TRƯỜNG ĐẠI HỌC SÀI GÒN
CÔNG NGHỆ THÔNG TIN

o0o

Đồ án môn học
Phát triển phần mềm mã nguồn mở
Đề tài phát triển phần mềm chẩn đoán ung thư phổi bằng hình ảnh

Giáo viên hướng dẫn: Từ Lãng Phiêu

Sinh viên thực hiện : Nguyễn Tư Nghĩa - 3121560060
Nguyễn Quốc Anh - 3121560010

Email : tunghia98@gmail.com

Thành phố Hồ Chí Minh - Tháng 5 Năm 2024

Mục lục

Lời cảm ơn

Chúng em xin cảm ơn thầy Từ Lãng Phiêu vì những giờ giảng dạy đầy nhiệt tình. Nếu như trong báo cáo còn có những thiếu sót, kính mong thầy và các bạn nhắc nhở để chúng em có thể cải thiện đồ án hơn.

Công việc của từng thành viên trong nhóm:

- Nguyễn Tư Nghĩa viết phần xử lý và viết báo cáo cho đồ án
- Nguyễn Quốc Anh viết giao diện cho đồ án

Chương 1

Giới thiệu

1.1 Giới thiệu đồ án

Trong thế kỷ 21, y tế số đã trở thành một lĩnh vực quan trọng và phát triển mạnh mẽ. Công nghệ thông tin và trí tuệ nhân tạo đã đóng vai trò quan trọng trong việc cải thiện chăm sóc sức khỏe và chẩn đoán bệnh. Trong ngành y tế, việc sử dụng hình ảnh y khoa, như hình ảnh CT phổi, đã trở nên phổ biến để hỗ trợ các bác sĩ trong việc đưa ra quyết định chẩn đoán và điều trị.

Đồ án này nhằm mục đích phát triển một ứng dụng chẩn đoán hình ảnh CT phổi bằng trí tuệ nhân tạo. Ứng dụng sẽ sử dụng một mô hình deep learning để dự đoán loại bệnh từ hình ảnh CT phổi, giúp giảm thời gian và công sức của các chuyên gia y tế trong quá trình chẩn đoán.

1.2 Mục tiêu đồ án

Mục tiêu của đồ án là phát triển một ứng dụng chẩn đoán hình ảnh CT phổi có khả năng dự đoán loại bệnh có thể có từ hình ảnh này. Cụ thể, các mục tiêu cụ thể bao gồm:

- Xây dựng một mô hình deep learning dựa trên kiến trúc EfficientNet để dự đoán loại bệnh từ hình ảnh CT phổi.
- Phát triển một giao diện người dùng trực quan và dễ sử dụng để người dùng có thể tải lên hình ảnh CT và xem kết quả dự đoán.
- Tối ưu hóa hiệu suất và độ chính xác của mô hình thông qua việc huấn luyện với tập dữ liệu lớn và tinh chỉnh siêu tham số.

1.3 Ý nghĩa đồ án

Đồ án này mang lại nhiều ý nghĩa quan trọng như sau:

- Nâng cao chất lượng chăm sóc sức khỏe: Ứng dụng có thể giúp các bác sĩ và chuyên gia y tế có được các dự đoán chính xác và nhanh chóng từ hình ảnh CT phổi, từ đó cải thiện chất lượng chăm sóc sức khỏe cho bệnh nhân.

- Tiết kiệm thời gian và chi phí: Việc sử dụng mô hình deep learning để tự động chẩn đoán hình ảnh CT có thể giảm thời gian và chi phí so với việc phải thực hiện quá trình này bằng tay bởi các chuyên gia y tế.

- Áp dụng thực tế: Đồ án đem lại giá trị thực tiễn khi có thể được triển khai và sử dụng trong các cơ sở y tế để hỗ trợ quyết định chẩn đoán và điều trị.

Tổng quan, đồ án này mang lại những đóng góp quan trọng trong việc ứng dụng trí tuệ nhân tạo vào lĩnh vực y tế, từ đó cải thiện chăm sóc sức khỏe và tiết kiệm tài nguyên y tế.

Chương 2

Cơ sở lý thuyết

2.1 Thư viện Torch

Giới thiệu

- Torch (hay PyTorch) là một thư viện mã nguồn mở được viết bằng Python cho tính toán khoa học và học máy, đặc biệt tập trung vào học sâu. Nó được phát triển bởi Facebook và được cộng đồng các nhà nghiên cứu và phát triển tích cực hỗ trợ.

Ưu điểm:

- **Dễ học và sử dụng:** Torch có cú pháp Python đơn giản và trực quan, dễ học và sử dụng cho người mới bắt đầu.
- **Linh hoạt:** Torch hỗ trợ nhiều mô hình học sâu khác nhau, bao gồm mạng nơ-ron tích chập (CNN), mạng nơ-ron hồi quy (RNN), và máy biến áp (transformer).
- **Hiệu suất cao:** Torch tận dụng tối đa phần cứng GPU để tăng tốc độ tính toán, giúp huấn luyện mô hình nhanh hơn.
- **Cộng đồng lớn:** Torch có cộng đồng người dùng và nhà phát triển lớn, cung cấp nhiều tài nguyên hữu ích, hướng dẫn và thư viện hỗ trợ.

