TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN

HCMUTE

BÁO CÁO CUỐI KÌ

MÔN: HỌC SÂU

Đề tài:

ỨNG DỤNG KIẾN TRÚC U-NET TRONG PHÂN ĐOẠN VÀ CHẨN ĐOÁN HÌNH ẢNH KHỐI U NỘI SỌ

GV: *TS. Trần Nhật Quang*

SVTH: *Nhóm 4*

Bùi Quốc Khang 21110202

Đặng Kim Thành 21110298

Thành phố Hồ Chí Minh, tháng 5 năm 2024

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN

HCMUTE

BÁO CÁO CUỐI KÌ

MÔN: HỌC SÂU

Đề tài:

ỨNG DỤNG KIẾN TRÚC U-NET TRONG PHÂN ĐOẠN VÀ CHẨN ĐOÁN HÌNH ẢNH KHỐI U NỘI SỌ

GV: *TS. Trần Nhật Quang*

SVTH: *Nhóm 4*

Bùi Quốc Khang 21110202

Đặng Kim Thành 21110298

Thành phố Hồ Chí Minh, tháng 5 năm 2024

DANH SÁCH THÀNH VIÊN THAM GIA ĐỒ ÁN

Đề tài: Ứng Dụng Kiến Trúc U-net trong Phân đoạn và Chẩn đoán Hình ảnh Khối U Não

---

STT	HỌ VÀ TÊN THÀNH VIÊN	MÃ SỐ SINH VIÊN	TỶ LỆ THAM GIA
1	Bùi Quốc Khang	21110202	100%
2	Đặng Kim Thành	21110298	100%

Ghi chú:

Tỷ lệ %: Mức độ phần trăm hoàn thành của từng sinh viên tham gia.

Trưởng nhóm*:* Bùi Quốc Khang

Nhận xét của giảng viên

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh - Tháng 5 năm 2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Trần Nhật Quang, thầy đã tận tình giảng dạy trong suốt quá trình tìm hiểu và học tập môn Học Sâu. Trong từng buổi học, thầy đã nhiệt huyết trong từng lời giải đáp, những kiến thức, kỹ năng, tài liệu quý giá mà thầy đã truyền đạt học không chỉ là nền tảng cho quá trình thực hiện bài tập cuối kì mà còn là kiến thức nền tảng cho nhiều phần sau này.

Rất nhiều công sức và nỗ lực đã được bỏ ra, nhưng do chưa có kinh nghiệm trong việc xây dựng đề tài và những hạn chế về kiến thức, bài báo cáo này không tránh khỏi sai sót. Chúng em luôn sẵn sàng tiếp thu những ý kiến từ thầy để cải thiện bài báo cáo cũng như đề tài lần này.

Một lần nữa, em xin bày tỏ lòng biết ơn đến thầy. Xin kính chúc thầy luôn dồi dào sức khỏe, đạt được nhiều thành công trong công việc.

Đại diện nhóm Bùi Quốc Khang

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG

PHẦN 1: TÌM HIỂU VỀ ĐẠO VĂN

1. Định nghĩa đạo văn

Từ điển Merriam Webster định nghĩa hành vi đạo văn "Plagiarism" là; "Ăn cắp và chuyển giao ý tưởng hoặc lời nói của người khác như là của riêng mình". Nói một cách đơn giản, đạo văn là quá trình lấy lời nói hoặc ý tưởng của người khác và giả vờ rằng chúng là của riêng mình. Theo pháp luật Việt Nam tại điều 2 khoản "Đạo văn là việc sử dụng có hoặc không có chủ ý của tác giả các sản phẩm học thuật về các câu văn, đoạn văn, bài viết, số liệu, hình ảnh, thông tin và ý tưởng của người khác vào các sản phẩm của mình mà không có những chỉ dẫn/thừa nhận tác giả của những nội dung đã sử dụng."

Đạo văn là một lỗi nghiêm trọng về tiêu chuẩn đạo đức khoa học, mà luật pháp (sở hữu trí tuệ) và các cơ sở đào tạo không thể bỏ qua. Đạo văn, tương tự như việc ăn cắp, không chỉ là hành động trái đạo đức mà còn là việc lấy mất sự cá nhân hóa và sáng tạo trong quá trình sáng tác. Khi chúng ta sao chép ý tưởng, công việc, và cấu trúc câu của người khác mà không đưa ra sự đóng góp cá nhân, chúng ta đang mất đi cơ hội để phát triển kỹ năng viết của bản thân. Nếu là sinh viên, trải nghiệm này có thể đánh mất sự hài lòng và tự hào khi đối diện với công trình mà mình đã ăn cắp. Đồng thời, việc đạo văn cũng phản ánh sự mù chữ về thông tin và thiếu tự tin trong khả năng diễn đạt suy nghĩ và ý tưởng cá nhân. Thay vì phát triển khả năng sáng tạo, đạo văn chỉ làm cho người viết trở nên phụ thuộc và thiếu độc lập trong quá trình học tập. Nó không chỉ là sự thừa nhận về khả năng mù chữ trong việc xử lý thông tin, mà còn là việc từ chối cơ hội để tỏa sáng và ghi dấu ấn cá nhân. Mặc dù có thể có những lợi ích ngắn hạn, như giảm áp lực hoặc tiết kiệm thời gian, nhưng hậu quả của đạo văn có thể rất đáng kể. Trong tương lai, khi sinh viên đối diện với thị trường lao động, khả năng sáng tạo và khả năng diễn đạt chính là những yếu tố quyết định sự thành công. Việc áp dụng kiến thức và kỹ năng một cách độc lập là chìa khóa để xây dựng sự nghiệp và đạt được thành công bền vững.

Theo pháp luật Việt Nam người được coi là đạo văn khi có các hành vi sau:

Sử dụng sản phẩm học thuật của người khác mà cam đoan rằng đó là của mình (được người viết thay tên); Sao chép (copy) quá nhiều từ một công trình (mặc dù có chỉ ra nguồn trích) để hình thành một phần lớn công trình của mình; Không dẫn nguồn đã trích khi thay đổi từ ngữ, di chuyển từ hoặc cụm từ, ý tưởng của tác giả khác; Cung cấp không chính xác về tác giả, nguồn của thông tin được trích dẫn, Sử dụng hơn 30% những sản phẩm học thuật của mình đã công bố vào những sản phẩm học thuật mới do mình là tác giả hoặc đồng tác giả mà không ghi rõ nguồn, gọi là tự đạo văn.

2. Những điều cần làm để tránh vấn đề đạo văn.

Hiểu rõ một số kiến thức cơ bản về vấn đề bản quyền.
Tham khảo nguồn tư liệu gốc.
Tìm hiểu kỹ về vấn đề mà bạn đang muốn nói tới.
Phân tích và tổng hợp thông tin.
Trích dẫn đoạn văn và nguồn của bài viết.
Diễn đạt lại nhiều lần bằng các cách khác nhau

Chúng em xin cam đoan dự án này do các thành viên trong nhóm thực hiện. Chúng em không sao chép, sử dụng bất kỳ tài liệu, mã nguồn… của người khác mà không ghi nguồn. Chúng em xin chịu hoàn toàn trách nhiệm nếu vi phạm đạo văn.

Các thành viên trong nhóm:
Bùi Quốc Khang - 21110202
Đặng Kim Thành - 21110298

Phần 2: PHẦN LÝ THUYẾT

1. Tổng quan về Deep Learning

1.1. Deep learning là gì và tại sao nên học nó?

Deep Learning (học sâu) là một nhánh của Machine Learning (học máy) trong lĩnh vực trí tuệ nhân tạo (AI). Deep learning đã được ra đời vào năm 1943 khi Warren McCulloch và Walter Pitts tạo ra một mô hình sử dụng các mạng nơ-ron sâu để mô phỏng hoạt động của bộ não người. Mô hình này sẽ sử dụng kết hợp toán học với các thuật toán và được xây dựng theo nhiều lớp, hay kiến trúc đào "sâu" mà mỗi lớp đều học được các đặc trưng khác nhau từ dữ liệu. Càng sâu vào trong mạng, các đặc trưng càng trở nên phức tạp và trừu tượng hơn. Theo sự tăng trưởng hướng tới tương lai, deep learning càng được ứng dụng rộng rãi trong nhiều lĩnh vực như nhận dạng giọng nói, dịch máy, xe tự lái và phân tích y tế,.. nhằm mục đích phục vụ cho con người.

Học deep learning sẽ đem lại nhiều lợi ích, kiến thức to lớn cho bản thân chúng ta: từ mở ra cơ hội nghề nghiệp với mức lương hấp dẫn trong các ngành công nghệ đến khả năng định hình và dẫn đầu xu hướng công nghệ mới. Kiến thức này không chỉ thúc đẩy sự sáng tạo khi giải quyết các vấn đề xã hội phức tạp qua ứng dụng AI mà còn mang lại kinh nghiệm khi vượt qua, hoàn thành các phạm trù mà con người hiện chưa thể xử lí tốt. Với vai trò là công cụ mạnh mẽ trong việc phân tích dữ liệu và tự động hóa, deep learning sẽ giúp bản thân ta góp phần vào những tiến bộ quan trọng trong nhiều lĩnh vực, từ y tế đến giao thông, làm phong phú thêm khả năng nghề nghiệp và sự nghiệp lâu dài của chính mình.

1.2. Giới thiệu về mạng nơ-ron nhân tạo (Artificial Neural Networks)

Với việc lấy cảm hứng từ cấu trúc và chức năng của mạng nơ-ron sinh học trong não người, mạng nơ-ron nhân tạo (ANNs) được phát triển để sử dụng rộng rãi trong việc mô phỏng các hành vi phức tạp và có khả năng học hỏi từ dữ liệu.

1.3. Sơ lược về nơ-ron sinh học và nơ-ron nhân tạo

❖ **Nơ-ron sinh học gồm:**

Dendrites: Những cấu trúc giống nhánh cây này nhận các tín hiệu điện từ các nơ-ron khác.

Nucleus: Là bộ phận chứa di truyền của tế bào nơ-ron.

Cell Body: Cũng được gọi là soma, là nơi chứa nucleus và các organelle khác, chịu trách nhiệm duy trì hoạt động sống của nơ-ron.

Axon: Một cấu trúc dài giống như sợi cáp, dẫn tín hiệu điện từ cell body tới các nơ-ron khác.

Axon Terminals: Đầu cuối của axon, chuyển tín hiệu điện tới dendrites của nơ-ron khác.

❖ **Nơ-ron nhân tạo gồm:**

Inputs (in_1, in_2, ..., in_n): Các giá trị đầu vào, tương ứng với các tín hiệu được nhận từ các nơ-ron khác hoặc từ dữ liệu ngoài. Mỗi đầu vào thường được nhân với một trọng số.

Processing Unit: Nơi xử lý tín hiệu, tương tự như cell body trong nơ-ron sinh học. Nó tính tổng có trọng số của các đầu vào, áp dụng một hàm kích hoạt, và sau đó phát sinh một đầu ra.

Output (out): Giá trị đầu ra của nơ-ron, được gửi tới nơ-ron tiếp theo trong mạng.

2. Mạng Nơ-ron nhân tạo và các hàm kích hoạt

❖ **Mạng nơ-ron nhân tạo:**

Nhận thấy được rằng việc xây dựng các nơ-ron nhân tạo thành một cấu trúc mạng lưới với các phần tử giống như các nơ-ron sinh học thu nhỏ đã trở thành một bước tiến quan trọng trong lĩnh vực trí tuệ nhân tạo và học máy. Việc xây dựng mạng nơ-ron nhân tạo này không chỉ giải quyết các bài toán thông thường mà còn có khả năng tự học hỏi và thích nghi với dữ liệu mới thông qua quá trình huấn luyện, tạo ra các mô hình có khả năng dự đoán và phân tích vượt trội. Với chi tiết từng lớp như sau:

- Lớp Đầu Vào (Input Layer)
  - Nhiệm vụ: Nhận các đầu vào từ dữ liệu bên ngoài.
  - Chức năng: Chuyển các giá trị đầu vào này tới lớp tiếp theo mà không thay đổi. Mỗi nơ-ron trong lớp đầu vào thường tương ứng với một thuộc tính trong dữ liệu đầu vào.
- Lớp Ẩn (Hidden Layer)
  - Nhiệm vụ: Xử lý thông tin từ lớp đầu vào. Lớp ẩn này có thể học các đặc trưng phức tạp hoặc các mối quan hệ từ dữ liệu.
  - Chức năng: Mỗi nơ-ron trong lớp ẩn nhận một tổng trọng số từ các nơ-ron của lớp trước, áp dụng một hàm kích hoạt, và gửi kết quả tới lớp tiếp theo.

- Lớp Đầu Ra (Output Layer)
  - Nhiệm vụ: Đưa ra dự đoán hoặc kết quả cuối cùng của mạng dựa trên thông tin được xử lý bởi lớp ẩn.
  - Chức năng: Giống như lớp ẩn, nơ-ron ở lớp đầu ra thực hiện tổng trọng số của đầu vào từ lớp ẩn và áp dụng một hàm kích hoạt để sinh ra đầu ra cuối cùng. Thường lớp đầu ra sẽ có bao nhiêu nơ-ron tùy thuộc vào loại bài toán và mục đích của người thiết kế.

Lưu ý: Kết nối giữa các nơ-ron ở đây là kết nối đầy đủ (Fully-connected): Mỗi nơ-ron trong một lớp được kết nối với mọi nơ-ron ở lớp tiếp theo, cho phép mạng có khả năng học được mọi mối quan hệ tiềm ẩn trong dữ liệu.

❖ **Hàm kích hoạt:**

Hàm kích hoạt là một thành phần thiết yếu trong mạng nơ-ron nhân tạo, giúp mô hình có khả năng học được các mối quan hệ phi tuyến từ dữ liệu. Có nhiều loại hàm kích hoạt khác nhau, mỗi loại đều có những đặc điểm và ứng dụng phù hợp:

- ReLU (Rectified Linear Unit): Là hàm kích hoạt phổ biến nhất, chủ yếu được sử dụng cho các lớp ẩn. ReLU có công thức là f(x) = max(0, x), giúp giảm vấn đề biến mất gradient khi huấn luyện mạng nơ-ron sâu.
- Sigmoid: Biến đổi đầu vào thành giá trị trong khoảng từ 0 đến 1, rất hữu ích cho các bài toán phân loại nhị phân tại lớp đầu ra. Tuy nhiên, hàm Sigmoid ít được sử dụng ở các lớp ẩn do vấn đề biến mất gradient.
- Tanh (Hyperbolic Tangent): Giống như hàm Sigmoid nhưng đầu ra của Tanh nằm trong khoảng từ -1 đến 1. Điều này làm cho đầu ra được chuẩn hóa tốt hơn so với Sigmoid, thường được sử dụng trong các mô hình RNN.

2.1. Thế nào là forward propagation và backpropagation

2.1.1. Forward propagation (theo chiều xuôi)

Forward propagation (theo chiều xuôi) là quá trình tính toán đầu ra của mạng nơ-ron dựa trên các đầu vào, trọng số và bias của các nơ-ron. Các dữ liệu được truyền vào từ input layer đến output layer thông qua các hidden layer, ở các hidden layers mỗi nơ-ron sẽ tính tổng trọng số của các đầu vào và bias, rồi áp dụng hàm kích hoạt (activation functions) để tạo ra đầu ra. Đầu ra của một nơ-ron sẽ là đầu vào cho các nơ-ron ở lớp tiếp theo. Khi quá trình này hoàn thành, mạng nơ-ron sẽ có một dự đoán cho đầu vào ban đầu.

2.1.2. Backpropagation (theo chiều ngược)

Backpropagation (theo chiều ngược) là quá trình cập nhật trọng số và bias của mạng nơ-ron để giảm thiểu sai số giữa đầu ra mong muốn và đầu ra thực tế. Backpropagation bao gồm hai bước: lan truyền ngược lỗi và cập nhật tham số. Lan truyền ngược lỗi là quá trình tính toán độ lỗi (loss) qua việc so sánh đầu ra dự đoán với giá trị đầu ra thực tế (như biểu đồ lỗi MSE) rồi sau đó đạo hàm riêng của hàm mất mát (cost function) theo từng trọng số và bias, sử dụng quy tắc dây chuyền (chain rule) để truyền lỗi từ lớp đầu ra ngược về các lớp trước. Các trọng số của mạng nơ-ron được cập nhật bằng cách sử dụng một thuật toán tối ưu hóa như Gradient Descent,

Stochastic Gradient Descent, hay Adam2, sử dụng đạo hàm riêng và giá trị độ lỗi để điều chỉnh trọng số. Quá trình này được lặp lại cho tất cả các mẫu huấn luyện trong tập dữ liệu và cho đến khi mạng nơ-ron hội tụ và đạt được kết quả tối ưu.

2.2. Đi sâu vào mạng nơ-ron nhân tạo (ANNs) và các siêu tham số (hyperparameters)

2.2.1. Tìm hiểu về loss functions (hàm mất mát)

Trong deep learning, hàm mất mát là một thành phần quan trọng của quá trình huấn luyện mạng nơ-ron, giúp đo lường sai số giữa giá trị dự đoán của mạng và giá trị thực tế. Mỗi hàm mất mát phù hợp với một loại bài toán cụ thể và việc lựa chọn đúng hàm mất mát có tác động đáng kể đến hiệu quả huấn luyện và hiệu suất của mô hình. Một vài hàm mất mát phổ biến trong các mô hình phân loại:

❖ **Sparse Categorical Crossentropy:**

Ứng dụng: Thường được sử dụng trong các bài toán phân loại nhiều lớp, nơi nhãn lớp là các số nguyên.

Mô tả: Hàm mất mát này tính toán sự chênh lệch giữa phân phối xác suất của nhãn dự đoán và nhãn thực tế. Nhãn thực tế sẽ được biểu diễn dưới dạng số nguyên. Điều này giúp giảm đáng kể lượng bộ nhớ sử dụng.

Công thức: $\mathbf{L} = -\sum_{\mathbf{i}} \mathbf{y_i} \cdot \mathbf{log}(\hat{\mathbf{y}}_\mathbf{i})$, trong đó y_i là nhãn thực tế dưới dạng số nguyên và $\hat{y}_i$ là xác suất dự đoán của mô hình cho lớp tương ứng.

❖ **Categorical Crossentropy:**

Ứng dụng: Sử dụng trong các bài toán phân loại nhiều lớp có nhãn dạng one-hot.

Mô tả: Giống như sparse_categorical_crossentropy nhưng đòi hỏi nhãn của dữ liệu thực tế phải ở dạng one-hot. Hàm này tính tổng entropy chéo giữa phân phối xác suất dự đoán và phân phối xác suất thực tế, mỗi nhãn được biểu diễn dưới dạng vector xác suất.

Công thức: $\mathbf{L} = -\sum_{\mathbf{i}} \mathbf{y_i} \cdot \mathbf{log}(\hat{\mathbf{y}}_\mathbf{i})$, trong đó y_i là vector one-hot của nhãn thực tế và $\hat{y}_i$ là xác suất dự đoán của mô hình cho từng lớp.

❖ **Binary Crossentropy:**

Ứng dụng: Dành cho các bài toán phân loại nhị phân.

Mô tả: Hàm này đo lường sự khác biệt giữa hai phân phối xác suất cho các trường hợp nhị phân - một cho mỗi lớp trong bài toán phân loại hai lớp.

Công thức: $\mathbf{L} = -\left(\mathbf{y} \cdot \mathbf{log}(\hat{\mathbf{y}}) + (\mathbf{1} - \mathbf{y}) \cdot \mathbf{log}\,(\mathbf{1} - \hat{\mathbf{y}})\right)$, trong đó y là nhãn thực tế (0 hoặc 1) và $\hat{y}$ là xác suất dự đoán của mô hình cho lớp tương ứng.

2.2.2. Giới thiệu và cách tinh chỉnh các hyperparameter (siêu tham số)

Hyperparameter là các thông số được thiết lập trước quá trình huấn luyện và có ảnh hưởng đáng kể đến hiệu suất và hiệu quả của mô hình học máy. Việc tối ưu hóa các hyperparameter này là bước thiết yếu để đạt được hiệu suất tối ưu từ mô hình. Phần này sẽ thảo luận về các chiến lược và phương pháp thường được sử dụng để tinh chỉnh các hyperparameter:

❖ **Tinh chỉnh learning rate (tham số học)**

Mô tả: Learning rate là một trong những hyperparameter quan trọng nhất trong huấn luyện mạng nơ-ron. Nó ảnh hưởng đến tốc độ học của mô hình, tức là mức độ nhanh chóng mà mô hình điều chỉnh các trọng số để giảm thiểu lỗi.

