MỤC LỤC

[Tóm tắt 2](#_Toc184842203)

[Danh sách hình vẽ 2](#_Toc184842204)

[1. Giới thiệu ( Introduction) 2](#_Toc184842205)

[**1.1 Tổng quan bài toán** 2](#_Toc184842206)

[**1.2 Bối cảnh và tầm quan trọng thực tiễn** 2](#_Toc184842207)

[**1.3 Vấn đề đặt ra** 2](#_Toc184842208)

[**1.4 Phương pháp, mô hình** 2](#_Toc184842209)

[**1.5 Mục tiêu mong muốn** 2](#_Toc184842210)

[2. Cơ sở lý thuyết (Background) 3](#_Toc184842211)

[2.1 Tổng quan mạng nơ-ron tích chập (CNN - Convolutional Neural Networks) 3](#_Toc184842212)

[2.1.1.1 Lịch sử của CNN 3](#_Toc184842213)

[2.1.1.2 CNN và bài toán thị giác máy tính 4](#_Toc184842214)

[2.1.1.3 Những layer chính của CNN 6](#_Toc184842215)

[***2.2 Mạng Recurrent Neural Network*** 12](#_Toc184842216)

[2.3 Mạng Long short term memory 14](#_Toc184842217)

[3. Phương pháp đề xuất (Methodology) 15](#_Toc184842218)

[**3.1 Mô hình và xử lý** 15](#_Toc184842219)

[3.1.1 Dữ liệu đầu vào 16](#_Toc184842220)

[3.1.2 Tiền xử lý dữ liệu 17](#_Toc184842221)

[3.1.3 Mô hình detect Deepfake 19](#_Toc184842222)

[3.2 Công cụ và môi trường 20](#_Toc184842223)

[**3.2.1 Công cụ** 20](#_Toc184842224)

[**3.2.2 Môi trường** 20](#_Toc184842225)

[4. Thực Nghiệm (Experiments) 21](#_Toc184842226)

[**4.1 Thiết lập thí nghiệm** 21](#_Toc184842227)

[4.2 Quy trình huấn luyện 21](#_Toc184842228)

[4.3 Kết quả và đánh giá 25](#_Toc184842229)

[6. Kết Luận (Conclusion) 27](#_Toc184842230)

[7. Tài liệu tham khảo 27](#_Toc184842231)

**Tóm tắt**

**Danh sách hình vẽ**

# 1. Giới thiệu ( Introduction)

**1.1 Tổng quan bài toán**

**1.2 Bối cảnh và tầm quan trọng thực tiễn**

**1.3 Vấn đề đặt ra**

**1.4 Phương pháp, mô hình**

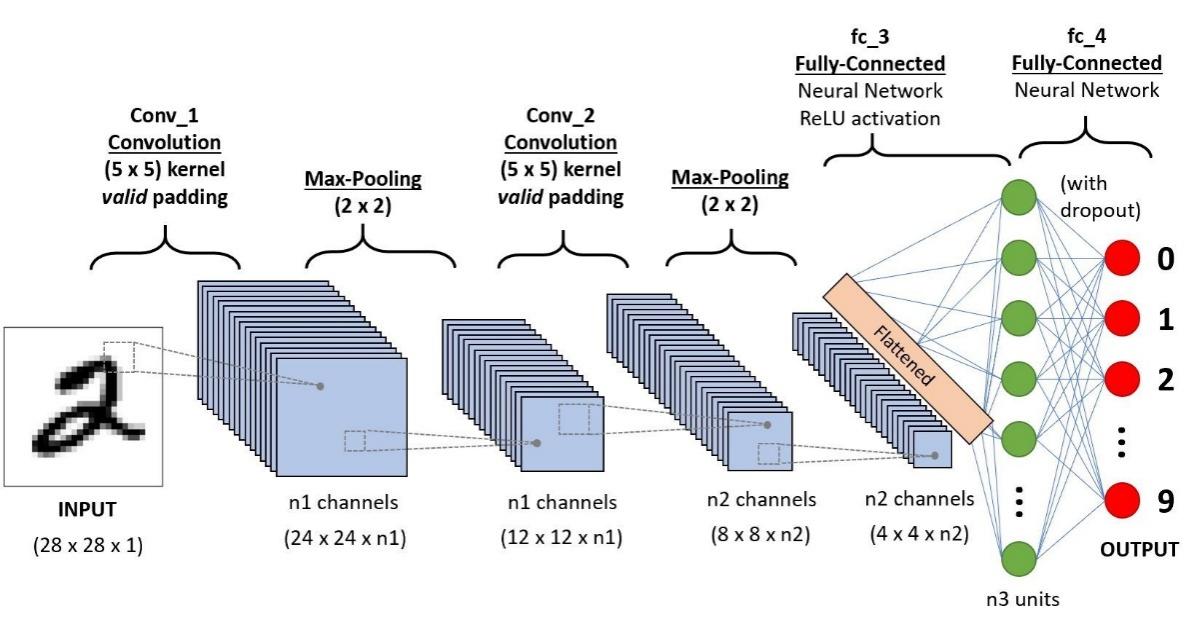
**1.5 Mục tiêu mong muốn**

# 2. Cơ sở lý thuyết (Background)

## 2.1 Tổng quan mạng nơ-ron tích chập (CNN - Convolutional Neural Networks)

Là một trong những kiến trúc mạng thần kinh phổ biến nhất trong thị giác máy tính. Các mạng CNN được thiết kế để xử lý dữ liệu có cấu trúc dạng lưới, chẳng hạn như hình ảnh.

Trong bài toán tổng hợp hình ảnh siêu phân giải, CNN giúp nhận diện các đặc trưng không gian của hình ảnh, chẳng hạn như các kết cấu, đường biên, và các đối tượng. Các lớp tích chập (convolution layers) trong CNN giúp mạng học các đặc trưng này từ dữ liệu huấn luyện. Trong SRGAN, mạng CNN được sử dụng để học cách chuyển đổi hình ảnh có độ phân giải thấp (LR) thành hình ảnh có độ phân giải cao (HR). Các lớp tích chập học các đặc trưng từ ảnh để tái tạo chi tiết và kết cấu của ảnh HR.



*Hình 6. Convolutional Neural Networks*

### 2.1.1.1 Lịch sử của CNN

Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) là một mô hình học máy có thể được sử dụng để xử lý dữ liệu hình ảnh và video. CNN được phát triển dựa trên nguyên tắc hoạt động của hệ thần kinh thị giác của con người. Lịch sử phát triển của CNN bắt đầu từ những năm 1960, khi Hubel và Wiesel phát hiện ra rằng các tế bào thần kinh trong não bộ của động vật có vú có thể nhận biết các tính năng cụ thể trong hình ảnh. Những phát hiện này đã truyền cảm hứng cho các nhà khoa học máy tính phát triển các mô hình CNN có thể học cách nhận ra các tính năng trong hình ảnh.

Một trong những mô hình CNN đầu tiên được phát triển bởi Fukushima vào năm 1979, mô hình này được sử dụng để nhận dạng chữ viết tay. Trong những năm 1980, CNN đã được sử dụng cho một số ứng dụng, bao gồm nhận dạng chữ viết tay, phân loại ảnh và phát hiện đối tượng. Tuy nhiên, CNN vẫn chưa được sử dụng rộng rãi vì các vấn đề về tính phức tạp và hiệu suất.

Năm 1998, LeCunetal.đã phát triển một mô hình CNN mới có tên là LeNet. Mô hình này đã đạt được thành công đáng kể trong việc nhận dạng chữ viết tay.

Năm 2012, AlexNet đã được phát triển bởi một nhóm nghiên cứu của Đại học Toronto gồm Alex Krizhevsky, Hya Sutskever và Geoffrey E. Hinton. AlexNet là một mô hình CNN lớn hơn và phức tạp hơn LeNet. AlexNet đã đạt được thành công vang dội trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC), một cuộc thi hàng năm về nhận dạng hình ảnh. Thành công của AlexNet đã thúc đẩy sự phát triển của CNN. Trong những năm gần đây, CNN đã được sử dụng rộng rãi cho nhiều ứng dụng, bao gồm nhận dạng hình ảnh, phân loại ảnh, phát hiện đối tượng, phân loại video và tạo hình ảnh. Sau sự xuất hiện của mạng AlexNet, nhiều mô hình mạng CNN nổi tiếng khác được ra đời bao gồm: VGGNet (2014), GoogleNet (2014), ResNet (2015), Inception V3 (2015), Xception (2016), DenseNet (2016),... Cho đến hiện nay, CNN hiện đang là một trong những mô hình học máy phổ biến nhất và được sử dụng trong nhiều ứng dụng thực tế, bao gồm: Nhận dạng khuôn mặt, Xác định đối tượng trong video, Phân loại ảnh y tế, Tạo hình ảnh chân dung,... CNN vẫn đang được tiếp tục nghiên cứu và phát triển. Các nhà khoa học vẫn đang liên tục tìm cách cải thiện hiệu suất của CNN và mở rộng ứng dụng của CNN sang các lĩnh vực mới.

### 2.1.1.2 CNN và bài toán thị giác máy tính

Trong học máy, trình phân loại sẽ gán nhãn lớp cho điểm dữ liệu. Ví dụ: bộ phân loại hình ảnh tạo ra nhãn lớp (ví dụ: chó mèo) cho những đối tượng tồn tại trong một hình ảnh. Mạng neural tích chập , hay gọi tắt là CNN, là một loại máy phân loại có khả năng giải quyết vấn đề này rất tốt!

