BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN CƠ SỞ
NGÀNH CÔNG NGHỆ THÔNG TIN

Đề tài

ỨNG DỤNG CÔNG NGHỆ DEEP LEARNING TRONG VIỆC XỬ LÝ HÌNH ẢNH PHÁT HIỆN DEEPFAKE

Sinh viên: Lâm Thế Vinh
Mã số: B2206022
Khóa: K48

Cần Thơ, 03/2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN CƠ SỞ
NGÀNH CÔNG NGHỆ THÔNG TIN

Đề tài

ỨNG DỤNG CÔNG NGHỆ DEEP LEARNING TRONG VIỆC XỬ LÝ HÌNH ẢNH PHÁT HIỆN DEEPFAKE

Cán bộ hướng dẫn
TS. Phạm Thế Phi

Sinh viên thực hiện
Lâm Thế Vinh
Mã số: B2206022
Khóa: 48

Cần Thơ, 03/2025

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn trân thành đến thầy TS. Phạm Thế Phi. Nếu không có sự hướng dẫn tận tình của thầy có lẽ bài báo cáo này đã không được hoàn thành. Ngoài ra, kiến thức và kinh nghiệm quý báo của thầy cũng đã giúp em học hỏi được rất nhiều trong suốt quá trình học tập và làm bài báo cáo.

Tiếp theo em muốn gửi lời cảm ơn tất cả các thầy, cô cán bộ giảng viên Trường Công nghệ thông tin và Truyền thông đã giảng dạy, chia sẽ kiến thức, kinh nghiệm của mình cho em và các bạn học trong suốt quá trình học tập.

Cuối cùng, em xin gửi lời cám ơn đến tất cả các bạn bè đã hỗ trợ, chia sẽ kiến thức để giúp em hoàn thành bài báo cáo.

Trân trọng,
Lâm Thế Vinh.

MỤC LỤC

DANH MỤC HÌNH ẢNH

GIỚI THIỆU ĐỀ TÀI

1. VẤN ĐỀ

Hiện nay với sự phát triển mạnh mẽ và đột phá của trí tuệ nhân tạo AI, nhiều lĩnh vực đã và đang phát triển, ứng dụng công nghệ này vào quy trình hoạt động mang lại kết quả đáng kinh ngạc đặc biệt là trong lĩnh vực nhận diện và xử lý hình ảnh.

Những thành tựu đặc sắc của việc ứng dụng AI trong lĩnh vực nhận diện và xử lý hình ảnh, thị giác máy tính [1] có thể kể đến như công nghệ tạo hình ảnh từ mô tả văn bản (Text-to-image) với các mô hình như DALL·E, MidJourney, Stable Diffusion,... Công nghệ tạo mô hình 3D từ ảnh 2D với các mô hình như Nvidia Instant NeRF, Meta 3D Gen,... Nổi bật nhất có lẽ là công nghệ Deepfake [2] giúp thay đổi khuôn mặt trong video một cách chân thực và rất khó để con người có thể phân biệt được. Deepfake được ứng dụng trong việc xây dựng phim, ảnh và truyền thông mang lại hiệu quả vô cùng to lớn. Tuy nhiên, vì thấy được tìm năng và hiệu quả của Deepfake nhiều kẻ xấu đã lợi dụng Deepfake cho các mục đích xấu như tung tin tức giả mạo, tống tiền, bôi nhọ danh dự của người khác,... Theo Báo công an nhân dân, tại Việt Nam đã có nhiều vụ việc sử dụng Deepfake ghép mặt người bị hại vào các clip nhạy cảm nhằm mục đích tống tiền với số tiền lên đến hàng trăm triệu đồng. Ngoài ra, còn có nhiều vụ việc dùng Deepfake để giả dạng người thân lừa gạt yêu cầu người bị hại chuyển tiền cũng đã xảy ra và đang được điều tra. Nhiều rủi ro là thế tuy nhiên việc ngăn chặn và phát hiện Deepfake ở Việt Nam cũng như trên toàn thế giới hiện nay vẫn còn rất hạn chế.

2. MỤC ĐÍCH

Việc nghiên cứu ứng dụng công nghệ học sâu Deep learning [3] trong phát hiện Deepfake sẽ là giải pháp hiệu quả để giúp cho người dùng tránh được rủi ro bị lợi dụng, giả mạo và tống tiền trên không gian mạng. Đồng thời, nghiên cứu cũng sẽ giúp thúc đẩy việc nghiên cứu ứng dụng AI trong nhiều lĩnh vực khác.

3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Nghiên cứu tập trung vào việc phát triển và tối ưu hóa các mô hình ResNext [4] và mạng nơ-ron hồi tiếp bộ nhớ dài ngắn hạn Long Short-Term Memory (LSTM) [5] cho nhiệm vụ phát hiện Deepfake. Điều này bao gồm việc thiết kế kiến trúc mô hình, lựa chọn siêu tham số, và áp dụng các kỹ thuật tiền xử lý dữ liệu để cải thiện hiệu suất của mô hình.

4. PHƯƠNG PHÁP NGHIÊN CỨU

Nghiên cứu mô hình ResNeXt để trích xuất các đặc trưng không gian từ từng khung hình, giúp nắm bắt các chi tiết quan trọng của hình ảnh. Sử dụng mạng nơ-ron hồi tiếp dài ngắn hạn Long Short-Term Memory (LSTM) để xử lý chuỗi các đặc trưng không gian từ ResNeXt, giúp phát hiện các bất thường theo thời gian trong video... Tiến hành huấn luyện và đánh giá mô hình dựa trên các tập dữ liệu công khai như FaceForensics++ chứa hàng ngàn video thật và giả mạo. Xây dựng, đánh giá và kiểm thử nền tảng ứng dụng mô hình.

NỘI DUNG

CHƯƠNG 1. ĐẶC TẢ YÊU CẦU

1. GIỚI THIỆU

Tài liệu đặc tả những yêu cầu cần thiết của hệ thống ứng dụng học sâu Deep learning trong việc phát hiện Deepfake thông qua mô hình chính ResNeXt và mạng nơ-ron hồi tiếp dài ngắn hạn Long Short-Term Memory (LSTM).

2. MÔ TẢ HỆ THỐNG

Hệ thống sẽ sử dụng mô hình ResNeXt để trích xuất các đặc trưng không gian từ từng khung hình, giúp nắm bắt các chi tiết hình ảnh quan trọng của hình ảnh. Sử dụng mạng nơ-ron hồi tiếp dài ngắn hạn Long Short-Term Memory (LSTM) để xử lý chuỗi các đặc trưng không gian từ ResNeXt, giúp phát hiện các bất thường theo thời gian trong video. Phân tích, đánh giá đưa ra kết quả xử lý dựa trên mô hình đã được huấn luyện.

3. YÊU CẦU

3.1. Yêu cầu chức năng

Chức năng chính: Hệ thống phân tích, đánh giá video và cho ra kết quả video đã được chỉnh sửa bởi trí tuệ nhân tọa Deepfake hay chưa dựa trên mô hình đã đưa huấn luyện sẵn:

- Đầu vào:
  + Hệ thống nhận vào mô hình phân tích, đánh giá hình ảnh đã được huấn luyện trước. Mô hình phải phù hợp với các framework nền tảng thông thường như TensorFlow, PyTorch.
  + Hệ thống nhận vào video do người dùng cung cấp.
- Tính toán và xử lý:
  + Sau khi nhận đầu vào, hệ thống tiến hành xử lý hình ảnh thông qua hai mô hình chính ResNeXt và LSTM.
  + Dựa trên mô hình đã được huấn luyện trước, hệ thống phân tích, đánh giá và tính toán để đưa ra kết quả video được cung cấp đã qua chỉnh sửa bằng Deepfake hay chưa.
  + Hiển thị, thông báo kết quả cho người dùng.

3.2. Yêu cầu phi chức năng

Hệ thống đạt yêu cầu về sản phẩm:

- Thời gian phân tích, đánh giá và đưa ra kết quả từ lúc người dùng cung cấp đầu vào là từ 30 đến 120 giây tùy thuộc vào độ dài của video.
- Độ dài tối đa video cung cấp đạt được là 2 phút.
- Số lượng khung hình phân tích tối đa là 100 khung hình.
- Khả năng mở rộng cao khi được phát triển trên nền tảng mã nguồn mở.

Hệ thống đạt yêu cầu về tổ chức:

- Giao diện đơn giản dễ sử dụng phù hợp với nhiều nhóm người dùng.
- Chức năng hệ thống dễ tiếp cận kể cả cho người dùng lần đầu.

Hệ thống đạt yêu cầu về bảo mật:

- Hệ thống không yêu cầu về thông tin người dùng. Các video được cung cấp cũng sẽ được bảo mật tránh rò rỉ thông tin người dùng.

3.3. Yêu cầu về môi trường vận hành hệ thống

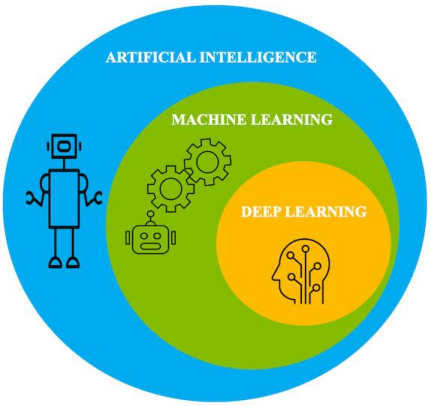
- Phần cứng:
  + Hệ thống yêu cầu máy chủ (Server) hoặc GPU Cloud: Cần phần cứng mạnh để xử lý video nhanh chóng, đặc biệt là GPU hỗ trợ TensorFlow, PyTorch như NVIDIA RTX 3090, A100 hoặc TPU trên Google Cloud.
  + Bộ nhớ (RAM & Ổ cứng SSD): RAM từ 16GB trở lên để xử lý dữ liệu lớn, ổ cứng SSD giúp tăng tốc độ đọc/ghi video.
  + Bộ xử lý (CPU): CPU đa nhân (Intel Xeon, AMD Ryzen Threadripper) để hỗ trợ tải công việc nặng
- Phần mềm:
  + Hệ điều hành: Windows hoặc Ubuntu.
  + Thiết lập môi trường ảo để dễ dàng vận hành và triển khai trên nhiều nền tảng.
  + Ngôn ngữ lập trình: yêu cầu cài đặt ngôn ngữ Python từ phiên bản 3.9 trở lên
  + Cài đặt các Framework học sâu như: TensorFlow và PyTorch để triển khai mô hình ResNeXt và LSTM, OpenCV để xử lý video, trích xuất khung hình, FFmpeg để chuyển đổi, trích xuất và xử lý video,…

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

1. CÁC KHÁI NIỆM LÝ THUYẾT ĐƯỢC SỬ DỤNG:

1.1. Deep learing (học sâu) và Ngôn ngữ lập trình Python:

Deep learning là một nhánh của trí tuệ nhân tạo (AI) và học máy (Machine learning). Deeplearning tập trung vào việc sử dụng các mạng nơ-ron nhân tạo nhiều lớp để mô phỏng bộ não con người trong việc xử lý dữ liệu và đưa ra quyết định. Các mô hình học sâu có khả năng tự học hỏi và trích xuất các đặc trưng từ dữ liệu lớn, giúp nhận điện và phân loại hình ảnh, âm thanh, văn bản và nhiều loại dữ liệu khác. Hiện nay, Deep learning đã và đang được ứng dụng trong nhiều lĩnh vực khác nhau như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh giọng nói, xe tự lái,...