Chiến lược tinh chỉnh: Bắt đầu với một learning rate nhỏ và từ từ tăng lên để tìm ra mức learning rate phù hợp mà không gây ra hiện tượng bùng nổ gradient hoặc hội tụ quá chậm.

Cách áp dụng thực tế: Khi thực hiện huấn luyện mô hình, việc chọn learning rate lớn vào ban đầu sẽ giúp mô hình nhanh chóng đi đến các điểm cực tiểu cục bộ rồi sau đó giảm learning rate sẽ mô hình kết thúc ở các điểm cực tiểu toàn cục trên hàm mất mát.

❖ **Chọn optimizer:**

Mô tả: Optimizer là thuật toán được sử dụng để cập nhật trọng số trong mạng dựa trên dữ liệu và hàm mất mát. Mỗi optimizer có những đặc điểm riêng biệt và phù hợp với các loại bài toán khác nhau.

Chiến lược tinh chỉnh: Thử nghiệm với các loại optimizer khác nhau, như SGD, Adam, hoặc RMSprop, để xem loại nào mang lại hiệu quả tốt nhất cho bài toán cụ thể.

❖ **Điều chỉnh số lượng lớp và nơ-ron mỗi lớp**

Mô tả: Số lượng lớp và số nơ-ron trong mỗi lớp ảnh hưởng trực tiếp đến khả năng học của mạng. Mạng sâu hơn hoặc có nhiều nơ-ron hơn có thể học được các mẫu phức tạp hơn nhưng cũng dễ gặp phải vấn đề quá khớp (overfitting).

Chiến lược tinh chỉnh: Bắt đầu với một mô hình đơn giản và từ từ tăng độ phức tạp của mô hình bằng cách thêm các lớp hoặc nơ-ron và quan sát ảnh hưởng của nó đến hiệu suất mô hình.

❖ **Chọn hàm kích hoạt**

Mô tả: Hàm kích hoạt quyết định một nơ-ron có được kích hoạt hay không dựa trên tổng trọng số đầu vào. Việc lựa chọn hàm kích hoạt phù hợp có thể cải thiện khả năng học và tốc độ hội tụ của mạng.

Chiến lược tinh chỉnh: Thử nghiệm với các hàm kích hoạt khác nhau như ReLU, Leaky ReLU, ELU, và SELU để tìm hàm kích hoạt phù hợp nhất cho từng loại bài toán.

❖ **Kích thước batch (batch size)**

Mô tả: Kích thước batch là số lượng mẫu dữ liệu được xử lý trước khi cập nhật các tham số mô hình, là một hyperparameter quan trọng ảnh hưởng đến quá trình học của mạng nơ-ron. Kính thước batch lớn có thể hỗ trợ đẩy nhanh việc tìm ra tham số gradient đạo hàm giúp mô hình tăng tốc tìm ra điểm cực tiểu toàn cục trên hàm mất mát nhưng lại yêu cầu cực mạnh mẽ về cấu hình máy cũng như khả năng tính toán của CPU (hoặc GPU).

Chiến lược tinh chỉnh: Sử dụng batch nhỏ với phạm vi từ 2 đến 32 sẽ giúp tăng khả năng tổng quát hóa, nhanh chóng phản ứng với dữ liệu mới nhưng lại quá trình huấn luyện lại không ổn định. Còn sử dụng batch lớn sẽ hỗ trợ lớn trong khả năng tối ưu, ổn định quá trình huấn luyện. Vì vậy khi chọn kích thước batch, cần cân nhắc

giữa tốc độ huấn luyện và khả năng tổng quát hóa của mô hình.
Ngoài ra còn có thể sử dụng các công cụ GridSearchCV và RandomizedSearchCV để tự động thử nghiệm và tìm kiếm tập hợp hyperparameter tối ưu. Hoặc Hyperband, Sklearn-Deap, và AutoML, …để tối ưu hóa hyperparameter, giúp quá trình tìm kiếm hiệu quả và nhanh chóng hơn.

2.2.3. Giới thiệu sơ về mạng nơ-ron phi tuần tự (nonsequential neural networks)

Trong khi các mạng nơ-ron truyền thống (sequential) có kiến trúc tuần tự từ đầu vào đến đầu ra, mạng nơ-ron phi tuần tự (nonsequential neural networks) cho phép các kết nối linh hoạt hơn, không tuân theo một trình tự chặt chẽ. Điều này tạo điều kiện cho mô hình có khả năng xử lý các dạng dữ liệu phức tạp và mối quan hệ đa tầng.

Với kiến trúc là sự kết hợp giữa hai kiến trúc mạng: một mạng rộng (wide) để học các quy tắc thô và một mạng sâu (deep) để học các biểu diễn phức tạp. Mô hình này thường được sử dụng rộng rãi trong các hệ thống đề xuất (recommender systems), nơi mà mạng cần học cả từ sự tương tác người dùng lịch sử (deep) và các tín hiệu ngữ nghĩa tức thì (wide). Ngoài ra mô hình này còn các biến thể như mạng "Wide & Deep" có khả năng xử lý nhiều loại đầu vào đồng thời, cho phép mô hình kết hợp thông tin từ nhiều nguồn khác nhau. Hoặc mạng "Wide & Deep" không chỉ xử lý đầu vào đa dạng mà còn cung cấp nhiều đầu ra cùng lúc.

3. Các kiến trúc thường thấy của mạng nơ-ron nhân tạo

3.1.1. Mô hình tĩnh

Mô hình tĩnh trong học sâu là một mạng nơ-ron có cấu trúc được xác định trước và không thay đổi trong suốt quá trình huấn luyện hoặc dự đoán. Các lớp và kết nối giữa chúng được định nghĩa một lần và không thay đổi. Do cấu trúc đã được xác định rõ ràng và không thay đổi, mô hình tĩnh thường dễ triển khai và duy trì hơn các mô hình động. Điều này làm giảm độ phức tạp trong quản lý và tối ưu hóa mô hình. Nhưng mô hình này lại thiếu linh hoạt và khó thích ứng với nhiệm vụ mới.

3.1.2. Mô hình động

Mô hình động trong học sâu không có cấu trúc cố định và cho phép các thay đổi trong kiến trúc mạng trong quá trình huấn luyện. Cấu trúc mạng có thể được điều chỉnh dựa trên dữ liệu đầu vào và tối ưu cho các tác vụ cụ thể. Với tính linh hoạt cao và khả năng thích ứng với nhiệm vụ, mô hình thích ứng tốt với các loại dữ liệu đa dạng và phức tạp như là xử lý ngôn ngữ tự nhiên hoặc xử lý tín hiệu thời gian thực. Mô hình động rất phù hợp với các ứng dụng AI tương tác, như robot học tập qua tương tác với môi trường hoặc phần mềm có khả năng thích ứng với hành vi người dùng.

3.2. Các vấn đề và kĩ thuật xử lí khi huấn luyện mạng nơ-ron nhân tạo

3.2.1. Các kĩ thuật callbacks phổ biến nhất

Trong quá trình huấn luyện mạng nơ-ron nhân tạo, các callbacks là các công cụ hữu ích cho phép can thiệp vào quá trình huấn luyện ở các giai đoạn khác nhau, từ

việc lưu mô hình, điều chỉnh tốc độ học, đến theo dõi và phân tích tiến trình huấn luyện. Các callbacks cung cấp khả năng mở rộng và tùy biến cao, giúp tối ưu hóa quá trình huấn luyện và đạt được kết quả tốt nhất.

❖ **ModelCheckpoint**

Mô tả: ModelCheckpoint là một callback cho phép lưu mô hình tại các thời điểm khác nhau trong quá trình huấn luyện. Điều này rất hữu ích để lưu lại phiên bản tốt nhất của mô hình dựa trên một số tiêu chí nhất định (ví dụ như độ chính xác trên tập kiểm định) giúp tiết kiệm không gian lưu trữ và tập trung vào mô hình tối ưu.

❖ **EarlyStopping**

Mô tả: EarlyStopping giúp ngừng quá trình huấn luyện sớm khi một số tiêu chí không cải thiện nữa (ví dụ như lỗi trên tập kiểm định). Điều này ngăn chặn hiện tượng quá khớp (overfitting) và tiết kiệm thời gian huấn luyện.

❖ **TensorBoard**

Mô tả: TensorBoard là một công cụ mạnh mẽ cho phép theo dõi và phân tích tiến trình huấn luyện bằng cách sử dụng các bảng điều khiển trực quan. Nó hiển thị các biểu đồ về lỗi huấn luyện, độ chính xác, và các thông số mạng khác theo thời gian giúp giám sát hiệu suất mô hình, thực hiện các điều chỉnh kịp thời, và đánh giá ảnh hưởng của các thay đổi đến hiệu suất mô hình.

3.2.2. Vấn đề vanishing và exploding gradients và các cách xử lí

3.2.2.1. Vấn đề vanishing và exploding gradients

Trong quá trình huấn luyện các mạng nơ-ron sâu, hai vấn đề phổ biến thường gặp là vanishing gradients (gradient biến mất) và exploding gradients (gradient bùng nổ). Những vấn đề này không chỉ ảnh hưởng đến tốc độ học tập của mạng mà còn có thể khiến quá trình huấn luyện trở nên không hiệu quả hoặc thậm chí là không thể hội tụ.

❖ **Vanishing Gradients**

- Mô tả: Vấn đề này xảy ra khi các gradients (đạo hàm của hàm mất mát theo các trọng số mạng) trở nên rất nhỏ trong quá trình lan truyền ngược, khiến cho các trọng số không được cập nhật đáng kể. Điều này thường xảy ra ở các lớp sâu trong mạng, đặc biệt khi sử dụng các hàm kích hoạt như sigmoid hoặc tanh, làm cho quá trình huấn luyện mất nhiều thời gian hoặc thậm chí dừng lại hoàn toàn.
- Giải pháp: Sử dụng các hàm kích hoạt như ReLU và các biến thể của nó (Leaky ReLU, ELU) có thể giảm thiểu vấn đề này vì chúng không bão hòa ở phạm vi giá trị rộng như tanh hoặc sigmoid.

❖ **Exploding Gradients**

- Mô tả: Ngược lại với vanishing gradients, exploding gradients là hiện tượng các gradients trở nên quá lớn trong quá trình huấn luyện. Điều này có thể dẫn đến việc cập nhật trọng số quá lớn, khiến mô hình trở nên không ổn định và dễ dàng diverge.

- Giải pháp: Áp dụng các kỹ thuật như gradient clipping (giới hạn giá trị gradient) trong quá trình huấn luyện có thể giúp kiểm soát và ngăn chặn vấn đề này. Gradient clipping thực hiện bằng cách giới hạn giá trị của gradients tới một ngưỡng nhất định, ngăn không cho chúng vượt quá mức cho phép.

3.2.2.2. Các cách xử lí

❖ **Kỹ thuật khởi tạo trọng số:**

- Mục đích: Một trong những nguyên nhân chính của vanishing và exploding gradients là cách thức khởi tạo trọng số của mạng.
- Phương pháp hiệu quả:
  - Khởi tạo Glorot/Xavier: Được sử dụng cho hàm kích hoạt tanh và sigmoid, phương pháp này dựa trên phân phối chuẩn khởi tạo trọng số sao cho output và gradients có phương sai giữ được cân bằng qua mọi lớp.
  - Khởi tạo He: Được thiết kế cho các hàm kích hoạt dựa trên ReLU và các biến thể của hàm như Leaky ReLU, Parametric ReLU (PReLU), hoặc Exponential Linear Unit (ELU), việc khởi tạo He đặt trọng số sao cho các nơ-ron không rơi vào trạng thái bão hòa quá sớm trong quá trình huấn luyện.

❖ **Áp dụng các hàm kích hoạt khác nhau (activation functions)**

ReLU là hàm kích hoạt phổ biến nhất trong các mạng nơ-ron hiện đại do tính đơn giản và hiệu quả trong việc giảm thiểu vấn đề vanishing gradient khi so sánh với các hàm sigmoid hoặc tanh. Tuy nhiên, nó có thể dẫn đến vấn đề "dying ReLU", nơi một số nơ-ron chỉ xuất ra giá trị zero và ngừng học hoàn toàn.

Leaky ReLU là một biến thể của ReLU nhằm giải quyết vấn đề "dying ReLU". Bằng cách cho phép một gradient nhỏ khi $x < 0$, Leaky ReLU giúp duy trì luồng gradient qua mạng, làm tăng khả năng học tập.

ELU cải thiện trên Leaky ReLU bằng cách sử dụng một hàm mũ cho các giá trị đầu vào âm. Điều này không chỉ giúp giảm vấn đề dying units mà còn giúp giảm thiểu noise trong tín hiệu đầu ra khi x nhỏ.

SELU là một biến thể của ELU có khả năng tự chuẩn hóa các đầu ra của các lớp. Khi được sử dụng với khởi tạo trọng số đặc biệt (LeCun normal initialization), SELU giúp các lớp trong mạng nơ-ron duy trì phương sai đầu vào qua mỗi lớp, làm giảm nhu cầu sử dụng chuẩn hóa batch và cải thiện hiệu quả huấn luyện.
Tùy thuộc vào vấn đề cụ thể, SELU > ELU > Leaky ReLU > ReLU có thể được coi là thứ tự ưu tiên.

❖ **Áp dụng kỹ thuật batch normalization**

Batch Normalization (BN) là một kỹ thuật rất quan trọng trong huấn luyện mạng nơ-ron sâu, giúp giảm thiểu vấn đề vanishing và exploding gradients bằng cách chuẩn hóa các đầu vào của mỗi lớp. Điều này không chỉ cải thiện tốc độ huấn luyện, mà còn cải thiện đáng kể khả năng tổng quát hóa của mô hình, giúp giảm bớt sự phụ thuộc của gradients vào tỷ lệ của các tham số trước đó, từ đó giảm thiểu hiện tượng unstable gradients và overfitting. Ngoài ra việc áp dụng batch normalization còn cho phép sử dụng tốc độ học lớn hơn mà không lo về việc mô hình sẽ diverge.

Với cơ chế chuẩn hóa đầu ra của một lớp bằng cách điều chỉnh theo tỷ lệ trung bình bằng 0 và độ lệch chuẩn bằng 1. Quá trình này được thực hiện độc lập cho mỗi batch dữ liệu, vì thế được gọi là "batch" normalization.

❖ **Kỹ thuật gradient clipping**

Gradient clipping là một kỹ thuật đơn giản được sử dụng để kiểm soát độ lớn của gradient trong quá trình huấn luyện một mạng nơ-ron áp dụng cho các mô hình học máy hoặc học sâu. Với mục đích để giảm thiểu vấn đề "exploding gradients" (gradient bùng nổ) và "vanishing gradients" (gradient biến mất). Với ý tưởng là sẽ đặt ra một ngưỡng giá trị của gradients và khi xảy ra sự bùng nổ các giá trị này sẽ luôn được giới hạn lại trong ngưỡng để vẫn đảm bảo tính chính xác và đáng tin cậy của mô hình. Giúp cho mô hình không bị vướng vào các trường hợp hội tụ chậm, sai lệch nhiều, các gradient trở nên quá lớn gây ra sự không ổn định.

Ngoài ra còn có adaptive gradient clipping là một biến thể của gradient clipping mà ngưỡng cắt được điều chỉnh tự động, thay đổi dựa trên lịch sử tính toán của gradient, thông tin mà mô hình thu thập được trong quá trình huấn luyện. Phương pháp này có thể được thay thế cho batch normalization trong việc xử lí gradient của mô hình. Phương pháp này sẽ giúp tăng tính ổn định và hiệu suất của mô hình, giúp tối ưu hóa mạng trên các phần của không gian hàm mất mát, cũng như làm cho mô hình trở nên linh hoạt hơn ở đa dạng bài toán.

3.2.3. Kỹ thuật transfer learning

Transfer learning là một kỹ thuật mạnh mẽ trong học sâu, cho phép tái sử dụng kiến thức (thường là các lớp đã được huấn luyện trước) từ một mô hình đã được huấn luyện trên một nhiệm vụ liên quan để cải thiện hiệu quả và hiệu suất huấn luyện trên một nhiệm vụ mới. Phương pháp này đặc biệt hữu ích khi tài nguyên dữ liệu hoặc tính toán bị hạn chế. Vậy nên khi áp dụng transfer learning cần phải để ý các vấn đề sau:

❖ **Tìm kiếm và chọn mô hình phù hợp**

Đánh giá các mô hình hiện có: Tìm kiếm một mô hình nơ-ron đã được huấn luyện trên một bộ dữ liệu lớn và liên quan đến nhiệm vụ cần giải quyết.
Lựa chọn lớp để tái sử dụng: Các lớp đầu tiên của một mạng nơ-ron được huấn luyện để nhận diện các đặc trưng chung như cạnh và màu sắc, có thể được tái sử dụng cho nhiều nhiệm vụ khác nhau.

❖ **Điều chỉnh mô hình cho nhiệm vụ mới**

Reuse the lower layers: Sử dụng lại các lớp thấp hơn của mô hình đã huấn luyện trước vì chúng đã học cách nhận dạng các đặc trưng chung, và thêm vào đó các lớp mới phù hợp với nhiệm vụ cụ thể.

Fine-tuning: Sau khi đã tái sử dụng các lớp, nên tiếp tục điều chỉnh một số lớp trên cùng của mô hình đã huấn luyện trước để làm cho chúng phù hợp hơn với nhiệm vụ mới. Thường thì việc này bao gồm việc unfreeze một số hoặc toàn bộ các lớp đã tái sử dụng và tiếp tục quá trình huấn luyện.

