TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY
BỘ MÔN CƠ ĐIỆN TỬ

ଓଓଓଓଓଓଓଓ 0o0 ଌଌଌଌଌଌଌଌ

BÁO CÁO CUỐI KỲ
TRÍ TUỆ NHÂN TẠO– 232ARIN337629
ĐỀ TÀI: MÔ HÌNH PHÂN ĐOẠN CÁC TIỂU VÙNG BÊN TRÊN KHỐI U NÃO

Nhóm: ARIN337629_23_2_03CLC

GVHD: TS Trần Vũ Hoàng

HỌ & TÊN SINH VIÊN	**MSSV**
Phạm Trung Hiếu	21146459
Lê Đình Tấn An	21143104
Đoàn Phương Tú	21143104

Tp. HCM, Ngày 4, tháng 5, năm 2024

Mục lục

I. Tổng quan

1.1. Lý do chọn đề tài

Đào tạo và hỗ trợ từ xa là một yếu tố quan trọng trong việc nâng cao chất lượng chăm sóc y tế ở các khu vực hẻo lánh. Các chuyên gia y tế có thể sử dụng các phân đoạn của khối u để đào tạo và hỗ trợ các bác sĩ địa phương. Cụ thể, họ có thể sử dụng hình ảnh phân đoạn để minh họa các trường hợp cụ thể, giải thích các nguyên tắc chẩn đoán và điều trị, cũng như cung cấp sự hỗ trợ trong việc đánh giá các hình ảnh MRI. Điều này không chỉ giúp nâng cao khả năng chẩn đoán và điều trị của các bác sĩ địa phương mà còn giảm bớt sự phụ thuộc vào các chuyên gia từ xa.

Trong một trường hợp cụ thể, một bệnh viện địa phương ở một khu vực hẻo lánh không có sẵn chuyên gia về nội soi và phẫu thuật não. Một bác sĩ địa phương đang gặp khó khăn trong việc đánh giá một hình ảnh MRI của một bệnh nhân với khối u não. Tuy nhiên, bác sĩ này có thể gửi hình ảnh MRI đó đến một chuyên gia ở một bệnh viện lớn thông qua hệ thống gửi hình ảnh y tế.

Chuyên gia này có thể sử dụng mô hình phân đoạn để chỉ ra các vùng khối u và giải thích chi tiết về kích thước, vị trí và tính chất của chúng. Họ có thể tạo ra các hình ảnh phân đoạn minh họa và trực quan để giải thích cho bác sĩ địa phương về sự phát triển của khối u và ảnh hưởng của nó lên cơ thể.

Hơn nữa, chuyên gia này có thể cung cấp hướng dẫn cụ thể về cách tiếp cận chuẩn đoán và điều trị khối u não. Họ có thể chia sẻ kinh nghiệm cá nhân và khuyến nghị các chiến lược điều trị dựa trên thông tin từ hình ảnh phân đoạn. Bằng cách này, bác sĩ địa phương sẽ được hỗ trợ để đưa ra quyết định chính xác hơn về điều trị của bệnh nhân, giảm thiểu nguy cơ và tối ưu hóa kết quả.

Trong trường hợp này, mục tiêu chính không phải là tạo ra phân đoạn với độ chính xác tuyệt đối mà là sử dụng phân đoạn để minh họa và giải thích. Việc tạo ra các hình ảnh phân đoạn có thể giúp chuyên gia từ xa truyền đạt thông tin một cách rõ ràng và trực quan đến các bác sĩ địa phương, giúp họ hiểu rõ hơn về tình trạng sức khỏe của bệnh nhân và cách tiếp cận điều trị.

Một phân đoạn không cần phải hoàn hảo mà chỉ cần đủ đáng tin cậy để minh họa vị trí và tính chất của khối u. Quan trọng nhất là thông tin được truyền đạt một cách hiệu quả và dễ hiểu, giúp bác sĩ địa phương có thể áp dụng những kiến thức này vào quá trình điều trị của bệnh nhân.

Việc chọn đề tài này không chỉ nhằm giải quyết vấn đề hiện tại trong việc đào tạo và hỗ trợ y tế từ xa, mà còn mở ra cơ hội cho việc áp dụng các công nghệ mới vào chăm sóc sức khỏe, đặc biệt là ở các khu vực khó khăn về y tế.

1.2. Mục tiêu

Mục tiêu xây dựng mô hình có khả năng phân đoạn tiểu vùng trên khối u não từ ảnh chụp MRI.

1.3. Giới hạn đề tài

- Đề tài này tập trung vào việc phát triển một hệ thống phân đoạn ảnh y tế để hỗ trợ chẩn đoán và điều trị từ xa, với mục tiêu chính là cải thiện chất lượng chăm sóc y tế ở các khu vực hẻo lánh. Tuy nhiên, do hạn chế về thời gian và nguồn lực, phạm vi nghiên cứu và phát triển của đề tài sẽ được giới hạn trong các khía cạnh sau:

    + Phát triển mô hình 3D U-Net: Đề tài sẽ tập trung vào việc xây dựng và huấn luyện một mô hình 3D U-Net để phân đoạn các khối u não trên hình ảnh MRI.

Đây là bước quan trọng để đảm bảo rằng mô hình có thể nhận diện và phân đoạn chính xác các vùng khối u.

+ Chuẩn bị dữ liệu: Bao gồm việc thu thập, tiền xử lý và chuẩn hóa dữ liệu MRI để đảm bảo tính nhất quán và độ tin cậy của mô hình phân đoạn. Các bước này sẽ bao gồm tải dữ liệu, đổi tên, loại bỏ các cột không cần thiết, chuẩn hóa, và lưu trữ dữ liệu dưới dạng các mảng Numpy.

+ Định nghĩa bộ sinh dữ liệu tùy chỉnh: Phát triển một bộ sinh dữ liệu tùy chỉnh để xử lý và tải các tập tin npy từ đĩa trong quá trình huấn luyện mô hình, do công cụ Keras ImageDataGenerator không hỗ trợ định dạng tệp này.

+ Huấn luyện và đánh giá mô hình: Mô hình sẽ được huấn luyện và đánh giá trên các tập dữ liệu đã chuẩn bị. Quá trình này bao gồm việc định nghĩa hàm mất mát, các chỉ số đo lường và biên dịch mô hình.

+ Dự đoán và trực quan hóa kết quả: Sau khi mô hình đã được huấn luyện, nó sẽ được sử dụng để dự đoán trên một số ảnh kiểm tra và trực quan hóa kết quả dự đoán để đánh giá hiệu quả của mô hình.

-Giới hạn của đề tài:

+Không triển khai trên quy mô lớn: Do giới hạn về thời gian và tài nguyên, mô hình sẽ không được triển khai và thử nghiệm trên quy mô lớn hoặc trong môi trường lâm sàng thực tế.

+Không bao gồm tích hợp hệ thống: Đề tài sẽ không bao gồm việc phát triển giao diện người dùng hoặc tích hợp mô hình vào hệ thống y tế hiện có. Mục tiêu chính là xây dựng và đánh giá mô hình phân đoạn.

+Không thực hiện nghiên cứu lâm sàng: Mô hình sẽ không được thử nghiệm trên các dữ liệu lâm sàng thực tế hoặc trên các bệnh nhân thực sự. Việc này yêu cầu thời gian, quy trình đạo đức và nguồn lực mà đề tài hiện tại không đáp ứng được.

+Hạn chế về tối ưu hóa: Mặc dù mô hình sẽ được tối ưu hóa để đạt hiệu quả cao nhất trong phạm vi nghiên cứu, việc tối ưu hóa để chạy trên các thiết bị phần cứng cụ thể như máy tính cá nhân với GPU sẽ không được thực hiện chi tiết.

II. Cơ sở lý thuyết

2.1. Thách thức của đề tài

Một trong những thách thức lớn nhất trong đề tài phân đoạn ảnh y học, đặc biệt là trong BraTS 2020 Challenge, là mất cân bằng dữ liệu. Vấn đề này có thể được hiểu rõ hơn qua các khía cạnh sau:

+ Các Lớp Phân Đoạn: Dữ liệu BraTS có các lớp phân đoạn bao gồm khối u tăng cường (enhancing tumor), khối u không tăng cường (non-enhancing tumor), và phù não (edema). Các vùng khối u có thể chiếm một phần nhỏ trong tổng thể hình ảnh, trong khi phần lớn hình ảnh là vùng não bình thường. Điều này dẫn đến sự mất cân bằng trong phân bố các lớp, làm cho việc huấn luyện mô hình trở nên khó khăn hơn khi mô hình có xu hướng học và dự đoán tốt hơn cho các lớp phổ biến (não bình thường) và bỏ qua các lớp ít xuất hiện (các loại khối u).

+Tần Suất Xuất Hiện: Các vùng khác nhau của khối u có thể xuất hiện với tần suất khác nhau trong tập dữ liệu, dẫn đến một số lớp có ít mẫu hơn đáng kể so với các lớp khác. Ví dụ, khối u tăng cường có thể xuất hiện ít hơn so với phù não hoặc mô não bình thường. Sự mất cân bằng này làm cho mô hình khó học được các đặc trưng của các lớp ít xuất hiện, gây ra sai lệch trong quá trình dự đoán và giảm hiệu quả của mô hình.

2.2. Phương pháp giải quyết

Để giải quyết vấn đề mất cân bằng dữ liệu và cải thiện hiệu suất của mô hình phân đoạn trong BraTS 2020 Challenge, các phương pháp sau đây có thể được áp dụng:

+ Tiền xử lý để loại bỏ các vùng không quan trọng:

Trước khi đưa dữ liệu vào mô hình, cần thực hiện các bước tiền xử lý để loại bỏ các vùng không quan trọng, giúp tập trung vào các vùng chứa thông tin cần thiết. Điều này bao gồm:

- Cắt và Chia nhỏ ảnh: Cắt bỏ các phần không chứa thông tin của ảnh MRI để giảm thiểu khối lượng tính toán và tăng hiệu suất.

-Chuẩn hóa dữ liệu: Chuẩn hóa cường độ của các ảnh MRI để đồng nhất các giá trị pixel, giúp mô hình học tốt hơn.

+Tính toán phù hợp trọng số cho từng lớp

Để giải quyết vấn đề mất cân bằng lớp, có thể điều chỉnh trọng số trong hàm mất mát (loss function) để mô hình tập trung hơn vào các lớp ít xuất hiện:

-Class Weighing: Điều chỉnh trọng số cho mỗi lớp trong hàm mất mát để tăng độ quan trọng của các lớp thiểu số, giúp mô hình học tốt hơn các đặc trưng của các lớp này.

-Focal Loss: Sử dụng hàm mất mát Focal Loss để giảm trọng số của các mẫu dễ và tăng trọng số của các mẫu khó, giúp mô hình tập trung vào các lớp thiểu số và các mẫu khó phân loại.