Hình ảnh 1. Deep learning

Python là ngôn ngữ lập trình bậc cao được Guido van Rossum phát triển và ra mắt lần đầu vào năm 1991. Theo các nghiên cứu nổi tiếng [3], Python được cho là ngôn ngữ hàng đầu trong việc xây dựng các mô hình máy học, Deep learning nhờ vào việc có nhiều thư viện chuyên dụng mạnh mẽ như TensorFlow, Keras và PyTorch. Với cú pháp đơn giản, dễ dàng tiếp cận bởi nhiều đối tượng người dùng Python ngày càng phổ biến và xây dựng được một cộng đồng lập trình viên đông đảo, Python mang lại sự hỗ trợ phong phú về tài liệu, hướng dẫn và giải đáp thắc mắc, tạo điều kiện thuận lợi cho việc học tập và phát triển các dự án deep learning.

Hình ảnh 2. Python

1.2. PyTorch

PyTorch là một framework học máy dựa trên thư viện Torch do Meta AI phát triển, được sử dụng trong lĩnh vực thị giác máy tính và xử lý ngôn ngữ tự nhiên. Với việc phát triển trên các nền tảng mã nguồn mở PyTorch hiện nay đang được cải tiến mạnh mẽ với sự đóng góp của cộng đồng lập trình viên đông đảo trên toàn thế giới. PyTorch nổi bật với khả năng xây dựng các mạng nơ-ron động dựa trên autograd, sự linh hoạt trong việc thay đổi cấu trúc mạng trong quá trình chạy, phù hợp cho nghiên cứu và phát triển các mô hình mới. Ngoài ra, PyTorch còn tích hợp sẵn khả năng hỗ trợ GPU, giúp tăng tốc độ tính toán và huấn luyện mô hình.

Hình ảnh 3. PyTorch

1.3. TensorFlow

TensorFlow là một thư viện phần mềm mã nguồn mở được phát triển bởi Google. TensorFlow thường được sử dụng cho các ứng dụng học máy (Machine learning)và học sâu (Deep learning). Thư viện này cho phép các lập trình viên tạo ra các biểu đồ luồng dữ liệu (dataflow graph), trong đó mỗi nút (node) đại diện cho một phép toán toán học, và các cạnh (edge) giữa các nút là các mảng dữ liệu đa chiều, được gọi là "tensor". Với khả năng mở rộng và hiệu suất cao, TensorFlow hỗ trợ các nhà phát triển xây dựng và triển khai các mô hình học máy phức tạp, đồng thời cung cấp các công cụ và tài nguyên phong phú để tạo ra các ứng dụng trí tuệ nhân tạo đa dạng.

Hình ảnh 4. Tensorflow

1.4. ResNeXt

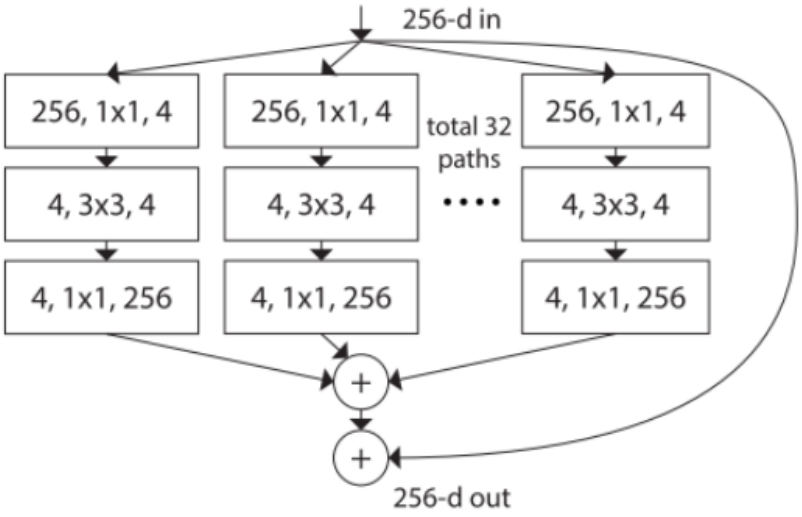
ResNeXt là một kiến trúc mạng nơ-ron tích chập (CNN) được giới thiệu nhằm cải thiện hiệu suất của các mô hình học sâu trong các nhiệm vụ như phân loại hình ảnh, phân đoạn và nhận dạng. Được phát triển dựa trên ResNet, ResNeXt bổ sung một yếu tố mới gọi là "cardinality" (độ phong phú), đại diện cho số lượng các nhánh biến đổi song song trong mỗi khối của mạng. Thay vì chỉ tăng chiều sâu hoặc chiều rộng của mạng, việc tăng độ phong phú giúp mô hình học được nhiều đặc trưng phức tạp hơn mà không làm tăng đáng kể số lượng tham số hoặc độ phức tạp tính toán.

"Cardinality block" là một phần mới được thêm vào trong kiến trúc mạng ResNeXt. Cardinality block có nhiệm vụ tạo sự phân chia (split) của các kênh đầu vào thành nhiều nhóm "Cardinality" (Cardinality groups). Mỗi nhóm đại diện cho một tập hợp các đặc trưng cụ thể mà mạng sẽ học và hoạt động độc lập với nhau. Điều này giúp mô hình có khả năng học đồng thời nhiều loại đặc trưng khác nhau, thu nhận thông tin đa dạng.

Các đặc trưng từ các nhóm được kết hợp lại thông qua phép biến đổi tổng hợp Aggregation transformation [6] để tạo ra đầu ra cuối cùng của khối. ResNeXt được xây dựng theo hướng "Network-in-Neuron". Thay vì tổng hợp tuyến tính tại mỗi path như neuron, mô hình ResNeXt sử dụng các function phi tuyến tính trên mỗi path. Ta có thể biểu diễn giai đoạn Transformation này theo công thức như sau:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x})$$

Trong đó $\mathcal{T}_i(\mathbf{x})$ là một hàm bất kì. *C* là cardinality. Giá trị *C* này kiểm soát độ phức tạp của giai đoạn chuyển đổi.



Hình ảnh 5. Kiến trúc mạng nơ-ron tích chập ResNeXt với Cardinality = 32

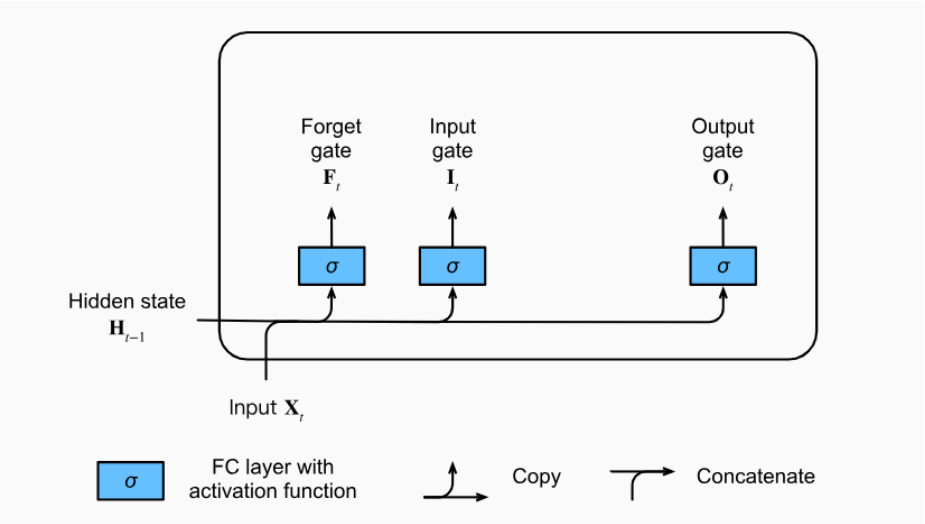
Trong khối ResNeXt trên, các đường dẫn (path) hoạt động độc lập, mỗi đường dẫn bao gồm ba lớp tích chập: *Conv 1×1*, *Conv3×3* và *Conv1×1*. Kích thước nội tại của mỗi đường dẫn là *d=4*, và độ lớn của nhóm (cardinality) là *C=32*. Tổng kích thước của các lớp *Conv3×3* là *d×C=4×32=128*. Sau đó, kích thước được tăng trực tiếp từ 4 lên 256, và các đường dẫn được cộng lại với nhau. Điều này giúp việc thiết kế ResNeXt path đơn giản hơn so với các kiến trúc khác như Inception-ResNet [7] và Grouped Convolution [8].

1.5. Bộ nhớ dài ngắn hạn Long Short-Term Memory (LSTM)

LSTM là một loại mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) được thiết kế để xử lý và dự báo các chuỗi dữ liệu có sự phụ thuộc theo thời gian, như văn bản, chuỗi thời gian tài chính, hoặc dữ liệu cảm biến. LSTM được giới thiệu bởi Hochreiter và Schmidhuber vào năm 1997 và đã được cải tiến bởi nhiều nhà nghiên cứu sau đó.

Kiến trúc RNN [9] tiêu chuẩn phạm vi của ngữ cảnh có thể được truy cập khá hạn chế. Vấn đề là do ảnh hưởng của đầu vào trên tầng ẩn, và vì thế trên đầu ra của mạng hoặc là suy giảm hoặc là tăng lên cấp số nhân theo chu kỳ xung quanh các kết nối hồi quy của mạng. Hiệu ứng này còn gọi là vấn đề biến mất đạo hàm (vanishing gradient problem). LSTM được thiết kế để giải quyết vấn đề "vanishing gradient" bằng cách sử dụng một cấu trúc đặc biệt gọi là "khối nhớ" (memory block). Mỗi khối nhớ bao gồm 3 cổng chính:

- Cổng quên (forget gate): Quyết định thông tin nào từ trạng thái trước đó cần được loại bỏ.
- Cổng đầu vào (input gate): Xác định thông tin nào từ đầu vào hiện tại sẽ được lưu trữ trong trạng thái ô nhớ.
- Cổng đầu ra (output gate): Quyết định phần nào của trạng thái ô nhớ sẽ được xuất ra ngoài.