3.2.4. Áp dụng các thuật toán faster optimizers

Thuật toán	Tính chất	Ưu điểm	Nhược điểm	Tốc độ hội	Chất lượng

				tụ	hội tụ
SGD (Stochastic Gradient Descent)	Cập nhật trọng số dựa trên gradient của một mẫu ngẫu nhiên	Đơn giản, dễ triển khai, điều chỉnh learning rate	Dễ bị mắc kẹt ở các điểm cực tiểu cục bộ, learning curve không mượt ở các hàm không đối xứng	Nhanh ở giai đoạn đầu, sau đó sẽ chậm	Cao
Adam (Adaptive Moment Estimation)	Kết hợp các ưu điểm của Momentum Optimization và RMSProp để tính toán, kiểm soát tốc độ học tập cũng như hội tụ.	Tự điều chỉnh tốc độ học tập, hiệu suất cao trên đa dạng các bài toán cũng như tránh bị kẹt ở các điểm cực tiểu cục bộ.	Phức tạp hơn các optimizer khác, yêu cầu 2 hyperparameter và điều chỉnh tham số	Nhanh	Khá hoặc cao
RMSprop (Root Mean Square Propagation)	Cập nhật tốc độ học tập riêng cho từng tham số dựa trên lịch sử gradient.	Hiệu quả với các hàm mất mát không đối xứng. Giảm tốc độ học tập cho các tham số không quan trọng. Learning rate ổn định.	Phức tạp hơn, cần xử lí việc điều chỉnh hyperparameter và tốc độ học	Nhanh	Khá hoặc cao
AdaGrad (Adaptive Gradient Algorithm)	Tự động điều chỉnh, tính toán, tối ưu hóa learning rate cho từng tham số dựa trên lịch sử các gradient.	Phù hợp cho các bài toán với dữ liệu thưa, các hàm mất mát không đối xứng	Learning rate giảm quá nhanh dẫn tới dễ mắc kẹt ở các điểm cục bộ (vanishing gradient). Không phù hợp gradient chậm.	Nhanh	Thấp (do dễ bị dừng sớm)
Nadam (Nesterov-accelerated Adaptive Moment	Là phiên bản nâng cấp của thuật toán Nesterov momentum	Có khả năng xử lí nhanh hơn trong việc tính toán, điều chỉnh tốc độ	Có độ phức tạp cao và có yêu cầu trong việc xử lí tham số	Nhanh	Khá hoặc cao

Estimation)	và Adam.	hội tụ, phù hợp với tập dữ liệu lớn			
Adam và Nadam Optimization	Là kết hợp các lợi ích của RMSprop và Momentum	Giúp ích lớn trong việc huấn luyện các tập dữ liệu lớn và hiệu quả ở hàm mất mát không đối xứng	Có độ phức tạp và có thể gặp phải các vấn đề liên quan đến việc chọn siêu tham số.	Nhanh	Cao
Momentum Optimization	Sử dụng đạo hàm trước đó để tính toán hướng và tốc độ cập nhật cho các tham số, gradient cần thiết và việc huấn luyện	Giúp giảm hiện tượng dao động (oscillation), thoát khỏi điểm cục bộ và tăng hiệu quả học tập	Phải lưu ý về tham số moment vì có thể gây ra sự mất mát	Khá	Cao

Bảng 1. So sánh các thuật toán optimizer

3.2.5. Kỹ thuật learning rate scheduling

Kỹ thuật	Đặc điểm	Ưu điểm	Nhược điểm
Time-based decay scheduling	- Giảm tốc độ học theo một tỉ lệ xác định sau mỗi epoch hoặc sau một số bước huấn luyện dựa trên hàm tuyến tính lr = lr_0 / (1 + decay * epoch). - Thích hợp các bài toán đơn giản, biết tỷ lệ giảm phù hợp	- Dễ dàng hiểu và triển khai, cung cấp một cách mạch lạc để giảm learning rate theo thời gian.	- Cần phải chọn hệ số giảm một cách cẩn thận; không cung cấp tính linh hoạt cao, không thích hợp cho mọi loại dữ liệu hoặc mô hình.
Exponential Scheduling	- Learning rate giảm dần theo một hàm mũ, thường là lr = lr0 * decay_rate ^ (epoch / decay_steps), với decay_rate là tỷ lệ giảm mà learning rate sẽ được áp dụng mỗi decay_steps epochs.	- Tăng tính linh hoạt so với time-based decay, giúp tối ưu hóa tốc độ học theo cách không đồng đều. - Giảm learning rate nhanh chóng ở giai đoạn đầu, giữ	- Hàm số mũ cần được điều chỉnh một cách thích hợp để đảm bảo tỉ lệ giảm learning rate phù hợp với tốc độ huấn luyện. - Cần chọn các

	- Thích hợp với các bài toán cần biết được tỉ lệ.	learning rate ở mức thấp ở giai đoạn sau. - Đơn giản và dễ triển khai.	siêu tham số cẩn thận, quá trình giảm có thể quá nhanh, khiến mô hình không đủ thời gian để tối ưu hóa.
Piecewise Constant Scheduling	- Learning rate được giữ ổn định tại các giá trị cố định trong các khoảng thời gian nhất định, sau đó giảm xuống một giá trị mới và ổn định tại đó trong một khoảng thời gian khác, và cứ tiếp tục như vậy. - Thích hợp với bài toán cần kiểm soát chi tiết tốc độ học.	- Cung cấp khả năng tùy chỉnh tốt, linh hoạt hơn cho từng giai đoạn cụ thể trong quá trình huấn luyện, có thể phù hợp với nhiều loại dữ liệu và mô hình.	- Yêu cầu xác định, đánh giá cẩn thận các điểm giảm learning rate và giá trị learning rate phù hợp cho từng giai đoạn thời gian hoặc số lượng epoch - Đòi hỏi sự can thiệp thủ công và điều chỉnh dựa trên kinh nghiệm hoặc thử nghiệm để xác định các tham số.
Performanc e scheduling	- Dựa trên hiệu suất của mô hình để điều chỉnh tốc độ học, áp dụng cụ thể trong trường hợp hiệu suất không cải thiện trên tập validation sau một số lượng epochs nhất định. - Thích hợp với bài toán phức tạp, dễ quá khớp	- Rất linh hoạt và phản ánh trực tiếp quá trình học của mô hình, giúp tối ưu hóa hiệu suất mô hình mà không cần giảm quá nhanh hoặc quá chậm. - Ngăn ngừa quá khớp (overfitting) và tránh bị kẹt ở điểm cực tiểu cục bộ.	- Do có thể phản ứng với sự biến động nhỏ nên yêu cầu theo dõi chi tiết và đánh giá cẩn thận hiệu suất của mô hình trong quá trình huấn luyện. - Cần xác định, tính toán đo lường thường xuyên ngưỡng hiệu suất và cách giảm learning rate dựa trên hiệu suất.
1cycle scheduling	- Là một phương pháp nâng cao mà trong đó learning rate trước tiên	- Đơn giản thiết lập hơn các phương pháp khác cùng với	- Cần chọn cẩn thận các giá trị learning rate ban

	tăng lên một cách nhanh chóng từ một giá trị thấp đến một giá trị cao, sau đó giảm dần về một giá trị thấp hơn trong suốt quá trình huấn luyện. - Thích hợp với bài toán cần cân bằng khám phá và khai thác dữ liệu áp dụng cho tốc độ học biến động (learning rate)	khả năng học nhanh và tinh chỉnh kỹ lưỡng. - Có khả năng tối ưu hóa tốt, hội tụ nhanh, giảm thiểu thời gian huấn luyện, giảm khả năng overfitting và cải thiện hiệu suất.	đầu và cuối cùng phù hợp. - Yêu cầu theo dõi và điều chỉnh learning rate theo lịch trình thời gian thay đổi 1cycle. - Không phù hợp cho mọi bài toán và tập dữ liệu.

Bảng 2. So sánh các kĩ thuật learning rate scheduling

3.2.6. Kỹ thuật regularization

3.2.6.1. Kỹ thuật l1 và l2 regularization

Đây là hai kỹ thuật phổ biến được sử dụng để giảm thiểu hiện tượng overfitting trong các mô hình học sâu. Cả hai phương pháp này thêm một thành phần phạt (penalty) vào hàm mất mát của mạng, giúp hạn chế độ lớn của các trọng số trong mô hình.

❖ **l1 Regularization (Lasso):**

Mô tả: l1 regularization thêm penalty tương ứng với tổng giá trị tuyệt đối của các trọng số vào hàm mất mát. Điều này thúc đẩy mô hình phát triển một giải pháp thưa (sparse solution), nơi phần lớn các trọng số sẽ bằng 0, làm giảm độ phức tạp của mô hình.

Lợi ích: Giúp giảm độ phức tạp mô hình và cải thiện khả năng tổng quát hóa bằng cách loại bỏ các trọng số của những đặc trưng không quan trọng.

❖ **l2 Regularization (Ridge):**

Mô tả: l2 regularization thêm penalty tương ứng với tổng bình phương của các trọng số vào hàm mất mát. Điều này không thúc đẩy một giải pháp thưa như l1 nhưng giúp kiểm soát độ lớn của các trọng số, từ đó giảm thiểu overfitting. Có nghĩa là tất cả các đặc trưng được giữ lại nhưng ảnh hưởng của chúng đến mô hình được giảm nhẹ.

Lợi ích: Đặc biệt hiệu quả trong việc giảm thiểu overfitting khi huấn luyện các mô hình lớn trên tập dữ liệu hạn chế.

❖ **l1_l2 Regularization (Elastic Net)**

Mô tả: Kết hợp cả l1 và l2 regularization. Mô hình này vừa thúc đẩy giải pháp thưa vừa kiểm soát độ lớn của các trọng số, mang lại lợi ích của cả hai phương pháp.

Lợi ích: Đặc biệt hiệu quả khi mô hình cần tính chất chọn lọc đặc trưng và một số biến quan trọng trong số chúng. Giúp tránh biến động không định kỳ khi số lượng

đặc trưng lớn hơn số lượng mẫu huấn luyện, hoặc khi một số đặc trưng có sự tương quan mạnh mẽ với nhau như trong Lasso. Điều này sẽ mang lại sự cân bằng giữa việc giảm độ phức tạp của mô hình và giữ được tính ổn định.

3.2.6.2. Kỹ thuật max-norm regularization

Max-norm regularization là một kỹ thuật hữu ích khác để giảm thiểu overfitting, đặc biệt là trong các mạng nơ-ron sâu. Phương pháp này đặt ra một giới hạn trên cho độ lớn của vector trọng số cho mỗi nơ-ron trong mạng. Điều này không chỉ giúp kiểm soát overfitting mà còn có thể cải thiện tính ổn định của mạng nơ-ron khi sử dụng các hàm kích hoạt như ReLU …

- Mô tả: Max-norm regularization sẽ ràng buộc giá trị tối đa của chuẩn (norm) của vector trọng số cho mỗi nơ-ron. Thường được xác định bởi một tham số r, là giá trị tối đa của chuẩn 2 của vector trọng số.
- Lợi ích: Ngăn chặn các trọng số trong mạng trở nên quá lớn, điều này có thể dẫn đến vấn đề overfitting và giúp giảm thiểu hiện tượng "exploding gradients."

3.2.6.3. Kỹ thuật dropout

Dropout là một kỹ thuật regularization mạnh mẽ, được sử dụng rộng rãi để ngăn chặn hiện tượng overfitting trong các mạng nơ-ron sâu. Kỹ thuật này hoạt động bằng cách tạm thời "loại bỏ" ngẫu nhiên một số nơ-ron trong quá trình huấn luyện, làm cho mạng không thể phụ thuộc quá nhiều vào bất kỳ nơ-ron cụ thể nào, từ đó tăng cường khả năng tổng quát hóa của mô hình.

- Mô tả: Trong mỗi bước huấn luyện, mỗi nơ-ron có một xác suất p được gọi là "dropout rate" để bị loại bỏ, tức là đầu ra của nơ-ron đó sẽ được thiết lập là 0 trong suốt quá trình lan truyền tiến đó.
- Lợi ích: Điều này tạo ra một mạng "thưa" trong mỗi lần lặp, với một cấu trúc ngẫu nhiên khác nhau, giúp mô hình ngăn chặn sự phụ thuộc không quá khớp với dữ liệu huấn luyện, buộc phải học cách phân phối trọng số hiệu quả hơn.

3.2.6.4. Các kỹ thuật khác

Ngoài ra, còn có thể sử dụng các kĩ thuật như batch normalization và early stopping đã để cập ở trên để hiệu chỉnh giúp cho mô hình được tối ưu và cho ra được các kết quả tốt nhất.

4. Các kĩ thuật tiền xử lí dữ liệu

4.1. Pipeline xử lý dữ liệu

Loading data và convert to tensorflow dataset: Chuyển đổi dữ liệu thành định dạng tensorflow dataset, cho phép tận dụng các chức năng xử lý dữ liệu tiên tiến của tensorflow. Điều này giúp tối ưu hóa việc nạp và xử lý dữ liệu trong quá trình huấn luyện.

Filtering data: Phương thức này cho phép lọc dữ liệu dựa trên một điều kiện cụ thể, chẳng hạn như giá trị của các mẫu dữ liệu.

Data transformations: Với phương thức “repeat()” được sử dụng để lặp lại dataset một số lần nhất định, giúp mô hình học tốt hơn bằng cách xem xét dữ liệu

nhiều lần và "batch()" được sử dụng để nhóm dữ liệu thành các batch, giúp tối ưu hóa việc huấn luyện mạng nơ-ron.

Random Shuffling: Với phương thức "shuffle()" sẽ giúp xáo trộn dữ liệu trong dataset để đảm bảo rằng mô hình không học theo thứ tự nào của dữ liệu, giúp mô hình học được bền vững và không phụ thuộc vào thứ tự của dữ liệu.

Custom transformations: Áp dụng một hàm đã định nghĩa trước lên từng phần tử của dataset. Điều này cho phép thực hiện các biến đổi tùy chỉnh trên dữ liệu, chẳng hạn như thay đổi quy mô của các giá trị hoặc áp dụng các biến đổi toán học.

4.2. Prefetching và Multithreading

Kỹ thuật "prefetch()" cho phép tối ưu hóa thời gian huấn luyện bằng cách chuẩn bị dữ liệu cho batch tiếp theo trong khi GPU đang xử lý batch hiện tại.

Kỹ thuật "multithreading" trong "interleave()" sẽ hỗ trợ khả năng đọc và xử lý dữ liệu từ nhiều file một cách song song, giảm thiểu thời gian chờ và tăng hiệu quả xử lý.

4.2.1. Kỹ thuật mã hóa dữ liệu Embeddings

Embedding là một kỹ thuật rất quan trọng trong học sâu, đặc biệt là khi làm việc với dữ liệu phân loại hoặc dữ liệu dạng văn bản. Mục đích chính của embedding là chuyển đổi từ đặc trưng phân loại (categorical) hoặc từ ngữ thành các vector số có thể xử lý được trong các mô hình học sâu.

- Định nghĩa: Embedding là việc học một ánh xạ từ các đối tượng rời rạc, chẳng hạn như từ vựng trong ngôn ngữ tự nhiên hoặc danh mục trong dữ liệu phân loại, sang các vector số thực.
- Mục đích: Mục đích của embedding là để biểu diễn dữ liệu đầu vào ở dạng mà mạng nơ-ron có thể dễ dàng xử lý, đồng thời giữ được mối quan hệ ngữ nghĩa hoặc logic giữa các đối tượng.

Ngoài ra kỹ thuật one-hot encoding cũng là một kỹ thuật cơ bản nhưng cực kỳ quan trọng trong việc chuẩn bị dữ liệu cho các mô hình học máy và học sâu, đặc biệt là khi làm việc với dữ liệu phân loại.

4.3. Khái niệm và kiến trúc mạng nơ-ron tích chập (convolutional neural networks)

4.3.1. Giới thiệu CNN (convolutional neural networks)

4.3.1.1. Lịch sử và phát triển của CNN

CNN là một thành tựu quan trọng trong lĩnh vực học sâu, với ứng dụng rộng rãi trong thị giác máy tính, nhận dạng hình ảnh và xử lý ngôn ngữ tự nhiên.

Khái niệm về CNN được phát triển từ nghiên cứu về mạng nơ-ron thị giác của não người, đặc biệt là từ những công trình nghiên cứu của David H. Hubel và Torsten Wiesel về tế bào thị giác

4.3.1.2. Cơ bản về kiến trúc CNN

CNN bao gồm các tầng tích chập (convolutional layer), tầng gộp (pooling layer), và các tầng kết nối đầy đủ cuối cùng (fully connected layers). Mỗi tầng tích

chập ở đây sẽ sử dụng một bộ lọc (filter) để quét qua ảnh đầu vào và tạo ra các bản đồ đặc trưng (feature map). Tầng gộp giúp giảm kích thước của các bản đồ đặc trưng, làm nổi bật các đặc điểm quan trọng và giảm thiểu số lượng tham số. Tầng kết nối đầy đủ cuối cùng sẽ giúp mô hình mạng trích xuất ra được output cần thiết cho vấn đề bài toán.

4.3.1.3. Điểm nổi bật của CNN

Khả năng tự học các bộ lọc: Trong quá trình huấn luyện, CNN có khả năng tự điều chỉnh các hệ số trong bộ lọc để tối ưu hóa hiệu suất cho từng nhiệm vụ cụ thể (tùy vào số lượng filter mà chúng ta định nghĩa sẽ có thể giúp CNN lấy ra bấy nhiêu đặc trưng cụ thể của đặc trưng đầu vào).

Hiệu quả trong việc trích xuất đặc trưng từ dữ liệu không gian cao như hình ảnh, nhờ vào việc giảm dần kích thước không gian mà vẫn giữ được thông tin quan trọng.

4.3.1.4. Ứng dụng của CNN

CNN hiện nay thường được sử dụng rộng rãi trong các ứng dụng nhận dạng hình ảnh, từ phân loại đối tượng, phát hiện vật thể cho đến phân tích video. Ngoài ra, CNN cũng được áp dụng trong các bài toán xử lý ngôn ngữ tự nhiên nhờ khả năng trích xuất đặc trưng từ dữ liệu tuần tự.

4.3.2. Cấu trúc của lớp tích chập (convolutional layer)

4.3.2.1. Trường tiếp nhận (Receptive Fields)

Receptive Rectangles (Receptive Fields): là một vùng không gian được tách ra từ không gian đầu vào trước đó, mà mỗi một nơ-ron trong convolutional layer xem xét để tính toán đầu ra của mình. Trong một convolutional layer, mỗi nơ-ron chỉ kết nối với một phần của không gian đầu vào. Receptive rectangles có kích thước cố định và di chuyển qua toàn bộ không gian đầu vào để thực hiện tính toán.

4.3.2.2. Padding (lớp đệm)

Padding (đệm): là quá trình thêm các giá trị đệm (các pixel giả) vào viền xung quanh của dữ liệu đầu vào trước khi thực hiện phép tích chập (thường là các giá trị 0). Mục đích của padding là để tăng kích thước hiệu quả của hình ảnh, cho phép nơ-ron ở biên cũng có đủ số lượng pixel để xem xét tính toán, áp dụng phép tích chập và để giúp giữ cho thông tin biên của hình ảnh không bị mất mát.

4.3.2.3. Strides (bước nhảy)

Strides (bước nhảy): Stride là bước di chuyển của receptive rectangles trên không gian đầu vào khi thực hiện phép tích chập. Stride xác định khoảng cách giữa hai vị trí liên tiếp của receptive rectangles. Khi stride có giá trị lớn hơn 1, receptive rectangles sẽ di chuyển nhanh hơn trên không gian đầu vào, làm giảm kích thước đầu ra của convolutional layer và tăng tốc độ tính toán, tuy nhiên có thể bỏ qua một số thông tin quan trọng.

4.3.2.4. Filters (bộ lọc)

Filters (bộ lọc) trong convolutional layer là các ma trận có kích thước nhỏ được áp dụng để tích chập với vùng không gian được trích xuất ra từ dữ liệu đầu vào trước đó. Mỗi filter thường sẽ chứa các trọng số có thể nhận diện một đặc trưng cụ thể trong dữ liệu đầu

vào. Các Filter sẽ là cố định trong một lớp convolutional layer và được tích chập lần lượt với các receptive field trượt qua toàn bộ không gian đầu vào.

4.3.2.5. Feature map (bản đồ đặc trưng)

Feature map là kết quả của phép tích chập giữa filter và các receptive field trượt qua toàn bộ không gian đầu vào. Nó là một mảng 2D của các giá trị, trong đó mỗi giá trị tương ứng với sự tính toán một nơ-ron trong convolutional layer. Feature map bao gồm những thông tin về sự xuất hiện và vị trí của các đặc trưng đã được filter nhận diện trong đầu vào.

Lưu ý: Nếu trong quá trình huấn luyện mạng CNN bị lỗi tràn bộ nhớ thì có thể xem xét các bước như giảm kích thước batch-size (giúp giảm độ lớn của tính toán), tăng stride (giúp cho ra bản đồ đặc trưng được cô đọng hơn, tốn ít bộ nhớ), tăng kernel_size (giúp giảm số lần receptive field phải tích chập với filter và lấy ra được các đặc trưng lớn hơn từ input).

4.3.3. Lý thuyết và chức năng của lớp gộp (pooling layer)

4.3.3.1. Giới thiệu về lớp gộp (pooling layer)

Pooling layers là một phần quan trọng trong kiến trúc của mạng nơ-ron convolutional (CNN). Chức năng chính của pooling layers là giảm kích thước của feature maps bằng cách lấy mẫu các giá trị đại diện quan trọng từ vùng lân cận. Việc giảm kích thước này giúp giảm độ phức tạp tính toán và giảm số lượng tham số, đồng thời tạo ra tính tổng quát hóa trong mô hình và giúp mô hình trở nên bền vững hơn đối với các biến đổi nhỏ trong dữ liệu đầu vào.

4.3.3.2. Các dạng của lớp gộp (pooling layer)

❖ **Max Pooling:**

Mô tả: Trong max pooling, giá trị lớn nhất trong một vùng nhất định (thường là 2x2 hoặc 3x3 pixel - với kích cỡ tự chọn) được chọn làm đại diện cho vùng đó trong feature map.

Ví dụ: Giả sử bạn có một vùng 2x2 trên feature map với giá trị [1,3], [4,2]. Max pooling sẽ chọn giá trị 4 làm đại diện.

Công dụng: Max pooling giữ lại những đặc trưng nổi bật nhất, giúp mô hình tập trung vào những phần quan trọng của dữ liệu.

❖ **Average Pooling:**

Mô tả: Average Pooling lấy giá trị trung bình của tất cả các giá trị trong vùng được chọn để đại diện cho vùng đó.

Ví dụ: Sử dụng cùng vùng 2x2 trên với giá trị [1,3], [4,2]. Average Pooling sẽ tính giá trị trung bình là (1 + 3 + 4 + 2) / 4 = 2.5 làm giá trị đại diện.

Công dụng: Giúp giảm noise và đảm bảo rằng các đặc trưng ít nổi bật hơn cũng được tính toán, có thể hữu ích trong việc giữ lại thông tin ngữ cảnh cũng như giúp mô hình có một cách nhìn tổng quát về tổng thể dữ liệu.

❖ **Global Pooling:**

Mô tả: Có hai loại chính là Global Max Pooling và Global Average Pooling. Khác với hai loại trên, Global Pooling xử lý toàn bộ feature map để tạo ra một giá trị duy nhất cho mỗi map, thay vì tạo ra một feature map mới với kích thước giảm đi.

