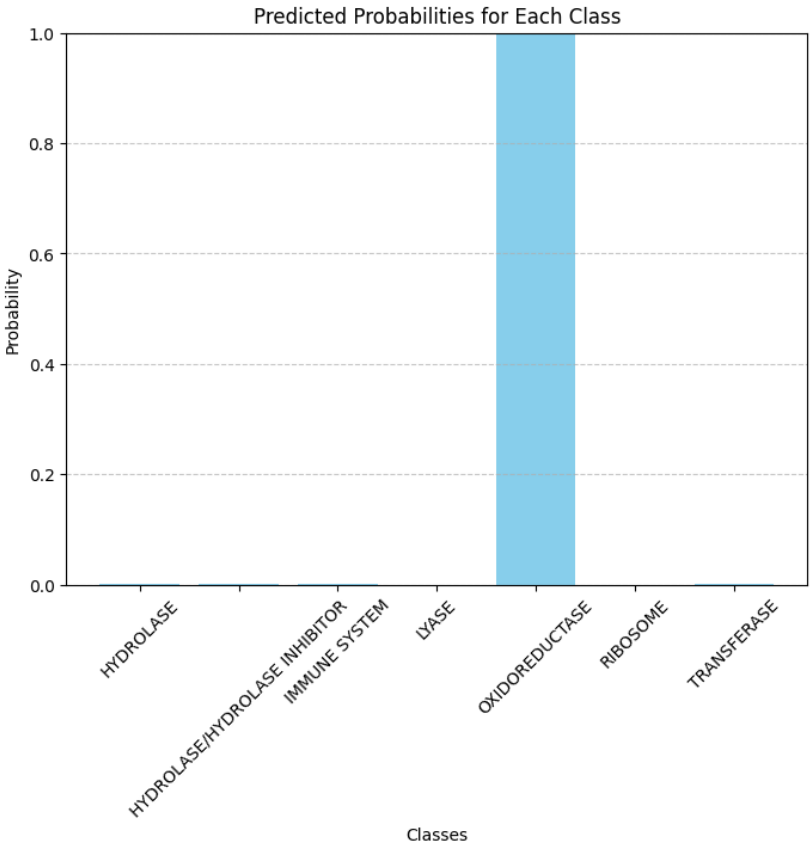


Hình 5.2: CNN Confusion matrix

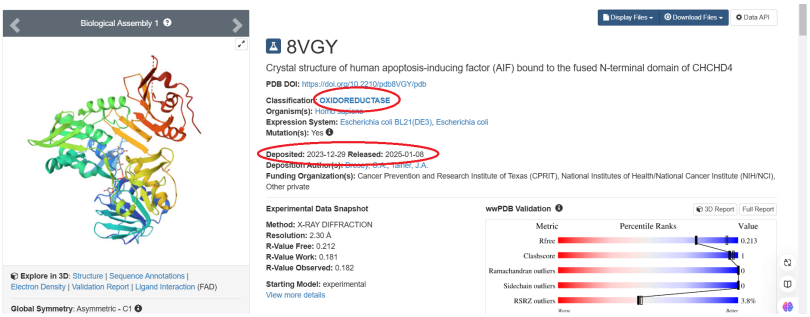
Kết quả của ma trận nhầm lẫn phản ánh đúng với classification report. Bên cạnh đó kết quả này còn chỉ ra hạn chế của mô hình đó là 2 lớp có độ chính xác thấp nhất như đã nói ở trên thường bị nhận diện nhầm lẫn. Trong đó tỷ lệ HYDROLASE/HYDROLASE INHIBITOR bị nhầm lẫn thành HYDROLASE chiếm tới 15% (93.75% trong các trường hợp nhầm lẫn). Điều này cho thấy mô hình vẫn còn hạn chế trong việc phân loại các protein có cấu tạo, chức năng tương tự nhau. Hiện tại mô hình hoạt động tốt với các lớp tách biệt về tính chất, cấu tạo.

Kịch bản 2: Thử nghiệm trên những protein không xuất hiện trong tập kiểm tra

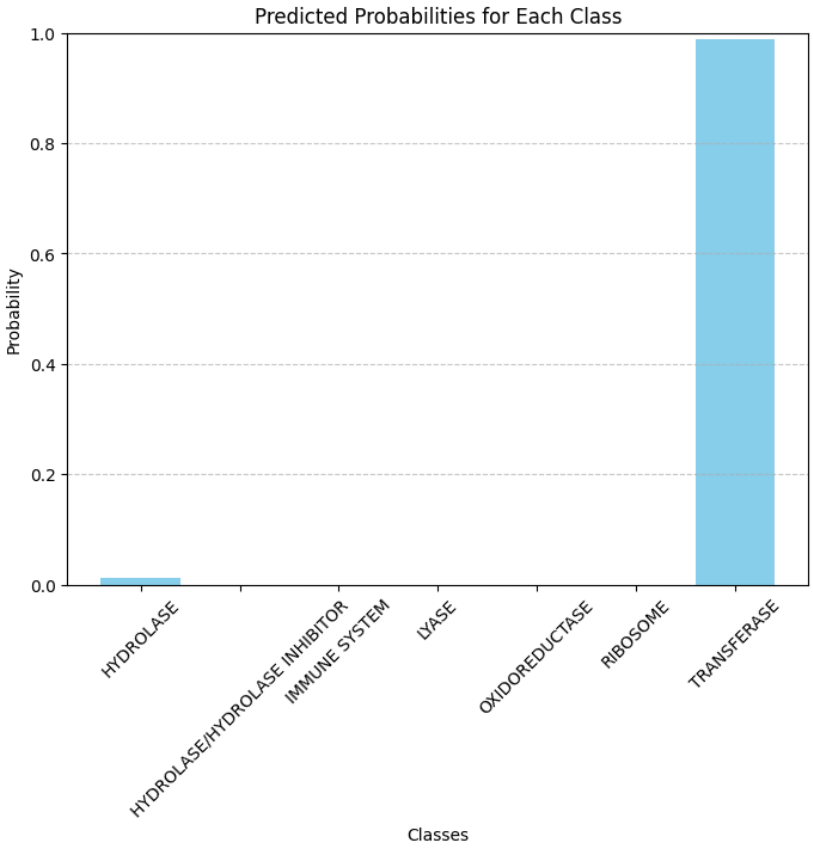
Chuỗi protein được lấy từ trang web https://www.rcsb.org/ được cập nhật thường xuyên.
Chúng ta sẽ thực nghiệm trên protein "8VGY" (nhập giá trị này vào thanh tìm kiếm và lựa chọn mục display files sau đó chọn FASTA sequence, chúng ta sẽ có được chuỗi aa của protein 8VGY). Đây là một protein mới được phát hiện và công bố trong năm 2024 (7 năm sau thời điểm bộ dữ liệu được sử dụng trong bài được thu thập)
Chuỗi aa của protein 8VGY:



Hình 5.3: Kết quả dự đoán phân lớp của protein 8VGY



Hình 5.4: Thông tin về protein 8VGY



Hình 5.5: Kết quả dự đoán phân lớp của protein 1A0F (True label: TRANSFERASE)

Kịch bản 3: Dự đoán những protein chưa được khám phá

Tạo ra chuỗi ngẫu nhiên, có độ dài ngẫu nhiên từ các kí tự đại diện cho các amino acid. Kiểm chứng trong tương lai