Ứng dụng của Torch:

- **Nhận dạng hình ảnh:** Phát hiện và phân loại đối tượng trong hình ảnh, như khuôn mặt, biển báo giao thông, v.v.
- **Xử lý ngôn ngữ tự nhiên:** Phân tích văn bản, dịch ngôn ngữ, tóm tắt văn bản, v.v.
- **Học tăng cường:** Huấn luyện các tác nhân học cách thực hiện các hành động để tối đa hóa phần thưởng trong môi trường phức tạp.

- **Tạo văn bản:** Tạo văn bản sáng tạo, như thơ, nhạc, kịch bản, v.v.
- **Phân tích dữ liệu:** Phân tích và khám phá dữ liệu, dự đoán xu hướng, v.v.

Các thành phần chính của Torch:

- **Tensors:** Cấu trúc dữ liệu đa chiều tương tự như ma trận trong NumPy, lưu trữ dữ liệu cho các phép tính học máy.
- **Mạng nơ-ron:** Các mô hình học máy được xây dựng bằng các lớp nơ-ron được kết nối với nhau, học cách thực hiện các nhiệm vụ từ dữ liệu.
- **Chức năng tổn thất:** Đo lường độ sai lệch giữa dự đoán của mô hình và dữ liệu thực tế.
- **Trình tối ưu hóa:** Cập nhật tham số mô hình để giảm thiểu hàm mất và cải thiện hiệu suất mô hình.
- **Dataloader:** Tải và xử lý dữ liệu cho mô hình huấn luyện và đánh giá.

Kết luận: Torch là một công cụ mạnh mẽ và linh hoạt cho học máy và học sâu. Với cú pháp dễ sử dụng, hiệu suất cao và cộng đồng lớn, Torch là lựa chọn phổ biến cho các nhà nghiên cứu, nhà phát triển và những người đam mê học máy.

2.2 Thư viện timm (Transformers and Image Models - Models)

Giới thiệu

- timm là một thư viện mã nguồn mở được viết bằng Python, cung cấp một bộ sưu tập phong phú các mô hình học sâu được đào tạo trước cho các nhiệm vụ xử lý ảnh và ngôn ngữ tự nhiên. Thư viện này được phát triển bởi Ross Wightman và cộng đồng, được đánh giá cao về hiệu suất cao, tính linh hoạt và khả năng dễ dàng tích hợp vào các dự án học sâu hiện có.

Ưu điểm của timm:

- **Đa dạng mô hình:** timm cung cấp nhiều mô hình học sâu tiên tiến cho các nhiệm vụ như phân loại ảnh, phân tích điểm ảnh, nhận dạng đối tượng, xử lý ngôn ngữ tự nhiên (NLP), v.v.
- **Hiệu suất cao:** Các mô hình trong timm được đào tạo trên tập dữ liệu khổng lồ, đạt được độ chính xác cao và hiệu suất vượt trội trên nhiều bài toán.
- **Dễ sử dụng:** timm có API đơn giản và trực quan, giúp bạn dễ dàng tích hợp các mô hình vào dự án của mình.
- **Linh hoạt:** timm hỗ trợ nhiều khung học sâu phổ biến như PyTorch, TensorFlow, JAX, v.v., cho phép bạn sử dụng các mô hình timm với nhiều nền tảng.
- **Cộng đồng tích cực:** timm có cộng đồng người dùng và nhà phát triển sôi động, luôn hỗ trợ lẫn nhau và chia sẻ kiến thức.

Ứng dụng của timm:

- Phân loại ảnh: Phân loại hình ảnh vào các lớp khác nhau, như động vật, hoa, đồ vật, v.v.
- Phân tích điểm ảnh: Phân tích và trích xuất thông tin từ hình ảnh, như xác định vị trí đối tượng, tính toán độ sâu, v.v.
- Nhận dạng đối tượng: Phát hiện và nhận dạng các đối tượng trong hình ảnh, như khuôn mặt, biển báo giao thông, logo, v.v.
- Xử lý ngôn ngữ tự nhiên (NLP): Xử lý và phân tích văn bản, như dịch máy, tóm tắt văn bản, phân tích tình cảm, v.v.
- Tạo văn bản: Tạo văn bản sáng tạo, như thơ, nhạc, kịch bản, v.v.