CNN là một mạng lưới thần kinh: một thuật toán được sử dụng để nhận dạng các mẫu trong dữ liệu. Mạng nơ-ron nói chung bao gồm một tập hợp các nơ-ron được tổ chức thành các lớp, mỗi lớp có trọng số và độ lệch có thể học được riêng. Hãy chia CNN thành các khối xây dựng cơ bản của nó

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ

Mô tả được tạo tự động

*Hình 7. Cấu trúc cơ bản của mạng CNN*

***1. Tensor (Tensor):*** Tensor có thể được hiểu như một cấu trúc dữ liệu đa chiều, tương tự như ma trận nhưng với số chiều n. Trong mạng CNN, tensor thường có ba chiều (chiều cao, chiều rộng và chiều kênh) cho đến khi đến tầng đầu ra. Tensor là một khái niệm quan trọng trong xử lý dữ liệu đa chiều, như hình ảnh trong mạng CNN.

***2. Neuron (Nơ-ron):*** Một neuron có thể được xem như một hàm số nhận nhiều đầu vào và cho ra một đầu ra duy nhất. Các neuron thực hiện các phép tính trên dữ liệu đầu vào để tạo ra các đặc trưng hữu ích cho việc phân loại ảnh.

***3. Layer (Tầng):*** Tầng trong mạng CNN là một nhóm các neuron thực hiện cùng một phép toán. Các neuron trong cùng một tầng thường có các siêu tham số giống nhau, và tầng thường là các khối xây dựng cơ bản trong kiến trúc mạng CNN.

***4. Kernel weights and biases*** (Trọng số và bias của kernel): Đây là các tham số cho mỗi neuron trong mạng CNN. Trọng số được sử dụng để trọng số hóa đầu vào, trong khi bias được sử dụng để điều chỉnh đầu ra của neuron. Các giá trị này được tinh chỉnh trong quá trình huấn luyện mạng và cho phép mạng thích nghi với dữ liệu cung cấp.

***5. Class score:*** Một CNN truyền tải một hàm số điểm khả vi, được biểu diễn dưới dạng điểm số lớp trong tầng đầu ra của nó. Hàm số điểm này có tính chất liên tục và có thể tích hợp và tạo đạo hàm, điều này quan trọng trong quá trình huấn luyện mạng nơ-ron.Trong tầng đầu ra của mạng CNN, các class score được tính toán cho mỗi lớp hoặc nhãn mà mạng được đào tạo để phân loại. Điểm số này thường đại diện cho mức độ tự tin của mạng đối với việc phân loại hình ảnh thành các lớp khác nhau. Bằng cách so sánh điểm số giữa các lớp, chúng ta có thể xác định lớp mà mạng dự đoán là chính xác nhất cho hình ảnh đầu vào.

CNNs đã chứng tỏ họ là một công cụ mạnh mẽ cho các ứng dụng thị giác máy tính và đã đạt được hiệu suất ấn tượng trong việc giải quyết các vấn đề liên quan đến xử lý hình ảnh. Tiếp sau đây ta sẽ đi vào việc hoạt động của các layers chính trong một mạng CNN.

### 2.1.1.3 Những layer chính của CNN

**a. Tầng đầu vào (Input layer)**

Tầng đầu vào (input layer) trong mạng nơ-ron tích chập (CNN) là phần đầu tiên của mạng, và nó có nhiệm vụ nhận và xử lý dữ liệu đầu vào, thường là hình ảnh hoặc dữ liệu có cấu trúc tương tự. Dưới đây là một số điểm quan trọng về tầng đầu vào trong CNN:

***1. Dữ liệu đầu vào:*** Tầng đầu vào chứa dữ liệu đầu vào, thường là một hình ảnh hoặc một tensor đa chiều thể hiện hình ảnh. Dữ liệu này thường có kích thước cố định và được xử lý bằng cách chuyển đổi nó thành các đặc trưng phù hợp cho việc trích xuất thông tin từ hình ảnh.

***2. Kích thước đầu vào:*** Mạng CNN cần biết kích thước của dữ liệu đầu vào để thiết lập kích thước tầng đầu ra cho các tầng tích chập sau. Thông thường, kích thước đầu vào được chỉ định trước khi xây dựng mạng.

***3. Kênh đầu vào:*** Hình ảnh màu thông thường có ba kênh (RGB), trong khi hình ảnh xám chỉ có một kênh. Tầng đầu vào phải có số kênh tương thích với đầu vào.

***4. Tiền xử lý đầu vào:*** Trong một số trường hợp, tiền xử lý có thể được thực hiện trên dữ liệu đầu vào để chuẩn bị nó cho việc đưa vào mạng CNN. Điều này có thể bao gồm việc chuẩn hóa giá trị pixel hoặc thay đổi kích thước hình ảnh để phù hợp với kích thước mạng.

***5. Thông tin đầu ra:*** Tầng đầu vào không thực hiện bất kỳ tính toán nào trên dữ liệu, nó chỉ đơn giản là nhận dữ liệu đầu vào và truyền nó xuống các tầng tích chập đầu tiên trong mạng CNN.

Tầng đầu vào là một phần quan trọng của mạng CNN vì nó đảm bảo rằng dữ liệu đầu vào được chuẩn bị đúng cách và truyền vào mạng để bắt đầu quá trình học tập và trích xuất đặc trưng từ hình ảnh.

**b. Lớp tích chập (Convolution Layer):**

CNNs sử dụng một loại tầng đặc biệt gọi là tầng tích chập (convolutional layer), điều này làm cho chúng rất thích hợp để học từ dữ liệu hình ảnh và dữ liệu giống hình ảnh. Về dữ liệu hình ảnh, CNN có thể được sử dụng cho nhiều nhiệm vụ thị giác máy tính khác nhau, như xử lý hình ảnh, phân loại, phân đoạn và phát hiện đối tượng.

Tầng tích chập trong CNN cho phép mạng học cách phát hiện các đặc trưng cụ thể trong hình ảnh, như cạnh, góc, hoặc các đặc điểm quan trọng khác. Điều này làm cho CNN trở nên rất hiệu quả trong việc xử lý hình ảnh vì chúng có khả năng chia sẻ trọng số (weight sharing) và sử dụng các tầng tích chập để trích xuất thông tin cục bộ từ hình ảnh.

***Phép toán tích chập***

* Tích chập một chiều (conv1D): Xét tín hiệu một chiều a(t) và bộ lọc (filter) w(t).Tích chập của tín hiệu và bộ lọc là một tín hiệu một chiều mới b(t) được xác định theo công thức:

Ảnh có chứa Phông chữ, chữ viết tay, màu trắng, thư pháp

Mô tả được tạo tự động

Nếu giả sử rằng a(t) và w(t) được định nghĩa chỉ trên giá trị nguyên của t, khi đó phép tích chập một chiều có thể được định nghĩa như sau:

Ảnh có chứa Phông chữ, màu trắng, văn bản, thiết kế

Mô tả được tạo tự động

Ví dụ về phép tích chập một chiều rời rạc (không có padding) được minh họa trong Hình 8.

Ảnh có chứa hàng, Hình chữ nhật

Mô tả được tạo tự động

*Hình 8. Tích chập một chiều*

* Tích chập 2 chiều (conv2D): Tương tự ta xét tín hiệu đầu vào 2 chiều I(i, j) và kernel 2 chiều K(i, j). Giả sử I, K được định nghĩa chỉ trên giá trị nguyên của i, j, tức là ma trận 2 chiều, khi đó, phép tích chập 2 chiều với các biến được định nghĩa như sau:

Ảnh có chứa Phông chữ, văn bản, màu trắng, thư pháp

Mô tả được tạo tự động

* Do tính giao hoán của phép tích chập, phép tích chập 2 chiều của I và K có thể được viết như sau:

Ảnh có chứa Phông chữ, văn bản, màu trắng, thư pháp

Mô tả được tạo tự động

* Khi bộ lọc hay kernel có tính đối xứng, phép tích chập trùng với phép tương quan (cross-correlation). Trong nhiều tài liệu học máy, phép tích chập được cài đặt bằng phép tương quan chéo:

Ảnh có chứa Phông chữ, văn bản, màu trắng, thuật in máy

Mô tả được tạo tự động

Khi thực hiện tích chập, ta cần phải quan tâm tới một số hyperparameter cho filter đó là size, stride và padding của filter định rõ cách mà filter di chuyển qua đầu vào và cách nó tương tác với dữ liệu để trích xuất đặc trưng.

***1. Kích thước (Size):*** Đây là kích thước của filter, thường được định nghĩ bằng chiều rộng và chiều cao của filter. Ví dụ, filter 3x3 là ma trận gồm 3 dòng, 3 cột. Nếu kernel là ma trận vuông, ta xét kernel có kích thước f × f.

***2. Bước nhảy (Stride):*** Stride là khoảng cách giữa các vị trí liền tiếp mà filter được áp dụng trên đầu vào. Stride xác định cách mà filter di chuyển qua dữ liệu đầu vào. Ví dụ, với stride bằng 1, filter di chuyển mỗi pixel một lần, trong khi với stride bằng 2, filter nhảy qua một pixel mỗi lần. Ký hiệu là s.

***3. Đệm (Padding),*** ký hiệu là p: Padding là việc thêm giá trị 0 hoặc giá trị khác vào xung quanh đầu vào trước khi filter được áp dụng. Padding có thể giúp duy trì kích thước đầu ra sau khi áp dụng filter, và nó cũng có thể giúp việc trích xuất đặc trưng từ biên của hình ảnh trở nên hiệu quả hơn. Có hai loại padding phổ biến: "same" (duy trì kích thước: p = (f − 1)/2) và "valid" (không padding: p = 0).