Hình ảnh 6. Cấu trúc khối nhớ của LSTM

Nhờ cấu trúc khối nhớ với các cổng điều khiển thông tin, LSTM có khả năng lưu giữ và truyền tải thông tin qua nhiều bước thời gian, giúp cải thiện khả năng học và dự báo chuỗi dài. Điều này giúp LSTM vượt trội hơn so với RNN truyền thống trong việc xử lý các chuỗi dữ liệu dài và phức tạp.

1.6. Bộ dữ liệu FaceForensics++

FaceForensis++ là bộ dữ liệu video, hình ảnh Deepfake dùng cho mục đích nghiên cứu và đào tạo các mô hình máy học liên quan đến Deepfake. Bộ dữ liệu này bao gồm hơn 1.000 chuỗi video gốc đã được chỉnh sửa bằng bốn phương pháp thao tác khuôn mặt tự động: Deepfakes, Face2Face, FaceSwap và NeutralTextures. Dữ liệu được lấy từ 997 video trên YouTube, tất cả đều chứa khuôn mặt chủ yếu ở gốc nhìn chính diện cho phép tạo ra các video Deepfake chân thực nhất. Với việc cung cấp các mặt nạ nhị phân, dữ liệu có thể được sử dụng cho các nhiệm vụ phân loại hình ảnh và video cũng như phân đoạn. Bộ dữ liệu là một lựa chọn vô cùng phù hợp để tiến hành huấn luyện mô hình Deep learning phát hiện Deepfake.

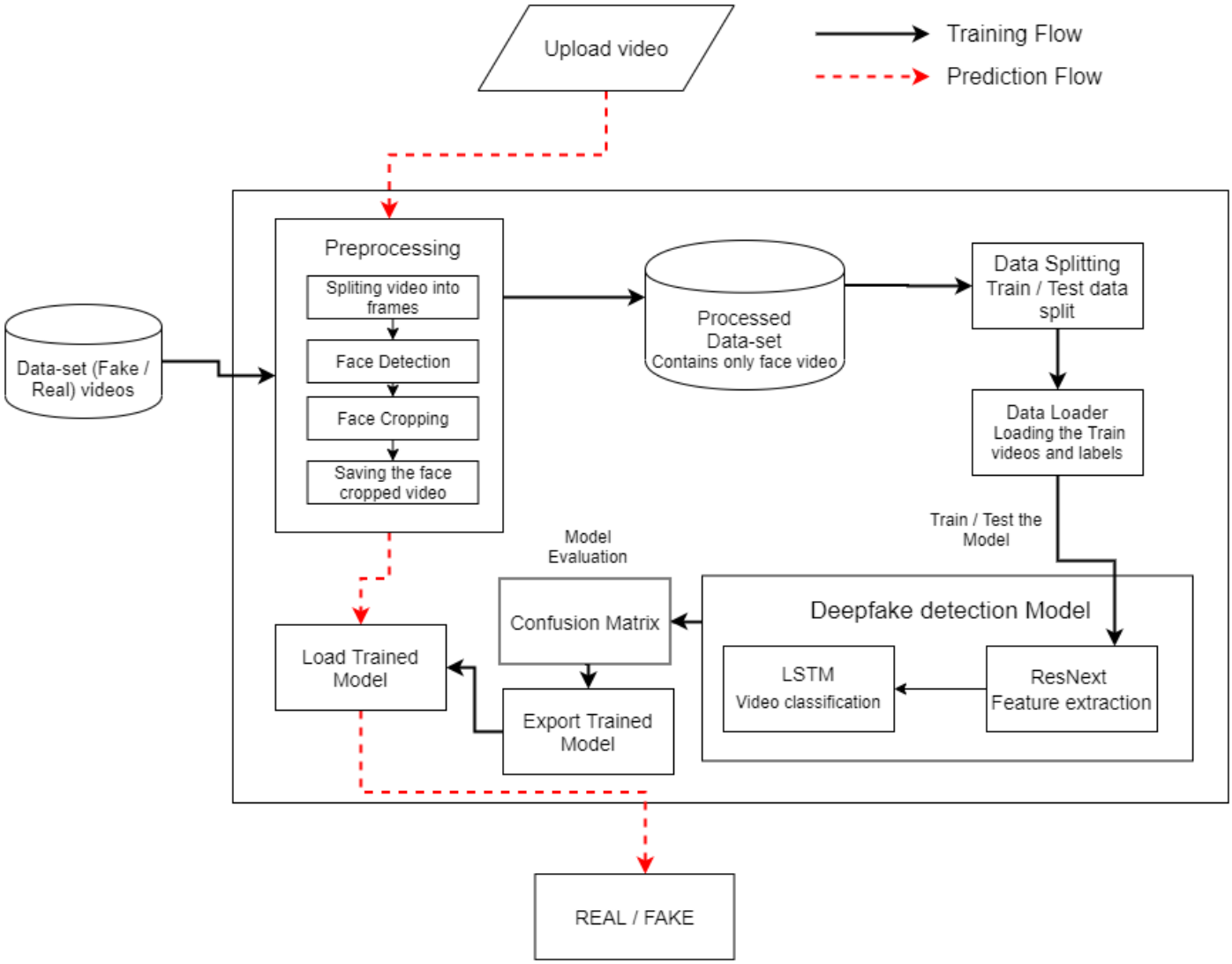
Hình ảnh 7. Bộ dữ liệu FaceForensics++

2. KẾT QUẢ VẬN DỤNG LÝ THUYẾT TRONG ĐỀ TÀI

Xây dựng một nền tảng phát hiện Deepfake thông qua video bằng phương pháp sử dụng ngôn ngữ lập trình Python và các thư viện như PyTorch hoặc TensorFlow để xây dựng mô hình học sâu Deep learning. Thu thập và chuẩn bị dữ liệu để huấn luyện mô hình phát hiện Deepfake hiệu quả thông qua bộ dữ liệu lớn FaceForensics++. Sử dụng mô hình ResNeXt để trích xuất các đặc trưng không gian trong từng khung hình, giúp nắm bắt các chi tiết hình ảnh quan trọng. Sử dụng mạng nơ-ron hồi tiếp ngắn hạn LSTM để xử lý các chuỗi đặc trưng không gian từ ResNeXt, giúp phát hiện các bất thường theo thời gian của video. Sử dụng ngôn ngữ Python để lập trình tính toán tự động đưa ra kết quả.

CHƯƠNG 3. CÀI ĐẶT GIẢI PHÁP VÀ KẾT QUẢ THỰC HIỆN

1. SƠ ĐỒ HỆ THỐNG

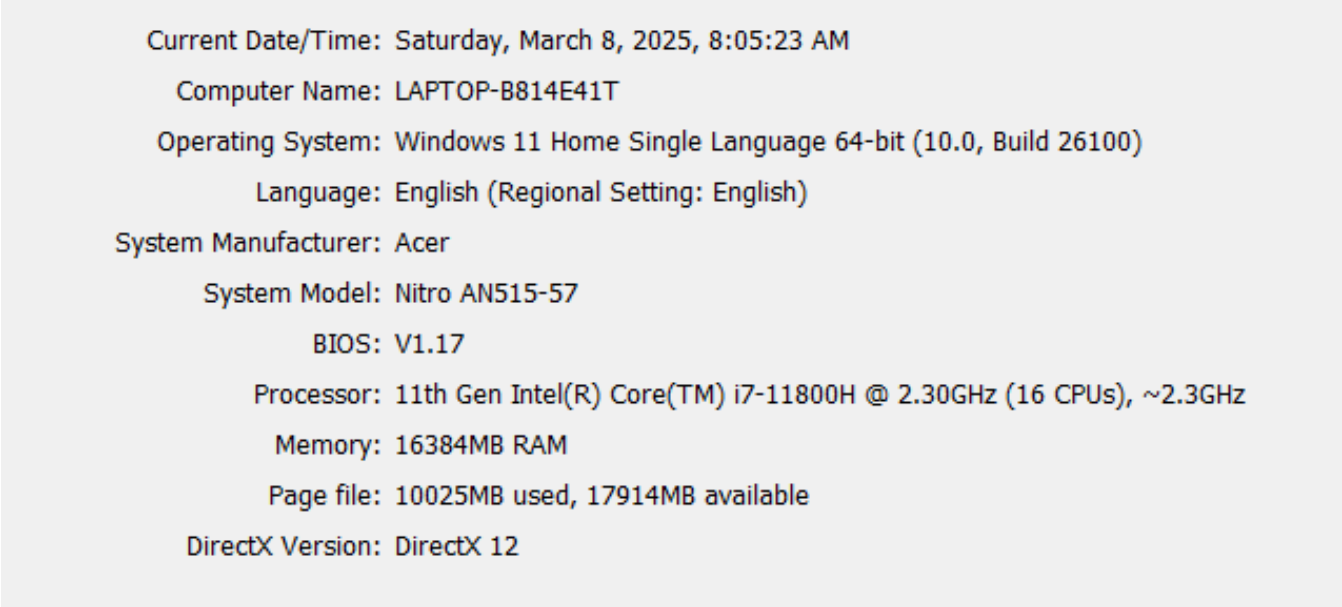


Hình ảnh 8. Sơ đồ hệ thống

Hệ thống sẽ đi theo hai luồng chính Training flow (luồng huấn luyện) dùng để huấn luyện mô hình và Prediction Flow (luồng dự đoán) dùng để dự đoán đưa ra kết quả REAL/FAKE dựa trên video Deepfake do người dùng cung cấp.

2. CÀI ĐẶT MÔI TRƯỜNG CẦN THIẾT

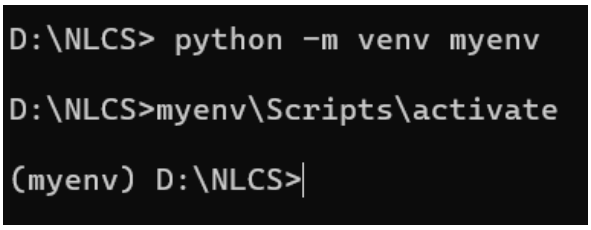
2.1. Thông tin máy chủ cài đặt hệ thống



Hình ảnh 9. Thông tin máy chủ

2.2. Tạo môi trường ảo để vận hành hệ thống

Môi trường ảo (Virtual Environment) là một không gian biệt lập trong hệ thống, nơi có thể cài đặt các thư viện và phụ thuộc mà không ảnh hưởng đến hệ thống toàn cục hoặc các dự án khác.