+Sử dụng các kỹ thuật tăng cường dữ liệu:

Tăng cường dữ liệu giúp tạo ra nhiều mẫu huấn luyện hơn từ các mẫu hiện có, cải thiện độ đa dạng của dữ liệu và giúp mô hình học tốt hơn:

- Tăng cường hình ảnh: Sử dụng các kỹ thuật tăng cường dữ liệu như xoay, lật, phóng to, thu nhỏ, thay đổi độ sáng và nhiễu để tạo ra các biến thể mới của ảnh huấn luyện.

- Oversampling: Tăng số lượng mẫu của các lớp thiểu số bằng cách nhân bản hoặc tạo thêm dữ liệu từ các mẫu hiện có để cân bằng tập dữ liệu.

+Sử dụng kiến trúc mạng chuyên biệt

Thiết kế và sử dụng các kiến trúc mạng chuyên biệt để cải thiện hiệu suất phân đoạn:

U-Net là một kiến trúc mạng nơ-ron sâu được thiết kế đặc biệt cho các tác vụ phân đoạn ảnh y học. Được giới thiệu bởi Olaf Ronneberger và các cộng sự vào năm 2015, U-Net đã trở thành một trong những mạng phổ biến nhất trong lĩnh vực này nhờ hiệu suất cao và khả năng học từ dữ liệu với số lượng tương đối ít.

-Kiến trúc của U-Net hình(0)

U-Net có một kiến trúc đối xứng hình chữ U, bao gồm hai phần chính: Contraction Path (Encoder) và Expansion Path (Decoder).

-Contraction Path (Encoder)

Phần này có nhiệm vụ trích xuất các đặc trưng từ ảnh đầu vào bằng cách sử dụng các lớp tích chập (convolution) và các lớp pooling để giảm dần kích thước không gian của ảnh. Cụ thể:

-Các lớp tích chập (Convolutional layers): Mỗi khối trong phần encoder bao gồm hai lớp tích chập 3x3 với hàm kích hoạt ReLU và padding 'same' để giữ nguyên kích thước đầu vào.

-Max Pooling: Sau các lớp tích chập, một lớp max pooling 2x2 với stride 2x2 được sử dụng để giảm kích thước không gian, đồng thời tăng số lượng bộ lọc.

-Expansion Path (Decoder)

Phần này có nhiệm vụ khôi phục lại kích thước không gian của ảnh đồng thời kết hợp thông tin từ các đặc trưng đã trích xuất. Cụ thể:

-Up-Convolution: Các lớp deconvolution hoặc transpose convolution 2x2 với stride 2x2 được sử dụng để tăng kích thước không gian của ảnh.

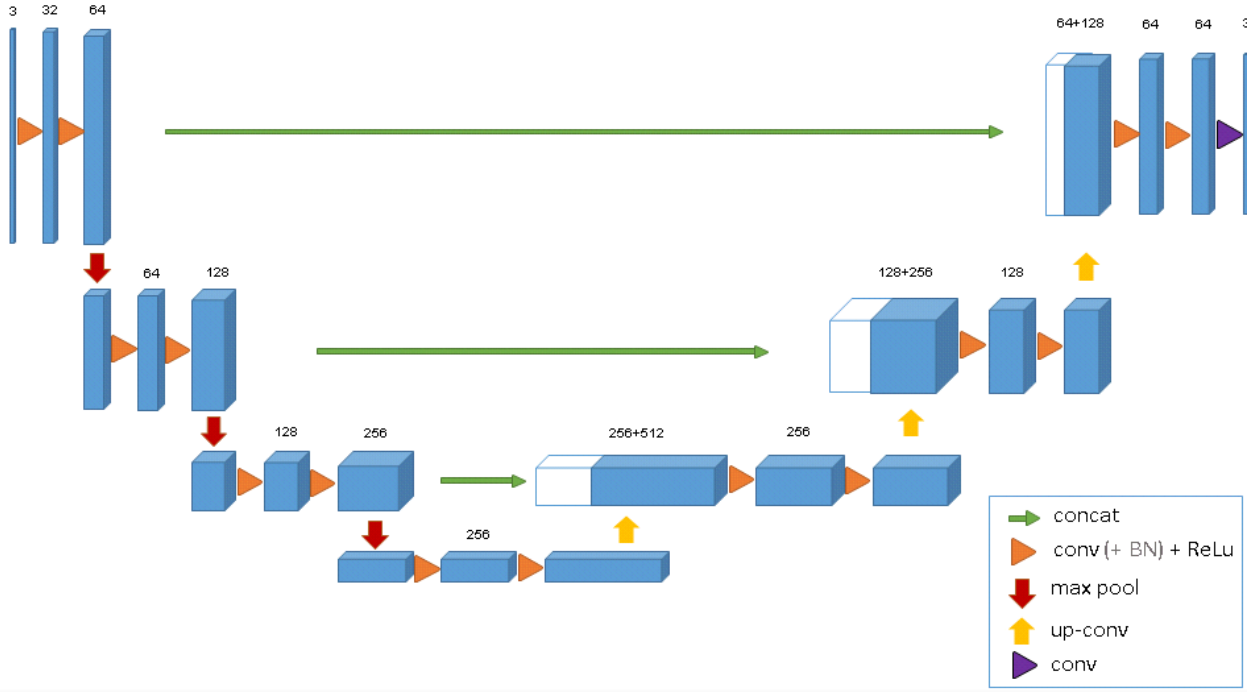
-Skip Connections: Các kết nối tắt (skip connections) giữa các lớp tương ứng ở phần encoder và decoder giúp giữ lại các thông tin chi tiết bị mất trong quá trình pooling. Các đặc trưng từ phần encoder được concatenate với các đặc trưng tương ứng ở phần decoder.

-Các lớp tích chập: Sau mỗi lớp up-convolution, hai lớp tích chập 3x3 với hàm kích hoạt ReLU được áp dụng.

-Output Layer:

Lớp cuối cùng của U-Net là một lớp tích chập 1x1 với số lượng bộ lọc bằng số lượng lớp cần phân đoạn, sử dụng hàm kích hoạt softmax (cho phân đoạn đa lớp) hoặc sigmoid (cho phân đoạn nhị phân).

Kết luận: U-Net là một công cụ mạnh mẽ và linh hoạt cho các tác vụ phân đoạn ảnh, đặc biệt là trong lĩnh vực y học. Với kiến trúc đối xứng và các kết nối tắt, U-Net có khả năng học và khôi phục lại các đặc trưng chi tiết từ ảnh đầu vào, mang lại hiệu suất cao trong việc phân đoạn ảnh phức tạp.



Hình 0 Kiến trúc U-net

III. Phương pháp đề xuất

3.1. Yêu cầu thiết kế

Hệ thống phân đoạn ảnh cần phải có các tiêu chí sau:

- Độ chính xác vừa phải nhưng đáng tin cậy.
  + Minh họa và giải thích: Mô hình không cần phải đạt được độ chính xác tối đa nhưng cần đủ tốt để minh họa các đặc điểm quan trọng của khối u.
  + Đáng tin cậy: Đảm bảo rằng các phân đoạn được tạo ra đủ tin cậy để hỗ trợ trong quá trình giảng dạy và hướng dẫn.
- Tốc độ suy luận ở mức độ hợp lý.
  + Thời gian suy luận hợp lý: Đảm bảo rằng thời gian suy luận trên một hình ảnh là hợp lý và không gây ra thời gian chờ đợi quá lớn. Việc này sẽ giúp cho các buổi hướng dẫn và thảo luận diễn ra mượt mà hơn mà không làm gián đoạn.

3.2. Phương pháp để so sánh và lựa chọn dựa trên yêu cầu thiết kế

-Độ Chính Xác Vừa Phải nhưng Đáng Tin Cậy

+ Cấu trúc U-Net: Mô hình này sử dụng cấu trúc U-Net, nổi tiếng với khả năng tốt trong việc phân đoạn các đặc điểm quan trọng.
+ Các lớp Conv3D: Các lớp Conv3D giúp mô hình học được các đặc điểm không gian phức tạp từ ảnh 3D.
+ Dropout: Sử dụng các lớp Dropout trong các tầng khác nhau giúp giảm overfitting, cải thiện khả năng tổng quát hóa của mô hình, và đảm bảo rằng các phân đoạn đủ tin cậy để sử dụng trong quá trình giảng dạy và hướng dẫn.

-Tốc Độ Suy Luận Nhanh

+ Sử dụng Conv3DTranspose: Các lớp Conv3DTranspose trong đường dẫn mở rộng giúp tăng độ phân giải không gian một cách hiệu quả, cải thiện tốc độ suy luận.
+ Kiến trúc đơn giản: Mặc dù mô hình có độ sâu đáng kể, việc thiết kế các tầng với số lượng filters hợp lý (16, 32, 64, 128, 256) giúp duy trì sự cân bằng giữa độ phức tạp và tốc độ.

3.3. Các bước được đề xuất cho hệ thống dựa trên yêu cầu thiết kế

Phương pháp đề xuất bao gồm các bước thực hiện cụ thể như sau:

1. Chuẩn bị dữ liệu

- Tải xuống và giải nén dữ liệu: Thu thập các tệp dữ liệu MRI từ nguồn BraTS2020 và giải nén chúng vào thư mục cụ thể.
- Chuyển đổi các tệp dữ liệu về dạng Numpy.
- Chuẩn hóa tất cả các tập dữ liệu bằng MinMaxScaler: Sử dụng MinMaxScaler để chuẩn hóa giá trị pixel của các ảnh MRI vào khoảng [0, 1].
- Kết hợp các kênh dữ liệu gốc thành một kênh duy nhất: Xếp chồng các ảnh MRI (Flair, T1, T1ce, T2) để tạo thành mảng 4D kết hợp.
- Cắt bỏ để loại bỏ các vùng không cần thiết: Cắt các ảnh kết hợp và mặt nạ tương ứng thành kích thước 128x128x128 để chỉ giữ lại vùng quan trọng.
- Bỏ các tập tin có ít chú thích: Lọc ra các tập tin có ít hơn 1% thể tích hữu ích.
- Lưu các tập tin hữu ích dưới dạng mảng Numpy: Lưu các ảnh và mặt nạ đã kết hợp dưới dạng tệp .npy để sử dụng sau này.
- Chia dữ liệu thành các tập train và tập validation

2. Định nghĩa bộ sinh dữ liệu tùy chỉnh.

- Định nghĩa bộ sinh dữ liệu tùy chỉnh: Do Keras ImageDataGenerator không hỗ trợ tệp .npy, chúng ta cần định nghĩa một bộ sinh dữ liệu tùy chỉnh để tải và xử lý các tệp này từ đĩa trong quá trình huấn luyện mô hình.

3. Thiết kế mô hình 3D U-net từ mạng U-net có sẵn

- Kiến trúc U-Net 3D: Định nghĩa mô hình U-Net 3D với các lớp Conv3D, MaxPooling3D, Conv3DTranspose, và Dropout.
- Khởi tạo kernel: Sử dụng 'he_uniform' để khởi tạo các lớp tích chập.

4. Huấn luyện và dự đoán

4.1 Huấn luyện

- Tải dữ liệu theo batch: Sử dụng bộ sinh dữ liệu tùy chỉnh để tải dữ liệu theo lô.
- Định nghĩa hàm mất mát và chỉ số đo lường: Chọn hàm mất mát và các chỉ số đo lường phù hợp cho nhiệm vụ phân đoạn.
- Định nghĩa và biên dịch mô hình 3D U-net: Biên dịch mô hình với optimizer, hàm mất mát và các chỉ số đo lường.