Sau khi áp dụng phép tích chập 2 chiều cho tín hiệu đầu vào I kích thước n × n với kernel f × f, bước nhảy s và giá trị đệm p, ta sẽ thu được ảnh mới. Ví dụ quá trình áp dụng phép tích chập 2 chiều cho một số giá trị f, s, p:

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

*Hình 9. Phép tích chập hai chiều ảnh 6 × 6*

* Tính toán trong Conv2D

Ảnh có chứa biểu đồ, ảnh chụp màn hình, Hình chữ nhật, hình vuông

Mô tả được tạo tự động

*Hình 10. Ví dụ tính toán trong Conv2D*

Lớp Conv2D chứa nf bộ lọc (kích thước 3×3×nc, 5×5×nc, . . .), với nc là số kênh của đầu vào. Đầu vào nhân chập với từng bộ lọc và cho qua hàm kích hoạt σt tạo ra các bản đồ đặc trưng (feature map). Ghép các bản đồ đặc trưng lại với nhau ta được ảnh đầu ra của lớp conv2D gồm nf kênh.

Ví dụ: Trong Hình 10, ảnh đầu vào có nc = 3 kênh và có nf = 2 bộ lọc, 16 nơ-ron trong bản đồ đặc trưng chia sẻ các trọng số trong bộ lọc 1, 16 nơ-ron trong bản đồ đặc trưng phía dưới chia sẻ trọng số trong bộ lọc 2.

Bản chất conv2D vẫn là 1 lớp mạng trong MLP (Multi layer perceptron) nhưng các trọng số được chia sẻ giữa các nơ-ron (làm giảm tổng số tham số). Đồng thời do tính chất của phép tích chập và kích thước bộ lọc nhỏ, kết quả tính toán thể hiện các tính chất cục bộ của ảnh đầu vào (dẫn đến tên gọi là bản đồ đặc trưng).

* ***Hàm kích hoạt (activation function)***

Sau khi tính toán tích chập, đầu ra sẽ được đưa qua 1 hàm activation function, minh họa trong Hình 11. Một phần lý do mà những mạng CNN đột phá này có thể đạt được độ chính xác khổng lồ là vì tính phi tuyến tính của hàm kích hoạt.

Ảnh có chứa biểu đồ, hàng, văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

*Hình 11. Sử dụng hàm kích hoạt trong conv2D.*

Tính phi tuyến ở hàm kích hoạt là cần thiết để tạo ra biến quyết định phi tuyến tính, để đầu ra không thể được biểu diễn như một tổ hợp tuyến tính của các đầu vào. Nếu không có hàm kích hoạt phi tuyến tính, kiến trúc mạng CNN sâu sẽ trở thành một lớp tích chập đơn, không hoạt động hiệu quả bằng cách sử dụng hàm kích hoạt phi tuyến tính.

***Hàm kích hoạt ReLU:*** Hàm kích hoạt ReLU được sử dụng cụ thể như một hàm kích hoạt phi tuyến tính, thay vì các hàm phi tuyến tính khác như Sigmoid, vì nhiều thực nghiệm trong các nghiên cứu chỉ ra rằng CNN sử dụng hàm kích hoạt ReLU huấn luyện có tốc độ nhanh hơn so với các hàm kích hoạt khác. Hàm kích hoạt này được áp dụng trên từng phần tử trên mọi giá trị từ Tensor đầu vào.

Ảnh có chứa Phông chữ, văn bản, màu trắng, thiết kế

Mô tả được tạo tự động

***Hàm kích hoạt Softmax:*** Hàm Softmax là hàm đảm bảo tổng đầu ra của CNN bằng 1. Vì điều này, hàm Softmax được sử dụng để chuyển đổi đầu ra của mạng thành một phân phối xác suất trên các lớp (hoặc các lựa chọn).

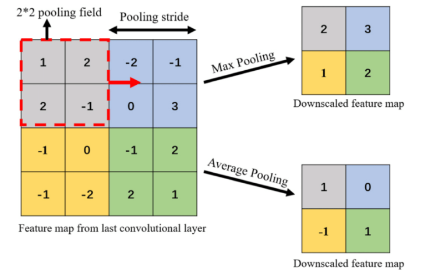
Ảnh có chứa Phông chữ, màu trắng, văn bản, chữ viết tay

Mô tả được tạo tự động

Lớp hoặc lựa chọn với xác suất cao nhất sau khi áp dụng Softmax thường được chọn là dự đoán của mô hình cho đầu vào cụ thể. Hàm Softmax giúp chuyển đổi kết quả thành xác suất, và do đó, nó rất hữu ích trong các tác vụ phân loại, nơi chúng ta muốn biết xác suất của việc một mẫu thuộc về từng lớp. Trong mạng CNN, tầng cuối cùng thường sử dụng hàm Softmax để trả về xác suất cho các lớp phân loại.

**c. Lớp Pooling**

Sau khi qua hàm kích hoạt từ Lớp tích chập, đầu ra (feature map) sẽ được cho qua 1 lớp gộp (Pooling Layer) để làm giảm kích thước dữ liệu tính toán xuống. Việc sử dụng lớp gộp giúp làm tăng tính bất biến, tính ổn định với các biến đổi nhỏ trong đầu vào. Một mạng CNN thường kết hợp nhiều lớp Conv2D và Pooling. Việc áp dụng Pooling sẽ áp dụng trên từng ma trận vuông con kích thước f × f với bước nhảy s.



*Hình 12. Average Pooling và Max Pooling với f = 2, s = 2*

Các lớp gộp phổ biến được xử dụng là Max Pooling, Average Pooling, Global Pool-ling. Ví dụ khi áp dụng Max Pooling và Average Pooling được minh họa trong Hình 12 với f = 2, s = 2. Còn Global Pooling đơn giản là áp dụng phép gộp trên ma trận vuông con có kích thước bằng feature map.

**d. Lớp làm phẳng (Flatten layer)**

Lớp làm phẳng (Flatten Layer) là lớp chuyển đổi lớp ba chiều trong mạng thành vector một chiều.

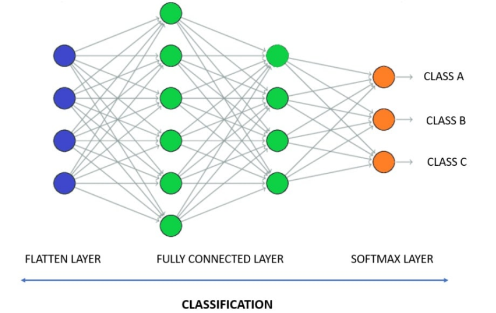
Ví dụ: một Tensor 5 × 5 × 2 sẽ được chuyển đổi thành vector có độ dài 5.5.2 = 50. Các lớp chập trước đó của mạng đã trích xuất các đặc điểm từ hình ảnh đầu vào, và bây giờ là lúc phân loại các đặc điểm. Chúng ta sử dụng hàm logistic/softmax để xác định độ tin cậy của các nhãn, hàm này yêu cầu đầu vào 1 chiều. Đây là lý do tại sao lớp làm phẳng là cần thiết.

**e. Lớp kết nối đầy đủ ( Fully Connected Layer)**

Lớp kết nối đầy đủ (còn được gọi là Lớp ẩn) là lớp cuối cùng trong mạng nơ-ron tích chập. Lớp này là sự kết hợp giữa hàm Affine và hàm phi tuyến tính:

* ***Hàm Affine:*** y= Wx+b
* ***Hàm phi tuyến:*** Sigmoid, Tanh, ReLU

Lớp kết nối đầy đủ lấy đầu vào từ Lớp Flatten. Dữ liệu đến từ Lớp Flatten trước tiên được chuyển đến hàm Affine và sau đó đến hàm Phi tuyến. Sự kết hợp của 1 hàm Affine và 1 hàm phi tuyến tính được gọi là 1 Fully Connected (Được kết nối hoàn toàn) hoặc 1 lớp ẩn. Lớp kết nối đầy đủ có thể thêm nhiều hidden layer như vậy dựa trên độ sâu mà ta muốn áp dụng mô hình phân loại của mình. Điều này hoàn toàn phụ thuộc vào tập dữ liệu huấn luyện.



*Hình 13. Lớp Fully Connected trong CNN*

Sự kết hợp của Lớp Flatten với Lớp Fully Connected và Lớp Softmax chính là mạng nơ-ron sâu (Deep Neural Network). Nếu chúng ta nhìn vào mạng lưới nơ-ron hoàn chỉnh, chúng ta sẽ thấy rằng các lớp ban đầu của mạng lưới nơ-ron tích chập bao gồm lớp tích chập Conv và lớp gộp Pooling. Lớp đầu ra trong mạng nơ-ron tích chập là lớp làm phẳng sử dụng hàm Softmax hoặc Sigmoid và tính toán chi phí (loss) bằng hàm Cross-entropy. Và cuối cùng sẽ có giá trị độ tin cậy của mỗi nhãn, nhãn có độ tin cậy cao nhất là nhãn cuối cùng của ảnh đầu vào. Kiến trúc mạng tổng quan của CNN được trình bày trong Hình 14.