Hình ảnh 10. Môi trường ảo

2.3. Cài đặt mô hình ResNeXt

Mô hình ResNeXt là một mô hình trong lĩnh vực Deep learing (học sâu) và Computer vision (thị giác máy tính). Để vận hành ResNeXt yêu cầu cài đặt ngôn ngữ lập trình Python và các thư viện hỗ trợ như PyTorch, OpenCV, NumPy, Matplotlib.

- Cài đặt Python 3 (phiên bản 3.10.11)

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
```

Hình ảnh 11. Kiểm tra cài đặt Python

- Cài đặt PyTorch **pip3 install torch torchvision**:

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
2.3.1+cpu
>>>
```

Hình ảnh 12. Kiểm tra cài đặt PyTorch

- Cài đặt OpenCV **pip3 install opencv-python**:

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
2.3.1+cpu
>>>
```

Hình ảnh 13. Kiểm tra cài đặt PyTorch

- Cài đặt NumPy **pip3 install numpy**:

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.10.0
>>>
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> print(np.__version__)
1.26.4
```

Hình ảnh 14. Kiểm tra cài đặt NumPy

- Cài đặt Matplotlib **pip3 install matplotlib:**

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> print(np.__version__)
1.26.4
```

Hình ảnh 15. Kiểm tra cài đặt Matplotlib

- Khởi tạo và cài đặt mô hình ResNeXt

```python
class Model(nn.Module):

    def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim = 2048, bidire
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained = True) # <-- ResNeXt50 được khởi tạo ở đây
        self.model = nn.Sequential(*list(model.children())[:-2])
        self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional)
        self.relu = nn.LeakyReLU()
        self.dp = nn.Dropout(0.4)
        self.linear1 = nn.Linear(2048,num_classes)
        self.avgpool = nn.AdaptiveAvgPool2d(1)

    def forward(self, x):
        batch_size,seq_length, c, h, w = x.shape
        x = x.view(batch_size * seq_length, c, h, w)
        fmap = self.model(x) # Trích xuất đặc trưng với ResNeXt
        x = self.avgpool(fmap)
        x = x.view(batch_size,seq_length,2048)
        x_lstm,_ = self.lstm(x,None) # Đưa qua LSTM để học theo thời gian
        return fmap,self.dp(self.linear1(x_lstm[:,-1,:]))
```

Hình ảnh 16. Khởi tạo và cài đặt mô hình ResNeXt

* Phân tích cài đặt:

- **models.resnext50_32x4d(pretrained=True):** Load mô hình ResNeXt-50 có trọng số được huấn luyện sẵn.

- **self.model = nn.Sequential(*list(model.children())[:-2]):** Loại bỏ các lớp fully connected của ResNeXt, chỉ giữ lại các lớp convolution để trích xuất đặc trưng.

- **self.lstm = nn.LSTM(latent_dim, hidden_dim, lstm_layers, bidirectional)**: Sau khi lấy đặc trưng từ ảnh, mô hình sử dụng LSTM để học chuỗi hình ảnh.

- **self.linear1 = nn.Linear(2048, num_classes)**: Lớp fully connected để phân loại.

+ **self.avgpool = nn.AdaptiveAvgPool2d(1):** Lớp pooling để giảm kích thước đầu ra

2.4. Cài đặt Mạng nơ-ron hồi tiếp bộ nhớ dài ngắn hạn Long Short-Term Memory (LSTM)

LSTM (Long Short-Term Memory) là một loại mạng nơ-ron hồi quy (RNN) giúp xử lý dữ liệu tuần tự như chuỗi thời gian, văn bản, âm thanh. Giống như ResNeXt để vận hành LSTM cần cài đặt các thư viện vận hành như PyTorch, Numpy, Pandas và Matplotlib. Ngoài ra, để triển khai LSTM cần cài đặt thêm các thư viện như scikit-learn, seaborn.

- Cài đặt Pandas **pip3 install pandas:**

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> print(pd.__version__)
2.2.2
```

Hình ảnh 17. Kiểm tra cài đặt Pandas

- Cài đặt Scikit-learn **pip3 install scikit-learn:**

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sklearn
>>> print(sklearn.__version__)
1.6.1
>>>
```

Hình ảnh 18. Kiểm tra cài đặt scikit-learn

- Cài đặt Scikit-learn **pip3 install seaborn**:

```
python
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import seaborn as sns
>>> print(sns.__version__)
0.13.2
```

Hình ảnh 19. Kiểm tra cài đặt Seaborn

- Cài đặt mạng nơ-ron hồi tiếp bộ nhớ dài ngắn hạn LSTM:

```python
class Model(nn.Module):

    def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim = 2048, bidirectional
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained = True)
        self.model = nn.Sequential(*list(model.children())[:-2])
        self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional) # Cài đặt lớp LSTM
        self.relu = nn.LeakyReLU()
        self.dp = nn.Dropout(0.4)
        self.linear1 = nn.Linear(2048,num_classes)
        self.avgpool = nn.AdaptiveAvgPool2d(1)

    def forward(self, x):
        batch_size,seq_length, c, h, w = x.shape
        x = x.view(batch_size * seq_length, c, h, w) # Đưa dữ liệu về dạng phù hợp để đưa vào ResNeXt
        fmap = self.model(x)
        x = self.avgpool(fmap)  # Áp dụng pooling để giảm kích thước
        x = x.view(batch_size,seq_length,2048) # Chuyển đổi về dạng phù hợp cho LSTM
        x_lstm,_ = self.lstm(x,None) # Đưa qua LSTM để học theo thời gian
        return fmap,self.dp(self.linear1(x_lstm[:,-1,:]))
```

Hình ảnh 20. Cài đặt LSTM

* Phân tích cài đặt:

- **self.lstm = nn.LSTM(latent_dim, hidden_dim, lstm_layers, bidirectional):**
  + **latent_dim=2048:** số chiều của vector đầu vào (từ ResNeXt).
  + **hidden_dim=2048:** số chiều của hidden state trong LSTM.
  + **lstm_layers=1:** số lớp của LSTM.
  + **bidirectional=False:** LSTM chỉ chạy theo 1 hướng.
- Hàm **forward(self, x)**: thực hiện quá trình xử lý dữ liệu từ đầu vào (video) qua ResNeXt, LSTM, trả về frame cuối cùng và lấy nó để dự đoán .

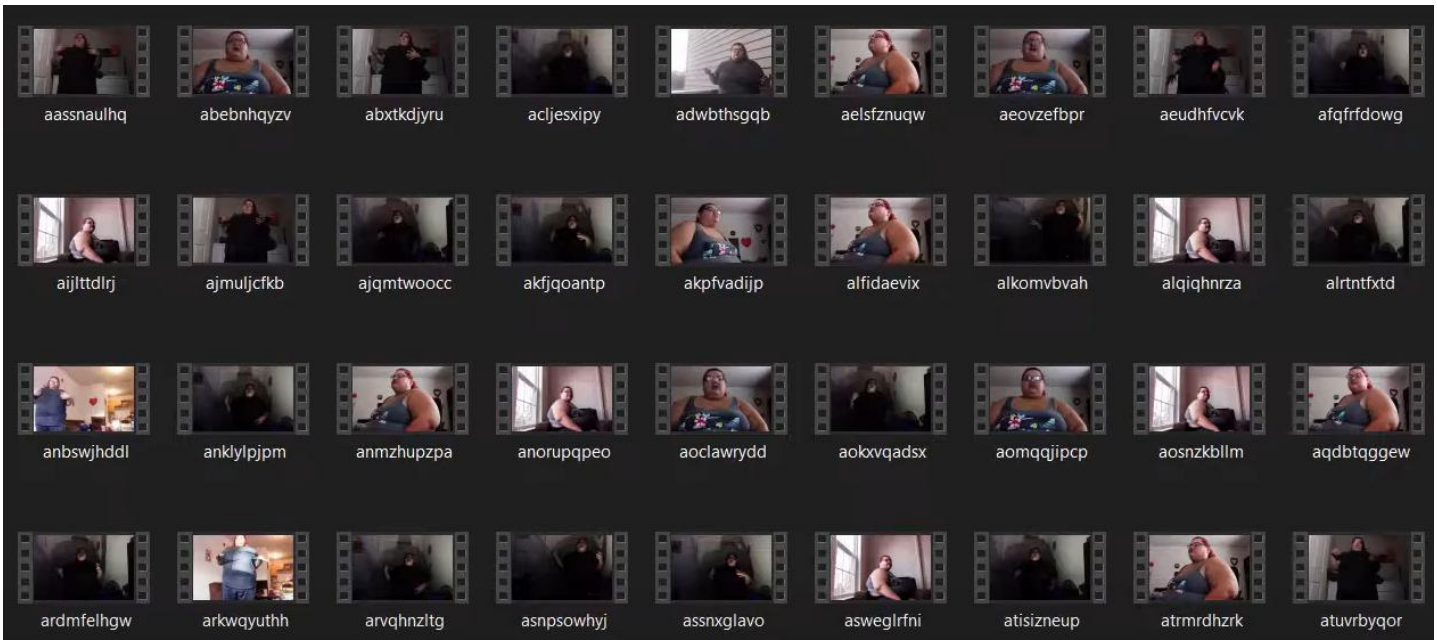
3. Cài đặt FaceForensics++

Để cài đặt FaceForensics++ dataset thực hiện chạy file cài đặt theo hướng dẫn của trang chủ FaceForensics++. Dưới đây là nội dung chính của file cài đặt

```python
def main(args):
    print(f'By pressing any key, you confirm that you agree to the FaceForensics terms of use: {args.tos_url}')
    input('Press any key to continue, or CTRL-C to exit.')