4.2 Dự đoán và trực quan hóa

- Tải mô hình đã huấn luyện: Tải mô hình đã được huấn luyện trước đó để sử dụng cho quá trình dự đoán.
- Dự đoán trên một số ảnh kiểm tra: Thực hiện dự đoán trên các ảnh kiểm tra.
- Trực quan hóa kết quả dự đoán: Sử dụng matplotlib để hiển thị kết quả dự đoán trên các ảnh kiểm tra nhằm đánh giá chất lượng của mô hình.

Phương pháp đề xuất này cung cấp một quy trình toàn diện từ việc chuẩn bị và chuẩn hóa dữ liệu, định nghĩa mô hình 3D U-Net, đến huấn luyện và dự đoán. Quy trình này đảm bảo độ chính xác vừa phải nhưng đáng tin cậy, và tốc độ suy luận nhanh, phù hợp với yêu cầu của hệ thống phân đoạn ảnh y tế.

IV. Kết quả và hướng phát triển

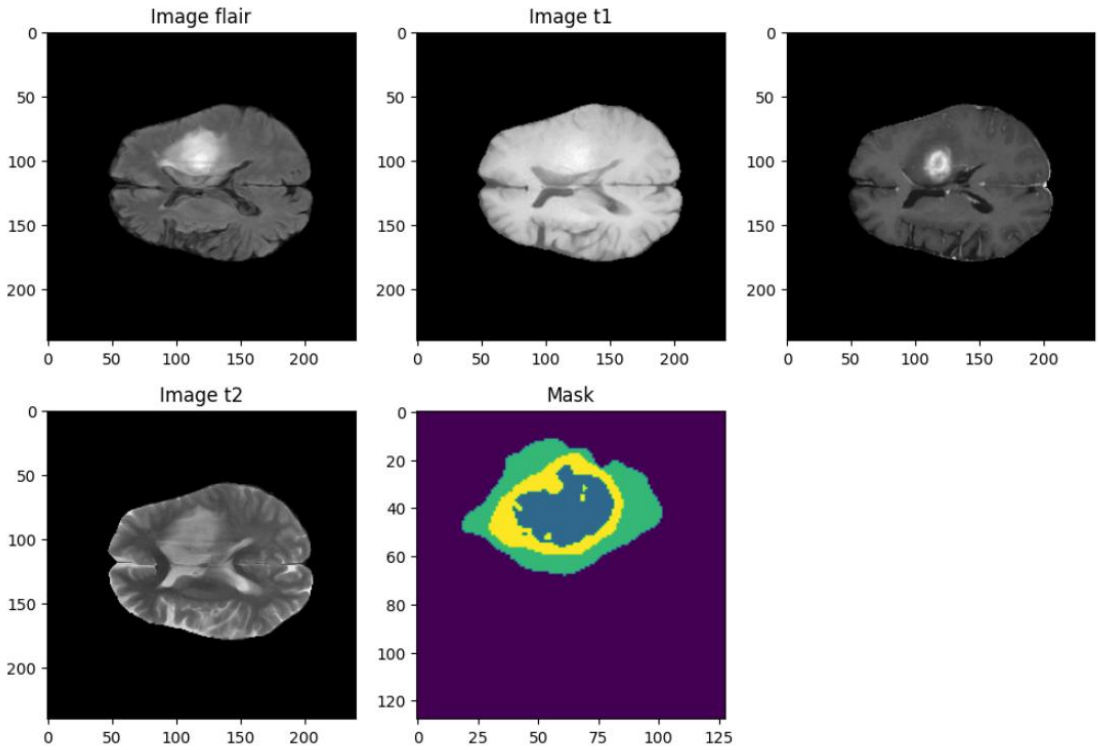
4.1. Tập dữ liệu hình (1)

Mô tả Dữ liệu (1):

Tập dữ liệu: Tập dữ liệu này chứa thông tin từ nhiều chế độ quét khác nhau được lưu dưới dạng các tệp NIfTI (. nii.gz). Có tổng cộng bốn "channel" thông tin khác nhau, mỗi kênh đại diện cho một loại quét khác nhau của cùng một vùng.

- Native (T1): Đây là thông tin từ quét T1 gốc, cung cấp hình ảnh về cấu trúc cơ bản của vùng quét. Quét T1 thường cung cấp thông tin về cấu trúc cơ bản của mô và mô, giúp xác định các đặc điểm anatomic.
- Post-contrast T1-weighted (T1CE): Đây là thông tin từ quét T1 sau khi tiêm chất đối lập, thường được sử dụng để phát hiện các bất thường hoặc khối u. Việc sử dụng chất đối lập giúp nổi bật lên các khu vực có sự thay đổi về dòng chảy máu hoặc thay đổi mô.
- T2-weighted (T2): Đây là thông tin từ quét T2, nơi mà mô và mô sẽ có cường độ tín hiệu khác nhau. Quét T2 thường hữu ích trong việc phát hiện các vấn đề liên quan đến nước, như viêm nhiễm hoặc tăng nước trong mô.
- T2 Fluid Attenuated Inversion Recovery (FLAIR): Đây là thông tin từ quét T2 FLAIR, một biến thể của quét T2 được thiết kế để loại bỏ tín hiệu từ dòng chảy máu. Quét T2 FLAIR thường được sử dụng để phát hiện và đánh dấu các vùng bất thường, như các vùng tổn thương hoặc bướu.

Dữ liệu này cung cấp một cái nhìn toàn diện và đa chiều về vùng u não, giúp cho việc phân tích và chẩn đoán trở nên hiệu quả hơn



Hình 1. Ví dụ về một u não từ tập dữ liệu huấn luyện BraTS 2020. Phía giữa bên dưới: Màu xanh đậm: u tăng nổi (ET), Màu xanh lá cây: u không tăng nổi / u nekrotic (NET/NCR), Màu vàng: sưng tấy xung quanh u (ED). Phía trên bên trái: chuỗi FLAIR, Phía trên bên phải: chuỗi T1CE, Phía dưới bên trái: chuỗi T2, Phía giữa bên trên: chuỗi T1 được tăng cường với bản đồ nhãn trên đè lên.

4.2. Phương pháp đánh giá

- Độ chính xác: đánh giá dựa trên chỉ số tính toán IoU trung bình trên tập validation, tính toán và hiển thị chỉ số IoU trung bình trên một batch hình ảnh từ tập kiểm tra, giúp đánh giá hiệu suất phân đoạn của mô hình.

- Đo thời gian suy luận trên các hình ảnh từ tập kiểm tra để đảm bảo rằng thời gian chờ đợi là hợp lý và không gây gián đoạn đến các hoạt động giảng dạy và thảo luận. Sử dụng các công cụ đo lường thời gian như Python's time module hoặc các tính năng tích hợp trong framework deep learning để đo và đánh giá thời gian suy luận.

4.3. Thực thi

1. Chuẩn bị dữ liệu

- Tải xuống và giải nén dữ liệu: Thu thập các tệp dữ liệu MRI từ nguồn BraTS2020 và giải nén chúng vào thư mục cụ thể.
- Chuyển đổi các tệp dữ liệu về dạng Numpy.

B1: Dùng phương thức get_fdata() chuyển đổi dữ liệu hình ảnh từ tệp NIfTI sang NumPy. Dữ liệu chuyển từ tệp NIfTI sang mảng NumPy sẽ có kích thước (240,240,155)

B2: Reshape kích thước từ mảng 3D thành mảng 2D

B3: Chuẩn hóa dữ liệu dùng MinMaxScaler(), để chuẩn hóa dữ liệu về khoảng cho phép [0 1].

B4: Reshape kích thước từ mảng 2D đã được chuẩn hóa dữ liệu trả về thành mảng 3D ban đầu.

+Dữ liệu sẽ có 4 kênh và chúng ta sẽ tiến hành sử lý cho tất cả 4 kênh ảnh (flair, t1, t1ce, t2).

+ Chúng chỉ tiến hành chuẩn hóa dữ liệu từ kênh ảnh, mà không chuẩn hóa cho mask.

B5: Sử dụng thư viện matplotlib để hiển thị trực quan ảnh đã được xử lý cho từng kênh ảnh, hình (2)

- Kết hợp các kênh dữ liệu gốc thành một kênh duy nhất: Xếp chồng các ảnh MRI (Flair, T1, T1ce, T2) để tạo thành mảng 4D kết hợp.

+ Từ việc sử lý và chuẩn hóa dữ liệu cơ bản từ tệp dữ liệu gốc, chúng ta sẽ tiến hình định dạng lại dữ liệu và chuyển bị dữ liệu để tiến hành train model

B1: Dữ liệu hiện có là 4 kênh ảnh (240, 240, 155), chúng ta sẽ tiến hành ghép các kênh ảnh trên thành duy nhất một kích đầu vào cho model để train dữ liệu. Cụ thể sẽ dùng hàm np.stack của thư viện numpy để tiến hành ghép cách kênh ảnh. Kích thước của dữ liệu lúc bây giờ sẽ là 1 kênh duy nhất (240, 240, 155,3)

- Lưu ý: mặc dù dữ liệu đầu nào là 4 kênh ảnh, nhưng chúng ta sẽ chỉ ghép 3 kênh lại với nhau vì kênh (t1) không cung cấp quá nhiều thông về dữ liệu, đồng thời việc bỏ bớt đi 1 kênh sẽ làm cho việc tính toán trở nên dễ dàng hơn.