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

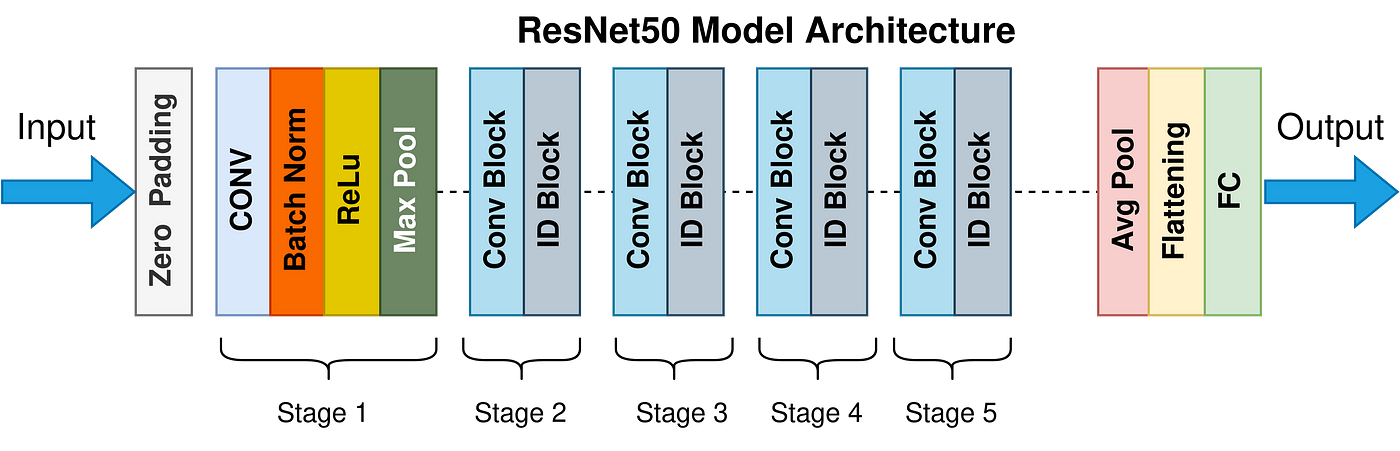
Mô tả được tạo tự động

*Hình 14. Kiến trúc mạng CNN*

*2.1.1.4 Một số mạng CNN điển hình*

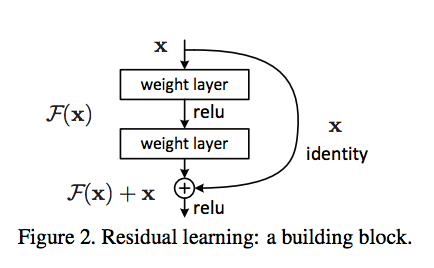
1. ***LeNet-5*** *(1998): Mạng CNN đầu tiên, được Yann LeCun đề xuất, sử dụng cho nhận diện chữ viết tay với cấu trúc gồm các lớp tích chập, pooling và fully connected.*
2. ***AlexNet*** *(2012): Đạt giải ImageNet Challenge, cải thiện hiệu năng nhờ ReLU, dropout và GPU, mở ra kỷ nguyên Deep Learning.*
3. ***VGGNet*** *(2014): Sử dụng tích chập 3x3 và tăng độ sâu mạng (16–19 lớp) để cải thiện khả năng trích xuất đặc trưng.*
4. ***GoogLeNet/Inception*** *(2014): Đề xuất khối Inception để trích xuất đa tỷ lệ đặc trưng, giảm tham số với cơ chế bottleneck.*
5. ***ResNet*** *(2015): Sử dụng residual blocks để giải quyết vấn đề vanishing gradient, cho phép mạng rất sâu (50–152 lớp).*
6. ***DenseNet*** *(2017): Kết nối dày đặc giữa các lớp, tối ưu hóa truyền thông tin và gradient, giảm số tham số.*

**2.2 Mạng Resnet**

ResNet-50 (Residual Network 50) là một mạng học sâu nổi tiếng được phát triển bởi nhóm nghiên cứu của Kaiming He vào năm 2015, với mục tiêu giải quyết vấn đề suy giảm hiệu năng khi độ sâu của mạng tăng lên. Đây là một phiên bản cải tiến của ResNet, với tổng cộng 50 lớp bao gồm cả lớp tích chập, lớp kết nối đầy đủ và các lớp Residual Blocks. 

**Kiến trúc mạng Resnet50**

Điểm nổi bật của ResNet-50 chính là khái niệm **"residual learning"**, trong đó mỗi khối (Residual Block) không chỉ học bản đồ đầu ra trực tiếp mà còn học phần dư (residual) của đầu vào và đầu ra, nhờ đó giúp mạng dễ dàng học được các đặc trưng quan trọng mà không làm mất thông tin đã có.



**Nút tắt để kết nối thông tin trước đó**

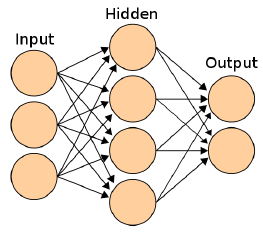
Kiến trúc của ResNet-50 bao gồm một chuỗi các khối tích chập với kích thước khác nhau, được kết hợp với các lớp chuẩn hóa (Batch Normalization) và các hàm kích hoạt phi tuyến (ReLU), đi kèm với các cơ chế bỏ mẫu (Pooling) và phức hợp (Fully Connected Layer) để tối ưu hóa quá trình học. ResNet-50 được chia thành năm giai đoạn chính (stages), mỗi giai đoạn bao gồm các khối residual với các lớp tích chập 1x1, 3x3 và 1x1, trong đó lớp 1x1 được dùng để giảm hoặc tăng số chiều dữ liệu, còn lớp 3x3 để trích xuất đặc trưng không gian.

Tổng cộng, mạng có hơn 23 triệu tham số, khiến nó đủ mạnh để học các đặc trưng phức tạp trên tập dữ liệu lớn như ImageNet, nhưng vẫn đảm bảo độ ổn định nhờ cơ chế residual. Một điểm khác biệt quan trọng của ResNet-50 so với các mô hình khác là khả năng duy trì hiệu năng cao ngay cả khi độ sâu mạng rất lớn, giúp nó trở thành một trong những mô hình phổ biến nhất trong các ứng dụng như nhận diện hình ảnh, phân loại đối tượng và các bài toán liên quan đến thị giác máy tính khác.

Cơ chế residual không chỉ giúp giảm vấn đề biến mất đạo hàm (vanishing gradient) mà còn cải thiện khả năng hội tụ, nhờ đó ResNet-50 đã trở thành một tiêu chuẩn cơ bản trong việc phát triển các mô hình học sâu hiện đại. Với cấu trúc linh hoạt và hiệu năng mạnh mẽ, ResNet-50 vẫn là sự lựa chọn hàng đầu trong nhiều nghiên cứu và ứng dụng thực tế ngày nay.

***2.3 Mạng Recurrent Neural Network***

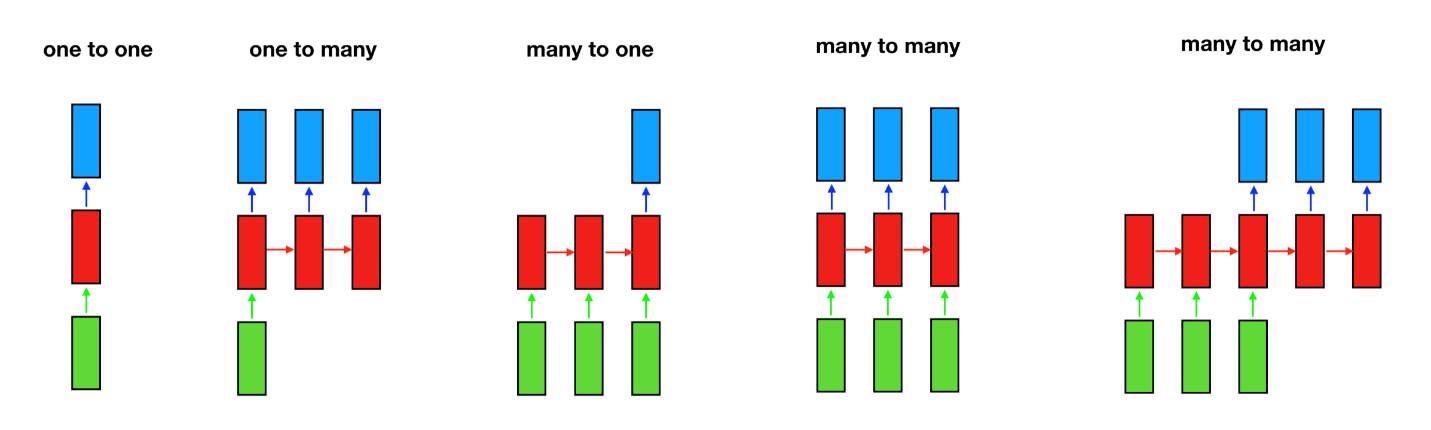
Neural Network truyền thống thường được xây dựng với kiến trúc gồm ba phần chính: lớp đầu vào (Input layer), lớp ẩn (Hidden layer) và lớp đầu ra (Output layer).



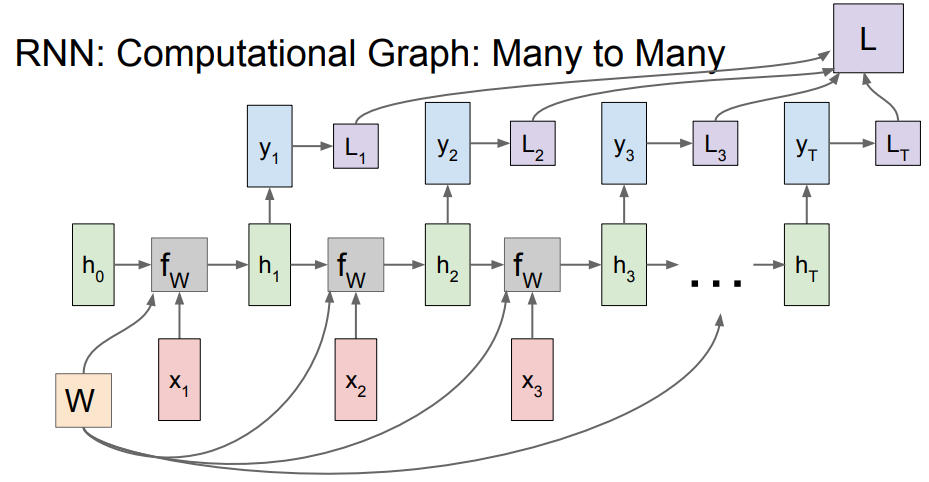
**Cấu trúc layer của mạng neural thông thường**

Tuy nhiên, trong cấu trúc này, đầu vào và đầu ra được xử lý như những đại lượng độc lập, không có sự liên hệ dài hạn giữa các bước tính toán. Điều đó dẫn đến việc mô hình này tỏ ra không phù hợp trong các bài toán xử lý chuỗi, nơi mà quá trình dự đoán không chỉ dựa trên một dữ liệu duy nhất tại thời điểm hiện tại mà còn phải xét đến vị trí và ngữ cảnh hình thành từ những bước trước. Chẳng hạn, trong bài toán xử lý ngôn ngữ tự nhiên, việc dự đoán từ tiếp theo trong một câu không thể bỏ qua thông tin từ các từ đi trước, do nghĩa, ngữ pháp hay ngữ cảnh thường thay đổi và phụ thuộc vào chuỗi từ đã xuất hiện trong thời gian trước đó.