    datasets = [args.dataset] if args.dataset != 'all' else ALL_DATASETS
    for dataset in datasets:
        dataset_path = DATASETS[dataset]
        output_dir = join(args.output_path, dataset_path, args.compression, args.type)
        os.makedirs(output_dir, exist_ok=True)
        print(f'Downloading {args.type} of dataset {dataset} to {output_dir}')

        filelist_url = args.base_url + FILELIST_URL
        filelist = json.loads(urllib.request.urlopen(filelist_url).read().decode('utf-8'))
        filelist = [f'{filename}.mp4' for filename in filelist]
        if args.num_videos:
            filelist = filelist[:args.num_videos]
        dataset_url = args.base_url + dataset_path + '/' + args.compression + '/' + args.type + '/'
        download_files(filelist, dataset_url, output_dir)

def download_files(filenames, base_url, output_path):
    os.makedirs(output_path, exist_ok=True)
    for filename in tqdm(filenames, desc='Downloading files'):
        download_file(base_url + filename, join(output_path, filename))
```

Hình ảnh 21. Cài đặt FaceForensics++ dataset

Dưới đây là hình ảnh một vài video của Dataset sau khi cài đặt thành công



Hình ảnh 22. Các video của dataset FaceForensics++ dataset

4. HUẤN LUYỆN MÔ HÌNH

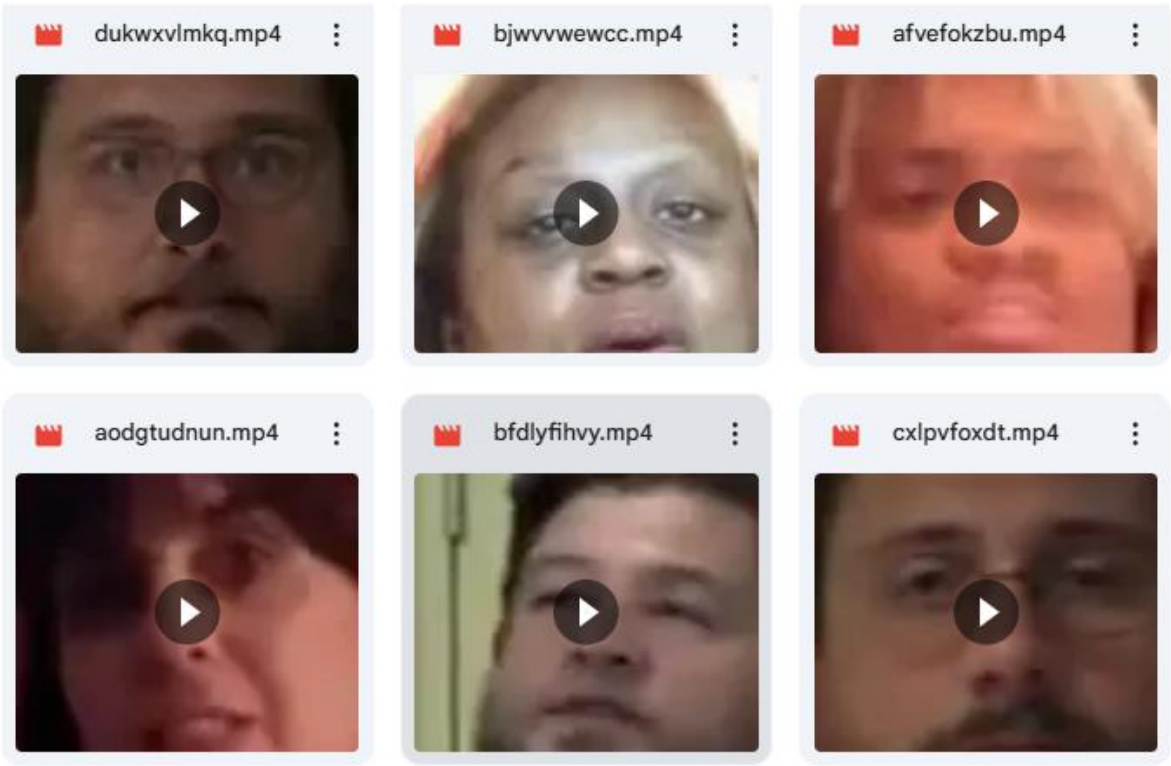
4.1. Tiền xử lý

Để đảm bảo kết quả huấn luyện chính xác, trước khi tiến hành huấn luyện, dữ liệu cần được chuẩn hóa để đạt tính thống nhất và rõ ràng. Giai đoạn này được gọi là tiền xử lý. Trong mô hình hiện tại, chúng ta cần chuyển đổi video về định dạng tối ưu, sao cho khuôn mặt của người trong video được hiển thị rõ nét nhất và xóa bỏ các bộ phận dư thừa.

```
def create_face_videos(path_list,out_dir):
  already_present_count =  glob.glob(out_dir+'*.mp4')
  print("No of videos already present " , len(already_present_count))
  for path in tqdm(path_list):
    out_path = os.path.join(out_dir,path.split('/')[-1])
    file_exists = glob.glob(out_path)
    if(len(file_exists) != 0):
      print("File Already exists: " , out_path)
      continue
    frames = []
    flag = 0
    face_all = []
    frames1 = []
    out = cv2.VideoWriter(out_path,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (112,112))
    for idx,frame in enumerate(frame_extract(path)):
      #if(idx % 3 == 0):
      if(idx <= 150):
        frames.append(frame)
        if(len(frames) == 4):
          faces = face_recognition.batch_face_locations(frames)
          for i,face in enumerate(faces):
            if(len(face) != 0):
              top,right,bottom,left = face[0]
            try:
              out.write(cv2.resize(frames[i][top:bottom,left:right,:],(112,112)))
            except:
              pass
          frames = []
    try:
      del top,right,bottom,left
    except:
```

Hình ảnh 23. Tiền xử lý video huấn luyện

Dưới đây là hình ảnh một vài video sau khi đã qua xử lý



Hình ảnh 24. Các video sau giải đoạn tiền xử lý

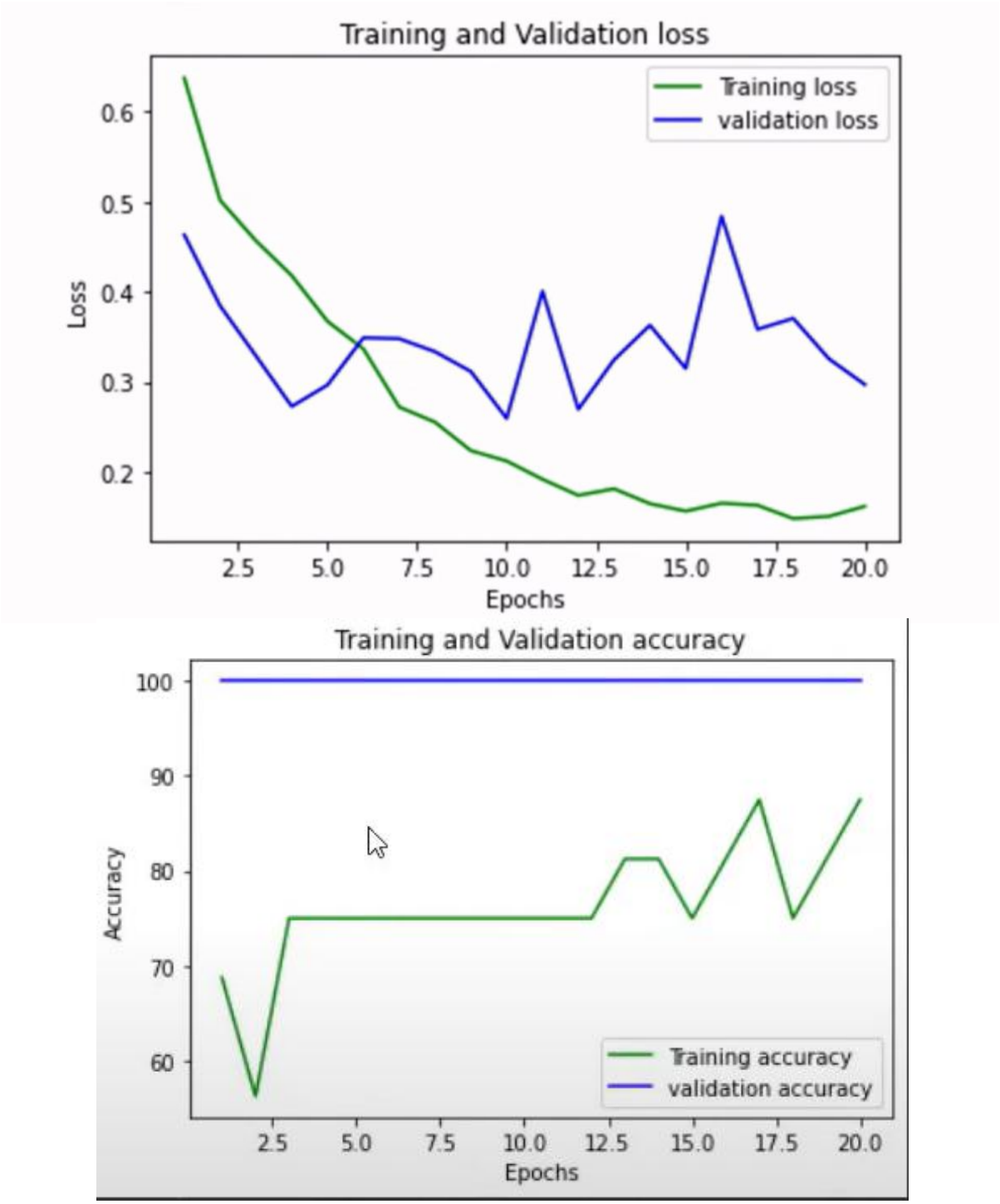
4.2. Tải video và huấn luyện mô hình

Đoạn code chính dùng để huấn luyện mô hình là hàm **train_epoch()**. Đây là hàm thực hiện quá trình huấn luyện trên từng epoch. Quy trình huấn luyện bắt đầu bằng việc đặt mô hình vào chế độ **train()**, sau đó lặp qua từng batch trong **data_loader**. Dữ liệu đầu vào được chuyển sang GPU nếu có, rồi đưa qua mô hình để dự đoán. Loss được tính bằng **criterion(outputs, targets)**, và độ chính xác được đo bằng **calculate_accuracy(outputs, targets)**. Các giá trị loss và accuracy trung bình được cập nhật thông qua **AverageMeter**. Tiếp theo, quá trình **backpropagation** được thực hiện bằng cách tính **gradient (loss.backward())** và cập nhật trọng số **(optimizer.step())**. Trong suốt quá trình huấn luyện, tiến trình được hiển thị theo từng batch. Cuối cùng, trọng số mô hình được lưu lại sau mỗi epoch để đảm bảo khả năng tiếp tục huấn luyện hoặc đánh giá sau này.