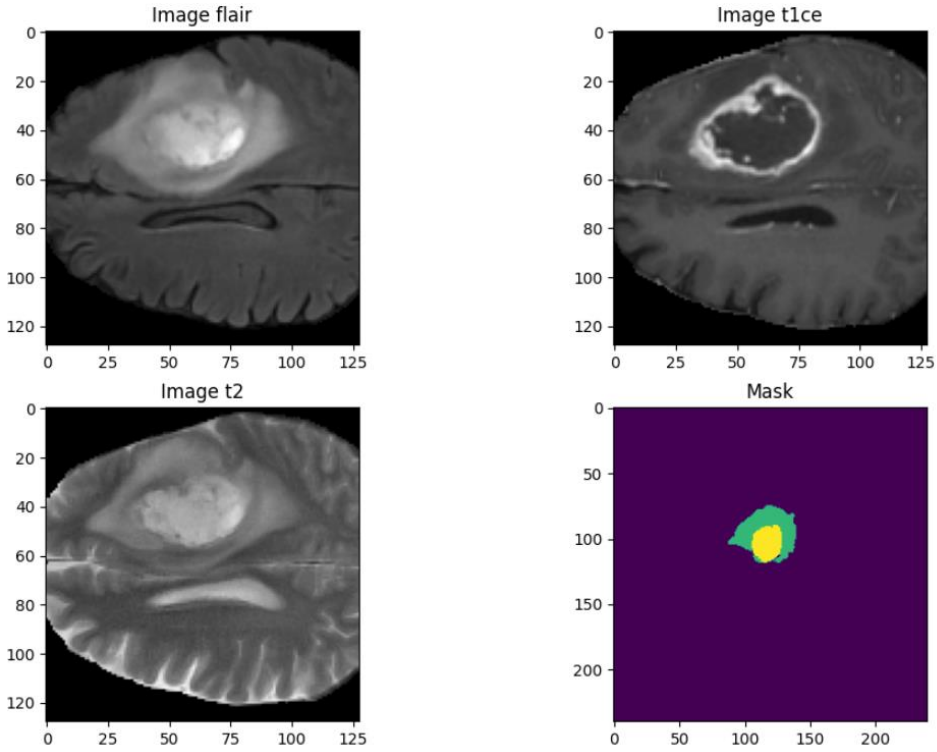
B2: Giới hạn lại kích thước đầu vào, thay vì (240, 240, 155, 3) thì kích thước đầu vào sẽ được giới hạn lại còn (128, 128, 128, 3), hình(2)

-Kết quả hình (3)

B3: Sử dụng thư viện matplotlib để hiển thị trực quan ảnh đã được xử lý cho từng kênh ảnh.

```
flair_shape (240, 240, 155)
```

Hình 1: Kích thước dữ liệu ảnh(flair) sau khi từ tệp NIfTI sang mảng NumPy



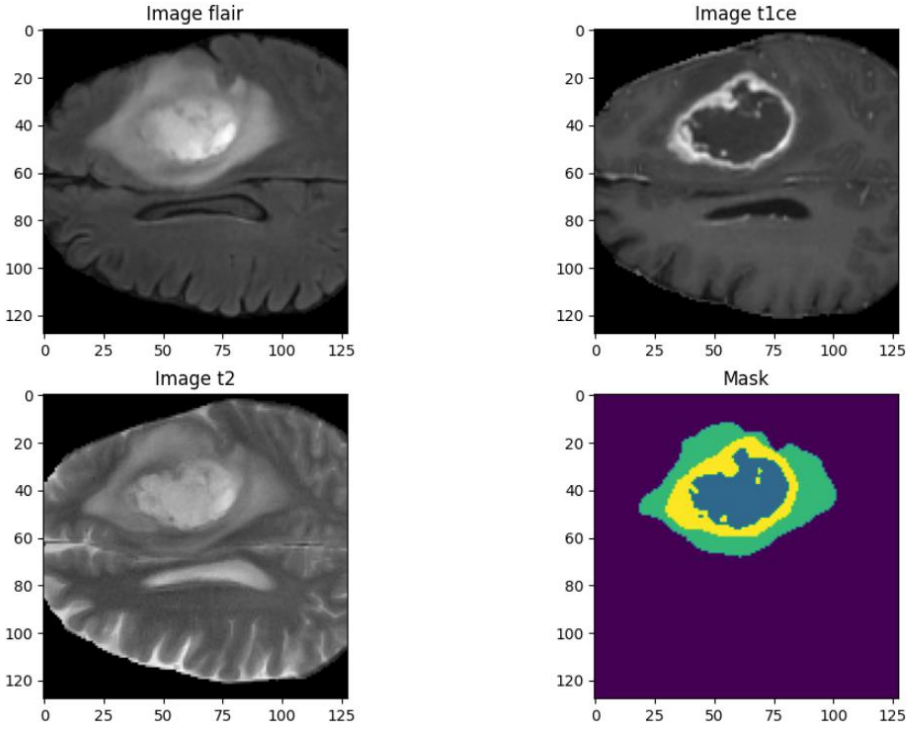
Hình 2: Hiển thị ảnh trực quan qua thư viện matplotlib

```
# Cắt ảnh đã kết hợp 3 kênh và mặt nạ
combined_x = combined_x[56:184, 56:184, 13:141]  # Cắt thành kích thước 128x128x128x3
test_mask = test_mask[56:184, 56:184, 13:141]  # Cắt mặt nạ tương tự 128x128x128
```

Hình 3: Lệnh Reshape kích thước đầu vào

```
combined_x_shape: (128, 128, 128, 3)
test_mask.shape: (128, 128, 128)
```

Hình 4: Kích thước dữ liệu ảnh sau khi Reshape



Hình 4: Hiển thị ảnh trực quan qua thư viện matplotlib

- Bỏ các tập dữ liệu có ít chú thích: Lọc và lưu lại những mask có giá trị hữu ích, tức là những mặt nạ có ít nhất 1% giá trị pixel không phải là giá trị 0. Nếu mặt nạ thỏa mãn điều kiện này, nó sẽ được chuyển đổi sang định dạng phân loại và được lưu trữ cùng với mảng hình ảnh tương ứng. Nếu không thỏa mãn, mặt nạ sẽ không được lưu lại. Điều này giúp loại bỏ những mặt nạ không có giá trị hoặc không có đủ thông tin cần thiết, đảm bảo chỉ lưu trữ những dữ liệu quan trọng và hữu ích, hình (5)

-Lưu các tập tin hữu ích dưới dạng mảng Numpy: Lưu các ảnh và mặt nạ đã kết hợp dưới dạng tệp .npy để sử dụng sau này, hình (5)

```
    val, counts = np.unique(temp_mask, return_counts=True)

    if (1 - (counts[0]/counts.sum())) > 0.01:  # Ít nhất 1% thể tích có giá trị hữu ích
        print("Save")
        temp_mask = to_categorical(temp_mask, num_classes=4)
        np.save(f'BraTS2020_TrainingData/input_data_3channels/images/image_{img}.npy', temp_combined_images)
        np.save(f'BraTS2020_TrainingData/input_data_3channels/masks/mask_{img}.npy', temp_mask)
    else:
        print("NotSave")
```

Hình 5: Loại bỏ các mask không có nhãn.

- Chia dữ liệu thành các tập train và tập validation. Chia tập dữ liệu đã được xử lý thành tập train(75%), tập valadition (25%), chuẩn bị cho quá trình train dữ liệu.

```
import splitfolders
# Định nghĩa thư mục vào
input_folder = 'BraTS2020_TrainingData/input_data_3channels/'
output_folder = 'BraTS2020_TrainingData/input_data_128/'
# Chia tập dữ liệu train trong output_forder
splitfolders.ratio(input_folder, output=output_folder, seed=42, ratio=(.75, .25), group_prefix=None)
```

Hình 6: Lệnh chia dữ liệu bằng thư viện splitfolders

2. Định nghĩa bộ sinh dữ liệu tùy chỉnh.

- Ở phần này chúng ta sẽ tiến hành định nghĩa các batch, chúng ta cần phải định nghĩa ra nó vì không thể load hết 1 lần dữ liệu lên để tiến hành train được, thời gian tính toán sẽ rất lâu.

+ Định nghĩa ra hàm load dữ liệu từ tập dữ liệu train và tập validation đã được định nghĩa ở mục 4.3.1. hình (6)

+ Định nghĩa ra một bộ tạo dữ liệu để lấy từng batch dữ liệu từ tập dữ liệu mà ta mong muốn, hình (7)

+ Từ 2 hàm tự định nghĩa trên, số lượng dữ liệu cho mỗi bath sẽ được quyết định thông qua batch_size, và thông qua hai biến img, msk, hình (8)

+ Một lần nữa kiểm tra kích thước đầu vào của dữ liệu hình (9)

```python
#Nhập Thư Viện và Định Nghĩa Hàm Tải Ảnh
import os
import numpy as np

def load_img(img_dir, img_list):# đối số đầu vào img_dir, img_list(đường dẫn chứa ảnh, danh sách)
    images = []# mảng để chứa
    for i, image_name in enumerate(img_list):# vòng lặp qua từng danh sách trong tệp ảnh
        if image_name.split('.')[1] == 'npy':# kiểm ra xem, có phải là tệp npy không
            image = np.load(img_dir + image_name)
            images.append(image)
    images = np.array(images)
    return images

# Định nghĩa hàm tải tải
# hàm này trả về kiểu numpy, chứa tất cả các ảnh được tải
```

Hình 6 Định nghĩa hàm load dữ liệu trong tập dữ liệu

```python
#Định Nghĩa Bộ Tạo Dữ Liệu
def imageLoader(img_dir, img_list, mask_dir, mask_list, batch_size):
    L = len(img_list)

    # Keras yêu cầu bộ tạo phải là vô hạn, vì vậy sử dụng while True
    while True:
        batch_start = 0
        batch_end = batch_size

        while batch_start < L:
            limit = min(batch_end, L)

            X = load_img(img_dir, img_list[batch_start:limit])
            Y = load_img(mask_dir, mask_list[batch_start:limit])

            yield (X, Y)  # một tuple với hai mảng numpy chứa batch_size mẫu

            batch_start += batch_size
            batch_end += batch_size
```

Hình 7 Định nghĩa bộ tạo dữ liệu trong tập dữ liệu

```
#Định nghĩa từng batch cho việc train và đánh giá trên tập validation

##################################
train_img_dir = "datasheet/content/data/BraTS2020_TrainingData/input_data_128/train/images/"
train_mask_dir ="datasheet/content/data/BraTS2020_TrainingData/input_data_128/train/masks/"
val_img_dir = "datasheet/content/data/BraTS2020_TrainingData/input_data_128/train/images/"
val_mask_dir = "datasheet/content/data/BraTS2020_TrainingData/input_data_128/train/images/"
train_img_list=os.listdir(train_img_dir)
train_mask_list = os.listdir(train_mask_dir)
val_img_list=os.listdir(val_img_dir)
val_mask_list = os.listdir(val_mask_dir)
##################################

############################################################################
batch_size = 2
train_img_datagen = imageLoader(train_img_dir, train_img_list,
                                train_mask_dir, train_mask_list, batch_size)
val_img_datagen = imageLoader(val_img_dir, val_img_list,
                                val_mask_dir, val_mask_list, batch_size)

# Dữ liệu sẽ được gọi thông qua img cho ảnh và msk cho nhãn
img, msk = train_img_datagen.__next__()
print(img.shape)
print(msk.shape)
############################################################################################
```

Hình 8 Load dữ liệu vào 2 biến (img, msk) để chuẩn bị train.

```
img_shape (2, 128, 128, 128, 3)
msk_shape (2, 128, 128, 128, 4)
```

Hình 9 Kiểm tra kích thước dữ liệu lần cuối trước khi train

3. Thiết kế mô hình 3D U-net từ mạng U-net có sẵn

3.1 Thông tin model

- Import các thư viện cần thiết
- Khởi tạo Kernel: khởi tạo trọng số cho các lớp Conv3D. he_uniform sẽ được sử dụng để cải thiện hiệu suất của mô hình.
- Hàm xây dựng mô hình U-Net 3D hình (10)

+IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH: Kích thước của ảnh đầu v ào.

+IMG_CHANNELS: Số kênh của ảnh đầu vào.