2.3 Thư viện sklearn.metrics

Giới thiệu Thư viện sklearn.metrics là một phần quan trọng của scikit-learn (sklearn), cung cấp nhiều hàm và công cụ để đánh giá hiệu suất của các mô hình học máy. Nó bao gồm các hàm để tính toán các chỉ số hiệu suất phổ biến như độ chính xác, độ chính xác, độ thu hồi, F1-score, ROC AUC và nhiều hơn nữa. **Ưu điểm của sklearn.metrics:**

- **Dễ sử dụng:** Thư viện cung cấp API đơn giản và trực quan, dễ dàng sử dụng cho người mới bắt đầu.
- **Đa dạng:** Hỗ trợ nhiều chỉ số hiệu suất cho nhiều loại mô hình học máy khác nhau.
- **Linh hoạt:** Cho phép bạn tùy chỉnh cách tính toán các chỉ số hiệu suất để phù hợp với nhu cầu cụ thể của bạn.
- **Tích hợp với sklearn:** Hoạt động liền mạch với các mô hình và dữ liệu sklearn khác.
- **Cộng đồng lớn:** Được hỗ trợ bởi cộng đồng sklearn rộng lớn, cung cấp tài nguyên và hướng dẫn phong phú.

Một số chỉ số hiệu suất phổ biến trong sklearn.metrics:

- Độ chính xác (Accuracy): Tỷ lệ dự đoán chính xác của mô hình trên tất cả các trường hợp.
- Độ chính xác (Precision): Tỷ lệ dự đoán dương tính chính xác của mô hình.
- Độ thu hồi (Recall): Tỷ lệ các trường hợp dương tính thực sự được mô hình dự đoán chính xác.
- F1-score: Trung bình hài hòa giữa độ chính xác và độ thu hồi.
- ROC AUC (Area Under the ROC Curve): Đo lường khả năng phân biệt giữa các lớp của mô hình.

2.4 Thư viện Numpy

NumPy là một thư viện mã nguồn mở được viết bằng Python dành cho tính toán khoa học. Nó cung cấp các cấu trúc dữ liệu hiệu suất cao để lưu trữ mảng đa chiều, cùng với nhiều hàm toán học và thao tác dữ liệu mạnh mẽ. NumPy là thư viện nền tảng cho nhiều thư viện học máy và khoa học dữ liệu phổ biến khác trong Python như Pandas, scikit-learn, Matplotlib, v.v.

Ưu điểm của NumPy:

- **Hiệu suất cao:** NumPy sử dụng các thuật toán được tối ưu hóa cho CPU và GPU, giúp tăng tốc độ tính toán cho các phép toán mảng.
- **Dễ sử dụng:** NumPy cung cấp API đơn giản và trực quan, dễ dàng học và sử dụng cho người mới bắt đầu.
- **Linh hoạt:** Hỗ trợ nhiều loại dữ liệu đa chiều, bao gồm mảng một chiều, hai chiều, N chiều.
- **Đa dạng:** Cung cấp nhiều hàm toán học và thao tác dữ liệu cho nhiều mục đích khác nhau như thống kê, đại số tuyến tính, xử lý tín hiệu, v.v.
- **Tích hợp với các thư viện khác:** Hoạt động liền mạch với các thư viện Python phổ biến khác như Pandas, scikit-learn, Matplotlib.

Các thành phần chính của NumPy:

- **Mảng NumPy:** Cấu trúc dữ liệu đa chiều hiệu suất cao để lưu trữ dữ liệu số.
- **Hàm toán học:** Cung cấp nhiều hàm toán học cho các phép toán cơ bản (cộng, trừ, nhân, chia), hàm lượng giác, hàm thống kê, v.v.
- **Thao tác dữ liệu:** Các hàm để thao tác với dữ liệu mảng như sắp xếp, lọc, trích xuất, v.v.
- **Linear algebra:** Cung cấp các hàm cho các phép toán đại số tuyến tính như giải phương trình tuyến tính, tính toán ma trận nghịch đảo, v.v.

Ngoài ra, NumPy còn có nhiều ứng dụng khác như xử lý ảnh, học máy, phân tích dữ liệu, mô phỏng, v.v.

2.5 Thư viện PIL

PPIL là một thư viện mã nguồn mở được viết bằng Python dành cho xử lý ảnh. Nó cung cấp nhiều công cụ và chức năng để thao tác với hình ảnh như mở, lưu, chỉnh sửa, chuyển đổi định dạng, v.v. PIL được sử dụng rộng rãi trong nhiều lĩnh vực như đồ họa, khoa học máy tính, học máy, xử lý ảnh kỹ thuật số, v.v.

Ưu điểm của PIL:

- **Dễ sử dụng:** PIL cung cấp API đơn giản và trực quan, dễ dàng học và sử dụng cho người mới bắt đầu.
- **Linh hoạt:** Hỗ trợ nhiều định dạng hình ảnh phổ biến như JPEG, PNG, GIF, BMP, TIFF, v.v.
- **Đa dạng:** Cung cấp nhiều chức năng để thao tác với hình ảnh như cắt, xoay, thay đổi kích thước, điều chỉnh độ sáng, độ tương phản, v.v.
- **Tích hợp với các thư viện khác:** Hoạt động liền mạch với các thư viện Python phổ biến khác như NumPy, Matplotlib, scikit-image.
- **Cộng đồng lớn:** Được hỗ trợ bởi cộng đồng người dùng và nhà phát triển PIL lớn, cung cấp nhiều tài nguyên và hướng dẫn hữu ích.