```
def train_epoch(epoch, num_epochs, data_loader, model, criterion, optimizer):
    model.train()
    losses = AverageMeter()
    accuracies = AverageMeter()
    t = []
    for i, (inputs, targets) in enumerate(data_loader):
        if torch.cuda.is_available():
            targets = targets.type(torch.cuda.LongTensor)
            inputs = inputs.cuda()
        _,outputs = model(inputs)
        loss  = criterion(outputs,targets.type(torch.cuda.LongTensor))
        acc = calculate_accuracy(outputs, targets.type(torch.cuda.LongTensor))
        losses.update(loss.item(), inputs.size(0))
        accuracies.update(acc, inputs.size(0))
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        sys.stdout.write(
                "\r[Epoch %d/%d] [Batch %d / %d] [Loss: %f, Acc: %.2f%%]"
                % (
                    epoch,
                    num_epochs,
                    i,
                    len(data_loader),
                    losses.avg,
                    accuracies.avg))
    torch.save(model.state_dict(),'/content/checkpoint.pt')
    return losses.avg,accuracies.avg
```

Hình ảnh 25. Đoạn code huấn luyện mô hình

4.3. Kết quả huấn luyện mô hình



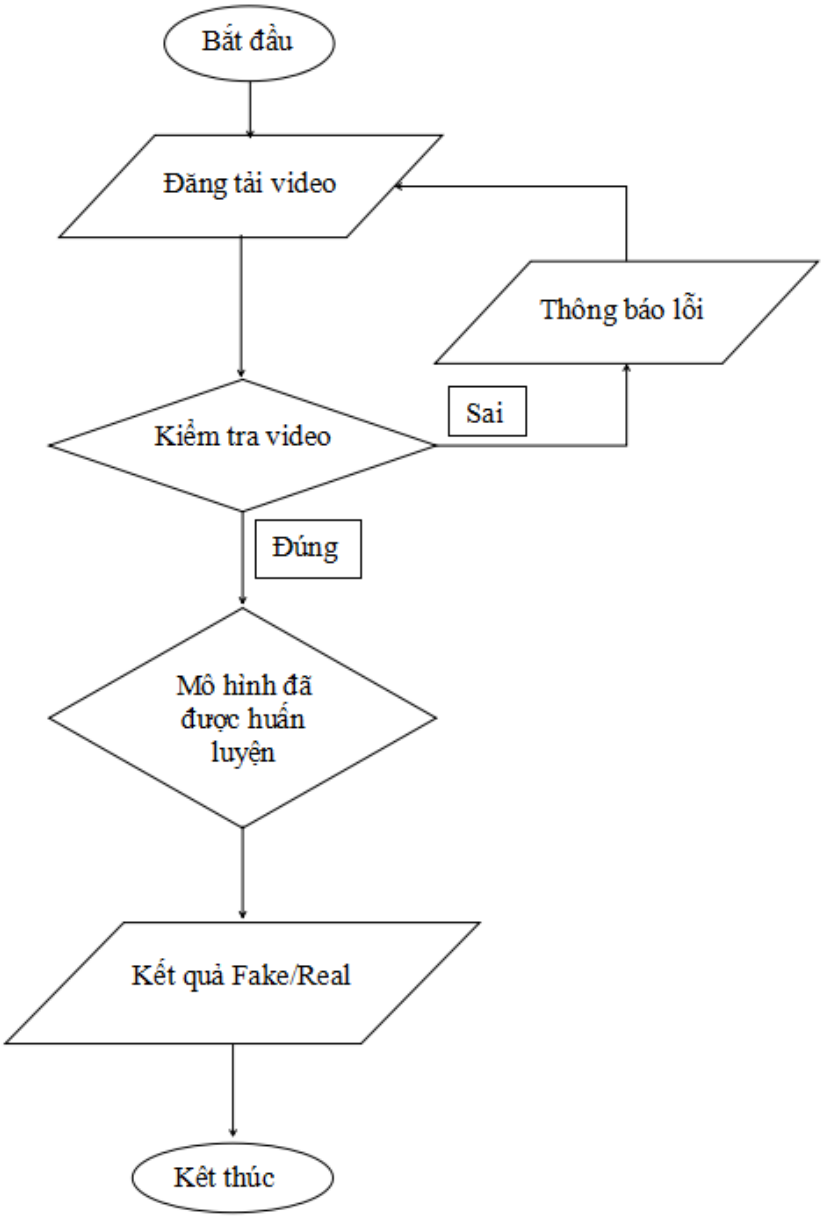
Hình ảnh 26. Kết quả huấn luyện mô hình

5. XÂY DỰNG TRANG WEB CƠ BẢN ĐỂ ỨNG DỤNG MÔ HÌNH

5.1. Mô tả

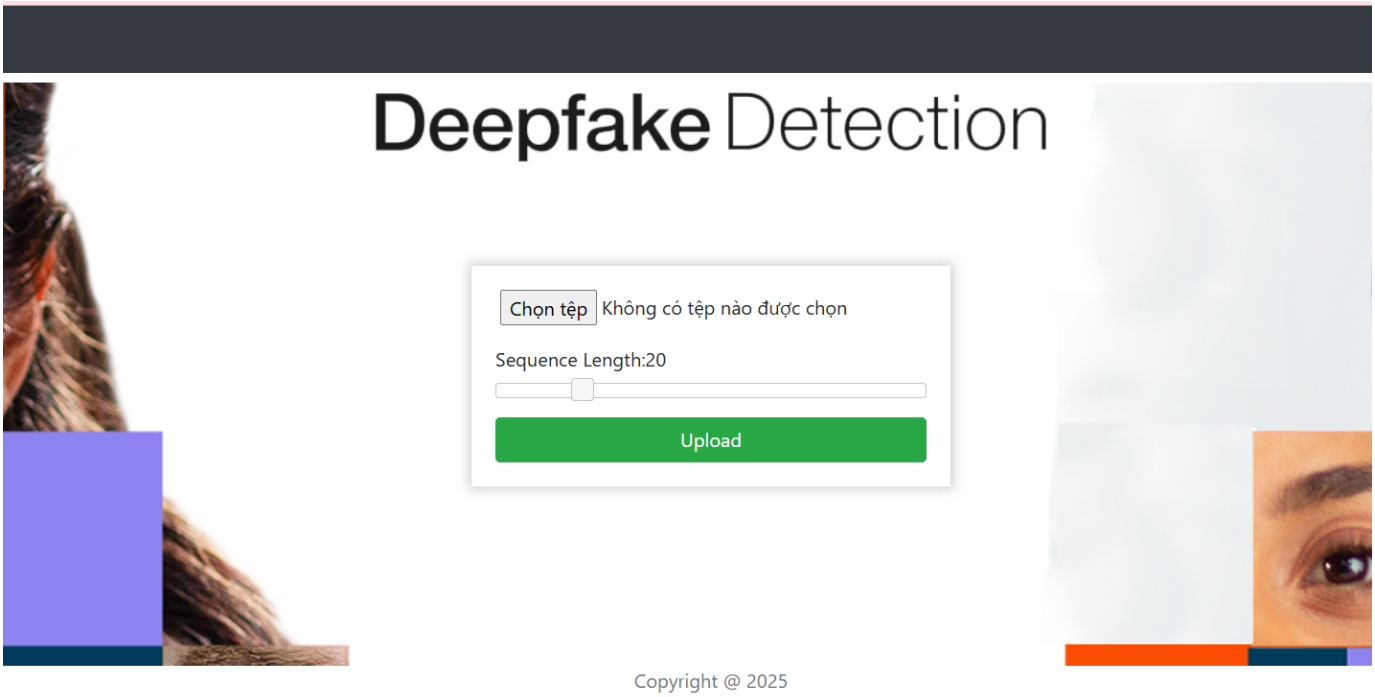
Trang web được xây dựng đơn giản bằng JS và CSS. Chức năng chính là giúp người dùng tải lên một video có thể là video Deepfake hoặc không. Sau đó hệ thống sẽ tiến hành phân tích đánh giá dựa trên các mô hình đã được huấn luyện trước và đưa ra kết quả dự đoán

5.2. Mô hình hệ thống



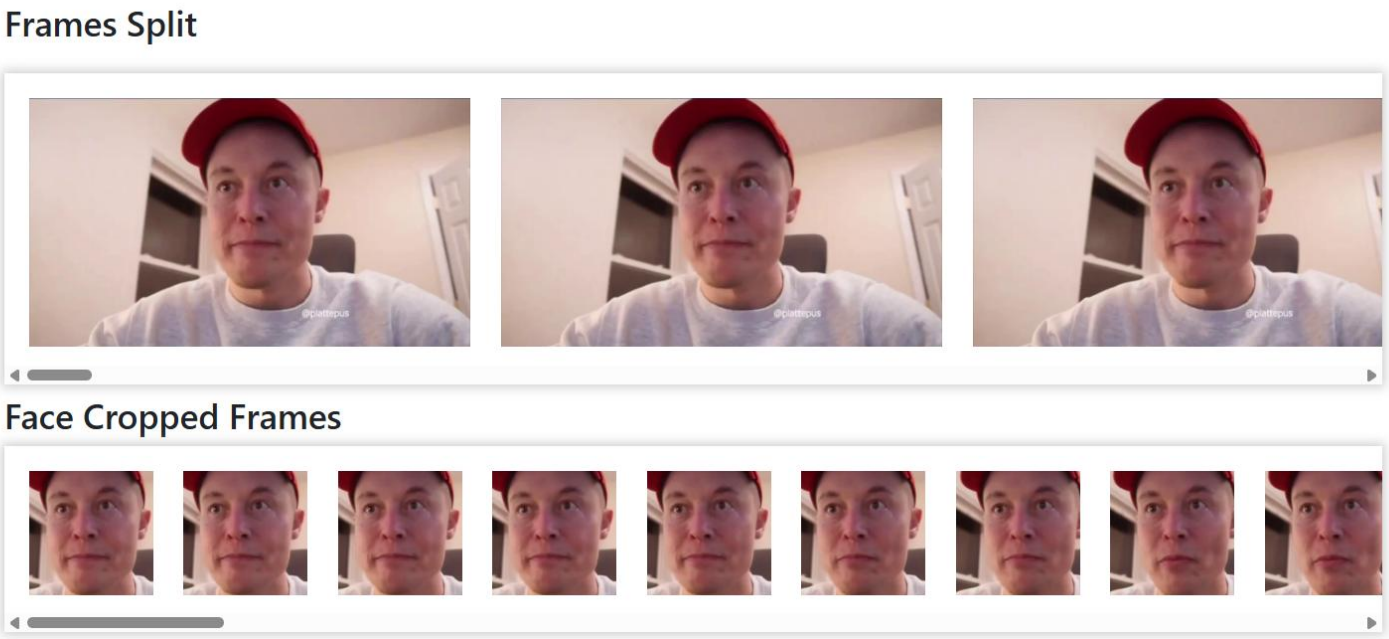
Hình ảnh 27. Sơ đồ tổng thể hệ thống

5.3. Giao diện hệ thống

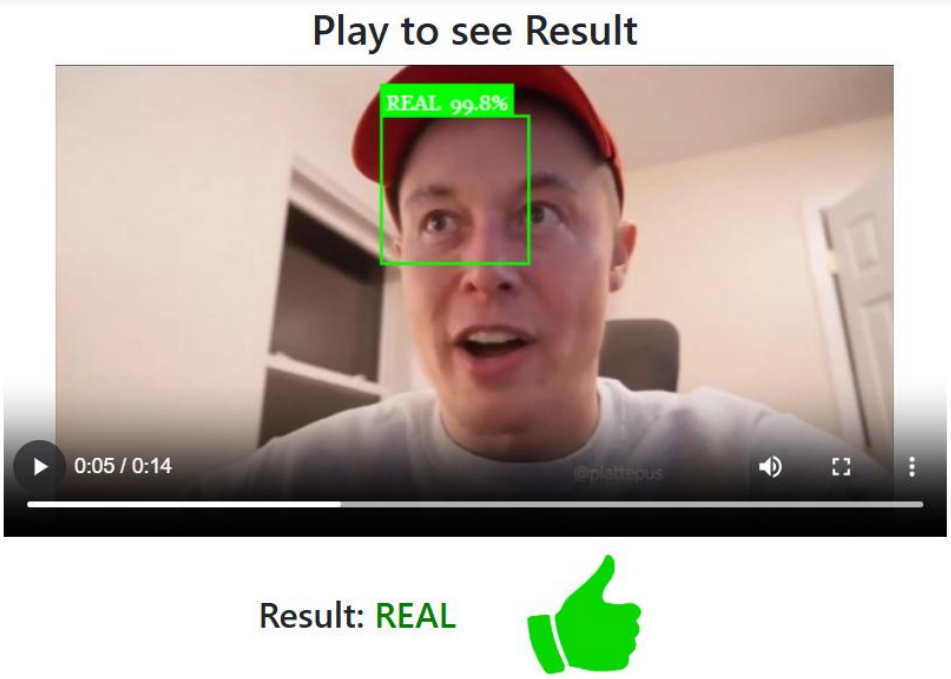


Hình ảnh 28. Giao diện chính

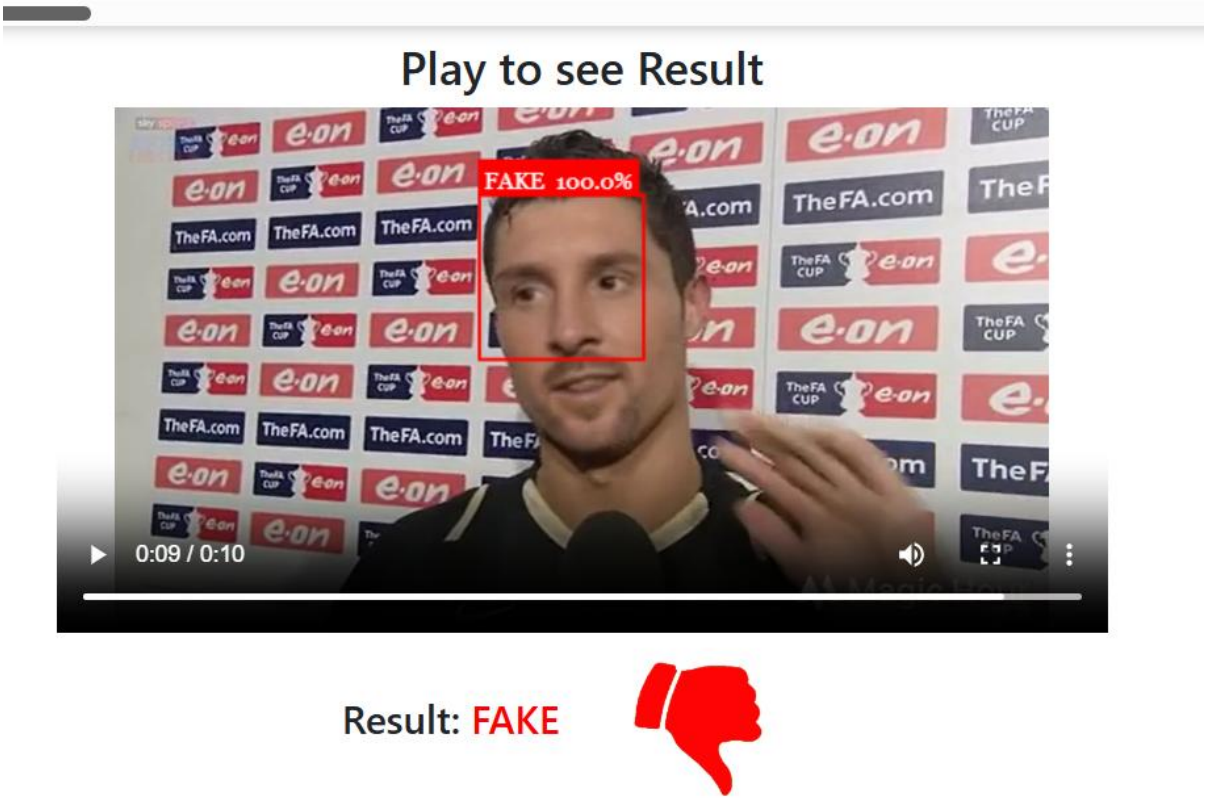
-



Hình ảnh 29. Hệ thống phân tích video



Hình ảnh 30. Hệ thống trả về kết quả Real



Hình ảnh 31. Hệ thống trả về kết quả Fake

CHƯƠNG 4. ĐÁNH GIÁ KIỂM THỬ

1. MỤC TIÊU KIỂM THỬ

Mục tiêu chính của việc kiểm thử là đánh giá độ chính xác của mô hình trong việc kiểm tra video Deepfake thật/giả. Ngoài ra, việc kiểm thử này cũng sẽ giúp tìm ra các điểm mạnh và điểm yếu của mô hình từ đó xây dựng kế hoạch nâng cấp, sửa chữa phù hợp cho tương lai.

2. KIỂM THỬ MÔ HÌNH

2.1. Kịch bản kiểm thử

Kiểm thử cần được thực hiện trên nhiều tập dữ liệu khác nhau với số khung khác nhau để cho ra một kết quả đánh giá chuẩn xác nhất.