Ví dụ: Đối với một feature map 4x4, Global Average Pooling sẽ tính trung bình của tất cả 16 giá trị để cho ra một giá trị đại diện duy nhất. Còn với Global Max

Pooling lấy giá trị lớn nhất của tất cả 16 giá trị để cho ra một giá trị đại diện duy nhất.

Công dụng: Rất hữu ích cho việc giảm kích thước đầu vào trước khi truyền vào các lớp fully connected ở cuối mạng, giúp giảm số lượng tham số và tránh overfitting.

4.3.4. Các kiến trúc CNN nổi tiếng

4.3.4.1. LeNet-5 (1998)

Khởi đầu của CNN: LeNet-5 là một trong những mạng CNN đầu tiên, được thiết kế bởi Yann LeCun. Kiến trúc này bao gồm các lớp tích chập xen kẽ với các lớp gộp và được sử dụng rộng rãi trong nhận dạng ký tự và ảnh.

Ứng dụng: Nhận dạng số viết tay trong bưu điện.

4.3.4.2. AlexNet (2012)

Đột phá ImageNet: AlexNet với sự cải tiến đáng kể về độ sâu và kích thước của mạng, đã giành chiến thắng trong cuộc thi ImageNet LSVRC-2012. Kiến trúc này sử dụng ReLU để tăng tốc độ học và áp dụng Dropout để giảm overfitting.

Tính năng nổi bật: Sử dụng các kỹ thuật như ReLU và Dropout.

4.3.4.3. GoogLeNet (Inception v1, 2014)

Cải tiến về mô-đun: GoogLeNet đã giới thiệu khái niệm Inception module, cho phép mạng tự quyết định kích thước của bộ lọc tại mỗi tầng.

Tối ưu hóa: Thiết kế mạng sâu hơn mà không làm tăng số lượng tham số nhờ vào sự kết hợp các bộ lọc của các kích thước khác nhau.

4.3.4.4. ResNet (2015)

Giải quyết vấn đề biến mất gradient: ResNet giới thiệu khái niệm kết nối dư (residual connections) giúp dẫn truyền gradient hiệu quả hơn trong quá trình huấn luyện các mạng sâu và còn giúp giảm mất mát thông tin sau khi qua các lớp tích chập.

Kiến trúc sâu vượt trội: Đạt tới độ sâu 152 lớp, ResNet đã giúp cải thiện đáng kể hiệu suất các mô hình trên nhiều tác vụ liên quan đến thị giác máy tính.

4.3.4.5. SENet (2017)

Cải tiến chức năng của kênh: SENet giới thiệu kỹ thuật Squeeze-and-Excitation ("Squeeze" - tóm lược thông tin trên mỗi kênh và "Excitation" - trọng số lại thông tin), là một cách tiếp cận để cải thiện hiệu suất của các mạng CNN bằng cách nhấn mạnh vào việc học các trọng số của các kênh quan trọng và bỏ qua những kênh không cần thiết trong một lớp tích chập.

Tối ưu hóa hiệu quả: SENet tập trung vào việc tự động học được sự quan trọng tương đối của mỗi kênh trong các đặc trưng để cải thiện độ chính xác của mô hình, giúp mô hình tinh chỉnh kĩ hơn và ít bị overfitting hơn.

4.3.4.6. Xception (Extreme Inception)

Được đề xuất bởi François Chollet vào năm 2016, Xception là một kiến trúc mở rộng của mô hình Inception, với khái niệm chính là lớp separable convolution, trong đó phân tách quá trình học không gian và kênh thành hai phần riêng biệt, giúp ghi nhớ sự tương quan không gian, các đặc trưng cục bộ và toàn cục một cách hiệu quả.

Xception tập trung vào việc tối ưu hóa việc sử dụng các kích thước kernel nhỏ bằng cách thực hiện các hoạt động convolution 1x1 (depthwise separable convolution) để giảm độ phức tạp tính toán cũng như làm cho mô hình trở nên nhỏ hơn nhưng vẫn giữ được độ chính xác.

Bảng so sánh giữa convolutional layers thường và separable convolutional layers:

Đặc điểm	Convolutional layers thường	Separable convolutional layers
Số lượng tham số	Lớn, do cần một ma trận kernel 2D	Nhỏ, do sử dụng các kernel 1D riêng biệt
Hiệu quả tính toán	Thấp, do phải tính toán phép tích chập trên toàn bộ đầu vào	Cao, do sử dụng các phép tích chập 1D riêng biệt
Chia sẻ thông tin giữa các kênh	Không chia sẻ thông tin giữa các kênh	Chia sẻ thông tin giữa các kênh
Kích thước kernel	Sử dụng kernel 2D	Sử dụng kernel 1D
Overfitting	Dễ xảy ra do số lượng tham số lớn	Giảm nguy cơ do số lượng tham số ít hơn và chia sẻ thông tin giữa các kênh
Kích thước mô hình	Lớn, do số lượng tham số lớn	Nhỏ, do số lượng tham số ít hơn và hiệu quả tính toán cao hơn
Khả năng, hiệu suất học	Cao	Giới hạn
Độ khó mô hình	Thấp	Cao, do cần nhiều sự điều chỉnh tham số

Bảng 3. So sánh convolutional layers thường và separable convolutional layers

4.3.4.7. EfficientNet

EfficientNet là một loạt các mô hình với cải tiến lớn về hiệu suất và độ chính xác so với các mô hình trước đó.

Kiến trúc EfficientNet được chú ý vì đã thực hiện việc mở rộng đồng quy mô, sử dụng phương pháp tỉ lệ hóa mô hình (compound scaling). Nó cân nhắc cả ba yếu tố: độ sâu, độ rộng và độ phân giải của mô hình để đạt được hiệu suất tốt nhất với số lượng tham số ít hơn (áp dụng được cho các thiết bị bộ nhớ, công suất nhỏ như IoT, mobile…).

4.3.5. Áp dụng kỹ thuật transfer learning with pretrained models vào thực tế

4.3.5.1. Sử dụng mô hình đã huấn luyện (using pretrained models)

Tiết kiệm thời gian và nguồn lực: Việc sử dụng các mô hình đã được huấn luyện trước trên các bộ dữ liệu lớn giúp tiết kiệm đáng kể thời gian và công sức cần

thiết cho việc huấn luyện mô hình từ đầu.

Chuẩn bị dữ liệu: Quá trình chuẩn bị dữ liệu để phù hợp với yêu cầu đầu vào của mô hình đã huấn luyện, bao gồm thay đổi kích thước ảnh và áp dụng các phương pháp tiền xử lý tương tự như khi huấn luyện mô hình (hoặc tùy vào loại mô hình dùng để pretrained sẽ có các hàm tiền xử lí thuộc về riêng chúng, ta nên đọc các tài liệu có liên quan đến mô hình để áp dụng cho đúng).

4.3.5.2. Transfer learning với mô hình đã huấn luyện

Nên sử dụng transfer learning khi có nhiệm vụ hoặc dữ liệu tương tự nhưng không đủ lớn để huấn luyện một mô hình từ đầu một cách hiệu quả.

Cách thức hoạt động: Transfer learning sẽ tái sử dụng các lớp thấp hơn của một mô hình đã được huấn luyện (thường là các lớp trích xuất đặc trưng) và chỉ huấn luyện lại các lớp trên cùng hoặc thêm vào để phù hợp với nhiệm vụ mới.

4.3.6. Object Detection trong CNN

4.3.6.1. Phân biệt classification với object detection

Ta có object detection và image classification là hai tác vụ quan trọng trong lĩnh vực thị giác máy tính phục vụ các mục đích và dự án khác nhau:

Đặc điểm	Image Classification	Object Detection
Mục tiêu	Image classification tập trung vào việc xác định lớp của một hình ảnh hoặc đối tượng chính trong hình ảnh là gì.	Object detection sẽ nhận diện lớp và xác định vị trí cụ thể của các đối tượng trong hình ảnh bằng cách vẽ bounding box xung quanh từng đối tượng.
Đầu vào	Một hình ảnh duy nhất	Một hình ảnh với nhiều đối tượng và thông tin về vị trí của các đối tượng đó.
Đầu ra	Nhãn của hình ảnh.	Bounding box và nhãn của mỗi đối tượng.
Độ phức tạp	Thấp, đơn giản, không đòi hỏi phát hiện hoặc đánh dấu các đối tượng riêng biệt.	Cao, phức tạp do kết hợp cả phân loại và định vị đối tượng.
Ứng dụng	Nhận diện khuôn mặt, xe cộ, biển báo giao thông.	Xe tự lái, nhận dạng khuôn mặt.
Mô hình	YOLO, Faster R-CNN, SSD…	VGG, ResNet, Inception…
Số liệu đánh giá	Accuracy, precision, recall.	Intersection over Union (IoU), mean Average Precision (mAP), precision, recall.
Tốc độ	Nhanh, áp dụng cho danh mục quan trọng.	Chậm, hữu ích cho xác định danh mục và vị trí đối tượng.

Bảng 4. So sánh Image classification và Object detection

4.3.6.2. Nhận diện đối tượng (object detection)

(Single) object detection là quá trình xác định vị trí và phân loại (một) đối tượng trong hình ảnh. Mục tiêu là phát hiện đối tượng và vẽ một khung giới hạn quanh nó.

Dữ liệu huấn luyện (training data) phục vụ cho object detector cần có tối thiểu 3 thành phần sau:

❖ **Hình ảnh:**

Là tập dữ liệu gồm các hình ảnh chứa các đối tượng mà bạn muốn object detector có thể phát hiện.

Hình ảnh có thể được thu thập ở nguồn khác nhau, nhiều định dạng khác nhau như JPG, PNG, TIFF, … Và các hình ảnh này cần phản ánh các điều kiện khác nhau, đa dạng góc độ… (có thể áp dụng làm giàu dữ liệu, tiền xử lí).

Chất lượng hình ảnh ảnh hưởng trực tiếp đến hiệu suất, khả năng tổng quát hóa của mô hình object detector.

❖ **Hộp bao (Bounding boxes):**

Là các hình chữ nhật được vẽ thủ công xung quanh mỗi đối tượng trong hình ảnh.

Bounding boxes xác định vị trí (góc trái trên và góc phải dưới hoặc …) và kích thước của đối tượng trong hình ảnh, giúp phân biệt với các đối tượng khác nhau.

Chất lượng bounding boxes ảnh hưởng đến độ chính xác của object detector vì vậy cần phải thực hiện việc vẽ bounding boxes có độ đảm bảo, tỉ lệ hình chuẩn và cân đối.

❖ **Nhãn lớp đối tượng (Class labels object):**

- Là tên, là dữ liệu đầu ra mong muốn cho mô hình phân loại các đối tượng trong hình ảnh.
- Mỗi bounding boxes cần được gán nhãn lớp chính xác để object detector có thể học cách phân biệt các đối tượng khác nhau.
- Có thể dùng các công cụ đánh nhãn dán chính như LabelImg, VGG Image Annotator…, sẽ giúp dễ dàng tạo nhãn dữ liệu cho quá trình huấn luyện mô hình phát hiện đối tượng .
- Ví dụ: nhãn lớp có thể là "chó", "mèo", "xe", "người", … trong một bức hình.

Ngoài 3 thành phần tối thiểu này, dữ liệu huấn luyện tốt có thể bao gồm: dữ liệu đa dạng (làm giàu dữ liệu), dữ liệu được trọn lọc, đánh nhãn cẩn thận, dữ liệu được cân bằng, bổ sung thêm các keypoint chính của đối tượng giúp dễ dàng hơn trong việc nhận diện và phân biệt. Hoặc bổ sung thêm các thông tin về bối cảnh nhằm hiểu rõ hơn về các đối tượng cần phân loại. Sau khi chuẩn bị được kĩ càng các dữ liệu tốt nhất, ta sẽ bắt đầu huấn luyện mô hình nhận diện đối tượng với các thông tin dữ liệu trên.

4.3.6.3. Phương pháp cơ bản phát hiện đa đối tượng (detecting multiple objects)

❖ **Sliding CNNs (CNN Trượt)**

- **Cơ chế Hoạt động:**

Sliding CNN còn gọi là convolutional implementation của sliding windows, là một kỹ thuật trong đó một mạng CNN được trượt qua toàn bộ ảnh để phát hiện các đối tượng tại nhiều vị trí và tỷ lệ khác nhau.

Khác với phương pháp truyền thống, CNN trượt áp dụng toàn bộ mạng cho từng vùng của ảnh, cho phép phát hiện đối tượng hiệu quả hơn và giảm thời gian tính toán.

Ngoài ra, sliding CNN còn áp dụng thêm một số phương pháp tiền xử lí dữ liệu rồi trượt qua lần nữa để tăng khả năng nhận diện được toàn bộ vật thể trong ảnh.

- **Lợi ích:**

Tối ưu hóa hiệu suất bằng cách giảm số lượng phép tính cần thiết so với phương pháp trượt cửa sổ truyền thống, nhờ vào việc chia sẻ tính toán giữa các vùng chồng chéo.

❖ **Non-max suppression (NMS)**

- **Xử lý Khung Chồng Chéo:**

Non-max suppression là một kỹ thuật được sử dụng để loại bỏ các khung giới hạn dư thừa, giữ lại chỉ khung có điểm tự tin cao nhất trong số các khung chồng chéo lên nhau.

NMS duyệt qua các khung giới hạn dựa trên điểm tự tin và loại bỏ các khung giới hạn có độ chồng lấn IoU (Intersection over Union) lớn hơn ngưỡng nhất định với khung có điểm cao nhất .

- **Bước Thực Hiện:**

Bước 1: Xếp hạng các khung giới hạn dựa trên điểm tự tin.

Bước 2: So sánh độ chồng lấn IoU của khung hiện tại với khung có điểm cao nhất.

Bước 3: Loại bỏ các khung có IoU cao hơn ngưỡng đã định.

Bước 4: Lặp lại cho đến khi tất cả các khung được xử lý.

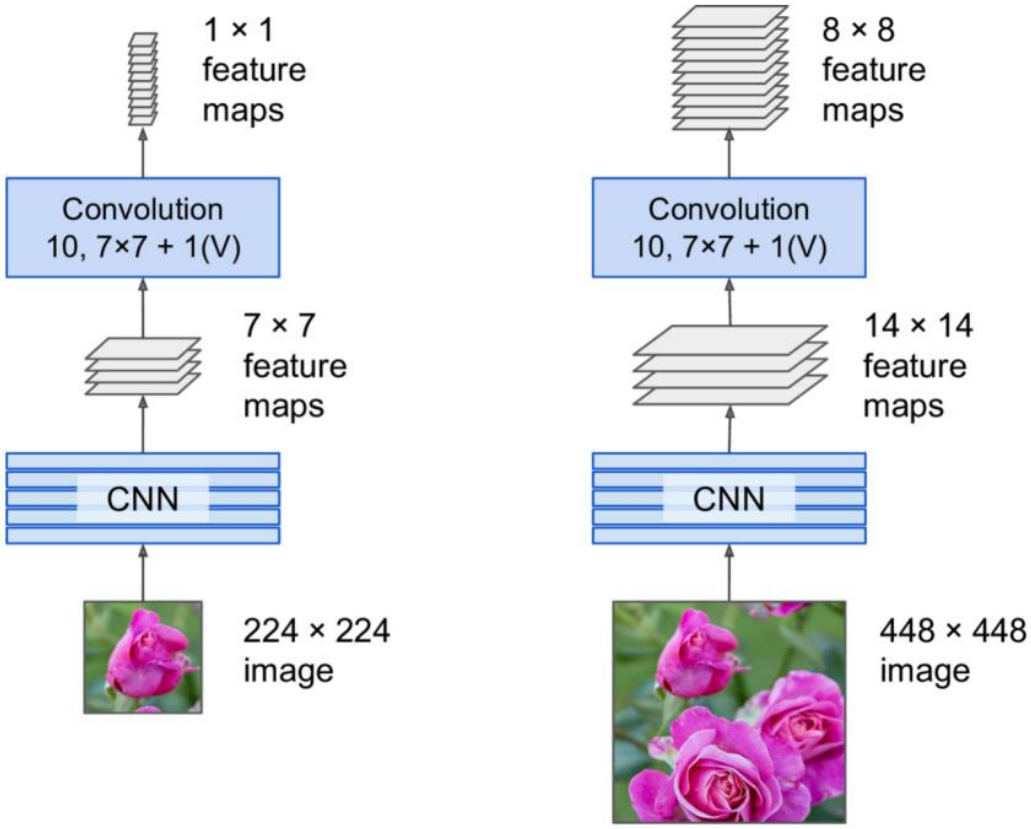
4.3.6.4. Mạng hoàn toàn tích chập (fully convolutional networks) dùng trong nhận diện đa đối tượng

❖ **Khái niệm:**

FCNs (hay fully convolutional networks) là một loại mạng nơ-ron tích chập được thiết kế để xử lý hình ảnh có kích thước bất kỳ, chuyên dụng cho việc phân đoạn hình ảnh (image segmentation) và phát hiện đối tượng (object detection).

Khác với các mạng CNN truyền thống, FCNs sẽ thay thế hoàn toàn các tầng kết nối đầy đủ bằng các tầng tích chập, cho phép đầu ra có kích thước không cố định. Với khả năng tự thích ứng với kích thước ảnh đó sẽ giúp FCNs trở nên lý tưởng cho các ứng dụng thực tế, nơi kích thước của đối tượng và hình ảnh có thể thay đổi đáng

kể.



Hình 1. The same fully convolutional network processing a small image (left) and a large one (right)

❖ **Cấu trúc:**

Kiến trúc: Bao gồm các tầng tích chập và tầng gộp xuống, sau đó là các tầng tích chập mở rộng (upsampling layers) để phục hồi kích thước ban đầu của ảnh hoặc bản đồ đặc trưng.

Upsampling: Sử dụng các kỹ thuật như tranposed convolution (deconvolution) để tăng kích thước của feature maps, giúp phục hồi chi tiết hình ảnh cho việc phân đoạn.

4.3.6.5. Một số mạng nhận diện đối tượng nổi tiếng (object detection NNs)

❖ **Faster R-CNN**

Giới thiệu: Faster R-CNN được cải tiến từ các phiên bản R-CNN trước đó bằng cách sử dụng một mạng đề xuất vùng quan tâm (Region Proposal Network - RPN) để tăng tốc quá trình phát hiện đối tượng.

Cơ chế hoạt động: RPN dùng để xác định các vùng quan tâm tiềm năng trên toàn bộ ảnh, sau đó kết hợp với mạng phân loại để xác định và định vị chính xác các đối tượng.

❖ **YOLO (You Only Look Once)**

Đặc điểm: YOLO là một mạng phát hiện đối tượng vô cùng nhanh, có thể xử lý ảnh trong thời gian thực với độ chính xác cao.

Cách thức: YOLO xem xét toàn bộ ảnh một lần duy nhất (do đó có tên "You Only Look Once") và phân chia hình ảnh thành các lưới, từng lưới phát hiện đối tượng.

❖ **SSD (Single Shot MultiBox Detector)**

Tối ưu hóa: SSD kết hợp lợi thế của YOLO về tốc độ và khả năng của Faster R-CNN trong việc định vị chính xác đối tượng.

Hiệu suất: SSD không chỉ nhanh mà còn chính xác, nhờ vào việc sử dụng nhiều bộ lọc ở các tỷ lệ khác nhau để phát hiện đối tượng ở nhiều kích cỡ khác nhau.

❖ **Mask R-CNN**

Phát triển từ Faster R-CNN: Mask R-CNN xây dựng trên kiến trúc của Faster R-CNN bằng cách thêm một nhánh để dự đoán mặt nạ phân đoạn cho mỗi vùng quan tâm.

Ứng dụng: Rất hiệu quả trong các tác vụ phân đoạn đối tượng nâng cao, nơi cần phát hiện và xác định hình dạng chính xác của các đối tượng.

4.3.6.6. Phương pháp đánh giá mạng nhận diện vật thể

❖ **Mean Average Precision (mAP)**

Định nghĩa: mAP là trung bình của precision trên tất cả các lớp đối tượng, ở các ngưỡng IoU khác nhau.

Tính toán:

- Tính precision và recall cho từng lớp đối tượng.
- Áp dụng ngưỡng intersection over union (IoU) để xác định độ chính xác của khung giới hạn.
- Tính average precision (AP) cho mỗi lớp, sau đó lấy trung bình cộng các AP để được mAP.

❖ **Intersection over Union (IoU)**

Công thức: $\text{IoU} = \frac{\text{Area of Union}}{\text{Area of Overlap}}$

Ứng dụng: Được sử dụng để đo lường độ chính xác của khung giới hạn dự đoán so với khung giới hạn thực tế. IoU cao cho thấy khung giới hạn dự đoán khớp chặt chẽ với đối tượng thực tế.