Chính vì thế, Mạng nơ-ron truy hồi (RNN) ra đời với ý tưởng cốt lõi là bổ sung một thành phần “bộ nhớ” dưới dạng trạng thái ẩn. Bộ nhớ này được cập nhật liên tục sau mỗi bước thời gian, giữ lại thông tin về những từ hoặc yếu tố ở bước tính trước, và từ đó hỗ trợ quá trình ra quyết định ở bước hiện tại. Nhờ có RNN, việc xử lý chuỗi trở nên khả thi hơn, cho phép mô hình nắm bắt được mối quan hệ phụ thuộc giữa các phần tử trong dữ liệu thời gian, như các từ trong một câu hay các khung hình trong một chuỗi video, từ đó mang lại kết quả dự đoán chính xác và tự nhiên hơn.



Các dạng cấu trúc của RNN



Hình này minh họa cấu trúc của một mạng nơ-ron truy hồi (RNN) theo dạng **Many-to-Many**, thường sử dụng trong các bài toán chuỗi như dịch máy hoặc phân loại chuỗi. Điểm chính trong hình là cách thông tin được lưu giữ và truyền qua các bước thời gian, cụ thể:

1. **Biến đầu vào x1,x2,x3,...,xT ​:** Đây là dữ liệu đầu vào theo chuỗi (ví dụ: từng từ trong một câu hoặc tín hiệu thời gian).
2. **Trạng thái ẩn:** Mỗi trạng thái ẩn tại thời điểm t (h1,h2...,hT​) được tính dựa trên dữ liệu đầu vào x(t) và​ trạng thái ẩn trước đó h(t-1)​. Nó hoạt động như một "bộ nhớ" để lưu thông tin ngữ cảnh từ các bước trước đó.
3. **Dự đoán đầu ra​:** Tại mỗi bước thời gian, mô hình tạo ra đầu ra tương ứng y1,y2,..,yT​ dựa trên trạng thái ẩn tại thời điểm đó.
4. **Hàm mất mát:** Được tính riêng tại mỗi bước để đánh giá sự khác biệt giữa đầu ra dự đoán và đầu ra thực tế, sau đó tổng hợp thành hàm mất mát tổng thể L

So với mạng truyền thống như CNN, RNN xử lý dữ liệu tuần tự bằng cách duy trì thông tin qua thời gian, cho phép mô hình học được các mối quan hệ phụ thuộc dài hạn trong dữ liệu. Cách kết nối thông tin qua các trạng thái ẩn chính là chìa khóa để xử lý chuỗi trong RNN.

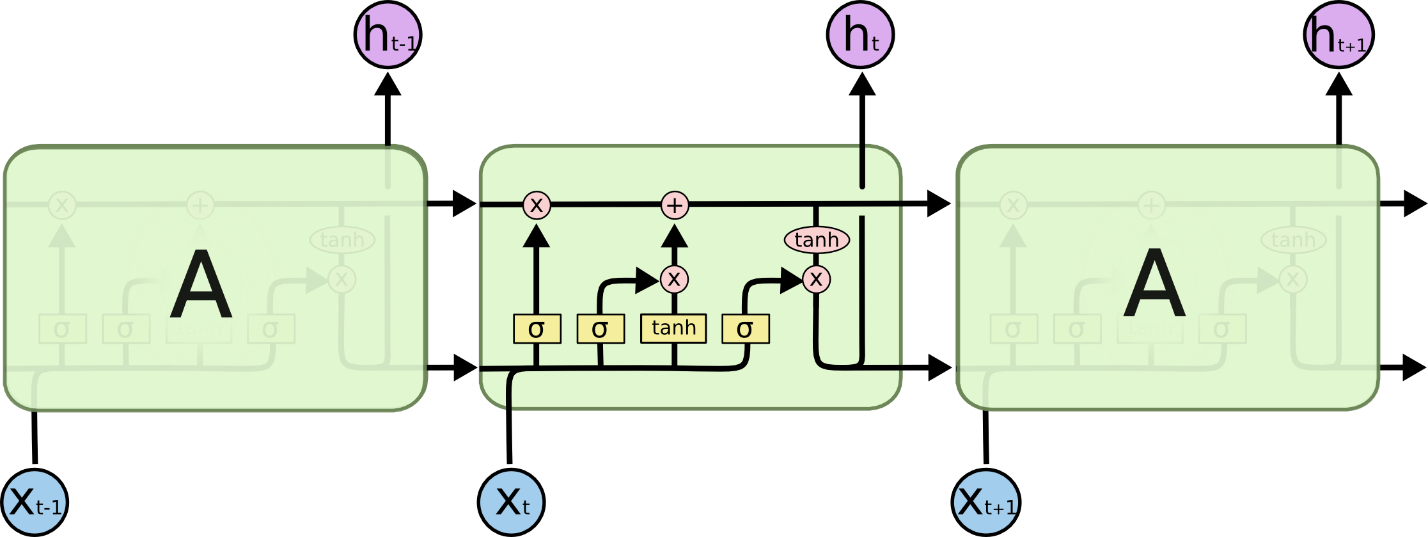
**Hạn chế**

RNN có khả năng kết nối thông tin từ quá khứ để thực hiện dự đoán hiện tại. Trong các ngữ cảnh ngắn hạn, RNN hoạt động hiệu quả, nhưng với các ngữ cảnh dài hạn (long-term dependencies), như việc dự đoán từ "ướt" dựa trên thông tin về "trời mưa" và "không mang áo mưa", RNN gặp khó khăn trong việc học. Nguyên nhân chính là hiện tượng triệt tiêu đạo hàm (vanishing gradient) khi xử lý chuỗi dài. Để khắc phục, LSTM ra đời với cơ chế đặc biệt, giúp lưu giữ thông tin dài hạn hiệu quả hơn.

## 2.4 Mạng Long short term memory

LSTM (Long Short-Term Memory) là một biến thể đặc biệt của mạng RNN, được thiết kế để khắc phục các hạn chế của RNN truyền thống, đặc biệt là hiện tượng **triệt tiêu đạo hàm (vanishing gradient)**, vốn làm giảm khả năng học các phụ thuộc dài hạn trong chuỗi dữ liệu. Nói ngắn gọn thay vì lấy toàn bộ dữ liệu ở phía trước như RNN , người ta sẽ tinh chỉnh để quên hoặc lấy lượng bao nhiêu đó trong số đó thôi và điều này giúp LSTM có thể xử lý tốt các chuỗi dữ liệu dài mà vẫn tập trung vào đặc trưng chính.

LSTM được thiết kế với một kiến trúc đặc biệt gồm ba cổng chính để quản lý thông tin một cách hiệu quả, khác biệt hoàn toàn so với RNN truyền thống.



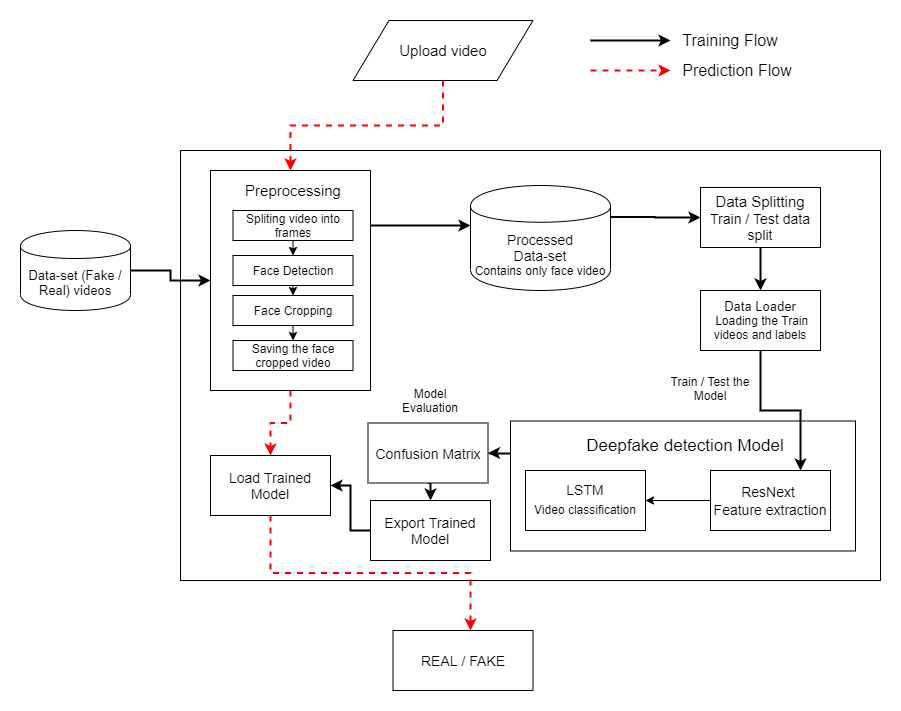
1. **Cổng quên (Forget Gate):** Đây là nơi quyết định xem thông tin nào từ trạng thái trước đó là không cần thiết và nên bị loại bỏ. Điều này giúp LSTM tập trung vào những thông tin quan trọng nhất, thay vì lưu giữ tất cả, như cách RNN làm, dẫn đến mất hiệu quả trong chuỗi dài.
2. **Cổng vào (Input Gate):** Cổng này chịu trách nhiệm thêm thông tin mới từ đầu vào hiện tại vào trạng thái bộ nhớ. Nó kiểm soát chính xác phần thông tin nào sẽ được lưu trữ để cập nhật bộ nhớ dài hạn, giúp LSTM học tốt hơn các phụ thuộc quan trọng trong chuỗi.
3. **Cổng đầu ra (Output Gate):** Sau khi xử lý thông tin, cổng đầu ra quyết định phần nào của trạng thái bộ nhớ sẽ được sử dụng để tạo ra đầu ra cho bước hiện tại. Điều này đảm bảo rằng chỉ những thông tin phù hợp với nhiệm vụ hiện tại được đưa vào dự đoán.