+num_classes: Số lớp đầu ra.num_classes: Số lớp đầu ra.

```
def simple_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH, IMG_CHANNELS, num_classes):
```

Hình 10: Hàm xây dựng model

- Xây dựng mô hình

+Khởi tạo lớp đầu vào với kích thước và số kênh xác định hình (10)

```
# Input layer
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH, IMG_CHANNELS))
s = inputs
```

Hình 11: Khởi tạo kích thước đầu vào của model

+ Tạo Contraction Path (Encoder), hình (11)

-c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(s)

+Conv3D: Lớp convolution 3D áp dụng các convolution 3D lên tensor đầu vào s.

+16: Số lượng bộ lọc trong lớp Conv3D là 16.

+(3, 3, 3): Kích thước cửa sổ convolution 3D (kích thước kernel) là 3x3x3.

+activation='relu': Hàm kích hoạt ReLU.

+kernel_initializer=kernel_initializer: Bộ khởi tạo trọng số cho kernel.

+padding='same': Đảm bảo rằng kích thước của đầu ra khớp với kích thước đầu vào sau khi convolution.

+(s): Sử dụng tensor đầu vào s.

-c1 = Dropout(0.1)(c1)

+Dropout: Kỹ thuật regularization để ngăn chặn overfitting.

+0.1: Tỷ lệ dropout là 10%.

+(c1): Sử dụng tensor c1 từ lớp trước đó.

-c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c1)

+Conv3D: Một lớp convolution 3D khác áp dụng lên tensor đầu vào c1.

+16: Số lượng bộ lọc trong lớp Conv3D là 16.

+(3, 3, 3): Kích thước cửa sổ convolution 3D (kích thước kernel) là 3x3x3.

+activation='relu': Hàm kích hoạt ReLU.

+kernel_initializer=kernel_initializer: Bộ khởi tạo trọng số cho kernel.

+padding='same': Đảm bảo rằng kích thước của đầu ra khớp với kích thước đầu vào sau khi convolution.

+(c1): Sử dụng tensor c1 từ lớp trước đó (sau khi dropout).

-p1 = MaxPooling3D((2, 2, 2))(c1)

+MaxPooling3D: Lớp pooling 3D áp dụng max pooling lên tensor đầu vào c1.

+(2, 2, 2): Kích thước của cửa sổ pooling là 2x2x2.

+(c1): Sử dụng tensor c1 từ lớp trước đó.

Tóm tắt:

-Conv3D: Áp dụng convolution 3D với 16 bộ lọc lên tensor đầu vào s, sử dụng kernel kích thước 3x3x3 và hàm kích hoạt ReLU. Kết quả là c1.

-Dropout: Áp dụng dropout với tỷ lệ 10% lên tensor c1 để giảm overfitting. Kết quả vẫn là c1.

-Conv3D: Áp dụng convolution 3D thứ hai với 16 bộ lọc lên tensor c1, sử dụng kernel kích thước 3x3x3 và hàm kích hoạt ReLU. Kết quả là c1.

-MaxPooling3D: Áp dụng max pooling với cửa sổ kích thước 2x2x2 lên tensor c1, giảm kích thước không gian của tensor. Kết quả là p1.

-Các bước này tạo thành một khối trong đường đi xuống của mô hình U-Net, nơi các đặc trưng được trích xuất và kích thước không gian của chúng giảm đi, giúp mô hình học được các đặc trưng quan trọng ở các cấp độ khác nhau.

- Làm tương tự cho các khối c2, c3, c4, c5 theo hình (11)

+ Tạo Expansion Path (Decoder) hình (12)

- u6 = Conv3DTranspose(128, (2, 2, 2), strides=(2, 2, 2), padding='same')(c5)

+ Conv3DTranspose: Lớp convolution chuyển vị để tăng kích thước không gian của đầu vào c5.

+ 128: Số lượng bộ lọc là 128.

+ (2, 2, 2): Kích thước kernel là 2x2x2.

+ strides=(2, 2, 2): Bước nhảy của convolution là 2 theo mỗi chiều, nghĩa là tăng kích thước không gian lên gấp đôi. padding='same': Kích thước đầu ra sẽ bằng kích thước đầu vào.

+ (c5): Sử dụng tensor đầu vào từ lớp trước đó c5

-u6 = concatenate([u6, c4])

+ concatenate: Hàm nối tensor u6 (đã được upsampled) với tensor c4 từ đường đi xuống.

+ [u6, c4]: Tensor u6 và tensor c4 được nối lại với nhau.

-c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u6)

+ Conv3D: Lớp convolution 3D.

+128: Số lượng bộ lọc là 128.

+(3, 3, 3): Kích thước kernel là 3x3x3.

+activation='relu': Hàm kích hoạt ReLU.

+kernel_initializer=kernel_initializer: Bộ khởi tạo trọng số cho kernel.

+padding='same': Kích thước đầu ra sẽ bằng kích thước đầu vào.

+u6: Sử dụng tensor đã được nối u6.

-c6 = Dropout(0.2)(c6)

+Dropout: Kỹ thuật regularization để ngăn chặn overfitting.

+0.2: Tỷ lệ dropout là 20%.

+(c6): Sử dụng tensor c6 từ lớp trước đó.

-c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c6)

+Conv3D: Lớp convolution 3D thứ hai.

+128: Số lượng bộ lọc là 128.

+(3, 3, 3): Kích thước kernel là 3x3x3.

+activation='relu': Hàm kích hoạt ReLU.

+kernel_initializer=kernel_initializer: Bộ khởi tạo trọng số cho kernel.

+padding='same': Kích thước đầu ra sẽ bằng kích thước đầu vào.

+(c6): Sử dụng tensor c6 từ lớp trước đó (kết quả sau khi dropout).

Tóm tắt:

-Lớp Conv3DTranspose: Tăng kích thước của tensor đầu vào c5.

-Hàm concatenate: Kết hợp tensor đã được upsampled u6 với bản đồ đặc trưng tương ứng từ đường đi xuống c4.

-Lớp Conv3D: Áp dụng một convolution 3D lên tensor đã được nối.

-Lớp Dropout: Áp dụng dropout để giảm overfitting.

-Lớp Conv3D: Áp dụng một convolution 3D khác lên tensor sau khi dropout.

-Các thao tác này là một khối điển hình trong đường đi lên của U-Net, tăng dần kích thước của các bản đồ đặc trưng đồng thời kết hợp chúng với các đặc trưng có độ phân giải cao từ đường đi xuống để tái tạo lại các kích thước không gian của đầu vào

-Làm tương tự cho các u7, u8, u9, với các chỉ số theo hình (12)

+ Tạo đầu ra hình (13)

- Lớp cuối cùng sử dụng Conv3D với kích thước bộ lọc 1x1x1 và hàm kích hoạt softmax để dự đoán phân đoạn.

+Khởi tạo, hiển thị và kiểm tra mô hình hình (14)

- Tạo mô hình từ lớp đầu vào và đầu ra, sau đó hiển thị tóm tắt cấu trúc mô hình.

- Khởi tạo mô hình với kích thước ảnh 128x128x128, 3 kênh đầu vào, và 4 lớp đầu ra. Sau đó, in ra kích thước đầu vào và đầu ra của mô hình để kiểm tra, hình

+ Kết quả của model

- Thông tin kết quả được thể hiện ở hình (15)

```
#Contraction path
c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(s)
c1 = Dropout(0.1)(c1)
c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c1)
p1 = MaxPooling3D((2, 2, 2))(c1)

c2 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p1)
c2 = Dropout(0.1)(c2)
c2 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c2)
p2 = MaxPooling3D((2, 2, 2))(c2)

c3 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p2)
c3 = Dropout(0.2)(c3)
c3 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c3)
p3 = MaxPooling3D((2, 2, 2))(c3)

c4 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p3)
c4 = Dropout(0.2)(c4)
c4 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c4)
p4 = MaxPooling3D(pool_size=(2, 2, 2))(c4)

c5 = Conv3D(256, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p4)
c5 = Dropout(0.3)(c5)
c5 = Conv3D(256, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c5)
```

Hình 11: Khối Contraction Path (Encoder)

```
#Expansive path
u6 = Conv3DTranspose(128, (2, 2, 2), strides=(2, 2, 2), padding='same')(c5)
u6 = concatenate([u6, c4])
c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u6)
c6 = Dropout(0.2)(c6)
c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c6)

u7 = Conv3DTranspose(64, (2, 2, 2), strides=(2, 2, 2), padding='same')(c6)
u7 = concatenate([u7, c3])
c7 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u7)
c7 = Dropout(0.2)(c7)
c7 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c7)

u8 = Conv3DTranspose(32, (2, 2, 2), strides=(2, 2, 2), padding='same')(c7)
u8 = concatenate([u8, c2])
c8 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u8)
c8 = Dropout(0.1)(c8)
c8 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c8)

u9 = Conv3DTranspose(16, (2, 2, 2), strides=(2, 2, 2), padding='same')(c8)
u9 = concatenate([u9, c1])
c9 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u9)
c9 = Dropout(0.1)(c9)
c9 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c9)
```

Hình 12: Khối Expansion Path (Decoder)

```
outputs = Conv3D(num_classes, (1, 1, 1), activation='softmax')(c9)
```

Hình 13: Output của model

```
    model = Model(inputs=[inputs], outputs=[outputs])
    #compile model outside of this function to make it flexible.
    model.summary()

    return model

model = simple_unet_model(128, 128, 128, 3, 4)
print('input_shape', model.input_shape)
print('output_shape', model.output_shape)
```

Hình 14 Hiển thị và kiểm tra mô hình hình

```
Total params: 5645828 (21.54 MB)
Trainable params: 5645828 (21.54 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
input_shape (None, 128, 128, 128, 3)
output_shape (None, 128, 128, 128, 4)
```

Hình 15 Kết quả của model

4 . Huấn luyện và dự đoán

4.1 Huấn luyện

- Tính toán trọng số cho từng lớp.

+ Tính toán số lượng pixel của từng lớp của mask(128x128x128x4), gán vào một datafram được tạo sẵn hình (16)

+ Dựa trên tổng số lượng pixel của từng lớp, tiến hành tính toán trọng số cho từng lớp để giảm thiểu vấn đề mất cân bằng dữ liệu, hình (17)

```
import pandas as pd
columns = ['0','1', '2', '3']
df = pd.DataFrame(columns=columns)
train_mask_list = sorted(glob.glob('BraTS2020_TrainingData/input_data_128/train/masks/*.npy'))
for img in range(len(train_mask_list)):
    print(img)
    temp_image=np.load(train_mask_list[img])#(128x128x128x4)
    temp_image = np.argmax(temp_image, axis=3)
    print( temp_image.shape)
    val, counts = np.unique(temp_image, return_counts=True)
    zipped = zip(columns, counts)
    conts_dict = dict(zipped)
    df = df.append(conts_dict, ignore_index=True)
print(df)
```

Hình 16 Tính toán số lượng mẫu từ (msk) trong mỗi lớp

```
label_0 = df['0'].sum()
label_1 = df['1'].sum()
label_2 = df['1'].sum()
label_3 = df['3'].sum()
total_labels = label_0 + label_1 + label_2 + label_3
n_classes = 4 ,

# Trọng số từng lớp= tổng số lượng mẫu / (số lớp * số lượng mẫu từng lớp)
wt0 = round((total_labels/(n_classes*label_0)), 2) #round to 2 decimals
wt1 = round((total_labels/(n_classes*label_1)), 2)
wt2 = round((total_labels/(n_classes*label_2)), 2)
wt3 = round((total_labels/(n_classes*label_3)), 2)
#Kết quả: wt0 =0.26, wt1 =22.53, wt2= 22.53, wt3 =26.21
```

Hình 17 Tính toán trọng số cho từng lớp

- Định nghĩa hàm mất mát và chỉ số đo lường: Chọn hàm mất mát và các chỉ số đo lường phù hợp cho nhiệm vụ phân đoạn.