Một số chức năng chính của PIL:

- Mở và lưu hình ảnh: PIL có thể mở hình ảnh từ tệp tin hoặc URL và lưu hình ảnh ở nhiều định dạng khác nhau.
- Chỉnh sửa hình ảnh: PIL cung cấp nhiều chức năng để chỉnh sửa hình ảnh như cắt, xoay, thay đổi kích thước, điều chỉnh độ sáng, độ tương phản, v.v.
- Chuyển đổi định dạng: PIL có thể chuyển đổi hình ảnh giữa nhiều định dạng khác nhau như JPEG, PNG, GIF, BMP, TIFF, v.v.
- Thêm hiệu ứng: PIL cung cấp nhiều hiệu ứng cho hình ảnh như làm mờ, thêm viền, tạo bóng đổ, v.v.
- Lọc ảnh: PIL cung cấp nhiều bộ lọc cho ảnh như lọc trung bình, lọc Gaussian, lọc Sobel, v.v.

2.6 Framework FastAPI

FastAPI là một framework web hiện đại, hiệu suất cao được xây dựng cho Python 3.6 trở lên, tập trung vào việc tạo API RESTful. Nó được thiết kế để dễ sử dụng, nhanh chóng và có thể mở rộng, giúp các nhà phát triển xây dựng API một cách nhanh chóng và hiệu quả.

Ưu điểm chính của FastAPI:

- **Hiệu suất cao:** FastAPI được tối ưu hóa cho tốc độ và hiệu suất, sử dụng Asynchronous Server Gateway Interface (ASGI) để xử lý các yêu cầu phi đồng bộ. Nhờ vậy, nó có thể xử lý nhiều yêu cầu đồng thời với độ trễ thấp.
- **Dễ sử dụng:** FastAPI sử dụng cú pháp Python đơn giản và trực quan, giúp bạn dễ dàng tạo API mà không cần nhiều boilerplate code.
- **Tự động hóa tài liệu:** FastAPI tự động tạo tài liệu API dựa trên mã của bạn, giúp người dùng dễ dàng hiểu và sử dụng API của bạn.
- **Hỗ trợ schema mạnh mẽ:** FastAPI hỗ trợ định nghĩa schema JSON cho các request và response, giúp đảm bảo tính chính xác và nhất quán của dữ liệu.
- **Linh hoạt:** FastAPI có thể được sử dụng để xây dựng nhiều loại API khác nhau, từ API RESTful đơn giản đến các ứng dụng web phức tạp.
- **Cộng đồng phát triển tích cực:** FastAPI có cộng đồng người dùng và nhà phát triển sôi động, luôn hỗ trợ lẫn nhau và chia sẻ kiến thức.

Chương 3

Thiết kế ứng dụng

3.1 Mô tả thiết kế

Ứng dụng "chẩn đoán ung thư phổi bằng hình ảnh" được thiết kế để chấp nhận hình ảnh CT phổi từ người dùng và trả về dự đoán về loại bệnh có thể có, chẳng hạn như ung thư phổi hoặc các vấn đề khác. **Mô-đun chính:**

- **FastAPI Backend Server:** Sẽ là trung tâm của ứng dụng, xử lý các yêu cầu từ phía người dùng và gửi chúng đến mô hình deep learning để dự đoán. FastAPI được chọn vì khả năng xử lý yêu cầu nhanh chóng và dễ dàng triển khai.
- **Deep Learning Model:** Mô hình deep learning sẽ được sử dụng để dự đoán loại bệnh từ hình ảnh CT phổi. Trong thiết kế này, chúng ta sử dụng một mô hình được huấn luyện trước sử dụng kiến trúc EfficientNet và sẽ được tinh chỉnh lại với dữ liệu của chúng ta.
- **Trang Web Giao diện người dùng (UI):** Giao diện người dùng sẽ cho phép người dùng tải lên hình ảnh CT phổi và hiển thị kết quả dự đoán từ mô hình.

Luồng làm việc:

1. **Người dùng tải lên hình ảnh CT phổi:** Người dùng truy cập trang web và tải lên hình ảnh CT phổi từ máy tính hoặc thiết bị của họ.
2. **FastAPI Server xử lý yêu cầu:** FastAPI nhận hình ảnh CT từ người dùng và chuyển nó đến mô hình deep learning để dự đoán.
3. **Mô hình dự đoán loại bệnh:** Mô hình deep learning sẽ xử lý hình ảnh CT và dự đoán loại bệnh có thể có.
4. **Kết quả trả về cho người dùng:** Kết quả của dự đoán sẽ được trả về cho người dùng thông qua giao diện người dùng, hiển thị loại bệnh được dự đoán từ hình ảnh CT.