❖ **Precision và Recall**

Precision: Tỉ lệ của các dự đoán đúng (true positives) so với tổng số dự đoán là đối tượng (true positives + false positives).

Recall: Tỉ lệ của các dự đoán đúng so với tổng số đối tượng thực tế (true positives + false negatives).

Tính toán: Sử dụng confusion matrix để xác định số lượng true positives, false positives, và false negatives.

❖ **F1 Score**

Định nghĩa: F1 Score là trung bình điều hòa của precision và recall.

Công thức: $\mathbf{F1} = \mathbf{2} \times \frac{\mathbf{Precision} + \mathbf{Recall}}{\mathbf{Precision} \times \mathbf{Recall}}$

Tầm quan trọng: F1 Score cung cấp một độ đo tổng hợp giúp đánh giá chất lượng của mô hình khi cả precision và recall đều quan trọng.

4.3.7. Sơ lược về semantic segmentation trong CNN

❖ **Lí thuyết:**

Semantic Segmentation (phân đoạn ngữ nghĩa) là một tác vụ xử lý hình ảnh phức tạp trong lĩnh vực thị giác máy tính, trong đó mỗi pixel của hình ảnh được phân loại theo lớp đối tượng mà nó thuộc về. Ví dụ, trong một hình ảnh, các pixel thuộc về các đối tượng như đường, xe hơi, người đi bộ hoặc tòa nhà được xác định và phân loại vào các danh mục riêng biệt.

Khác với việc thực hiện object detection - nơi mục tiêu là phát hiện ra các đối tượng và ranh giới của chúng thì semantic segmentation cung cấp một bộ phân loại chi tiết hơn ở cấp độ pixel cho từng đối tượng và ranh giới chính xác của chúng.

❖ **Thách thức:**

Một thách thức chính trong semantic segmentation là việc mất độ phân giải không gian khi hình ảnh đi qua các lớp của mạng nơ-ron tích chập (CNN). Quá trình này xảy ra chủ yếu là trong các lớp gộp (pooling) hoặc các lớp có bước nhảy lớn hơn một, thường làm mất đi các chi tiết không gian quan trọng. Điều này dẫn đến nhu cầu phải phục hồi thông tin không gian một cách chính xác để đảm bảo mỗi pixel được phân loại đúng.

❖ **Giải pháp:**

Một cách tiếp cận phổ biến để giải quyết thách thức này bao gồm sử dụng một mạng hoàn toàn tích chập (FCN), có thể xử lý hình ảnh có kích thước bất kỳ và thường được điều chỉnh từ các CNN đã được huấn luyện trước bằng cách chuyển đổi các lớp dày đặc thành các lớp tích chập.

Sự điều chỉnh này bao gồm việc thực hiện kỹ thuật upsampling, giúp lấy lại độ phân giải không gian đã mất trong các lớp trước, do đó cải thiện độ chính xác của phân loại ở cấp độ pixel. Hoặc áp dụng các kĩ thuật nâng cấp mạng CNN gốc, bổ sung thêm các lớp sau thích hợp, hoặc thậm chí có thể mở rộng vượt quá kích thước của hình ảnh gốc. Tất cả các điều này có thể được sử dụng để tăng độ phân giải của hình ảnh, đây là một kỹ thuật được gọi là super-resolution.

4.4. Preprocess sequences using RNN

❖ **Dự đoán tương lai**

Dự đoán tương lai sẽ làm tất cả mọi thứ người dùng làm. Chẳng hạn khi người dùng nhắn tin bằng bàn phím ảo trên điện thoại, thì bàn phím có hiển thị lên những từ gợi ý sau đó.

❖ **Sequential data**

RNN hoạt động trên các chuỗi có độ dài tùy ý.

4.4.1. RNN's fundamental concepts

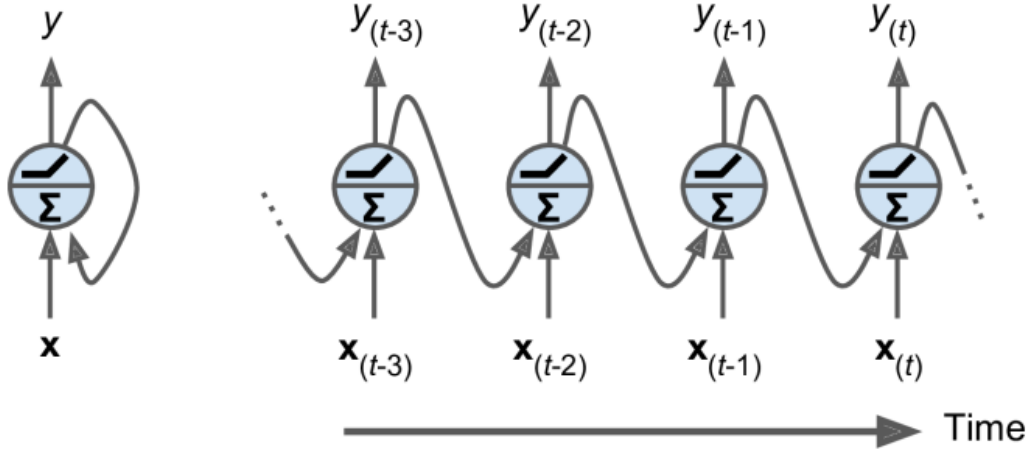
❖ **Recurrent neuron**

Mạng neuron hồi quy trông rất giống mạng neuron tiếp nối, ngoại trừ việc nó cũng có các kết nối hướng ngược lại.

RNN đơn giản nhất bao gồm một neuron nhận các input, sản sinh ra một output và gửi lại output về chính nó. Ở mỗi bước thời gian t (hay còn gọi là frame), recurrent neuron nhận các input là $x_{(t)}$ cũng như là output từ thời gian trước đó, $y_{(t-1)}$. Bởi vì không có ouput trước đó ở lần đầu tiên nên đặt $y_0 = 0$.

Hình 2. A recurrent neuron (left)

❖ **Unrolling the network through time:** Là cùng một neuron hồi quy được biểu diễn một lần trong mỗi bước thời gian



Hình 3. A recurrent neuron (left) unrolled through time (right)

❖ **Recurrent layers**

Trong RNN, lớp tái phát bao gồm các nút mà mỗi nút của nó lưu giữ trạng thái từ một bước thời gian trước đó, và sử dụng trạng thái này cùng với dữ liệu đầu vào hiện tại để tính toán trạng thái hiện tại.

❖ **Weight of recurrent layer**

Mỗi neuron có 2 tập weight: một tập dành cho các input x_t được gọi là $\mathbf{w_x}$ và một tập dành cho các output của bước thời gian trước $y_{(t-1)}$ được gọi là $\mathbf{w_y}$

Nếu xem xét toàn bộ recurrent layer thay vì chỉ một recurrent neuron, có thể xem

toàn bộ vector trong hai ma trận weight là W_x và W_y. Vector output của toàn bộ recurrent layer được tính toán như sau:

$$\mathbf{y}_{(t)} = \phi\left(\mathbf{W}_x{}^{\mathsf{T}}\mathbf{x}_{(t)} + \mathbf{W}_y{}^{\mathsf{T}}\mathbf{y}_{(t-1)} + \mathbf{b}\right)$$

Hình 4. Output of a recurrent layer for a single instance

Trong đó: b là vector bias, ϕ là hàm hoạt động (activation function)

❖ **Function y_t**

Cũng như các mạng neuron nối tiếp, có thể tính toán output của recurrent layer trong một lần chụp toàn bộ mini-batch bằng cách đặt tất cả input tại bước thời gian t vào ma trận input

$$\begin{aligned}\mathbf{Y}_{(t)} &= \phi\left(\mathbf{X}_{(t)}\mathbf{W}_x + \mathbf{Y}_{(t-1)}\mathbf{W}_y + \mathbf{b}\right)\\ &= \phi\left(\begin{bmatrix}\mathbf{X}_{(t)} & \mathbf{Y}_{(t-1)}\end{bmatrix}\mathbf{W} + \mathbf{b}\right) \text{ with } \mathbf{W} = \begin{bmatrix}\mathbf{W}_x\\ \mathbf{W}_y\end{bmatrix}\end{aligned}$$

Hình 5. Outputs of a layer of recurrent neurons for all instances in a mini-batch

Trong đó:

- $Y_{(t)}$ là một ma trận m x $n_{neurons}$ chứa các output của các lớp tại bước thời điểm t cho mỗi bản thể trong mini-batch (m là số bản thể có trong mini-batch và $n_{neurons}$ là số lượng neuron)
- $X_{(t)}$ là một ma trận m x n_{inputs} chứa các input cho tất cả các bản thể (n_{inputs} là số lượng các đặc trưng đầu vào)
- W_x là một ma trận n_{inputs} x $n_{neurons}$ chứa chứa trọng số kết nối cho các input của bước thời gian hiện tại
- W_y là một ma trận $n_{neurons}$ x $n_{neurons}$ chứa chứa trọng số kết nối cho các output của bước thời gian trước đó.
- b là một vector có kích thước là $n_{neurons}$ chứa bias của mỗi neuron
- Các ma trận trọng số W_x và W_y thường được ghép theo chiều dọc thành một ma trận trọng số W có dạng (n_{inputs} + $n_{neurons}$)x $n_{neurons}$
- Ký hiệu [$X_{(t)}$ $Y_{(t-1)}$] thể hiện sự nối ngang của ma trận $X_{(t)}$ và $Y_{(t-1)}$

❖ **Memory Cells:**

Bởi vì output của một recurrent neuron ở bước thời gian t là một hàm của tất cả các input từ bước thời gian trước đó nên có thể nói đó là một dạng của memory.

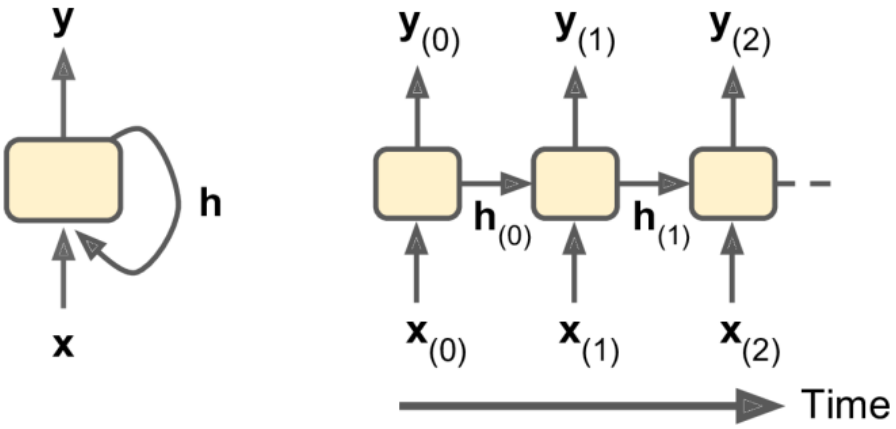
Một phần của mạng lưới thần kinh bảo toàn một số trạng thái qua các bước thời gian được gọi là một memory cell (hoặc đơn giản là cell)

Một tế bào thần kinh tái phát đơn lẻ hoặc một lớp tế bào thần kinh tái phát là một tế bào rất cơ bản, chỉ có khả năng học các mẫu ngắn (thường dài khoảng 10 bước, nhưng điều này thay đổi tùy theo nhiệm vụ).

❖ **Trạng thái của cell:**

Tại mỗi bước thời gian t trạng thái của một cell là một hàm của một số input tại bước thời gian đó và trạng thái của nó ở bước thời gian trước đó: $\mathbf{h_{(t)} = f(h_{(t-1)}, x_{(t)})}$.

Output của nó ở bước thời gian t, ký hiệu là y, cũng là một hàm của trạng thái trước đó và các input hiện tại.



Hình 6. A cell's hidden state and its output may be different

❖ **Input and Output Sequences:**

Một mạng RNN có thể được sử dụng để thực hiện các loại mạng khác nhau phụ thuộc vào cách mà dữ liệu được đưa vào và đầu ra được sử dụng.

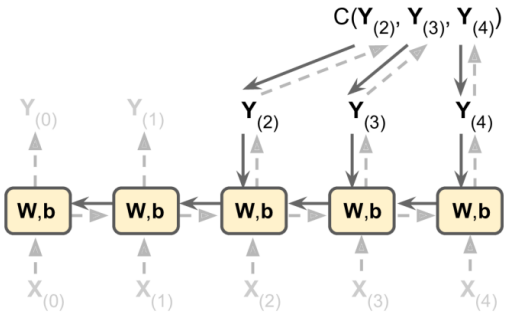
- Mạng từ dãy đến dãy: Có thể dùng để dự đoán các chuỗi dữ liệu, ví dụ như giá cổ phiếu. Đưa vào dãy giá cổ phiếu qua N ngày gần nhất và đầu ra sẽ là giá cổ phiếu của ngày tiếp theo.
- Mạng từ dãy đến vector: Đưa vào một dãy dữ liệu và chỉ quan tâm đến đầu ra cuối cùng, ví dụ như phân tích cảm xúc từ một đoạn văn.
- Mạng từ vector đến dãy: Đưa vào cùng một vector đầu vào ở mỗi bước thời gian và tạo ra một dãy đầu ra, ví dụ như việc tạo ra một phụ đề cho một hình ảnh.
- Mạng từ dãy đến vector, và từ vector đến dãy: Sử dụng một phần mạng để mã hóa dữ liệu từ dãy thành một vector và một phần khác để giải mã từ vector này thành dãy mới. Ví dụ, dịch câu từ một ngôn ngữ sang ngôn ngữ khác, nơi mà cần phải xem xét toàn bộ câu trước khi dịch để đảm bảo tính chính xác.

4.4.2. Training RNN

Để huấn luyện một RNN, một cách thường được sử dụng là "unroll" nó qua thời gian và sau đó đơn giản sử dụng thuật toán lan truyền ngược thông thường. Chiến lược này được gọi là backpropagation through time (BPTT)

Quy trình này bao gồm các bước cơ bản sau đây. Đầu tiên, chúng ta "unroll" mạng RNN qua thời gian để tạo ra một mạng trải rộng với nhiều bước thời gian. Tiếp theo, chúng ta thực hiện một lần lan truyền xuôi qua mạng đã được "unroll". Sau đó, dãy đầu ra được đánh giá bằng một hàm chi phí, có thể bỏ qua một số đầu ra nếu cần thiết. Đồng thời, chúng ta lan truyền ngược các đạo hàm của hàm chi phí qua mạng "unroll" để tính toán các gradient. Các tham số của mô hình được cập nhật bằng cách sử dụng các đạo hàm tính được. Điều đáng lưu ý là các đạo hàm chảy ngược không chỉ qua đầu ra cuối cùng mà còn qua tất cả các đầu ra được sử dụng bởi hàm chi phí. Với việc sử dụng cùng các tham số ở mỗi bước thời gian, lan truyền ngược tự động tính tổng qua tất cả các bước thời gian, giúp cập nhật hiệu quả các tham số của mạng

RNN



Hình 7. Backpropagation through time

4.4.3. Implement RNNs

❖ **Time series**

Một chuỗi gồm một hoặc nhiều giá trị trong mỗi bước thời gian.

- Giá trị đơn cho mỗi bước thời gian được gọi là univariate (đơn biến)
- Nhiều giá trị cho mỗi bước thời gian được gọi là multivariate (đa biến)

Nhiệm vụ điển hình của time series là dự đoán các giá trị trong tương lai, được gọi là forecasting. Một nhiệm vụ phổ biến khác là điền vào chỗ trống: predict (hay đúng hơn là "postdict") các giá trị còn thiếu trong quá khứ. Điều này được gọi là sự quy kết (imputation).

❖ **Baseline Metrics**

Trước khi bắt đầu sử dụng RNNs, việc tạo ra và đánh giá một số chỉ số cơ sở thường là một ý tưởng tốt. Nếu không có các chỉ số cơ sở này, có thể xảy ra trường hợp chúng ta nghĩ rằng mô hình của mình hoạt động tốt trong khi thực tế nó lại kém hơn các mô hình cơ bản. Phương pháp này được gọi là dự báo ngây thơ (naive forecasting)

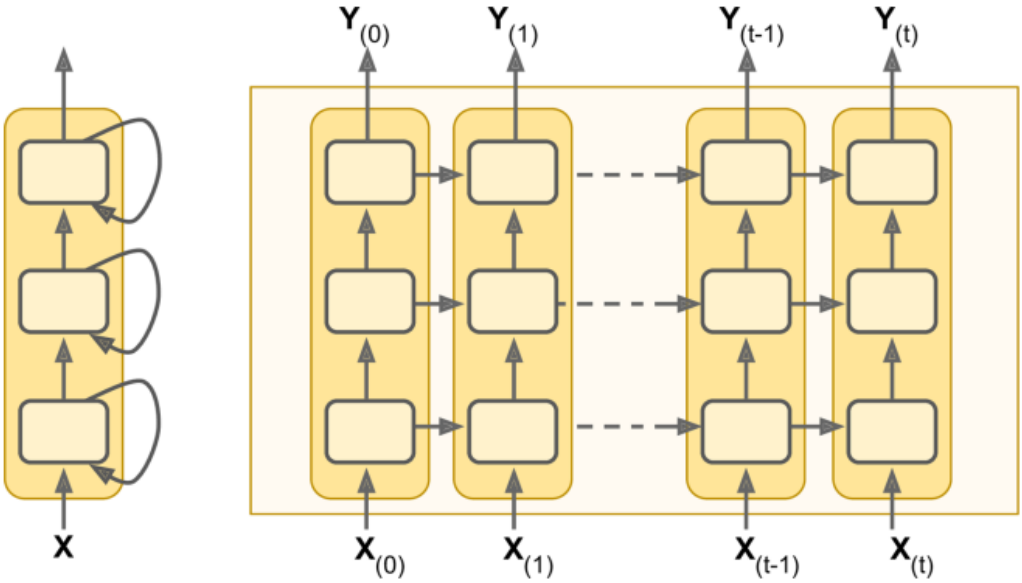
❖ **Simple RNN**

Mô hình RNN đơn giản nhất bao gồm một lớp duy nhất với một neuron. Trong mô hình này, không cần xác định độ dài của chuỗi đầu vào vì RNN có thể xử lý bất kỳ số lượng bước thời gian nào. Lớp SimpleRNN sử dụng hàm kích hoạt hyperbolic tangent mặc định.

Quá trình hoạt động của mô hình bao gồm việc thiết lập trạng thái ban đầu là 0, truyền nó vào một neuron hồi tiếp duy nhất cùng với giá trị của bước thời gian đầu tiên. Neuron tính tổng trọng số của các giá trị này, áp dụng hàm kích hoạt hyperbolic tangent, và đầu ra này cũng là trạng thái mới. Trạng thái mới này được truyền đến neuron hồi tiếp cùng với giá trị đầu vào tiếp theo, và quá trình lặp lại cho đến khi đến bước thời gian cuối cùng. Cuối cùng, mô hình chỉ đơn giản là đầu ra giá trị cuối cùng. Tất cả các bước này được thực hiện đồng thời cho mỗi chuỗi thời gian.

❖ **Deep RNN**

Rất phổ biến khi xếp nhiều lớp neuron hồi tiếp lên nhau. Điều này tạo ra một mạng RNN sâu.



Hình 8. Deep RNN (left) unrolled through time (right)

4.4.4. Các biến thể của LSTM cells và NNS khác xử lý dữ liệu tuần tự

4.4.4.1. RNNs with 1D conv layers

LSTM và GRU là một trong những lý do chính sau thành công của mạng nơ-ron hồi tiếp (RNN). Tuy nhiên, mặc dù chúng có thể xử lý các chuỗi dài hơn nhiều so với các RNN đơn giản, chúng vẫn có bộ nhớ ngắn hạn khá hạn chế, và gặp khó khăn trong việc học các mẫu dài hạn trong các chuỗi có 100 bước thời gian trở lên, như các mẫu âm thanh, chuỗi thời gian dài, hoặc các câu dài. Một cách để giải quyết vấn đề này là làm ngắn đi các chuỗi đầu vào, ví dụ bằng cách sử dụng các lớp tích chập 1D.