+ Định nghĩa hàm Loss. Hàm Loss được tính toán dựa trên dice_loss và focal_loss trong thư viện segmentation_models_3D

- Dice Loss (Loss Dice) là hàm mất mát được sử dụng trong bài toán phân đoạn hình ảnh (image segmentation), đo lường sự tương đồng giữa dự đoán và nhãn thực tế bằng cách tính tỷ lệ giữa diện tích giao nhau và tổng diện tích của hai vùng, sẽ tính tính toán ứng với trọng số của từng lớp wt0, wt1, wt2, wt3.
- Focal Loss là một hàm mất mát được thiết kế để giảm hiện tượng mất cân bằng giữa các lớp trong bài toán phân loại đa lớp.
-Tổng Loss (Total Loss): total_loss là tổng của Dice Loss và Focal Loss, với trọng số 1 cho Focal Loss.

+ Định nghĩa metrics, optimizer để tiến hành train dữ liệu, hình (18)

-Metrics được sử dụng là Accuracy và IOU

-Optimizers được sử dụng là Adam với learing_rate được khởi tạo là 0.0001

```
#Define loss, metrics and optimizer to be used for training
wt0, wt1, wt2, wt3 = 0.26, 22.53, 22.53, 26.21
import segmentation_models_3D as sm
dice_loss = sm.losses.DiceLoss(class_weights=np.array([wt0, wt1, wt2, wt3]))
focal_loss = sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)

metrics = ['accuracy', sm.metrics.IOUScore(threshold=0.5)]

LR = 0.0001
optim = keras.optimizers.Adam(LR)
```

Hình 18: Định nghĩa hàm loss(total_loss) và các tham số(metrics, optimmizer)

- Biên dịch mô hình 3D U-net: Biên dịch mô hình với optimizer, hàm mất mát và các chỉ số đo lường, load ảnh theo từng batch và tiến hành train model. Lưu kết quả lại vào file hdf5, hình (19), hình (20)

```
# Train model
steps_per_epoch = len(train_img_list)//batch_size
val_steps_per_epoch = len(val_img_list)//batch_size


#from  simple_3d_unet import simple_unet_model

model = simple_unet_model(IMG_HEIGHT=128,
                          IMG_WIDTH=128,
                          IMG_DEPTH=128,
                          IMG_CHANNELS=3,
                          num_classes=4)
model.compile(optimizer = optim, loss=total_loss, metrics=metrics)
print(model.summary())
print(model.input_shape)
print(model.output_shape)
history=model.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=100,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
model.save('brats_3d.hdf5')
```

Hình 19 Train dữ liệu

4.4. Kết quả

4.4.1 Định lượng

1. Quá trình huấn luyện mô hình

Dữ liệu được huấn luyện với 100 epoch.

Các chỉ số chính tại epoch 1:

- Accuracy: 0.7879

Độ chính xác ban đầu là 78.79%, cho thấy mô hình đã có khả năng phân loại ban đầu nhưng còn khá thấp.

-IoU Score: 0.1612

Chỉ số IoU thấp (16.12%) thể hiện khả năng phân đoạn chính xác các tiểu vùng còn rất hạn chế ở giai đoạn đầu.

-Loss: 0.9693

Mất mát cao (96.93%) cho thấy mô hình vẫn đang học và còn nhiều lỗi cần phải điều chỉnh.

-Validation Accuracy: 0.9426

Độ chính xác trên tập validation là 94.26%, cho thấy mô hình hoạt động khá tốt trên dữ liệu chưa thấy trong quá trình huấn luyện nhưng có thể bị overfitting.

-Validation IoU Score: 0.1789

Chỉ số IoU trên tập validation là 17.89%, cũng khá thấp, cần cải thiện.

-Validation Loss: 0.8186

Mất mát trên tập validation là 81.86%, cao nhưng thấp hơn so với tập huấn luyện, có thể do overfitting hoặc dữ liệu validation dễ hơn.

Các chỉ số chính tại epoch 100 (hình 21):

-Accuracy: 0.9991

Độ chính xác rất cao (99.91%), cho thấy mô hình đã học rất tốt và gần như không có lỗi.

-IoU Score: 0.7931

Chỉ số IoU tăng lên 79.31%, chứng tỏ khả năng phân đoạn các tiểu vùng được cải thiện đáng kể.

-Loss: 0.7843

Mất mát giảm xuống còn 78.43%, cho thấy mô hình đã tối ưu hóa tốt hơn, mặc dù vẫn còn cao hơn mong đợi cho một mô hình tốt.

-Validation Accuracy: 0.9812

Độ chính xác trên tập validation là 98.12%, cao nhưng không cao bằng tập huấn luyện, cho thấy mô hình có khả năng tổng quát hóa tốt.

-Validation IoU Score: 0.6594

Chỉ số IoU trên tập validation đạt 65.94%, cho thấy mô hình đã cải thiện đáng kể trên dữ liệu chưa thấy.

-Validation Loss: 0.8186

Mất mát trên tập validation không thay đổi nhiều so với epoch 1, điều này có thể do dữ liệu validation đặc thù hoặc các yếu tố khác không thay đổi nhiều.

Nhận xét:

1. Độ chính xác (Accuracy):

   Sự tăng từ 78.79% lên 99.91% cho thấy mô hình đã học rất tốt qua các epoch, tăng đáng kể khả năng phân loại chính xác.

2. Chỉ số IoU (IoU Score):

   IoU tăng từ 0.1612 lên 0.7931, chứng minh khả năng phân đoạn đã được cải thiện nhiều, mô hình đã học cách phân đoạn các tiểu vùng một cách chính xác hơn.

3. Mất mát (Loss)

   Mặc dù mất mát giảm từ 0.9693 xuống 0.7843, nó vẫn tương đối cao, cho thấy còn có thể cải thiện thêm.

4. Validation Metrics:

   Độ chính xác trên tập validation và IoU cũng tăng lên, cho thấy mô hình không chỉ học tốt trên tập huấn luyện mà còn có khả năng tổng quát hóa tốt trên dữ liệu mới.

   Mặc dù validation loss không thay đổi nhiều, điều này có thể do tập dữ liệu validation đặc thù hoặc các yếu tố khác mà mô hình cần phải học thêm để tối ưu hóa tốt hơn.

Kết luận:

-Sự Cải Thiện Toàn Diện: Các chỉ số đều cải thiện đáng kể từ epoch 1 đến epoch 100, cho thấy mô hình đã học và tối ưu hóa tốt qua quá trình huấn luyện.

-Khả Năng Tổng Quát Hóa: Các chỉ số trên tập validation cũng cải thiện, chứng minh mô hình có khả năng tổng quát hóa tốt và không bị overfitting.

-Cần Cải Thiện Thêm: Mất mát vẫn còn cao, cần có biện pháp tối ưu hóa thêm để giảm thiểu mất mát và cải thiện hiệu suất mô hình.

4.4.2 Định tính

1. Trực quan hóa trên ảnh của tập Validation:

-Hình ảnh kết quả dự đoán trên tập validation (hình 25) cho thấy mô hình đã phân đoạn chính xác các tiểu vùng bên trên khối u não.

-Các vùng phân đoạn trong hình ảnh validation trùng khớp với các vùng thực tế, thể hiện qua việc mô hình có thể nắm bắt được các đặc điểm quan trọng của dữ liệu huấn luyện.

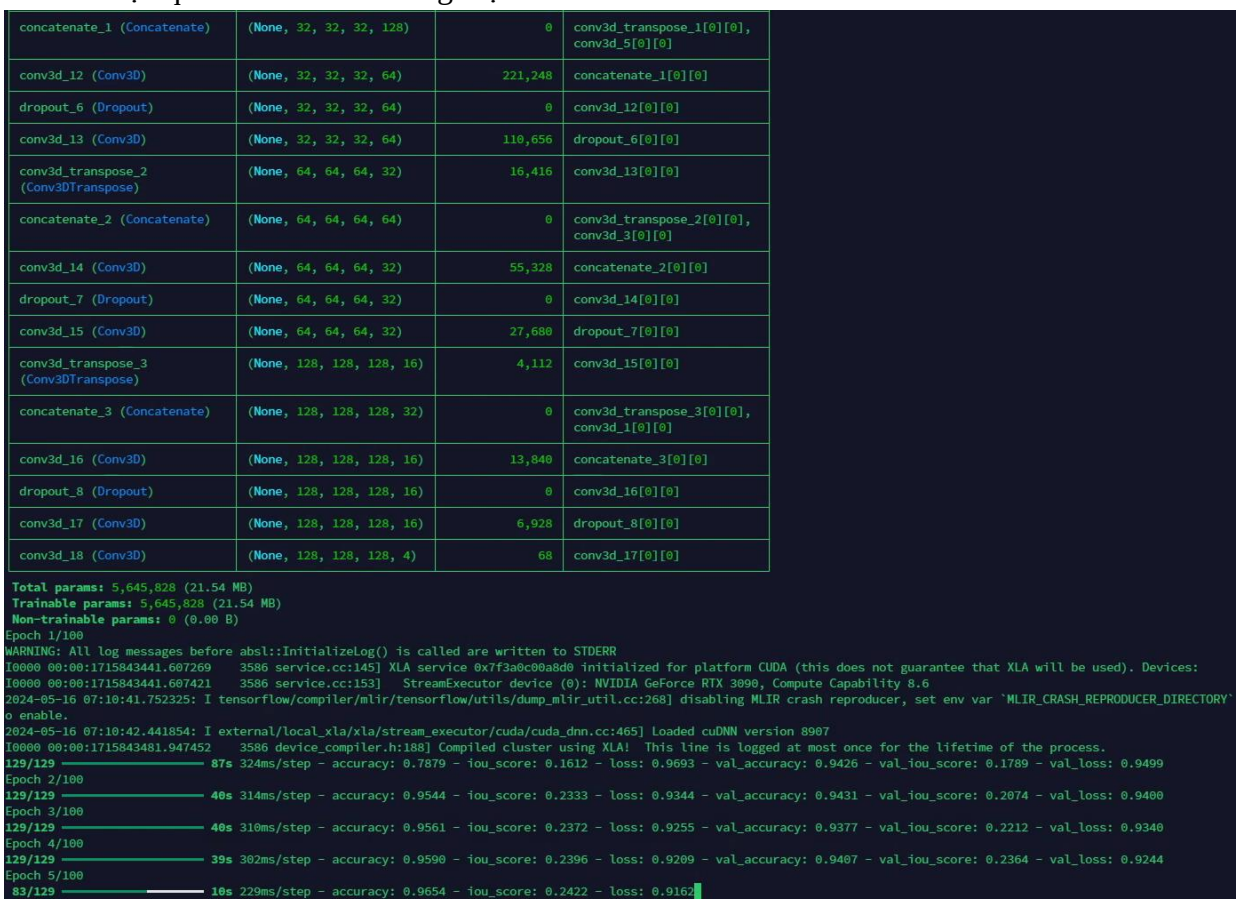
2. Trực quan hóa trên ảnh của tập Test:

-Hình ảnh kết quả dự đoán trên tập test (hình 26) thể hiện khả năng của mô hình trong việc phân đoạn chính xác trên dữ liệu mới mà nó chưa từng thấy.