**Trạng thái bộ nhớ (Cell State):** Đây là một thành phần chính của LSTM, cho phép thông tin quan trọng được truyền qua nhiều bước thời gian mà không bị thay đổi quá nhiều. Điều này giúp khắc phục vấn đề mất mát thông tin mà RNN gặp phải.

So với RNN truyền thống, sự khác biệt lớn nhất của LSTM là khả năng duy trì và quản lý thông tin dài hạn thông qua cơ chế cổng và trạng thái bộ nhớ.

# 3. Phương pháp đề xuất (Methodology)

**3.1 Mô hình và xử lý**



Kiến trúc này là quy trình phát hiện deepfake từ video, bao gồm các bước chính:

* **Dữ liệu đầu vào**: Tập video (thật/giả) được đưa vào hệ thống.
* **Tiền xử lý**: Video được chia thành các khung hình, sau đó thực hiện phát hiện và cắt phần khuôn mặt, cuối cùng lưu lại video chỉ chứa khuôn mặt.
* **Xử lý dữ liệu**: Dữ liệu được chia thành tập huấn luyện và kiểm tra, sau đó tải vào hệ thống thông qua Data Loader.
* **Mô hình phát hiện deepfake**: Sử dụng ResNext để trích xuất đặc trưng từ video và LSTM để phân loại chuỗi video, xác định video thật hoặc giả.
* **Đánh giá mô hình**: Hệ thống đánh giá hiệu suất bằng ma trận nhầm lẫn và xuất mô hình đã huấn luyện.
* **Dự đoán**: Video đầu vào được phân loại thông qua luồng dự đoán, kết quả là thật hoặc giả.

Quy trình gồm hai luồng chính: huấn luyện (training flow) và dự đoán (prediction flow).

3.1.1 Dữ liệu đầu vào

Tập dữ liệu **Celeb-DF (Celeb-Deepfake)** là một tập dữ liệu nổi bật và được thiết kế đặc biệt để phục vụ nghiên cứu và phát triển các thuật toán phát hiện video giả mạo (deepfake). Celeb-DF được tạo ra để khắc phục một số hạn chế của các tập dữ liệu deepfake trước đây, như chất lượng hình ảnh thấp hoặc các đặc điểm giả mạo dễ nhận ra, từ đó nâng cao tính thực tế và độ khó trong việc phát hiện deepfake.

**Links:** [**Celeb-DF.zip - Google Drive**](https://drive.google.com/file/d/10NGF38RgF8FZneKOuCOdRIsPzpC7_WDd/view)

Celeb-DF bao gồm **590 gốc video thật** của các nhân vật nổi tiếng được trích xuất từ các cuộc phỏng vấn và các chương trình truyền hình.

Từ các video gốc, tập dữ liệu tạo ra hơn **5.639 video deepfake giả mạo**, với chất lượng cao hơn nhiều so với các tập dữ liệu deepfake khác như DeepFake-TIMIT.

Các video giả mạo trong Celeb-DF được tạo bằng công nghệ tiên tiến, mô phỏng sát với các đặc điểm của video thật. Hệ thống tạo deepfake được cải tiến để tránh các lỗi dễ phát hiện như:

Mô phỏng chuyển động miệng khớp với âm thanh.

Loại bỏ hiện tượng nhòe xung quanh vùng mặt.

Cải thiện các kết nối giữa vùng khuôn mặt giả mạo và nền (seamless blending).

Do đó, video giả mạo trong Celeb-DF rất thực tế, khó phân biệt bằng mắt thường.

Celeb-DF chứa các video có **nhiều điều kiện khác nhau** về ánh sáng, góc quay, chuyển động khuôn mặt, và biểu cảm.Bao gồm cả video ở các độ phân giải khác nhau, từ đó tăng tính đa dạng và thách thức cho các thuật toán phát hiện deepfake.

Celeb-DF được chia thành hai tập chính: **Celeb-DF v1** và **Celeb-DF v2**.

**Celeb-DF v1**: Phiên bản ban đầu, tuy nhiên bị hạn chế về chất lượng và tính đại diện

**Celeb-DF v2**: Phiên bản cải tiến, cung cấp video có chất lượng cao hơn, với sự tinh chỉnh về kỹ thuật tạo deepfake để tăng độ chân thực.

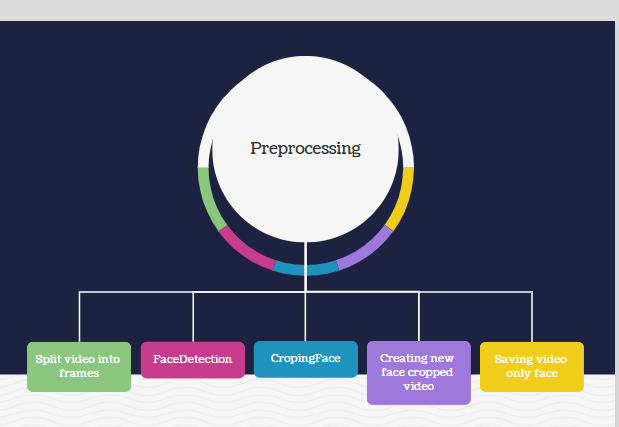
Celeb-DF được đánh giá cao vì:

* Chất lượng video giả mạo tốt hơn so với các tập dữ liệu khác.
* Độ khó cao, giúp phát triển các mô hình phát hiện deepfake mạnh mẽ hơn.
* Tính đa dạng và sự tương đồng với dữ liệu thực tế, hỗ trợ nghiên cứu ứng dụng sâu rộng.

Celeb-DF chủ yếu tập trung vào video của các nhân vật nổi tiếng, dẫn đến **thiếu tính đại diện** cho các dữ liệu video phổ thông.Số lượng video gốc thật còn tương đối hạn chế so với các tập dữ liệu lớn khác. Ngoài ra data chỉ chứa các video khuôn mặt, không hỗ trợ các nghiên cứu liên quan đến giả mạo toàn bộ cơ thể.

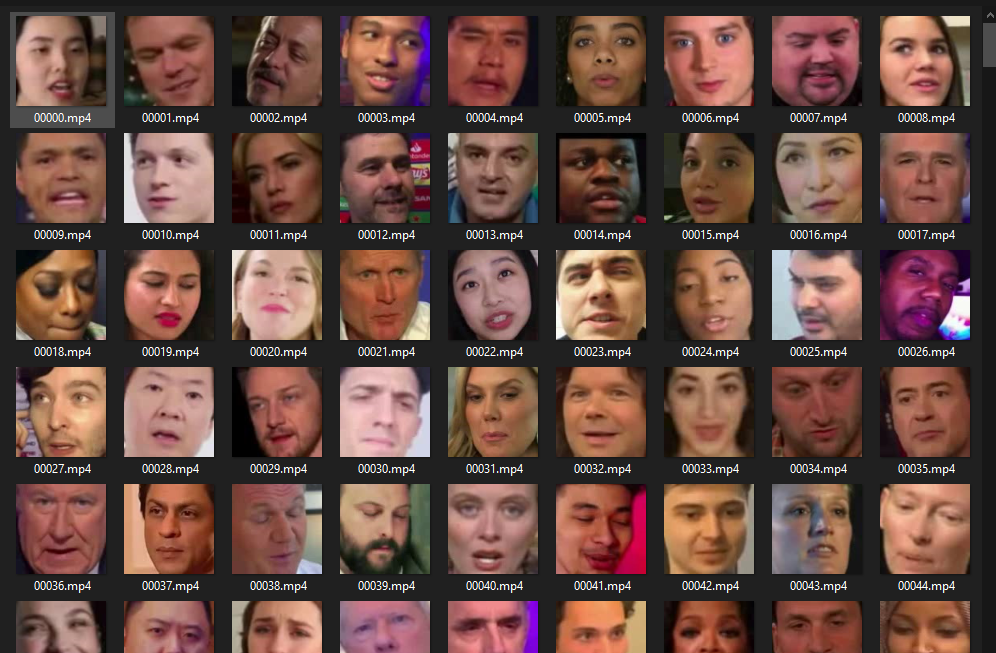
Celeb-DF là một tập dữ liệu được thiết kế chuyên biệt cho bài toán phát hiện deepfake, với các video giả mạo chất lượng cao và tính thực tế vượt trội. Nhờ độ khó cao và tính đa dạng, Celeb-DF là một trong những nguồn dữ liệu quan trọng nhất trong lĩnh vực nghiên cứu deepfake, góp phần phát triển các hệ thống phát hiện giả mạo hiện đại và hiệu quả hơn.