Công nghệ sử dụng:

- **FastAPI:** Sử dụng để tạo và triển khai backend server nhanh chóng và dễ dàng.
- **PyTorch và Timm:** Sử dụng PyTorch để xây dựng và huấn luyện mô hình deep learning, và sử dụng Timm để sử dụng kiến trúc mô hình EfficientNet.
- **HTML/CSS/JavaScript:** Sử dụng để xây dựng giao diện người dùng trực quan và thân thiện với người dùng.

Tính năng dự kiến:

- **Tải lên hình ảnh CT:** Người dùng có thể tải lên hình ảnh CT phổi từ máy tính hoặc thiết bị của họ.
- **Hiển thị kết quả dự đoán:** Kết quả của dự đoán sẽ được hiển thị cho người dùng, cho họ biết loại bệnh có thể có từ hình ảnh CT.

3.2 Cấu trúc mã nguồn

- **dataset:** Thư mục chứa dữ liệu cho việc huấn luyện và kiểm tra mô hình.
- **data loader:** Thư mục chứa các hàm data loader tự định nghĩa.
- **train.py:** File chứa mã nguồn cho việc huấn luyện mô hình.
- **evaluate.py:** File chứa mã nguồn cho việc kiểm tra và đánh giá mô hình.
- **app.py:** File chính của ứng dụng FastAPI, nơi mà các endpoint được định nghĩa và xử lý.
- **index.html:** Tệp HTML chứa cấu trúc và giao diện của trang web.
- **index.css:** Mã CSS để tạo giao diện người dùng trực quan và thân thiện.
- **index.js:** Mã JavaScript để xử lý tải lên hình ảnh và hiển thị kết quả dự đoán.
- **requirements.txt :** Danh sách các thư viện Python cần thiết và phiên bản tương ứng để triển khai ứng dụng.

3.3 Mô hình ứng dụng

3.3.1 Mô hình CNN

- Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.
- CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày).
- Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

3.3.2 Mô hình EfficientNet B3

EfficientNet B3 là một kiến trúc mạng nơ-ron được thiết kế để đạt hiệu suất cao trong các tác vụ nhận dạng hình ảnh với chi phí tính toán thấp. Nó thuộc họ EfficientNet, một tập hợp các mô hình được phát triển bởi Google AI với mục tiêu tối ưu hóa hiệu quả giữa độ chính xác và tốc độ xử lý.

Đặc điểm chính của EfficientNet B3 bao gồm:

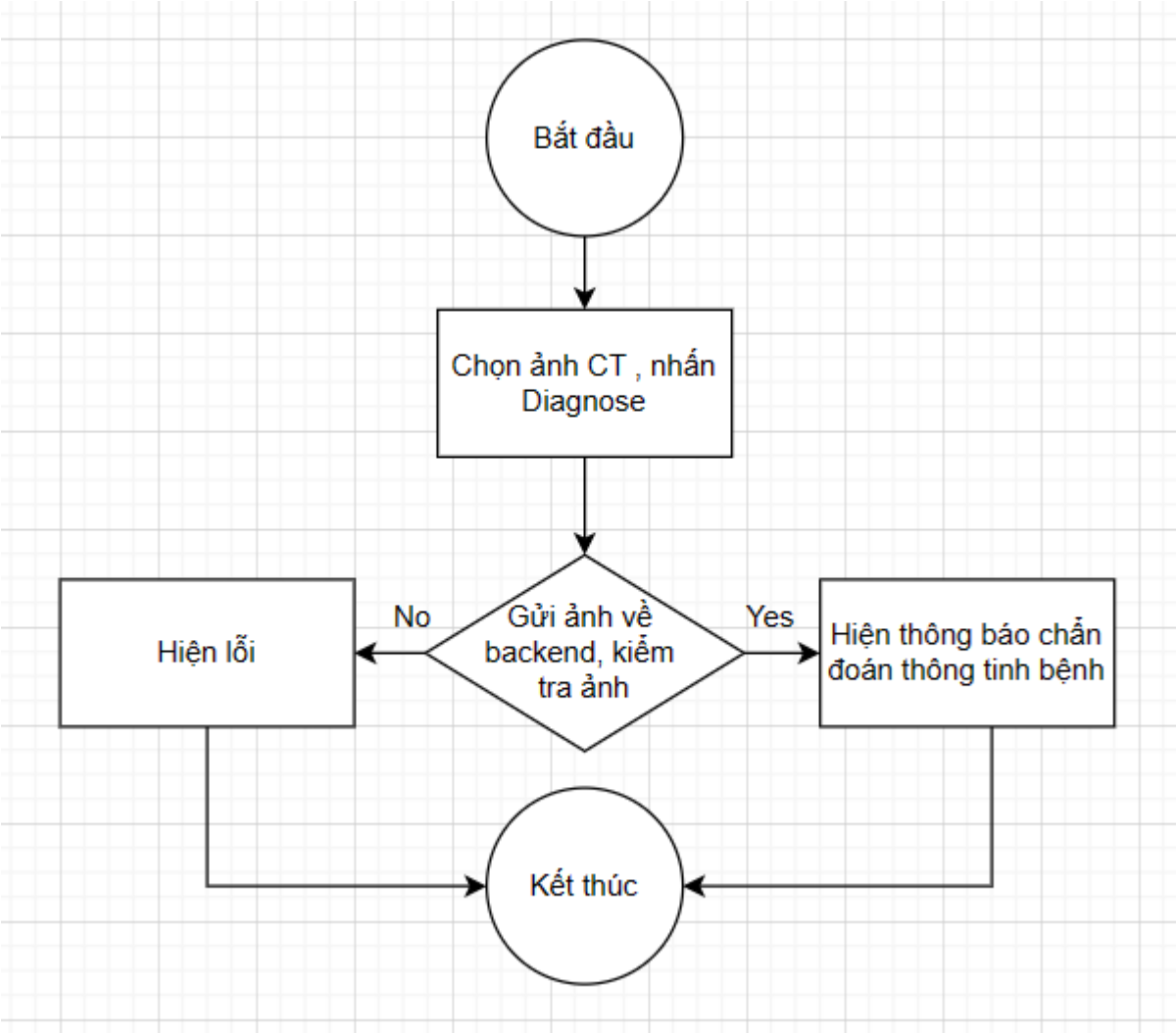
- Sử dụng kiến trúc dựa trên việc tìm kiếm tự động (NAS), giúp tự động tìm kiếm cấu trúc mạng tối ưu cho các nhiệm vụ cụ thể.
- Áp dụng kỹ thuật nén mô hình như compound scaling và squeeze-excitation để giảm kích thước mô hình và chi phí tính toán mà vẫn duy trì hiệu suất cao.
- Có thể được tinh chỉnh cho nhiều tác vụ nhận dạng hình ảnh khác nhau như phân loại, phát hiện đối tượng và phân đoạn ảnh.