-Các vùng phân đoạn trong hình ảnh test cũng cho thấy sự tương đồng với vùng thực tế, điều này minh chứng cho khả năng tổng quát hóa và tính khả thi của mô hình trong ứng dụng thực tế.

Kết luận:

- Định lượng: Các chỉ số đánh giá như accuracy, IoU score, loss cho thấy mô hình đã cải thiện đáng kể qua các epoch và có khả năng phân đoạn chính xác.
- Định tính: Trực quan hóa kết quả dự đoán trên tập validation và test cho thấy mô hình có thể nhận diện và phân đoạn chính xác các tiểu vùng bên trên khối u não, khẳng định tính hiệu quả của mô hình trong thực tế.

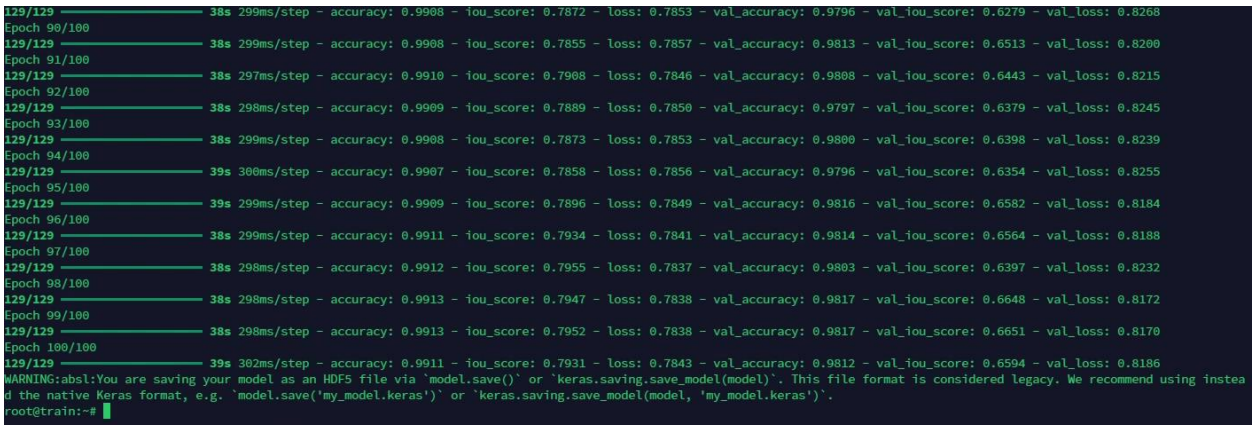


concatenate_1 (Concatenate)	(None, 32, 32, 32, 128)	0	conv3d_transpose_1[0][0], conv3d_5[0][0]
conv3d_12 (Conv3D)	(None, 32, 32, 32, 64)	221,248	concatenate_1[0][0]
dropout_6 (Dropout)	(None, 32, 32, 32, 64)	0	conv3d_12[0][0]
conv3d_13 (Conv3D)	(None, 32, 32, 32, 64)	110,656	dropout_6[0][0]
conv3d_transpose_2 (Conv3DTranspose)	(None, 64, 64, 64, 32)	16,416	conv3d_13[0][0]
concatenate_2 (Concatenate)	(None, 64, 64, 64, 64)	0	conv3d_transpose_2[0][0], conv3d_3[0][0]
conv3d_14 (Conv3D)	(None, 64, 64, 64, 32)	55,328	concatenate_2[0][0]
dropout_7 (Dropout)	(None, 64, 64, 64, 32)	0	conv3d_14[0][0]
conv3d_15 (Conv3D)	(None, 64, 64, 64, 32)	27,680	dropout_7[0][0]
conv3d_transpose_3 (Conv3DTranspose)	(None, 128, 128, 128, 16)	4,112	conv3d_15[0][0]
concatenate_3 (Concatenate)	(None, 128, 128, 128, 32)	0	conv3d_transpose_3[0][0], conv3d_1[0][0]
conv3d_16 (Conv3D)	(None, 128, 128, 128, 16)	13,840	concatenate_3[0][0]
dropout_8 (Dropout)	(None, 128, 128, 128, 16)	0	conv3d_16[0][0]
conv3d_17 (Conv3D)	(None, 128, 128, 128, 16)	6,928	dropout_8[0][0]
conv3d_18 (Conv3D)	(None, 128, 128, 128, 4)	68	conv3d_17[0][0]

Total params: 5,645,828 (21.54 MB)
Trainable params: 5,645,828 (21.54 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1715843441.607269    3586 service.cc:145] XLA service 0x7f3a0c00a8d0 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
I0000 00:00:1715843441.607421    3586 service.cc:153]   StreamExecutor device (0): NVIDIA GeForce RTX 3090, Compute Capability 8.6
2024-05-16 07:10:41.752325: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY`
o enable.
2024-05-16 07:10:42.441854: I external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:465] Loaded cuDNN version 8907
I0000 00:00:1715843481.947452    3586 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
129/129 ━━━━━━━━━━━━━━━━━━━━ 87s 324ms/step - accuracy: 0.7879 - iou_score: 0.1612 - loss: 0.9693 - val_accuracy: 0.9426 - val_iou_score: 0.1789 - val_loss: 0.9499
Epoch 2/100
129/129 ━━━━━━━━━━━━━━━━━━━━ 40s 314ms/step - accuracy: 0.9544 - iou_score: 0.2333 - loss: 0.9344 - val_accuracy: 0.9431 - val_iou_score: 0.2074 - val_loss: 0.9400
Epoch 3/100
129/129 ━━━━━━━━━━━━━━━━━━━━ 40s 310ms/step - accuracy: 0.9561 - iou_score: 0.2372 - loss: 0.9255 - val_accuracy: 0.9377 - val_iou_score: 0.2212 - val_loss: 0.9340
Epoch 4/100
129/129 ━━━━━━━━━━━━━━━━━━━━ 39s 302ms/step - accuracy: 0.9590 - iou_score: 0.2396 - loss: 0.9209 - val_accuracy: 0.9407 - val_iou_score: 0.2364 - val_loss: 0.9244
Epoch 5/100
 83/129 ━━━━━━━━━━━━━━━━━━━━ 10s 229ms/step - accuracy: 0.9654 - iou_score: 0.2422 - loss: 0.9162
```

Hình 20: Model với một vài epoch đầu tiên



Hình 21: Model ở epoch thứ 100

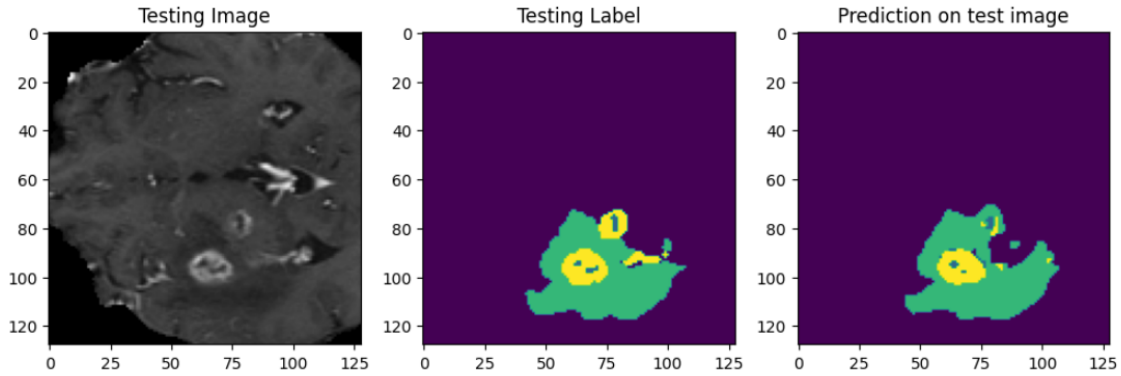
```
from keras.models import load_model
model = load_model('/content/drive/MyDrive/Colab Notebooks/brats_3d.hdf5',compile=False)
```

Hình 22: Load model đã train

```
###########################################
# Dự đoán một ảnh trên tập validation
img_num = 100
test_img = np.load("/content/datasheet/content/data/BraTS2020_TrainingData/input_data_128/val/images/image_"+str(img_num)+".npy")
test_mask = np.load("/content/datasheet/content/data/BraTS2020_TrainingData/input_data_128/val/masks/mask_"+str(img_num)+".npy")
test_mask_argmax=np.argmax(test_mask, axis=3)
test_img_input = np.expand_dims(test_img, axis=0)
test_prediction = model.predict(test_img_input)
test_prediction_argmax=np.argmax(test_prediction, axis=4)[0,:,:,:]

print(test_prediction_argmax.shape)
print(test_mask_argmax.shape)
print(np.unique(test_prediction_argmax))


#Plot individual slices from test predictions for verification
from matplotlib import pyplot as plt
import random

#n_slice=random.randint(0, test_prediction_argmax.shape[2])
n_slice = 55
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,n_slice,1], cmap='gray')
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(test_mask_argmax[:,:,n_slice])
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(test_prediction_argmax[:,:, n_slice])
plt.show()

###########################################################
```

Hình 23: Dự đoán một ảnh trên tập Validation.

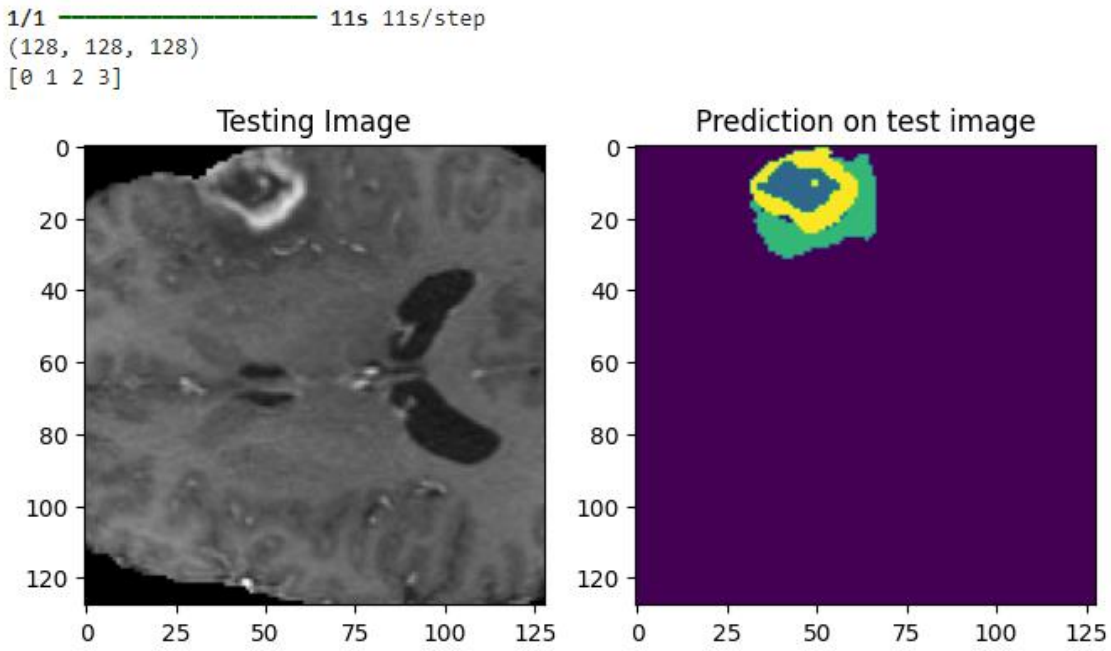
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 13s 13s/step
(128, 128, 128)
(128, 128, 128)
[0 1 2 3]
```