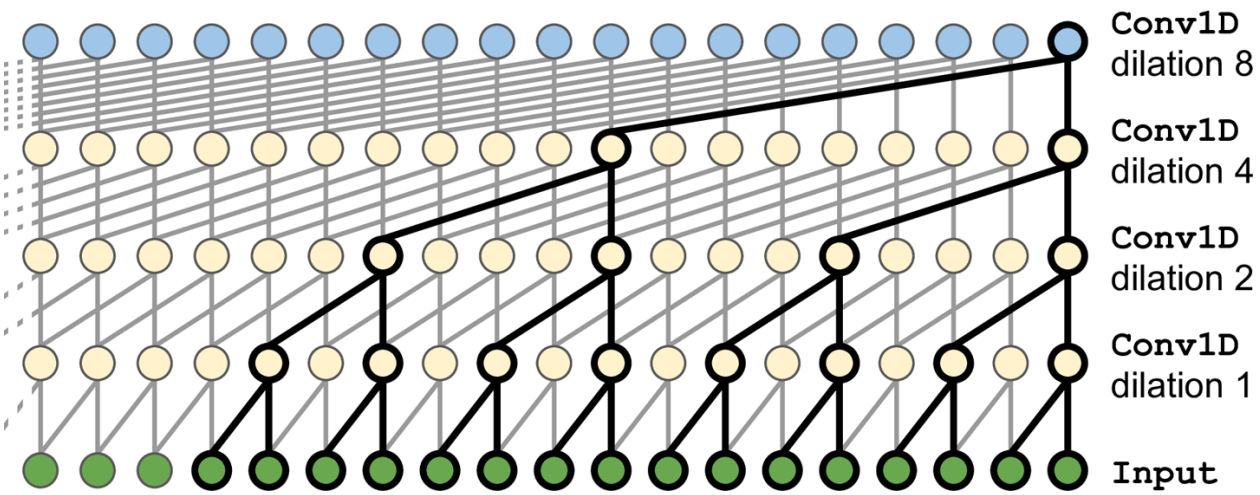
❖ **RNNs with 1D conv layers**

Một lớp tích chập 2D hoạt động bằng cách trượt một số lõi (hoặc bộ lọc) khá nhỏ qua một hình ảnh, tạo ra nhiều bản đồ đặc trưng 2D (mỗi lõi tương ứng với một bản đồ đặc trưng). Tương tự, một lớp tích chập 1D trượt một số lõi qua một chuỗi, tạo ra một bản đồ đặc trưng 1D cho mỗi lõi. Mỗi lõi sẽ học cách phát hiện một mẫu tuần tự rất ngắn (không dài hơn kích thước của lõi).

Xây dựng một mạng nơ-ron bao gồm một sự kết hợp của các lớp hồi tiếp và các lớp tích chập 1D (hoặc thậm chí là các lớp lọc 1D). Nếu bạn sử dụng một lớp tích chập 1D với bước nhảy là 1 và "same" padding, thì đầu ra của chuỗi sẽ có cùng độ dài với đầu vào. Nhưng nếu bạn sử dụng "valid" padding hoặc một bước nhảy lớn hơn 1, thì đầu ra của chuỗi sẽ ngắn hơn đầu vào, vì vậy hãy đảm bảo bạn điều chỉnh các mục tiêu tương ứng.

4.4.4.2. WaveNet

WaveNet một kiến trúc mạng nơ-ron sử dụng các lớp tích chập 1D được xếp chồng lên nhau. Tăng tỷ lệ dilation (khoảng cách giữa các đầu vào của mỗi neuron) lên gấp đôi ở mỗi lớp, cho phép mạng học các mẫu ngắn hạn ở các lớp thấp và các mẫu dài hạn ở các lớp cao hơn. Nhờ tỷ lệ dilation tăng gấp đôi, mạng có khả năng xử lý các chuỗi cực kỳ lớn một cách hiệu quả.

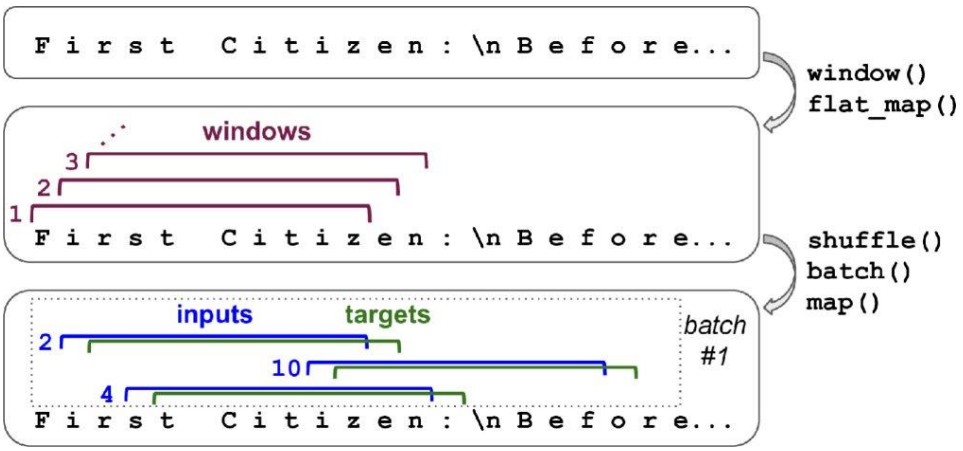


Hình 9. WaveNet architecture

4.5. Natural Language Processing

4.5.1. Char-RNN

Char-RNN có thể được sử dụng để tạo ra văn bản mới, một ký tự mỗi lần. Điều ấn tượng ở đây là mô hình đã có thể học được từ vựng, ngữ pháp, dấu câu chính xác và nhiều điều khác chỉ bằng cách học dự đoán ký tự tiếp theo trong một câu.



Hình 10. Preparing a dataset of shuffled windows

4.5.2. Stateful RNNs

❖ **Stateless vs Stateful RNNs**

Stateless RNN bắt đầu mỗi lần lặp huấn luyện với trạng thái ẩn mới và loại bỏ nó sau mỗi lần xử lý lô dữ liệu, tiêu tốn ít tài nguyên tính toán hơn.

Stateful RNN lưu giữ trạng thái cuối cùng giữa các lô dữ liệu huấn luyện, giúp học mẫu dài hạn.

❖ **Stateful RNNs**

Stateful RNN là một loại mạng RNN mà giữ lại trạng thái cuối cùng sau khi xử lý một lô dữ liệu huấn luyện và sử dụng nó làm trạng thái ban đầu cho lô dữ liệu huấn luyện tiếp theo. Điều này cho phép mô hình học các mẫu dài hạn mặc dù chỉ lan truyền ngược qua các chuỗi ngắn.

PHẦN 3: LẬP TRÌNH

CHƯƠNG 1: GIỚI THIỆU

1.1. Tổng quan về khối u nội sọ và tầm quan trọng của Segmentation

Khối u não, một trong những bệnh lý ung thư nguy hiểm và phức tạp, đòi hỏi việc chẩn đoán và theo dõi chính xác qua các phương pháp hình ảnh như MRI và CT scan. Trong đó, việc áp dụng phân đoạn hình ảnh, hay segmentation, là bước thiết yếu để xác định kích thước, vị trí và ranh giới của khối u, từ đó hỗ trợ các bác sĩ trong việc lên kế hoạch điều trị và đánh giá hiệu quả liệu pháp. Công nghệ AI và học máy đã làm tăng đáng kể độ chính xác của quá trình này, góp phần cải thiện kết quả điều trị và giám sát bệnh tiến triển.

1.1.1. Định nghĩa và phân loại khối u nội sọ

Có hai loại khối u não chính:

- U não nguyên phát, bao gồm các khối u phát triển trực tiếp từ não hoặc các mô liên quan như u thần kinh đệm (bao gồm u tế bào hình sao, u thần kinh đệm ít gai và u màng não thất) và u lympho ở hệ thần kinh trung ương. Khối u này còn có thể phát triển từ các cấu trúc nằm ngoài dây thần kinh, như u màng não và u dây thần kinh tiền đình ốc tai.
- U não thứ phát hoặc di căn, xuất phát từ các mô ngoài não và sau đó di căn đến não.

Trong thực tế, u não di căn xuất hiện nhiều hơn đến 10 lần so với các khối u nguyên phát.

1.1.2. Các phương pháp điều trị hiện hành

Trong điều trị u não, việc bảo vệ đường thở là ưu tiên hàng đầu, đặc biệt đối với bệnh nhân trong tình trạng hôn mê hoặc có phản xạ đường thở giảm. Việc sử dụng dexamethasone giúp giảm áp lực nội sọ, trong khi mannitol được chỉ định cho trường hợp thoát vị não do khối u. Thuốc chống động kinh cũng cần thiết để xử lý tình trạng động kinh.

Đối với các khối u não, các phương pháp điều trị có thể bao gồm phẫu thuật cắt bỏ, xạ trị, hóa trị, liệu pháp nhắm đích và liệu pháp miễn dịch. Đặc biệt, phẫu thuật thường được ưu tiên để cải thiện kết quả điều trị cho bệnh nhân có khối u di căn.

Về xạ trị, có hai phương pháp chính là xạ trị phù hợp và xạ trị lập thể, tùy theo mục tiêu bảo vệ mô não bình thường. Các khối u như u thần kinh đệm có thể được điều trị bằng xạ trị phù hợp, trong khi các trường hợp di căn phù hợp với xạ trị lập thể hoặc xạ trị khu trú.

Tuy nhiên, độc tính thần kinh sau xạ trị là một vấn đề nghiêm trọng, bao gồm độc tính cấp tính và xuất hiện muộn. Độc tính cấp tính thường bao gồm triệu chứng như đau đầu, buồn nôn, và nôn. Độc tính xuất hiện muộn có thể gây ra các biến chứng như bệnh tủy sống, teo vỏ não, và tổn thương chất trắng, đòi hỏi các phương pháp chẩn đoán và điều trị phức tạp để giảm thiểu tác động đến bệnh nhân.

1.2. Lý do chọn kiến trúc U-net cho việc Segmentation

U-Net, một kiến trúc mạng neural tích chập được phát triển chuyên biệt cho nhiệm vụ phân đoạn hình ảnh, đã nhanh chóng trở thành một trong những công cụ chuẩn trong lĩnh vực y tế. Đặc biệt, U-Net có nhiều ưu điểm khiến nó trở nên phù hợp cho việc xử lý các hình ảnh y tế, từ phân đoạn tổ chức đến phân tích cấu trúc trong các loại hình ảnh như CT và MRI.

Một trong những đặc điểm nổi bật của U-Net là khả năng làm việc hiệu quả ngay cả với lượng dữ liệu huấn luyện nhỏ. Điều này rất quan trọng trong lĩnh vực y tế, nơi mà dữ liệu thường khan hiếm và việc thu thập cũng như gán nhãn dữ liệu có thể tốn kém và thời gian.

Kiến trúc của U-Net bao gồm hai phần chính: phần thu hẹp (contracting path) giúp học các đặc trưng sâu và phần mở rộng (expansive path) giúp khôi phục kích thước hình ảnh, từ đó cải thiện độ chính xác trong việc phân đoạn. Phần mở rộng được hỗ trợ bởi các kết nối qua (skip connections), giúp truyền dẫn thông tin từ các lớp đầu vào đến các lớp đầu ra, qua đó bảo toàn được thông tin quan trọng cho việc phục hồi chi tiết của hình ảnh.

U-Net đã được áp dụng rộng rãi trong nhiều ứng dụng y tế như phân đoạn tổ chức và cấu trúc trong hình ảnh chụp cắt lớp vi tính (CT) và cộng hưởng từ (MRI). Ngoài ra, nó còn được sử dụng trong phân tích các loại hình ảnh khác như nội soi hoặc ảnh vi mô. Sự linh hoạt và hiệu quả của U-Net cũng được chứng minh qua khả năng thích ứng với các bối cảnh khác ngoài y tế, như phân đoạn ảnh từ vệ tinh hoặc trong các ứng dụng nhận diện đối tượng.

1.3. Mục tiêu và phạm vi của báo cáo

Báo cáo này nhằm mục tiêu khảo sát và phân tích hiệu quả của kiến trúc mạng nơ-ron tích chập U-Net trong việc phân đoạn và chẩn đoán hình ảnh y tế, đặc biệt là các khối u nội sọ từ hình ảnh chụp cộng hưởng từ (MRI). Phạm vi nghiên cứu tập trung vào việc áp dụng U-Net, một mô hình được điều chỉnh và tối ưu hóa để xử lý các thách thức đặc thù trong phân đoạn hình ảnh y tế chính xác và nhanh chóng.

CHƯƠNG 2: LÝ THUYẾT Y HỌC VỀ KHỐI U NỘI SỌ

2.1. Sinh lý bệnh của khối u nội sọ

2.1.1. Quá trình phát triển của khối u nội sọ

Rối loạn chức năng thần kinh trong trường hợp u nội sọ có thể bắt nguồn từ các nguyên nhân sau:

- Sự xâm nhập và phá hủy các mô não bởi khối u.
- Áp lực trực tiếp từ khối u đè nén các mô lân cận.
- Tăng áp lực trong hộp sọ do khối u chiếm không gian.
- Chảy máu xảy ra trong hoặc xung quanh khối u.
- Sưng não.
- Tắc nghẽn tại các xoang tĩnh mạch của màng cứng, thường gặp ở các khối u di căn đến xương hoặc màng cứng bên ngoài.
- Tắc nghẽn đường dẫn lưu dịch não tủy (CSF), thường thấy ở khối u não thất 3 hoặc các khối u ở vùng hố sau.
- Cản trở quá trình hấp thụ CSF, có thể gặp trong bệnh bạch cầu hoặc ung thư biểu mô màng não.
- Tắc nghẽn động mạch.
- Hiếm gặp, có thể phát triển hội chứng cận ung thư.

Khối u ác tính có khả năng phát triển mạch máu mới bên trong, có thể gây chảy máu hoặc tắc nghẽn, dẫn đến hoại tử và biểu hiện lâm sàng tương tự như đột quỵ. Chảy máu là biến chứng phổ biến của các khối u di căn, đặc biệt là trong các trường hợp u melanin, ung thư biểu mô tế bào thận, ung thư đường mật, ung thư tuyến giáp, ung thư phổi hoặc ung thư vú.

Trái lại, các khối u lành tính phát triển chậm và có thể trở nên khá lớn trước khi xuất hiện triệu chứng, một phần do thiếu phù não. Các khối u nguyên phát ác tính tăng trưởng nhanh chóng nhưng hiếm khi lan rộng ra khỏi hệ thần kinh trung ương. Cái chết thường do sự phát triển của khối u tại chỗ và/hoặc chảy máu liên quan đến khối u, dù đó là khối u lành tính hay ác tính.

2.1.2. Các triệu chứng và dấu hiệu của u nội sọ

Triệu chứng và dấu hiệu của khối u não bao gồm các biểu hiện giống nhau từ khối u nguyên phát và khối u di căn, chủ yếu gây ra bởi áp lực tăng trong sọ.

Đau đầu là biểu hiện thường gặp nhất, đặc biệt nghiêm trọng khi bệnh nhân thức dậy từ giấc ngủ không REM, khi đó sự thông khí giảm và lưu lượng máu tới não tăng lên, làm tăng áp lực nội sọ. Đau đầu có thể trở nên trầm trọng hơn khi nằm nghiêng hoặc khi thực hiện nghiệm pháp Valsalva. Khi áp lực nội sọ cực cao, đau đầu có thể đi kèm với nôn mửa và cảm giác buồn nôn.

Suy giảm trạng thái tinh thần là triệu chứng phổ biến thứ hai, bao gồm cảm giác buồn ngủ, mệt mỏi, thay đổi tính cách, rối loạn hành vi và giảm nhận thức, đặc biệt là với các khối u não ác tính. Khả năng phản xạ đường thở có thể bị ảnh hưởng.

Rối loạn chức năng não bộ khu trú có thể gây ra các triệu chứng như rối loạn thần kinh khu trú, rối loạn nội tiết hoặc động kinh cục bộ, phụ thuộc vào vị trí của khối u. Các dấu hiệu thần kinh khu trú thường cho thấy vị trí của khối u nhưng đôi khi lại không khớp với vị trí dự đoán. Ví dụ, liệt cơ thẳng ngoài hoặc liệt nửa người

có thể xảy ra do áp lực trong sọ tăng lên.

Khối u có thể gây ra phản ứng viêm ở vùng màng não, dẫn đến viêm màng não bán cấp hoặc mạn tính.

2.2. Các phương pháp chẩn đoán Brain Tumor

2.2.1. Chụp MRI và các kỹ thuật hình ảnh khác

Chẩn đoán khối u nội sọ thường bao gồm việc sử dụng MRI trọng số T1 có tiêm chất đối quang gadolinium hoặc chụp CT có tiêm chất cản quang. Trong một số trường hợp, sinh thiết cũng có thể được thực hiện.

Khối u não thường bị bỏ qua ở giai đoạn sớm. Bất kỳ triệu chứng nào sau đây cũng nên được coi là dấu hiệu cảnh báo và xem xét khả năng tồn tại u não:

- Rối loạn chức năng não bộ cục bộ hoặc toàn thể tăng dần.
- Cơn co giật mới xuất hiện.
- Đau đầu kéo dài, không thể giải thích được hoặc mới xuất hiện gần đây, đặc biệt nghiêm trọng hơn trong khi ngủ.
- Dấu hiệu của áp lực nội sọ tăng lên như phù gai hoặc nôn mửa không giải thích được.
- Các vấn đề nội tiết do tuyến yên hoặc khu vực dưới đồi.

Các triệu chứng này cũng có thể xuất hiện do các khối u nội sọ khác như áp xe, phình động mạch, dị dạng động-tĩnh mạch, chảy máu nội sọ, tụ máu dưới màng cứng, u hạt, u nang ký sinh (như bệnh ấu trùng sán dây lợn ở não), hoặc đột quỵ do thiếu máu não cục bộ.

Một khám thần kinh toàn diện, chẩn đoán hình ảnh thần kinh và chụp X-quang ngực (để tìm nguồn di căn) là cần thiết. MRI trọng số T1 với gadolinium là phương pháp được ưu tiên. Chụp CT với chất cản quang là một lựa chọn thay thế. MRI thường phát hiện các khối u như u tế bào hình sao và u thần kinh đệm ít nhánh bậc thấp sớm hơn so với CT và cung cấp hình ảnh rõ ràng hơn về các cấu trúc não gần xương. Nếu chẩn đoán hình ảnh thông thường không cung cấp đủ chi tiết cho khu vực cần xem xét, các hình ảnh cận cảnh hoặc các góc nhìn đặc biệt của khu vực này nên được thực hiện. Nếu nghi ngờ áp lực nội sọ tăng cao mà hình ảnh thần kinh bình thường, việc kiểm tra áp lực nội sọ nguyên phát nên được cân nhắc và có thể yêu cầu làm thủ thuật dò thắt lưng.

Trong trường hợp nghi ngờ về loại khối u dựa trên đặc điểm điện quang trên MRI mà không có tính khẳng định, sinh thiết não có thể được yêu cầu, đôi khi là phẫu thuật sinh thiết.

Các xét nghiệm chuyên biệt như chỉ điểm khối u di truyền và phân tử trong máu và dịch não tủy (CSF) có thể hữu ích trong một số trường hợp. Đối với bệnh nhân AIDS, xét nghiệm virus Epstein-Barr trong CSF thường tăng đáng kể khi u lymphoma của hệ thần kinh trung ương phát triển.

2.2.2. Phân biệt các loại u nội sọ qua hình ảnh

Chẩn đoán các loại u nội sọ qua hình ảnh MRI đòi hỏi việc phân tích kỹ các đặc điểm của tổn thương như hình dạng, vị trí, tính chất của tín hiệu, và phản ứng của chúng với chất tương phản. Các chuỗi xung khác nhau như T1, T2, T2 FLAIR, và Diffusion cung cấp thông tin quan trọng về tổn thương. Ví dụ, u thần kinh đệm thường

có tín hiệu giảm trên T1 và tăng trên T2, trong khi T2 FLAIR hữu ích trong việc phát hiện các dịch tễ ở chất trắng.

Khi tiêm gadolinium, các khối u thường nổi bật hơn do tích tụ chất tương phản tại các khu vực có hàng rào máu não bị hỏng. Điều này là rất quan trọng trong việc phân biệt các loại u, với các khối u như lymphoma và di căn não thường bắt chất tương phản mạnh. Vị trí của khối u cũng gợi ý về nguồn gốc của chúng; chẳng hạn, u tế bào đệm thường phát triển ở vùng thùy, trong khi u nguyên bào tủy có thể xuất hiện trong não thất.