Bước đầu tiên, chuẩn bị các mô hình đã được huấn luyện sẵn với số lượng khung khác nhau lần lượt là 10, 20, 40, 60, 80, 100 khung hình. Tiếp theo, tiến hành thử nghiệm các mô hình này với tập dữ liệu video gồm 6000 videos có chứa cả video thật và video giả. Ghi lại kết quả của các lần đánh giá. Cuối cùng, tổng hợp tính toán thống kê kết quả kiểm thử.

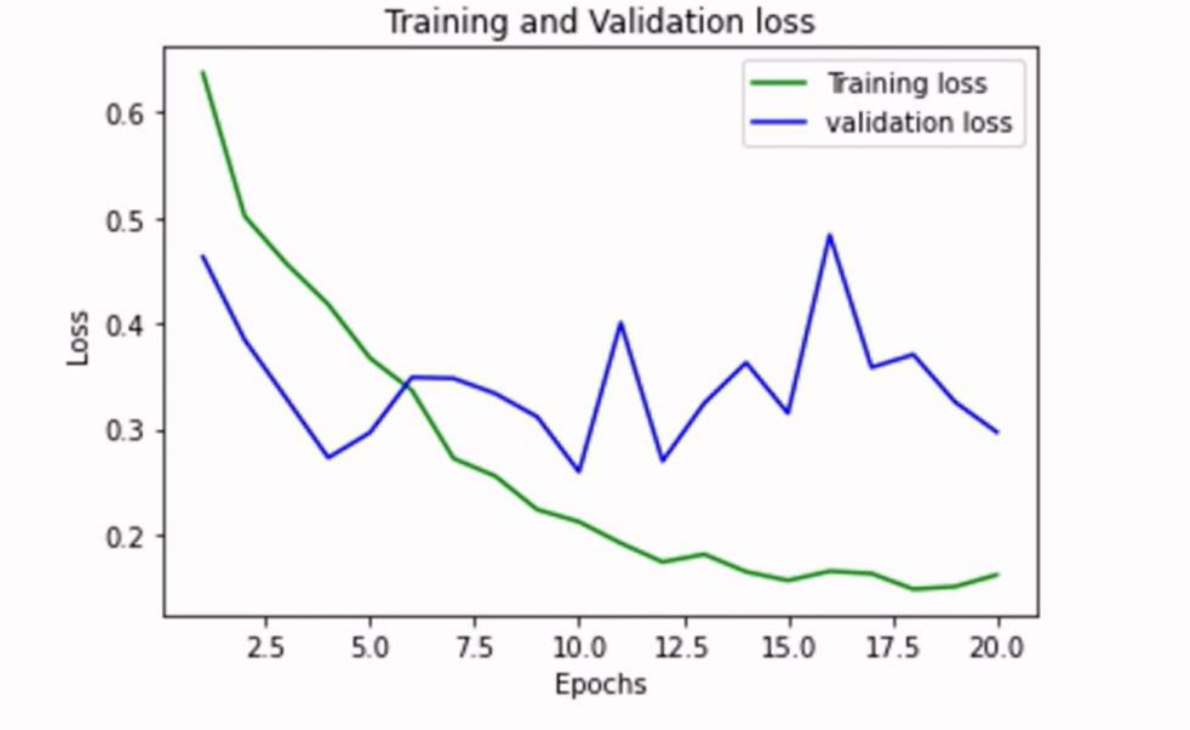
2.2. Kiểm thử độ chính xác mô hình

Dưới đây là bảng số liệu độ chính xác của các mô hình dựa trên số lượng frames được phân tách. Có thể thấy các model được huấn luyện với số frames càng lớn thì đạt kết quả càng cao. Đặc biệt với model có số lượng frames phân tách là 100 có độ chính xác lên đến 93.58796%.

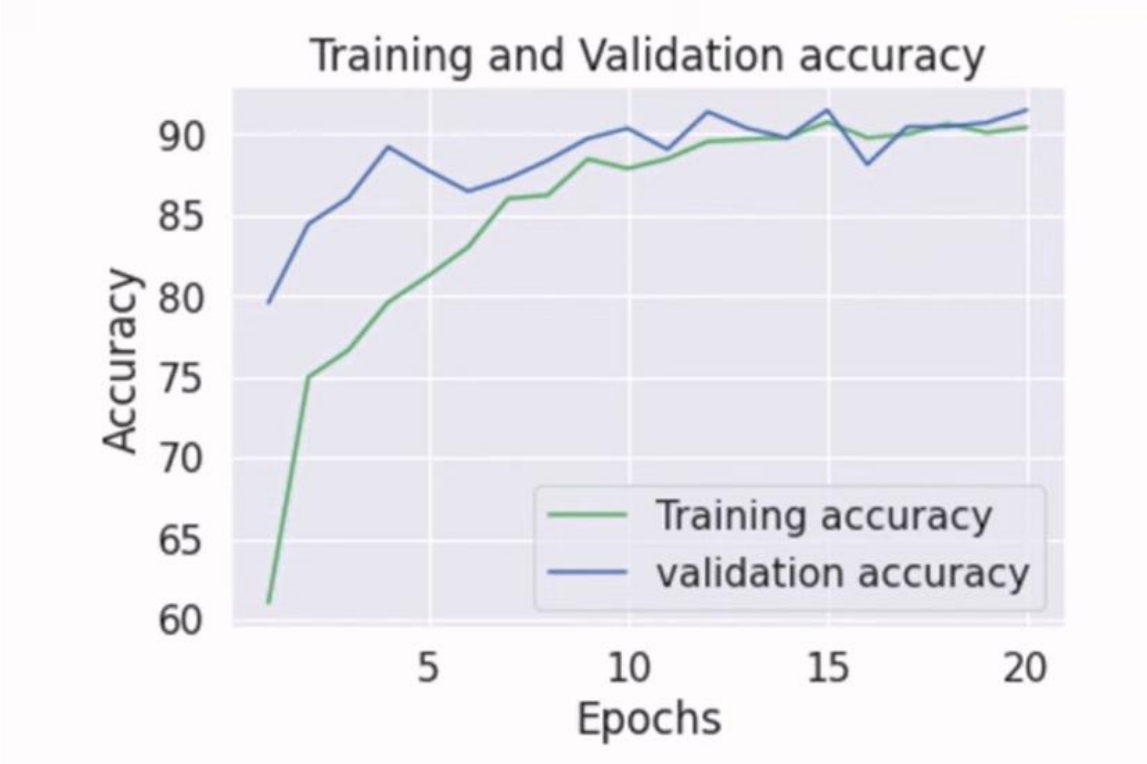
Model name	No of videos	No of Frames	Accuracy
model_84_acc_10_frames_final_data.pt	6000	10	84.21461
model_87_acc_20_frames_final_data.pt	6000	20	87.79160
model_89_acc_40_frames_final_data.pt	6000	40	89.34681
model_90_acc_60_frames_final_data.pt	6000	60	90.59097
model_91_acc_80_frames_final_data.pt	6000	80	91.49818
model_93_acc_100_frames_final_data.pt	6000	100	93.58794

Bảng 1. Đánh giá độ chính xác của model

Dưới đây là kết quả huấn luyện mô hình với số lượng khung hình là 80 và số lượng video huấn luyện là 6000 video bao gồm cả video thật và video giả.



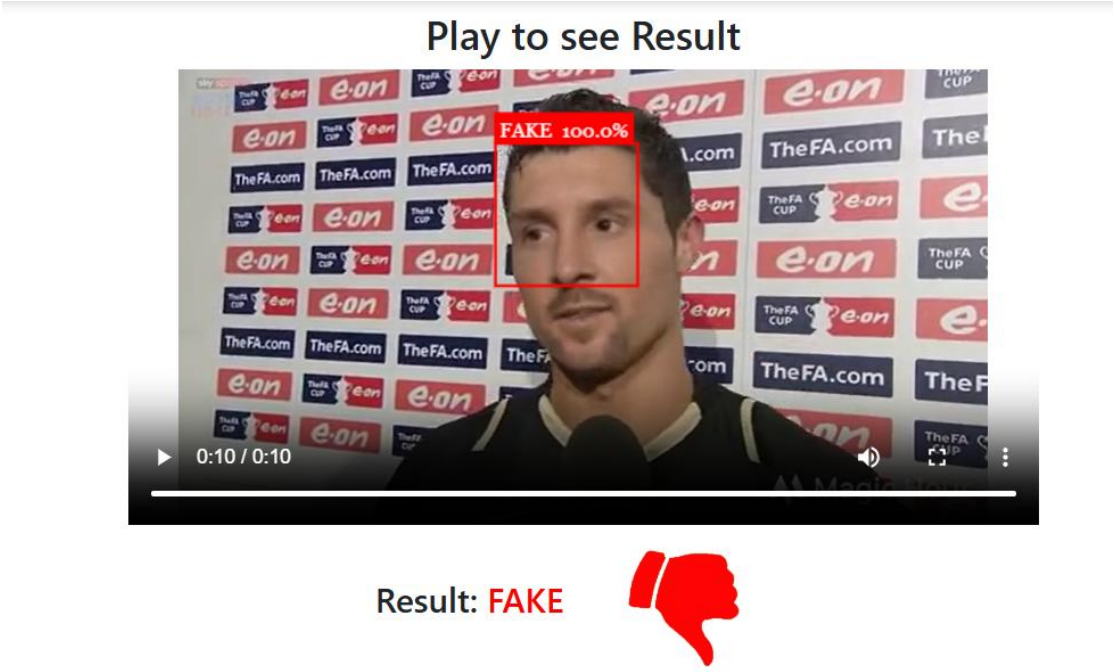
Hình ảnh 32. Độ sai trên tập huấn luyện và tập xác thực



Hình ảnh 33. Độ chính xác trên tập huấn luyện và tập xác thực

2.3. Kiểm thử độ chính xác của hệ thống dự đoán dựa trên mô hình

Kiểm thử hệ thống với model 60 khung hình trên một video giả thời lượng 10s



Hình ảnh 33. Dự đoán video Deepfake model 60 khung hình

Kiểm thử hệ thống với model 10 khung hình cùng một video với model 60 khung hình

Hình ảnh 34. Dự đoán video Deepfake trên model 10 khung hình

3. KẾT LUẬN KIỂM THỬ

Có thể thấy thông qua kết quả kiểm thử, các mô hình đã đạt được kết quả vô cùng khả quan. Xét *Bảng 1. Đánh giá độ chính xác của model* các mô hình được huấn luyện với số lượng khung phân tách càng cao sẽ đưa ra kết quả càng chuẩn xác. Nhìn chung tất cả các mô hình đều đưa ra được con số đánh giá rất đáng mong đợi với độ chính xác thấp nhất là 84.21461% của model phân tách 10 khung hình và độ chính xác cao nhất là 93.58794% của model phân tách 100 khung hình.

Khi ứng dụng mô hình phát hiện Deepfake trên một hệ thống thực tế cũng cho thấy rằng mô hình có thể kết hợp tốt với các môi trường khác và cho ra kết quả dự đoán đáng mong đợi *Hình ảnh 33. Dự đoán video Deepfake model 60 khung hình* và *Hình ảnh 34. Dự đoán video Deepfake model 10 khung hình.* Cả hai mô hình đều dự đoán chính xác kết quả là 100% và 96,4%.

KẾT LUẬN

1. KẾT QUẢ ĐẠT ĐƯỢC