Hình 24: Kết quả trên tập Validation

```
##############################################
# Dự đoán duy nhất 1 lần trên ảnh test
img_num = 1

test_img = np.load("/content/data/BraTS2020_ValidationData/input_data_3channels/images/image_"+str(img_num)+".npy")


test_img_input = np.expand_dims(test_img, axis=0)
test_prediction = model.predict(test_img_input)
test_prediction_argmax=np.argmax(test_prediction, axis=4)[0,:,:,:]


print(test_prediction_argmax.shape)

print(np.unique(test_prediction_argmax))


#Plot individual slices from test predictions for verification
from matplotlib import pyplot as plt
import random

#n_slice=random.randint(0, test_prediction_argmax.shape[2])
n_slice = 55
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,n_slice,1], cmap='gray')
plt.subplot(232)
plt.title('Testing Label')
plt.title('Prediction on test image')
plt.imshow(test_prediction_argmax[:,:, n_slice])
plt.show()

##############################################################
```

Hình 25 Dự đoán 1 lần duy nhất trên ảnh test



Hình 26: Kết quả trên ảnh test

4.5. Hạn chế

1. Overfitting:

Mất mát cao trên tập huấn luyện và validation: Mặc dù độ chính xác cao, nhưng mất mát vẫn khá cao, điều này có thể do overfitting. Mô hình có thể học quá tốt trên tập huấn luyện nhưng không tổng quát hóa tốt trên dữ liệu mới.

2. Độ chính xác trên tập validation và IoU vẫn còn thấp so với tập huấn luyện:

Chênh lệch giữa tập huấn luyện và validation: Độ chính xác và IoU trên tập validation thấp hơn so với tập huấn luyện, cho thấy mô hình có thể vẫn chưa học được tất cả các đặc tính từ dữ liệu.

3. Chưa tối ưu hóa về mặt computational cost:

Thời gian suy luận và tài nguyên tính toán: Mặc dù thời gian suy luận nhanh (0.5 giây mỗi ảnh 3D), mô hình vẫn có thể được tối ưu hóa hơn nữa về tài nguyên tính toán để giảm thiểu chi phí và tăng tốc độ.

V. Kết luận và hướng phát triển

5.1. Kết luận

Trong quá trình thực hiện đề tài, chúng tôi đã phát triển và huấn luyện thành công mô hình U-Net 3D để phân đoạn các tiểu vùng bên trong khối u não từ các ảnh MRI. Các kết quả cho thấy mô hình có khả năng phân đoạn khá chính xác với các chỉ số đo lường ở mức cao sau 100 epoch huấn luyện: độ chính xác (accuracy) đạt 0.9991, chỉ số IoU (Intersection over Union) đạt 0.7931, và loss giảm xuống 0.7843. Các chỉ số trên tập kiểm tra cũng cho thấy kết quả khả quan với độ chính xác đạt 0.9812 và IoU đạt 0. 6594. Những nhận xét ở phần kết quả đã phần nào chứng minh những tiêu chí về yêu cầu thiết kế đã phần nào đạt được, cụ thể:

1. Độ chính xác vừa phải nhưng đáng tin cậy:

-Minh họa và giải thích: Mô hình không cần phải đạt được độ chính xác tối đa nhưng cần đủ tốt để minh họa các đặc điểm quan trọng của khối u.

Với độ chính xác và IoU đạt được, mô hình đã chứng minh khả năng phân đoạn các đặc điểm quan trọng của khối u não một cách rõ ràng.

-Hình ảnh minh họa: Ở phần kết quả định tính (hình 25), mô hình đã hoàn toàn đạt được độ chính xác đủ tốt để minh họa các đặc điểm quan trọng của khối u. Ba vùng quan trọng của khối u được phân biệt rõ ràng trên ảnh dự đoán từ tập test, thể hiện sự phân đoạn chính xác và rõ nét, đáp ứng yêu cầu thiết kế về độ tin cậy và khả năng minh họa.

2. Tốc độ suy luận ở mức độ hợp lý:

- Thời gian suy luận hợp lý: Mô hình cũng đã thỏa mãn được tiêu chí về tốc độ suy luận hợp lý. Thời gian suy luận trên mỗi ảnh thông qua mô hình chỉ mất từ 10 đến 15 giây (hình 25). Thời gian này là đủ hợp lý để thỏa mãn tiêu chí về tốc độ suy luận của mô hình, đảm bảo rằng các buổi hướng dẫn và thảo luận diễn ra mượt mà mà không bị gián đoạn bởi thời gian chờ đợi dài.

Kết luận: Mô hình U-Net 3D mà chúng tôi phát triển đã đáp ứng đầy đủ các yêu cầu thiết kế ban đầu. Độ chính xác vừa phải nhưng đáng tin cậy, cùng với tốc độ suy luận hợp lý, làm cho mô hình trở thành một công cụ hữu ích trong việc phân đoạn các tiểu vùng bên trong khối u não từ ảnh MRI. Điều này không chỉ hỗ trợ trong việc chẩn đoán và điều trị y học mà còn trong các hoạt động giảng dạy và nghiên cứu.

5.2. Hướng phát triển của đề tài

Dựa trên các hạn chế đã được xác định trong quá trình thực hiện, chúng tôi đề xuất các hướng phát triển sau đây để hoàn thiện và mở rộng đề tài:

1. Tích hợp hệ thống và phát triển giao diện người dùng:

Phát triển giao diện người dùng: Tạo ra một giao diện người dùng thân thiện, giúp các chuyên gia y tế dễ dàng sử dụng mô hình để phân đoạn ảnh MRI. Giao diện này cần bao gồm các tính năng như tải ảnh, hiển thị kết quả phân đoạn, và cung cấp các tùy chọn để tinh chỉnh và đánh giá kết quả.

Tích hợp vào hệ thống y tế hiện có: Để mô hình có thể được sử dụng trong các quy trình chẩn đoán và điều trị hiện tại, cần phát triển các phương pháp tích hợp mô hình vào các hệ thống y tế hiện có. Điều này bao gồm việc đảm bảo tính tương thích với các hệ thống quản lý dữ liệu y tế (EMR/EHR).

2. Tối ưu hóa mô hình cho các thiết bị phần cứng cụ thể:

Tối ưu hóa hiệu suất trên các thiết bị GPU: Phát triển và tinh chỉnh mô hình để chạy hiệu quả trên các thiết bị phần cứng cụ thể như máy tính cá nhân với GPU hoặc các hệ thống máy chủ mạnh mẽ, đảm bảo rằng mô hình có thể hoạt động hiệu quả và nhanh chóng trong môi trường thực tế.

Phát triển phiên bản nhẹ cho các thiết bị di động: Để mở rộng phạm vi ứng dụng, nghiên cứu và phát triển phiên bản mô hình nhẹ có thể chạy trên các thiết bị di động hoặc các thiết bị phần cứng hạn chế khác, giúp các khu vực hẻo lánh hoặc các cơ sở y tế có điều kiện hạn chế cũng có thể sử dụng được.

3. Nâng cao chất lượng và độ chính xác của mô hình:

Cải tiến thuật toán và kỹ thuật huấn luyện: Áp dụng các thuật toán và kỹ thuật huấn luyện tiên tiến hơn để nâng cao chất lượng và độ chính xác của mô hình, như sử

dụng các phương pháp học sâu mới nhất hoặc kết hợp với các mô hình khác để cải thiện hiệu quả phân đoạn.

Mở rộng bộ dữ liệu huấn luyện: Liên tục cập nhật và mở rộng bộ dữ liệu huấn luyện với các trường hợp mới và đa dạng, bao gồm cả các biến thể khác nhau của khối u não, để mô hình có khả năng tổng quát hóa tốt hơn.

4. Mở rộng quy mô thử nghiệm:

Triển khai trên quy mô lớn: Để xác nhận tính khả thi và hiệu quả của mô hình trên nhiều trường hợp khác nhau, cần triển khai và thử nghiệm mô hình trên quy mô lớn hơn. Điều này sẽ bao gồm việc thu thập và huấn luyện mô hình trên một lượng lớn dữ liệu từ nhiều nguồn khác nhau, đảm bảo tính đa dạng và toàn diện của dữ liệu.

5. Tăng cường khả năng tự động hóa và thông minh của hệ thống:

Áp dụng học chuyển giao (Transfer Learning): Sử dụng các mô hình học sâu đã được huấn luyện trước trên các tập dữ liệu lớn và chuyển giao các kiến thức này sang mô hình của chúng ta để tăng cường hiệu quả huấn luyện và giảm thời gian huấn luyện.

Phát triển các phương pháp học không giám sát (Unsupervised Learning): Điều tra và phát triển các phương pháp học không giám sát để mô hình có thể học hỏi từ dữ liệu chưa được gán nhãn, mở rộng khả năng áp dụng mô hình trong các tình huống thiếu dữ liệu được gán nhãn chi tiết.

Tài liệu tham khảo

[1] Shukor Bin Abd Razak, Ahmad Fahrulrazie Bin Mohamad “Identification of Spam Email Based on Information from Email Header” 13th International Conference on Intelligent SystemsDesign and Applications (ISDA), 2013.

[2] Mohammed Reza Parsei, Mohammed Salehi “E-Mail Spam Detection Based on Part of Speech Tagging” 2nd International Conference on Knowledge Based Engineering and Innovation (KBEI), 2015.

[3] Sunil B. Rathod, Tareek M. Pattewar “Content Based Spam Detection in Email using Bayesian Classifier”, presented at the IEEE ICCSP 2015 conference.

[4] Aakash Atul Alurkar, Sourabh Bharat Ranade, Shreeya Vijay Joshi, Siddhesh Sanjay Ranade, Piyush A. Sonewa, Parikshit N. Mahalle, Arvind V. Deshpande “A Proposed Data Science Approach for Email Spam Classification using Machine Learning Techniques”, 2017.

[5] Kriti Agarwal, Tarun Kumar “Email Spam Detection using integrated approach of Naïve Bayes and Particle Swarm Optimization”, Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018.