Tác động của tổn thương lên các cấu trúc lân cận cũng rất quan trọng trong chẩn đoán hình ảnh. Phù nề do tích tụ dịch có thể làm tăng áp lực trong sọ và đẩy các cấu trúc não xung quanh, trong khi giãn não thất có thể chỉ ra rằng khối u đang gây tắc nghẽn dòng chảy dịch não tủy. Đẩy các cấu trúc lân cận và tụt não cũng là những dấu hiệu quan trọng cho thấy sự thay đổi vị trí của các rãnh vỏ não và các cấu trúc trung tâm, thường xuất hiện do các khối u lớn.

Để chẩn đoán chính xác các loại u não bằng MRI đòi hỏi sự hiểu biết chuyên sâu về giải phẫu học não và các đặc điểm bệnh lý của từng loại u, cũng như kinh nghiệm trong việc đánh giá hình ảnh y tế.

CHƯƠNG 3: KIẾN TRÚC U-NET

3.1. Nguyên lý hoạt động của U-net

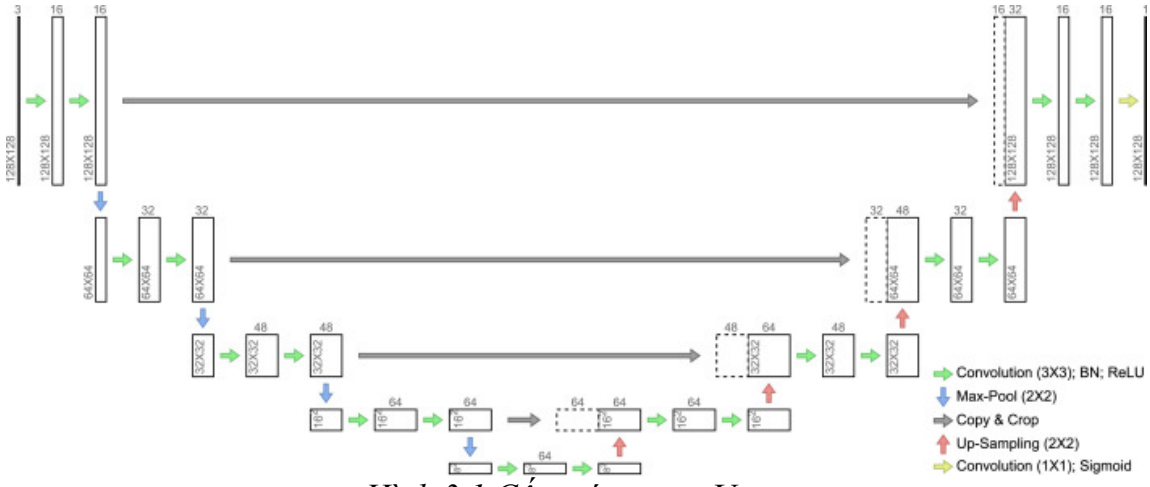
U-Net là một kiến trúc mạng nơ-ron tích chập (CNN) hoàn toàn kết nối, được thiết kế đặc biệt để phân đoạn hình ảnh một cách hiệu quả. Kiến trúc này thích hợp với nhiều nhiệm vụ phân tích hình ảnh trong các ứng dụng đa dạng, từ quan sát Trái Đất, video đến hình ảnh y tế.

U-Net sử dụng kiến trúc tự mã hóa tích chập, với hai phần chính: mã hóa và giải mã. Phần mã hóa bao gồm một chuỗi các lớp tích chập và lớp gộp, giúp nén hình ảnh đầu vào vào một biểu diễn không gian ẩn. Biểu diễn này là một phiên bản nén của hình ảnh, cho phép mô hình nắm bắt các điểm dữ liệu gần nhau. Phần giải mã sử dụng các phép tích chập đảo để tái tạo lại hình ảnh từ biểu diễn nén, cho phép định vị chính xác của các chi tiết trong hình ảnh.

U-Net không sử dụng các lớp kết nối đầy đủ mà chỉ dựa vào các lớp tích chập và lớp gộp tối đa. Ban đầu, U-Net được thiết kế cho hình ảnh 572 × 572 pixels nhưng có thể được điều chỉnh dễ dàng cho các kích thước hình ảnh khác nhau. Mạng này được tối ưu để phân đoạn, với các lớp tích chập chồng lên nhau giúp học các đặc điểm tốn hơn từ hình ảnh nén.

U-Net hoạt động dựa trên giả định rằng hình ảnh đầu vào và bản đồ nhị phân tương ứng có cùng kích thước. Mô hình này tái tạo hình ảnh đầu vào từ biểu diễn không gian ẩn, sao cho hình ảnh tái tạo phải khớp chính xác với kích thước ban đầu. Hiệu suất của U-Net phụ thuộc vào chất lượng của hình ảnh đầu vào và được đánh giá qua các chỉ số như lỗi biến dạng, lỗi rand và lỗi điểm ảnh. Ban đầu, U-Net đã đạt được kết quả vượt trội so với các mạng tích chập cửa sổ trượt, và các phiên bản sau đã cải thiện đáng kể hiệu suất phân đoạn hình ảnh.

3.1.1. Cấu trúc của mạng U-net



Hình 3.1 Cấu trúc mạng U-net

Mạng này bao gồm hai phần chính: mã hoá (encoder) và giải mã (decoder), được kết nối với nhau thông qua một cổng trung gian.

Phần encoder gồm nhiều khối, mỗi khối chứa hai lớp tích chập 3x3, mỗi lớp theo sau là chuẩn hóa theo batch và hàm kích hoạt ReLU. Các lớp gộp cực đại 2x2 được sử dụng sau mỗi khối để giảm dần kích thước không gian của các đặc trưng, điều này không chỉ giúp tăng trường nhận thức mà còn giảm bớt số lượng tham số

trong mạng.

Trong phần decoder, mỗi khối bắt đầu bằng một lớp lấy mẫu lên 2x2, theo sau là các lớp tích chập để tinh chỉnh đặc trưng. Đặc biệt, đặc trưng từ phần encoder được sao chép và nối với đầu vào của lớp lấy mẫu lên qua các phép sao chép và cắt xén, điều này giúp phục hồi thông tin không gian mất mát trong quá trình co lại.

Ở giữa hai đường dẫn, mạng sử dụng một lớp tích chập 3x3 và một lớp tích chập 1x1 với hàm kích hoạt Sigmoid để tạo ra đầu ra cuối cùng. Cấu trúc đặc biệt này của U-Net cho phép nó hoạt động hiệu quả ngay cả với lượng dữ liệu giới hạn, đồng thời bảo toàn các đặc trưng không gian quan trọng, làm cho nó trở nên lý tưởng cho việc phân đoạn ảnh y tế.

3.1.2. Sự đặc biệt của U-net

- U-Net có cấu trúc đối xứng U-hình với đường dẫn co lại ở một bên và đường dẫn mở rộng ở bên kia. Điều này giúp mô hình không chỉ thu nhận thông tin thông qua việc giảm chiều dữ liệu mà còn có khả năng phục hồi chi tiết ảnh qua quá trình mở rộng.
- U-Net có kết nối trực tiếp (Skip Connection). Mạng sử dụng các kết nối trực tiếp giữa các lớp của đường dẫn co lại và đường dẫn mở rộng. Điều này giúp truyền tải thông tin không gian một cách trực tiếp đến các lớp sau, làm tăng khả năng phục hồi các đặc điểm quan trọng của ảnh, nhất là trong các tác vụ phân đoạn chính xác các vùng biên dạng phức tạp.
- U-Net hiệu quả với lượng dữ liệu nhỏ. Nhờ cấu trúc đặc biệt và việc sử dụng các kết nối trực tiếp, U-Net có thể đạt được hiệu suất tốt ngay cả với lượng dữ liệu huấn luyện tương đối nhỏ. Điều này rất quan trọng trong lĩnh vực y tế, nơi mà dữ liệu huấn luyện có nhãn có thể khan hiếm và đắt đỏ.
- Chất lượng phân đoạn cao. U-Net rất phù hợp cho việc phân đoạn ảnh y tế vì nó có thể phân đoạn các cấu trúc y tế tinh vi và nhỏ mà không mất mát thông tin quan trọng, giúp cho các bác sĩ có thể nhận diện chính xác và hiệu quả hơn các bệnh lý từ hình ảnh chẩn đoán.
- Tính linh hoạt và thích ứng cao của U-Net. U-Net không chỉ giới hạn trong lĩnh vực y tế mà còn được áp dụng rộng rãi trong các tác vụ xử lý ảnh khác như phân đoạn đối tượng trong video, nhận dạng cảnh vật trong ảnh vệ tinh, và thậm chí trong các ứng dụng như dự báo thời tiết.

3.2. Lợi ích của U-net trong lĩnh vực y học

Mô hình U-Net có thể được sử dụng để phân đoạn các ảnh chụp CT, MRI, X-ray, và ảnh siêu âm, hỗ trợ chẩn đoán và lên kế hoạch phẫu thuật. Chẳng hạn, nó có thể giúp xác định chính xác khu vực cần phẫu thuật hoặc điều trị, làm tăng tỷ lệ thành công của các thủ thuật y tế. U-Net còn có thể tự động hóa việc phân đoạn và phân tích ảnh y tế, giúp giảm bớt gánh nặng công việc cho các chuyên gia y tế và cho phép họ tập trung vào việc chăm sóc bệnh nhân.

CHƯƠNG 4: Chuẩn bị dữ liệu và Tiền xử lý

4.1. Thu thập dữ liệu

4.1.1. Nguồn dữ liệu

Tập dữ liệu: BraTS2020
Tổng quan về tập dữ liệu: BraTS luôn tập trung vào việc đánh giá các phương pháp tiên tiến trong việc phân đoạn khối u não dựa trên ảnh chụp cộng hưởng từ đa mô thức (MRI). Trong khuôn khổ BraTS 2020, các ảnh MRI được thu thập trước khi tiến hành phẫu thuật từ nhiều cơ sở khác nhau đã được sử dụng. Chương trình này chủ yếu tập trung vào việc phân đoạn các khối u não glioma, những khối u có đặc điểm đa dạng về hình thái, bề ngoài và mô bệnh học. Đồng thời, BraTS 2020 cũng nhấn mạnh đến tầm quan trọng lâm sàng của việc phân đoạn này bằng cách đưa ra nhiệm vụ dự đoán tổng thời gian sống của bệnh nhân, phân biệt giữa tình trạng tiến triển giả của bệnh và tái phát của u thực sự. Nhiệm vụ này được thực hiện thông qua việc phân tích tích hợp các đặc điểm radiomic và sử dụng các thuật toán học máy. Cuối cùng, BraTS 2020 cũng quan tâm đến việc đánh giá sự không chắc chắn trong các thuật toán phân đoạn khối u.

4.1.2. Định dạng và quy mô

Tất cả các ảnh chụp đa mô thức của BraTS đều được lưu trữ dưới định dạng tệp NIfTI (.nii.gz). Các ảnh này mô tả bốn chế độ chụp khác nhau: a) chế độ chụp tự nhiên T1, b) chế độ chụp T1 sau khi tiêm thuốc cản quang (T1Gd), c) chế độ chụp T2, và d) ảnh chụp T2-FLAIR (thể tích phục hồi đảo chiều giảm độ nhạy chất lỏng). Các ảnh này được thu thập tại nhiều cơ sở (tổng cộng 19 cơ sở) theo các quy trình lâm sàng khác nhau và từ nhiều loại máy quét khác nhau, và những cơ sở này được ghi nhận như là những người đóng góp dữ liệu.

Các bộ dữ liệu hình ảnh đã được phân đoạn thủ công bởi từ một đến bốn chuyên gia đánh giá, tuân theo một quy trình chú thích nhất quán. Các chú thích này sau đó được các bác sĩ chẩn đoán hình ảnh thần kinh có kinh nghiệm phê duyệt. Các chú thích bao gồm khối u tăng cường GD (ET - nhãn 4), phù nề xung quanh khối u (ED - nhãn 2), và lõi khối u hoại tử và không tăng cường (NCR/NET - nhãn 1), như đã được mô tả trong các bài báo BraTS từ năm 2012-2013 và bài báo tổng kết BraTS mới nhất. Các dữ liệu được cung cấp đã qua tiền xử lý bao gồm đăng ký trên cùng một mẫu giải phẫu, được nội suy đến cùng một độ phân giải (1 mm^3) và đã được loại bỏ phần sọ.

CHƯƠNG 5: QUY TRÌNH HUẤN LUYỆN MODEL

5.1. Cài đặt môi trường và công cụ

Đối với dự án này, chúng ta sử dụng Python phiên bản 3.10.12, với một loạt các thư viện bổ trợ được quản lý thông qua một tệp requirements.txt. Tệp này bao gồm các phiên bản cụ thể của các thư viện như nibabel để xử lý ảnh y tế, matplotlib cho việc vẽ đồ thị, scikit-image và scikit-learn cho xử lý và học máy, opencv-python cho xử lý ảnh, tensorflow và keras cho học sâu, và pandas cho xử lý dữ liệu. Môi trường này còn được tăng cường bởi sức mạnh phần cứng từ card đồ họa NVIDIA Geforce RTX 3050 và GPU T4 từ Google Colab.

5.2. Chi tiết kỹ thuật huấn luyện

5.2.1. Cấu hình tham số

```
VOLUME_START_AT = 60
VOLUME_SLICES = 75
DATA_PATH =
str(current_directory)+"/datasets/BraTS2020_TrainingData/MICCAI_Bra
TS2020_TrainingData"
EXCLUDED_FILES = ["survival_info.csv", "name_mapping.csv"]
TRAIN_SIZE = 0.85
VAL_SIZE = 0.2
RANDOM_STATE = 42
IMG_SIZE = 128

def load_samples(data_path):
    try:
        samples = os.listdir(data_path)
    except FileNotFoundError:
        print(f"Error: Đường dẫn {data_path} không tồn tại.")
        return []
    except PermissionError:
        print(f"Error: Không có quyền truy cập {data_path}.")
        return []

    samples = [sample for sample in samples if sample not in
EXCLUDED_FILES]
    return samples

def split_data(samples):
    samples_train, samples_val = train_test_split(samples,
test_size=VAL_SIZE, random_state=RANDOM_STATE)
    samples_train, samples_test = train_test_split(samples_train,
test_size=(1 - TRAIN_SIZE), random_state=RANDOM_STATE)

    return samples_train, samples_val, samples_test

def print_distribution(samples_train, samples_val, samples_test):
```

```
    print(f"Độ dài tập Train: {len(samples_train)}
({len(samples_train)/len(samples)*100:.2f}%)")
    print(f"Độ dài tập Validation: {len(samples_val)}
({len(samples_val)/len(samples)*100:.2f}%)")
    print(f"Độ dài tập Test: {len(samples_test)}
({len(samples_test)/len(samples)*100:.2f}%)")


samples = load_samples(DATA_PATH)
if samples:
    samples_train, samples_val, samples_test = split_data(samples)
    print(f"Tổng các samples: {len(samples)}")
    print_distribution(samples_train, samples_val, samples_test)
```

Trong đoạn code được sử dụng trong bài này chúng em đã thiết lập một số tham số cấu hình quan trọng cho quá trình huấn luyện. Điểm bắt đầu là sử dụng các biến như `VOLUME_START_AT`, `VOLUME_SLICES`, `IMG_SIZE` để xác định kích thước và phạm vi dữ liệu đầu vào. Các hằng số như `TRAIN_SIZE`, `VAL_SIZE`, và `RANDOM_STATE` được dùng để chia dữ liệu thành các tập huấn luyện, kiểm thử và xác thực. Điều này đảm bảo tính ngẫu nhiên và đại diện cho dữ liệu trong mỗi tập.

```
# Triển khai mô hình U-Net cho BraTS 2019 của tác giả Naomi Fridman,
https://naomi-fridman.medium.com/multi-class-image-segmentation-
a5cc671e647a
def build_unet_naomi_fridman(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(inputs)
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv1)

    pool = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv)

    pool1 = MaxPooling2D(pool_size=(2, 2))(conv)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv2)

    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv3)

    pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)
```

```
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool4)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv5)
    drop5 = Dropout(dropout)(conv5)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(UpSampling2D(size = 2)(drop5))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv8))
    merge9 = concatenate([conv,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv9)

    up = Conv2D(32, 2, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(UpSampling2D(size = 2)(conv9))
    merge = concatenate([conv1,up], axis = 3)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv)

    conv10 = Conv2D(4, 1, activation = 'softmax')(conv)

    return Model(inputs = inputs, outputs = conv10)
```

Mô hình được cấu hình để sử dụng kiến trúc U-Net cùng với các hàm kích hoạt, tỷ lệ Dropout, và các hàm khởi tạo trọng số, nhằm tối ưu hóa quá trình học. Tham số dropout giúp ngăn chặn hiện tượng overfitting bằng cách loại bỏ ngẫu nhiên một số đơn vị trong quá trình huấn luyện, trong khi `ker_init` hỗ trợ việc khởi đầu hiệu quả cho quá trình lan truyền ngược.

5.2.2. Quy trình validation và test

```
def split_data(samples):
    samples_train, samples_val = train_test_split(samples,
test_size=VAL_SIZE, random_state=RANDOM_STATE)
    samples_train, samples_test = train_test_split(samples_train,
test_size=(1 - TRAIN_SIZE), random_state=RANDOM_STATE)

    return samples_train, samples_val, samples_test
```

Quy trình kiểm định và thử nghiệm được thực hiện thông qua việc sử dụng các tập dữ liệu riêng biệt, mà quá trình chia tập này được thực hiện thông qua hàm `split_data`. Hàm này chia bộ dữ liệu thành ba phần: huấn luyện, xác thực, và kiểm thử, với tỷ lệ phân chia được xác định bởi các tham số như `TRAIN_SIZE` và `VAL_SIZE`. Độ phân giải của các hình ảnh được chuẩn hóa ở kích thước 128x128, điều này quan trọng để đảm bảo tính nhất quán và độ chính xác của dữ liệu đầu vào cho mô hình.

```
def flatten_channels(y, channel_index):
    return K.flatten(y[:,:,:,channel_index])

def calculate_intersection(y_true_f, y_pred_f):
    return K.sum(y_true_f * y_pred_f)

def dice_coef(y_true, y_pred, smooth=1.0):
    class_num = 4
    total_loss = 0
    for i in range(class_num):
        y_true_f = flatten_channels(y_true, i)
        y_pred_f = flatten_channels(y_pred, i)
        intersection = calculate_intersection(y_true_f, y_pred_f)
        loss = ((2. * intersection + smooth) / (K.sum(y_true_f) +
K.sum(y_pred_f) + smooth))
        total_loss += loss
    return total_loss / class_num

def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    return true_positives / (predicted_positives + K.epsilon())

def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())

def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred),
0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())
```

Mỗi bước trong quá trình validation và test không chỉ đánh giá hiệu suất mô

hình trên dữ liệu mới mà còn đảm bảo rằng mô hình có khả năng tổng quát hóa tốt trên dữ liệu ngoài mẫu. Các metrics được tính toán trong quá trình này bao gồm độ chính xác, độ chính xác từng lớp (MeanIoU), và các chỉ số đánh giá mô hình như **dice_coef**, **precision**, **sensitivity**, và **specificity**, cung cấp một cái nhìn toàn diện về hiệu suất mô hình trên các lớp khác nhau.

```
callbacks = [
    keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
factor=0.2,
                                  patience=2, min_lr=0.000001,
verbose=1),

    keras.callbacks.ModelCheckpoint(filepath =
'/content/drive/MyDrive/Colab
Notebooks/BraTS2020/model_.{epoch:02d}-{val_loss:.6f}.m5',
                                 verbose=1, save_best_only=True,
save_weights_only = True),

    CSVLogger('/content/drive/MyDrive/Colab
Notebooks/BraTS2020/training.log', separator=',', append=False)
]
```

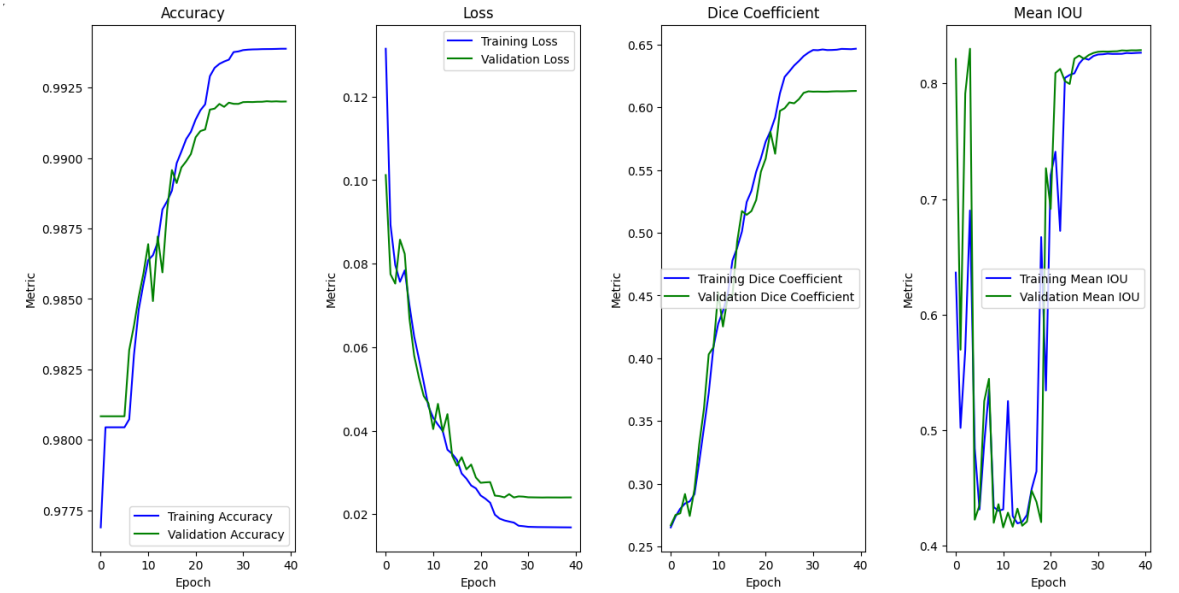
Hệ thống callbacks sử dụng trong Keras giúp tối ưu hóa quá trình huấn luyện bằng cách tự động điều chỉnh tốc độ học tập và lưu mô hình tại thời điểm có kết quả tốt nhất, nhằm đạt được hiệu quả cao nhất trong quá trình huấn luyện.