EfficientNet B3 đã đạt được hiệu suất ấn tượng trong các bài kiểm tra chuẩn về nhận dạng hình ảnh. Ví dụ, trên tập dữ liệu ImageNet, EfficientNet B3 đạt được độ chính xác Top-1 là 84.0

Ưu điểm của EfficientNet B3 bao gồm hiệu suất cao, tốc độ xử lý nhanh và kích thước mô hình nhỏ. Tuy nhiên, điều này cũng đi kèm với nhược điểm như độ phức tạp cao và yêu cầu dữ liệu đào tạo lớn.

EfficientNet B3 có thể được sử dụng cho nhiều ứng dụng nhận dạng hình ảnh như phân loại, phát hiện đối tượng, phân đoạn ảnh và chỉnh sửa ảnh.

3.4 Lưu đồ luồng dữ liệu

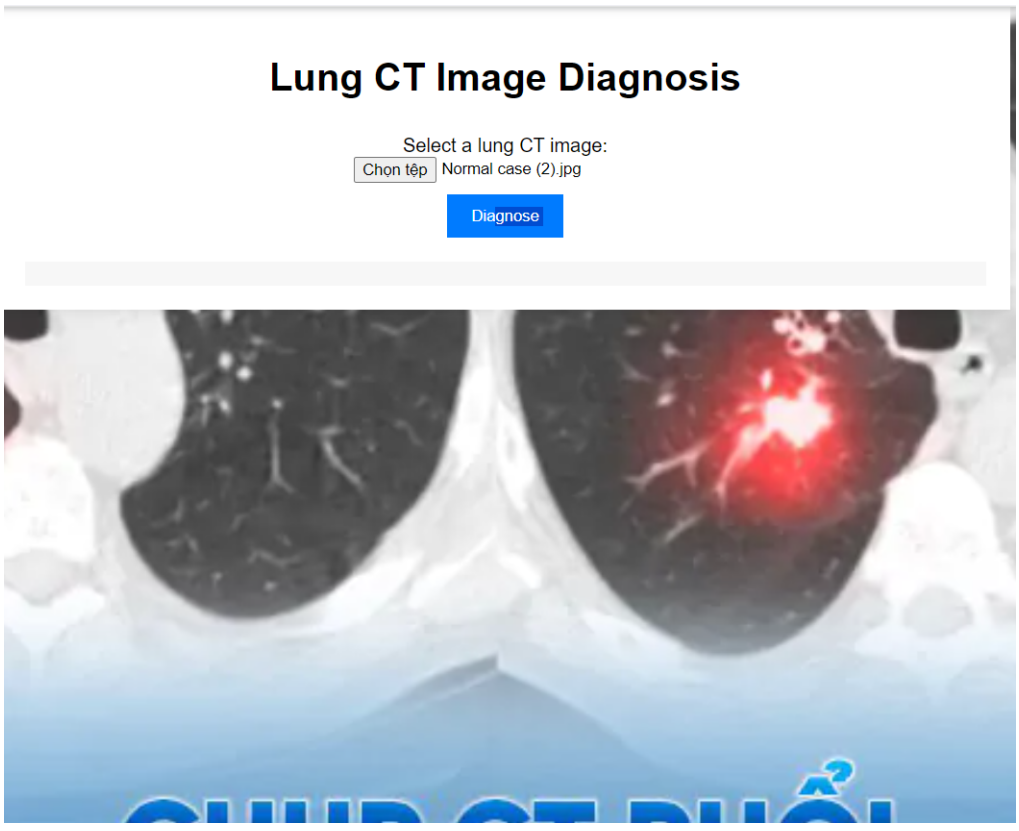


Hình 3.1: Lưu đồ luồng dữ liệu dự đoán ung thư phổi bằng ảnh CT

Chương 4

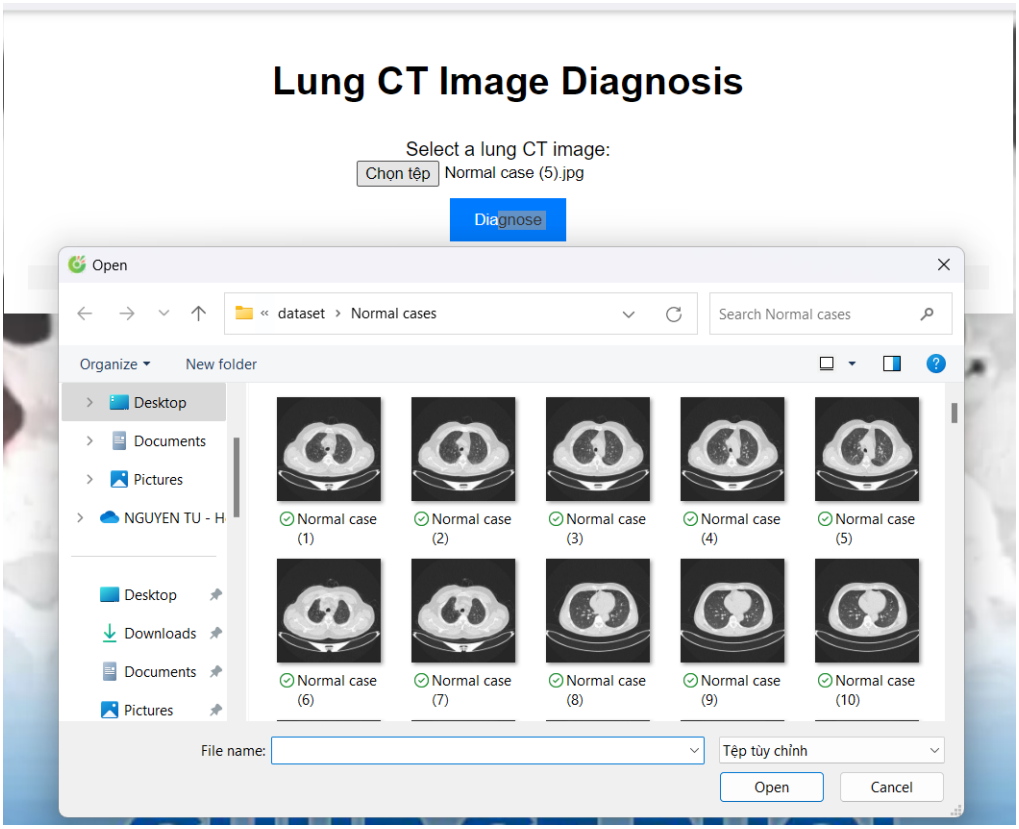
Hiện thực

4.1 Trang chủ



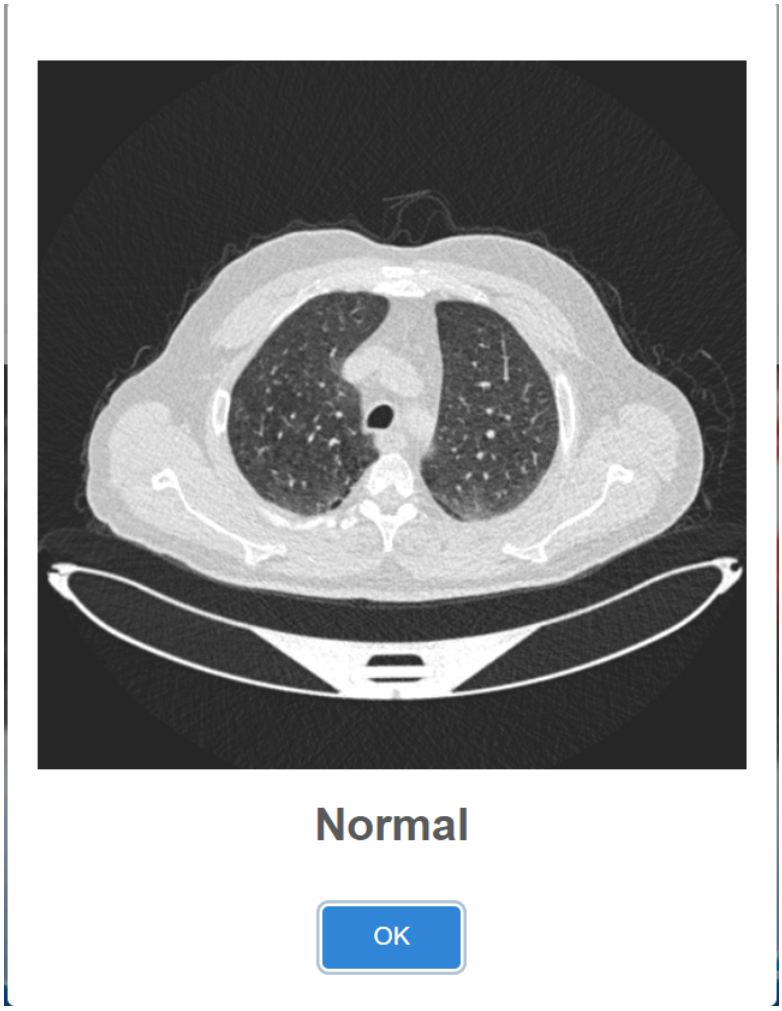
Hình 4.1: Màn hình trang chủ

4.2 Chọn file ảnh CT



Hình 4.2.1: Màn hình chọn ảnh CT

4.3 Hiện kết quả



Hình 4.3: Màn hình thông báo kết quả

Chương 5

Cách thức cài đặt ứng dụng, môi trường chạy ứng dụng

5.1 Yêu cầu

- torch
- timm
- fastapi
- uvicorn[standard]
- scikit-learn