Kết quả nghiên cứu của chúng tôi đã thành công trong việc ứng dụng công nghệ Deeplearning trong việc phát hiện Deepfake dựa trên mô hình chính là ResNeXt và mạng nơ-ron hồi quy Long Short-Term Memory. Thông qua quá trình đánh giá kiểm thử cho thấy các mô hình được huấn luyện đạt được độ dự đoán chính xác vô cùng cao vượt qua kì vọng ban đầu. Cuối cùng nghiên cứu cũng đã thành công trong việc ứng dụng mô hình vào trong một hệ thống thực tế mở ra tiềm năng ứng dụng và phát triển mạnh mẽ của mô hình trong tương lai.

2. PHƯƠNG HƯỚNG PHÁT TRIỂN

Trong tương lai, nghiên cứu có thể được mở rộng theo nhiều hướng để nâng cao hiệu quả và tính ứng dụng. Trước tiên, việc tối ưu hóa mô hình ResNeXt và LSTM, cũng như áp dụng transfer learning, sẽ giúp cải thiện độ chính xác và hiệu suất. Bên cạnh đó,việc mở rộng và đa dạng hóa tập dữ liệu sẽ giúp mô hình nhận diện được nhiều dạng Deepfake phức tạp hơn. Ngoài ra, để tăng tính thực tế, hệ thống cần được tối ưu hóa tốc độ xử lý và tích hợp vào các nền tảng phát hiện Deepfake theo thời gian thực, đồng thời phát triển một giao diện hoặc ứng dụng trực tuyến giúp người dùng dễ dàng kiểm tra nội dung giả mạo. Hơn nữa, nghiên cứu có thể kết hợp các phương pháp đa phương thức, như nhận diện giọng nói hoặc chuyển động khuôn mặt, nhằm tăng cường khả năng phát hiện Deepfake. Cuối cùng, việc ứng dụng các mô hình AI tiên tiến hơn, chẳng hạn như Vision Transformers (ViTs), cũng là một hướng đi tiềm năng. Với những phương hướng này, hệ thống sẽ trở thành một công cụ phát hiện Deepfake mạnh mẽ, có khả năng ứng dụng cao trong thực tế.

NGUỒN TÀI LIỆU THAM KHẢO

[1] David Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, 2002.

[2] Nina Schick, *Deepfakes: The Coming Infocalypse*, 2020.

[3] Manning Publications Co.,*Deep Learning with Python*, 2021

[4] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He: Aggregated Residual Transformations for Deep Neural Networks, 2016.

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep Learning, 2016.

[6] Saining Xie, Ross Girshick Piotr Dollar, Zhuowen Tu, Kaiming He, UC San Diego, Facebook AI Research, *Aggregated Residual Transformations for Deep Neutral Networks*, 2016.

[7] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, *"Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning"*, 2016.

[8] Zhuo Su, Linpu Fang, Wenxiong Kang, Dewen Hu, Matti Pietikäinen, and Li Liu, "Dynamic Group Convolution for Accelerating Convolutional Neural Networks," Proceedings of the European Conference on Computer Vision (ECCV), 2020.

[9] Quản Thành Thơ, *Mạng nơ-ron nhân tạo: từ hồi quy đến học sâu*, Nhà xuất bản Đại học Quốc gia Thành phố Hồ Chí Minh, 2021