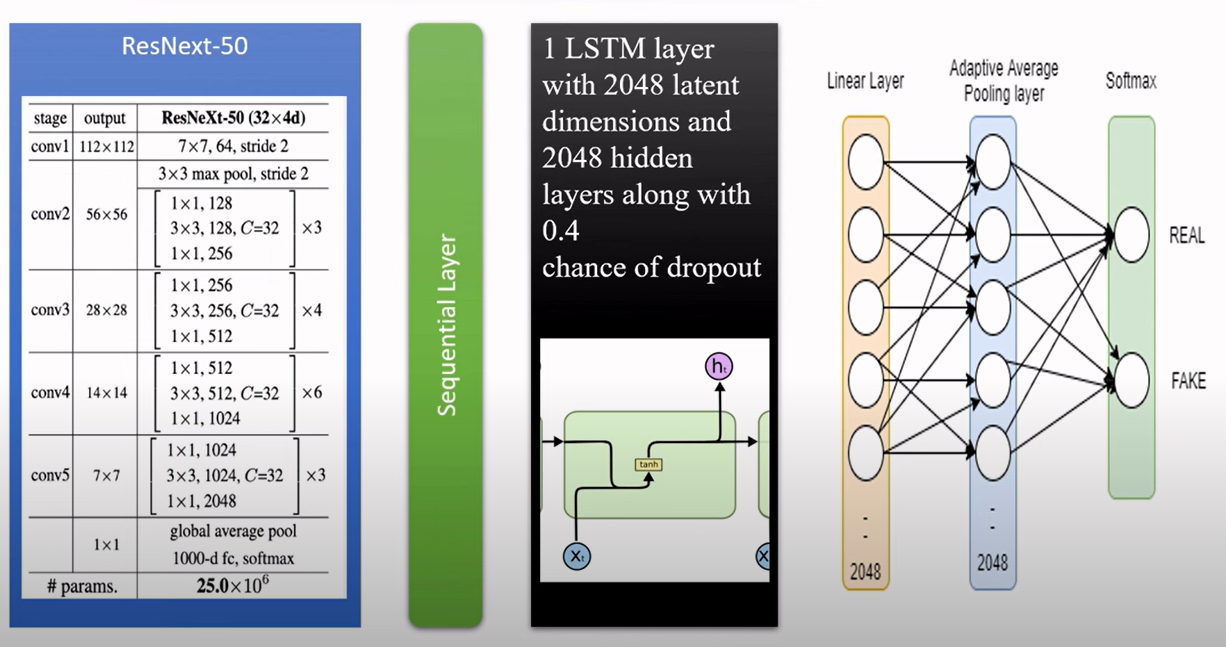
3.1.2 Tiền xử lý dữ liệu



Hình ảnh minh họa quy trình xử lý trước (preprocessing) trong một pipeline phát hiện deepfake, bao gồm năm bước chính.

Đầu tiên, video được chia thành các khung hình riêng lẻ (split video into frames) để chuẩn bị cho các bước xử lý tiếp theo. Sau đó, các khung hình được đưa vào bước phát hiện khuôn mặt (face detection), nơi thư viện sử dụng các thuật toán CNN để xác định vị trí khuôn mặt trong mỗi khung hình. Tiếp theo, khuôn mặt được cắt ra từ các khung hình này (cropping face) và chuyển thành các hình ảnh chỉ chứa khuôn mặt, giúp loại bỏ các yếu tố không liên quan trong nền video. Sau khi các khuôn mặt đã được trích xuất, hệ thống tạo ra một video mới chỉ chứa các khuôn mặt đã được cắt gọn (creating new face cropped video), đảm bảo video tập trung vào đối tượng quan trọng nhất là khuôn mặt. Cuối cùng, video chứa khuôn mặt được lưu lại (saving video only face) để sử dụng trong các bước phân tích và huấn luyện mô hình phát hiện deepfake tiếp theo, giúp cải thiện độ chính xác của hệ thống. Quy trình này không chỉ tối ưu hóa dữ liệu đầu vào mà còn giúp giảm thiểu nhiễu, tập trung vào đặc điểm chính phục vụ phát hiện deepfake.

3.1.3 Mô hình detect Deepfake



Hình ảnh minh họa kiến trúc mô hình phát hiện deepfake bao gồm bốn thành phần chính. Đầu tiên, mô hình sử dụng ResNeXt-50 làm bộ trích xuất đặc trưng với các tầng convolution (conv1 đến conv5) để trích xuất đặc trưng không gian từ video face. Sau đó, một tầng LSTM với 2048 chiều tiềm ẩn, 2048 chiều ẩn, và tỷ lệ dropout 0.4 được sử dụng để học thông tin thời gian từ chuỗi khung hình. Tiếp theo, tầng pooling thích ứng (Adaptive Average Pooling) giảm chiều dữ liệu trước khi đi qua tầng fully connected (Linear Layer). Cuối cùng, lớp Softmax dự đoán nhãn "REAL" hoặc "FAKE" dựa trên đầu ra của mạng. Kiến trúc này kết hợp mạnh mẽ giữa đặc trưng không gian và thời gian để phát hiện deepfake chính xác.

## 3.2 Công cụ và môi trường

**3.2.1 Công cụ**

Thư viện Python cho học sâu và xử lý ảnh:

+ TensorFlow/Keras: Dùng để xây dựng và huấn luyện mô hình nhờ vào khả năng tối ưu hóa GPU và hỗ trợ các cấu trúc mạng phức tạp.

+ PyTorch: Lựa chọn thay thế cho TensorFlow trong các thử nghiệm khác, với ưu điểm là dễ dàng kiểm soát và tùy chỉnh trong quá trình phát triển.

+ OpenCV: Sử dụng để xử lý dữ liệu hình ảnh đầu vào (chuyển đổi định dạng, resize, chuẩn hóa) và hiển thị kết quả sau khi tái tạo.

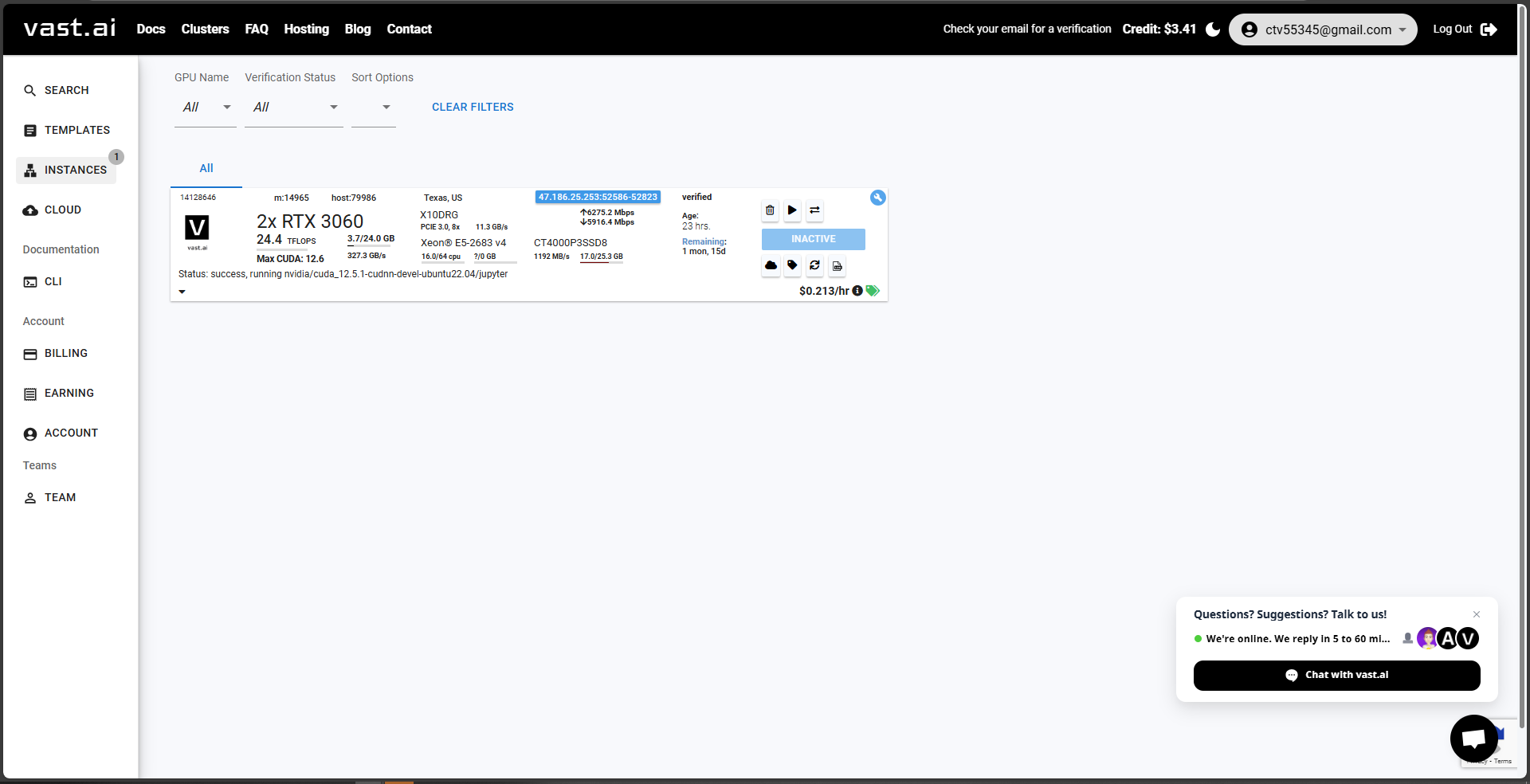
+ NumPy: Hỗ trợ các thao tác tính toán ma trận, cần thiết cho việc tiền xử lý và đánh giá mô hình.