- python-multipart

5.2 Cách thực hiện

- Mở command prompt hoặc power shell trong máy tính
- Tải bản zip hoặc tạo bản sao để bắt đầu làm việc với dự án
  git clone https://github.com/tunghia98/Lung_Cancer.git
- Trỏ đường dẫn tới thư mục
  cd Lung_Cancer
- Tạo và cài đặt môi trường python3
  python3 -m venv env source venv/bin/activate

- Cài đặt các thư viện chưa có

  pip install -r requirements.txt

- Dowload dataset

  Tải xuống tập dữ liệu từ liên kết sau:`https://www.kaggle.com/datasets/antonixx/the-iqothnccd-lung-cancer-dataset/data` và đặt nó vào thư mục 'dataset' theo cấu trúc này dataset

  - Bengin cases
  - Malignant cases
  - Normal case
  - split_info.json

- Huấn luyện Model

  python train.py

- Đánh giá Model

  python evaluate.py

- Chạy FastAPI

  python -m uvicorn app:app –reload

- Chạy chương trình

  run index.html

Chương 6

Tổng kết

Đồ án đã đạt được mục tiêu ban đầu là phát triển một ứng dụng chẩn đoán hình ảnh CT phổi bằng trí tuệ nhân tạo, đồng thời cũng đã mô tả chi tiết về quá trình hiện thực và những bước tiếp theo để phát triển và cải thiện ứng dụng. tuy nhiên, việc triển khai và cải thiện ứng dụng vẫn còn một số thách thức và tiềm ẩn. Dưới đây là một số điểm cần lưu ý:

- Tinh chỉnh mô hình: Mặc dù đã triển khai một mô hình chẩn đoán sử dụng EfficientNet, nhưng việc tinh chỉnh và cải thiện hiệu suất của mô hình vẫn cần thời gian và nỗ lực. Có thể cần thêm dữ liệu hoặc thực hiện các kỹ thuật tinh chỉnh mô hình để đạt được hiệu suất tốt nhất.

- Thiết kế giao diện người dùng: Giao diện người dùng có thể cần được cải thiện để tăng tính thân thiện và trực quan. Có thể cần thêm các tính năng như hướng dẫn sử dụng, ghi chú và thông tin thêm về bệnh lý.

Tài liệu tham khảo

[1] PyTorch Documentation
https://pytorch.org/docs/2.3/

[2] CustomTkinter Documentation
https://timm.fast.ai/

[3] Sklearn Documentation
https://scikit-learn.org/0.21/documentation.html

[4] Numpy Documentation
https://numpy.org/doc/

[5] PIL Documentation
https://pillow.readthedocs.io/en/stable/

[6] The IQ-OTHNCCD Lung Cancer Dataset
https://www.kaggle.com/datasets/antonixx/the-iqothnccd-lung-cancer-dataset/data