CHƯƠNG 6: KẾT QUẢ VÀ ĐÁNH GIÁ

6.1. Hiệu suất của model

6.1.1. Đánh giá model

Chúng ta sẽ tiến hành vẽ đồ thị và thông tin của model sau khi train



Độ chính xác (Accuracy):

- Độ chính xác của mô hình tăng đều và ổn định qua các kỳ, với độ chính xác trên tập huấn luyện (màu xanh) và tập xác thực (màu xanh lá) gần như trùng nhau vào cuối quá trình. Điều này cho thấy mô hình có khả năng tổng quát hóa tốt và không bị overfitting nặng.

Độ mất mát (Loss):

- Độ mất mát trên cả hai tập (huấn luyện và xác thực) giảm nhanh trong những kỳ đầu, sau đó tiếp tục giảm dần ở mức độ chậm hơn và ổn định. Điều này cho thấy quá trình học đang diễn ra hiệu quả.

Hệ số dice (Dice Coefficient):

- Hệ số Dice tăng đều qua từng kỳ, với sự chênh lệch nhỏ giữa tập huấn luyện và xác thực, cho thấy mô hình cũng không bị overfitting đáng kể về mặt đo lường này.

IOU trung bình (Mean IOU):

- Biểu đồ IOU trung bình cho thấy có sự biến động mạnh giữa các kỳ đánh giá, đặc biệt là trên tập xác thực. Sự biến động này có thể chỉ ra rằng mô hình vẫn còn gặp khó khăn trong việc học các đặc điểm nhất định từ dữ liệu, hoặc có thể đề xuất rằng phương pháp đánh giá IOU có sự nhạy cảm cao với những thay đổi nhỏ trong việc phân đoạn.

Sau đó chúng ta tiến hành đánh giá model trên tập test

```
45/45 [==============================] - 12s 255ms/step - loss: 0.0197 - accuracy: 0.9932 - mean_io_u: 0.8301 - dice_coef: 0.6376 - precision: 0.9936 - sensitivity: 0.9915 - specificity: 0.9978

Kết quả đánh giá model trên tập test:
====================================
Loss: 0.0197
Accuracy: 0.9932
MeanIOU: 0.8301
Dice Coefficient: 0.6376
Precision: 0.9936
Sensitivity: 0.9915
Specificity: 0.9978
```

1. **Loss (Độ mất mát): 0.0197**

Độ mất mát của mô hình rất thấp, cho thấy mô hình rất hiệu quả trong việc dự đoán các lớp đúng của dữ liệu, với sự sai lệch giữa dự đoán và thực tế rất nhỏ.

2. **Accuracy (Độ chính xác): 0.9932**

Độ chính xác cực kỳ cao, gần như hoàn hảo, chỉ ra rằng mô hình có khả năng phân loại đúng các pixel vào lớp đúng của chúng.

3. **MeanIOU (IOU trung bình): 0.8301**

IOU Trung bình đạt 83%, là một chỉ số tốt, đặc biệt là trong các tác vụ phân đoạn hình ảnh y khoa. Chỉ số này cho thấy mức độ chồng chéo giữa các vùng phân đoạn của mô hình và vùng thật trên ảnh.

4. **Dice Coefficient (Hệ số dice): 0.6376**

Hệ số Dice thấp hơn so với các chỉ số khác, cho thấy có một số khác biệt giữa các vùng phân đoạn của mô hình và vùng thật. Sự chênh lệch này có thể do một số đặc điểm của ảnh y khoa khó phân đoạn chính xác.

5. **Precision (Độ chính xác của dự đoán tích cực): 0.9936**

Độ chính xác rất cao, cho thấy phần lớn các dự đoán tích cực của mô hình là chính xác, tức là ít false positives.

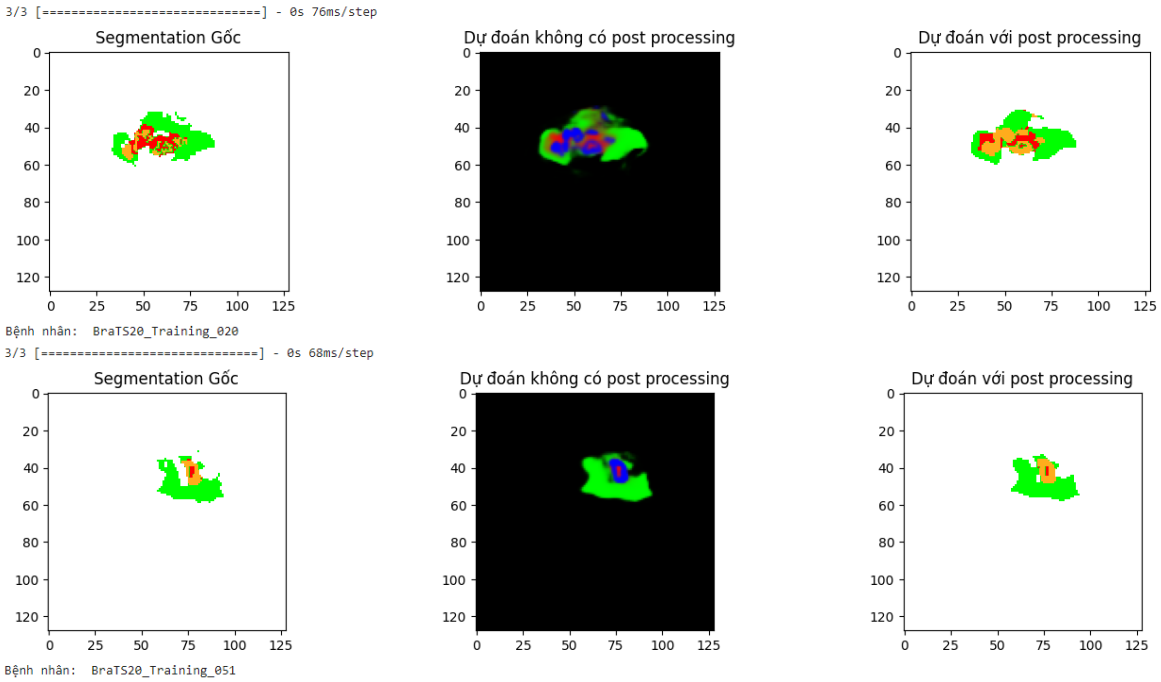
6. **Sensitivity (Độ nhạy): 0.9915**

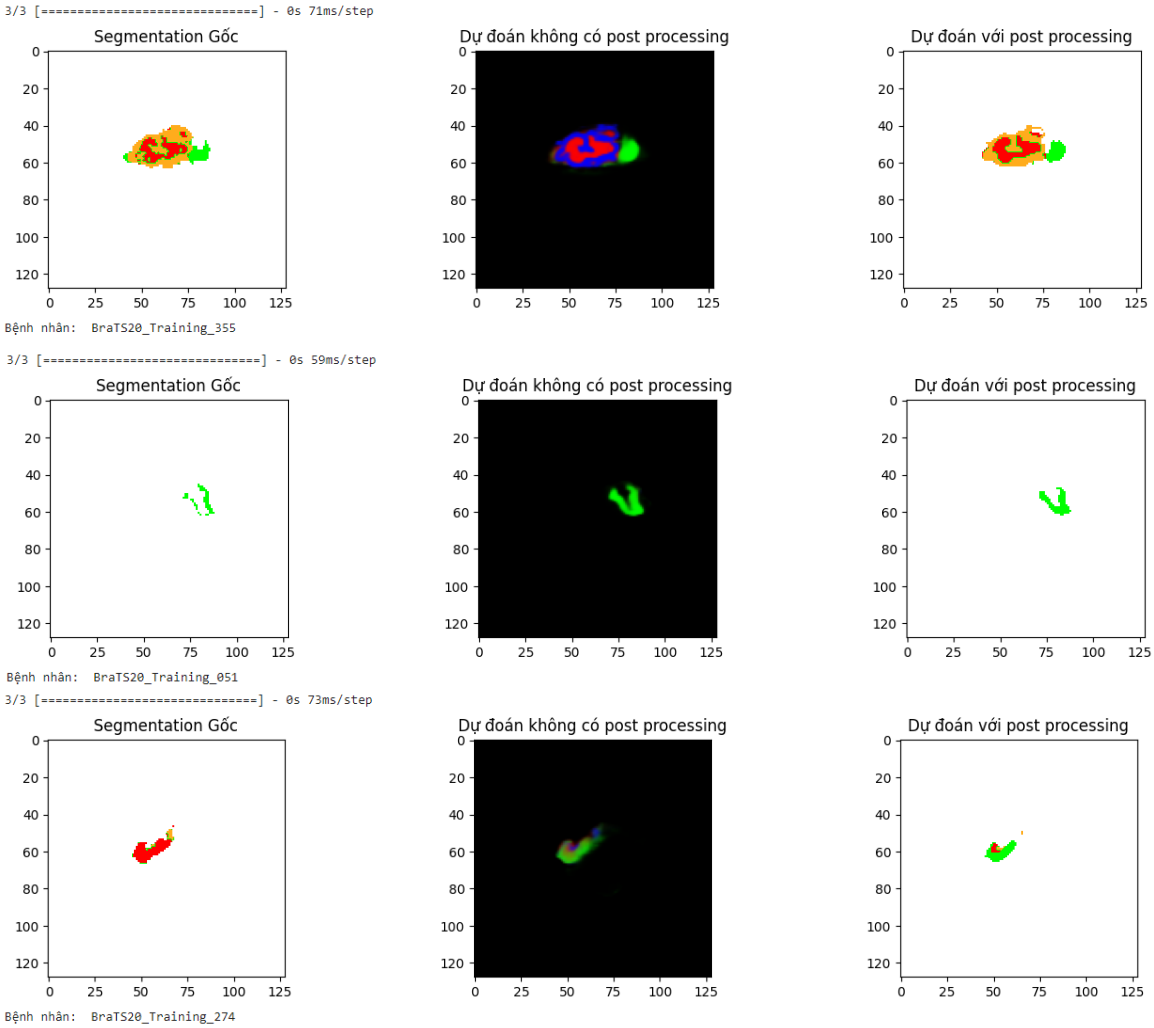
Độ nhạy cũng rất cao, nghĩa là mô hình rất tốt trong việc phát hiện các trường hợp thật (true positives), tức là ít false negatives.

7. **Specificity (Độ đặc hiệu): 0.9978**

Độ đặc hiệu cực kỳ cao, chỉ ra rằng mô hình hiệu quả trong việc xác định các trường hợp không phải là dương tính, tức là rất ít false positives.

6.1.2. Một số hình ảnh dự đoán thực tế của model





6.2. Thách thức và hạn chế

6.2.1. Vấn đề dữ liệu

Sự khác biệt và biến đổi trong dữ liệu:

Tập dữ liệu BRaTS2020, mặc dù đã được chuẩn hóa và tập hợp từ nhiều trung tâm, vẫn tồn tại sự khác biệt đáng kể về mặt chất lượng hình ảnh và cách ghi nhận dữ liệu giữa các trung tâm. Điều này có thể gây khó khăn trong việc huấn luyện một mô hình học sâu có khả năng tổng quát hóa cao trên các dữ liệu thực tế.

Sự mất cân bằng trong lớp dữ liệu:

Một thách thức phổ biến trong các tập dữ liệu y khoa là sự mất cân bằng lớp, nơi số lượng pixel biểu thị u nội sọ ít hơn nhiều so với số lượng pixel không phải u. Điều này dẫn đến sự thiên vị trong quá trình huấn luyện, làm cho mô hình khó phát hiện các trường hợp ít gặp hơn nhưng lại rất quan trọng.

Nhiễu và độ chính xác của nhãn:

Việc phân đoạn ảnh y khoa thường phụ thuộc vào chất lượng và độ chính xác của nhãn được gán cho dữ liệu huấn luyện. Những sai sót hoặc thiếu sót trong việc gán nhãn có thể dẫn đến việc huấn luyện mô hình sai lệch.

6.2.2. Giới hạn của kiến trúc U-net

Chiều sâu và độ phức tạp của mô hình:

Kiến trúc U-net có cấu trúc tổng hợp thông tin từ các lớp trước đó thông qua các đường nối qua, nhưng vẫn có giới hạn trong việc xử lý các vấn đề phức tạp nhất định do độ sâu và số lượng tham số của mô hình. Điều này có thể hạn chế khả năng của mô hình trong việc nắm bắt các đặc điểm phức tạp của dữ liệu.

Quá khớp (overfitting):

Mặc dù U-net đã cho thấy khả năng phân đoạn ảnh y khoa tốt, nhưng nó có xu hướng overfitting, đặc biệt khi số lượng dữ liệu huấn luyện hạn chế hoặc khi mô hình được huấn luyện quá lâu trên tập dữ liệu có tính chất đặc thù.

CHƯƠNG 7: HƯỚNG PHÁT TRIỂN TƯƠNG LAI

7.1. Cải tiến kiến trúc U-net

Việc thiết kế các lớp sâu hơn hoặc thêm các đường nối qua tinh vi hơn có thể giúp mô hình nắm bắt được các mẫu phức tạp hơn và cải thiện độ chính xác. Việc áp dụng kỹ thuật "Residual Learning" hoặc "Dense Connections" có thể giúp làm giảm vấn đề mất mát thông tin qua từng lớp.

Tích hợp cơ chế chú ý vào kiến trúc U-net giúp mô hình tập trung vào những vùng quan trọng của ảnh, từ đó nâng cao khả năng phân đoạn chính xác hơn. Điều này rất hữu ích trong việc phân đoạn các vùng biên động hoặc nhỏ.

Áp dụng các kỹ thuật huấn luyện tiên tiến như huấn luyện đa tác vụ, học chuyển giao, và tăng cường dữ liệu động có thể giúp cải thiện độ chính xác và khả năng tổng quát hóa của mô hình.

7.2. Áp dụng trong các lĩnh vực khác của y học

Kiến trúc U-net có thể được tùy biến để phục vụ trong việc phân đoạn các cấu trúc khác của cơ thể như phổi, gan, và tim. Mỗi cấu trúc này có những thách thức riêng về hình ảnh và mô hình, và việc tùy chỉnh U-net cho từng trường hợp cụ thể sẽ là một lĩnh vực nghiên cứu quan trọng.

Ngoài nhận diện khối u, U-net có thể được ứng dụng để theo dõi sự thay đổi của bệnh lý theo thời gian, giúp các bác sĩ đánh giá hiệu quả điều trị và thực hiện các quyết định lâm sàng chính xác hơn.

U-net có thể được tích hợp với các hệ thống AI khác như hệ thống phân tích dữ liệu lớn, học máy cho dữ liệu chuỗi thời gian, và các mô hình dự đoán để tạo thành một hệ thống y tế thông minh toàn diện.

Tài liệu tham khảo

1. Gaspar L, Prabhu R, Hdeib A, et al: Congress of Neurological Surgeons systematic review and evidence-based guidelines on the role of whole brain radiation therapy in adults with newly diagnosed metastatic brain tumors. Neurosurgery 84 (3):E159–E162, 2019. doi: 10.1093/neuros/nyy541
2. Vogelbaum MA, Brown PD, Messersmith H, et al: Treatment for brain metastases: ASCO-SNO-Astro guideline. J Clin Oncol 40 (5):492–516, 2022 doi: 10.1200/JCO.21.02314 Xuất bản điện tử ngày 21 tháng 12 năm 2021.
3. Long GV, Atkinson V, Lo S, et al: Combination nivolumab and ipilimumab or nivolumab alone in melanoma brain metastases: A multicentre randomised phase 2 study. Lancet Oncol 19 (5):672–681, 2018 doi: 10.1016/S1470-2045(18)30139-6 Xuất bản điện tử ngày 27 tháng 3 năm 2018
4. Moss NS, Tosi U, Santomasso BD, et al: Multifocal and pathologically-confirmed brain metastasis complete response to trastuzumab deruxtecan. CNS Oncol 11 (3):CNS90, 2022 doi: 10.2217/cns-2022-0010 Xuất bản điện tử ngày 8 tháng 6 năm 2022.
5. Ovh/AI-trAIning-examples. (n.d.). GitHub. https://github.com/ovh/ai-training-examples
6. Walsh, J., Othmani, A., Jain, M., & Dev, S. (2022). Using U-net network for efficient brain tumor segmentation in MRI images. Healthcare Analytics, 2, 100098. https://doi.org/10.1016/j.health.2022.100098
7. Hướng dẫn cách đọc kết quả MRI sọ não. (n.d.). Bệnh viện Đa khoa Quốc tế Vinmec | Vinmec. https://www.vinmec.com/vi/tin-tuc/thong-tin-suc-khoe/suc-khoe-tong-quat/huong-dan-cach-doc-ket-qua-mri-so-nao/
8. Tổng quan về các khối u trong sọ. (2023, May 5). Cẩm nang MSD - Phiên bản dành cho chuyên gia.
9. Devansh Sharma, Image Classification Using CNN | Step-wise Tutorial. 12 Jan, 2024. From: https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/
10. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.
11. Activation functions in Neural Networks. 17 Feb, 2023. From: https://www.geeksforgeeks.org/activation-functions-neural-networks/
12. Chitra G Desai. Comparative Analysis of Optimizers in Deep Neural Networks. Oct 2020. From: https://www.researchgate.net/publication/345381779_Comparative_Analysis_of_Optimizers_in_Deep_Neural_Networks
13. Vaibhav Balloli. Adaptive Gradient Clipping. 31 March, 2021. From: https://tourdeml.github.io/blog/posts/2021-03-31-adaptive-gradient-clipping/

14. Katherine (Yi) Li. How to Choose a Learning Rate Scheduler for Neural Networks. 9 August, 2023. From: https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler
15. Đặng Thị Hằng | Phạm Duy Tùng. Tìm Hiểu Về Dropout Trong Deep Learning, Machine Learning. 5 May 2019. From: https://www.phamduytung.com/blog/2019-05-05-deep-learning-dropout/
16. Afshine Amidi và Shervine Amidi (Dịch bởi Phạm Hồng Vinh và Đàm Minh Tiến). 17 May 2020. Mạng neural tích chập cheatsheet. From: https://github.com/afshinea/stanford-cs-230-deep-learning/blob/master/vi/cheatsheet-convolutional-neural-networks.pdf
17. savyakhosla. 21 Apr, 2023. CNN | Introduction to Pooling Layer. From: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/
18. Maël Fabien. XCeption Model and Depthwise Separable Convolutions. From: https://maelfabien.github.io/deeplearning/xception/#
19. Zhe Ming Chng. 10 Aug, 2020. Using Depthwise Separable Convolutions in Tensorflow. From: https://machinelearningmastery.com/using-depthwise-separable-convolutions-in-tensorflow/
20. Yuliia Kniazieva. 7 Sep, 2023. Between Image Classification & Object Detection?. From: https://labelyourdata.com/articles/object-detection-vs-image-classification#:~:text=While%20image%20classification%20focuses%20on%20assigning%20a%20single,the%20positions%20of%20numerous%20objects%20within%20an%20image.
21. Lyudmil Vladimirov. 2020. Training Custom Object Detector. From: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html