+ Matplotlib và Seaborn: Vẽ biểu đồ và trực quan hóa quá trình huấn luyện

+Gradio: Phát triển giao diện để sử dụng

**3.2.2 Môi trường**

Phần cứng và phần mềm hỗ trợ:



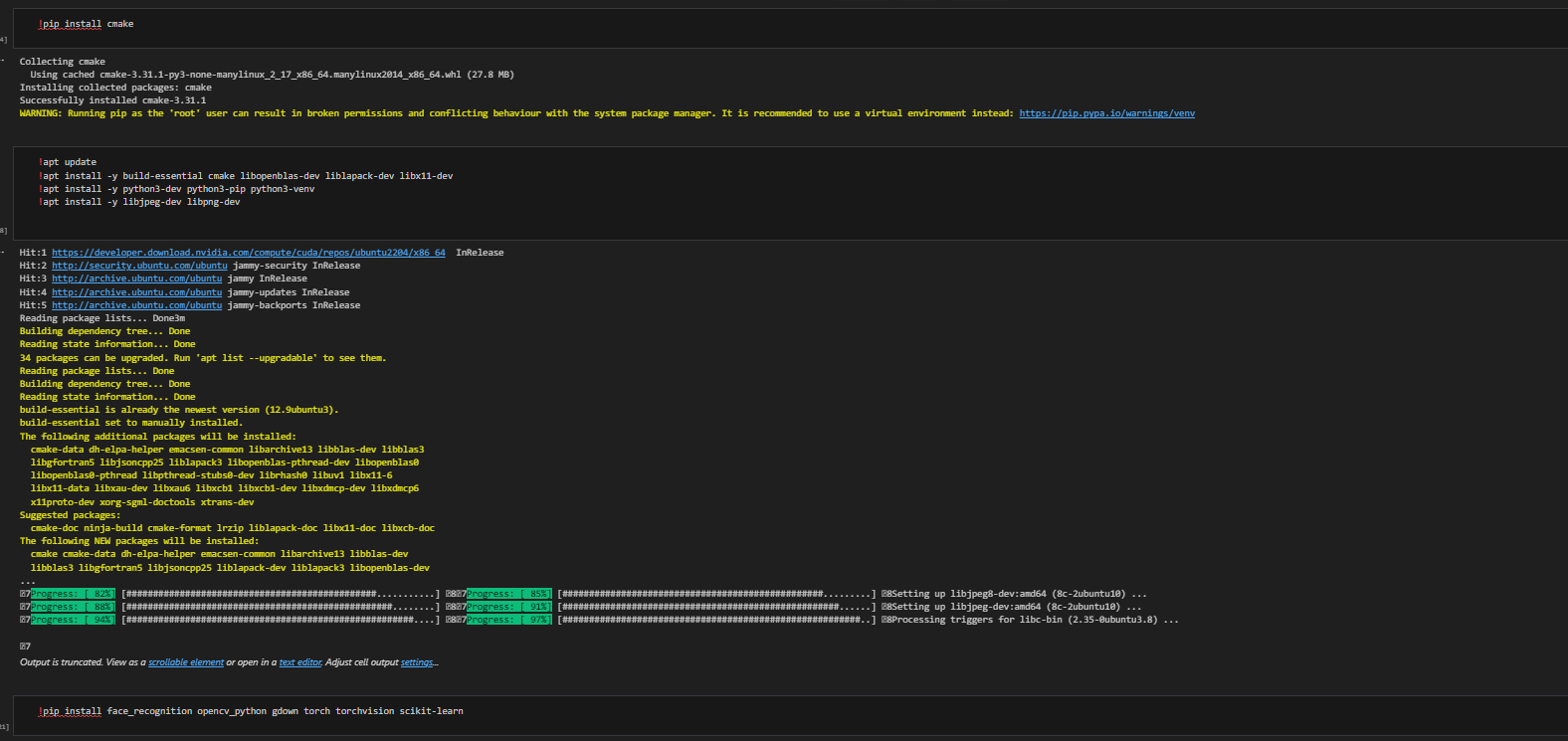
Hình ảnh minh họa môi trường chạy mô hình sử dụng dịch vụ Vast.ai, một nền tảng cung cấp tài nguyên GPU cho các tác vụ tính toán hiệu năng cao với chi phí hợp lý.

Cụ thể, trong thiết lập này, tôi đã chọn một phiên làm việc với GPU RTX 3060 có công suất tính toán lên đến 24.4 TFLOPS và dung lượng bộ nhớ CUDA tối đa 12.6 GB. Máy chủ được vận hành trên hệ điều hành Ubuntu 22.04, hỗ trợ CUDA 12.5.1 và cuDNN, cung cấp tài nguyên phù hợp cho các ứng dụng như huấn luyện mô hình học sâu. Với chi phí $0.213/giờ, môi trường này đảm bảo hiệu suất tốt cho các tác vụ yêu cầu cao như phát hiện deepfake hoặc xử lý video trong thời gian thực, tối ưu cả chi phí lẫn hiệu quả tính toán.

# 4. Thực Nghiệm (Experiments)

**4.1 Thiết lập thí nghiệm**

Trong thí nghiệm phát hiện deepfake, mã code được thiết kế để xử lý toàn bộ pipeline từ giai đoạn tiền xử lý dữ liệu đến huấn luyện và đánh giá mô hình. Đầu tiên, môi trường được thiết lập bằng cách cài đặt các thư viện cần thiết như cmake, opencv, torch, torchvision, face\_recognition, scikit-learn, cùng các gói hỗ trợ khác như build-essential, libopenblas-dev, liblapack-dev. Những công cụ này đảm bảo khả năng xử lý video, trích xuất khuôn mặt, và thực hiện huấn luyện mô hình trên GPU hiệu năng cao.



## 4.2 Quy trình huấn luyện

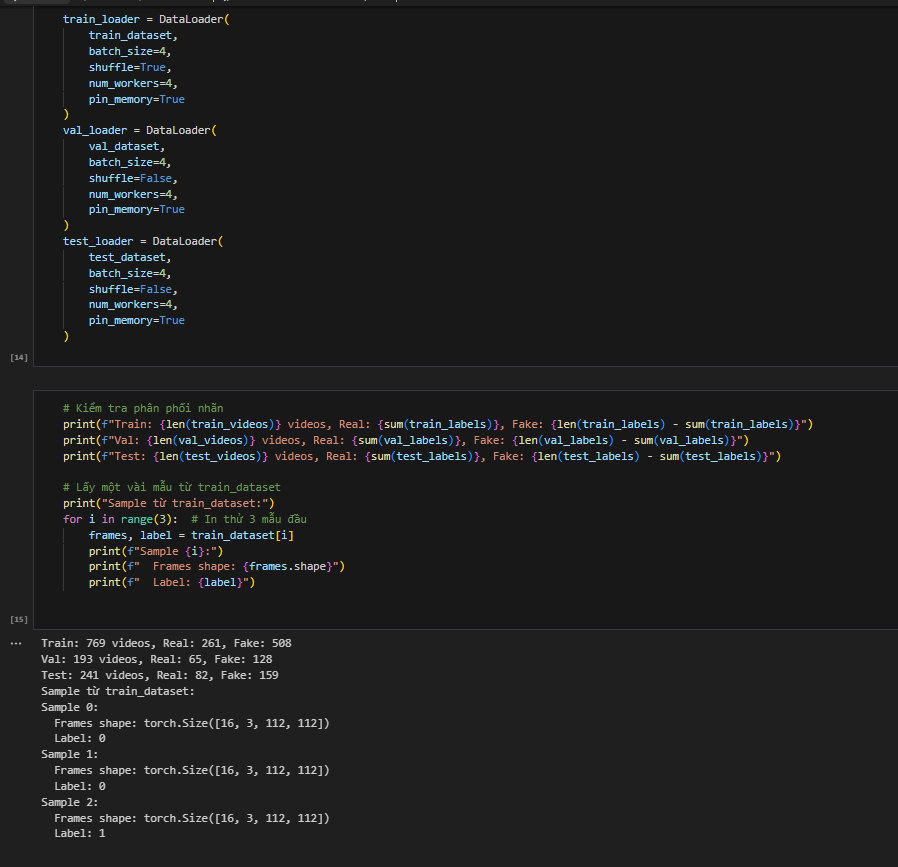
Dữ liệu thí nghiệm bao gồm các video gốc (real) từ tập "Celeb-real" và "YouTube-real", cùng các video giả (fake) từ "Celeb-synthesis". Các video này được tải xuống, giải nén và phân loại với nhãn tương ứng: 1 cho video thật (real) và 0 cho video giả (fake). Quá trình này đảm bảo rằng các nhãn được chuẩn bị đầy đủ và nhất quán trước khi chuyển sang bước tiền xử lý dữ liệu.

Quá trình tiền xử lý dữ liệu được thực hiện qua năm bước chính. Đầu tiên, mỗi video được tách thành các khung hình riêng lẻ bằng cách sử dụng OpenCV, cho phép xử lý từng khung hình độc lập. Sau đó, các khuôn mặt trong từng khung hình được phát hiện bằng thư viện face\_recognition, giúp xác định chính xác vị trí khuôn mặt trong mỗi khung. Tiếp theo, khuôn mặt được cắt từ các khung hình và chuẩn hóa kích thước về 112x112 pixel để đảm bảo đồng nhất dữ liệu. Những khuôn mặt đã được cắt sẽ được ghép lại thành các video chỉ chứa khuôn mặt, tạo điều kiện cho mô hình học tập tốt hơn. Cuối cùng, các video chứa khuôn mặt được lưu vào thư mục processed\_videos để sử dụng trong quá trình huấn luyện và kiểm định



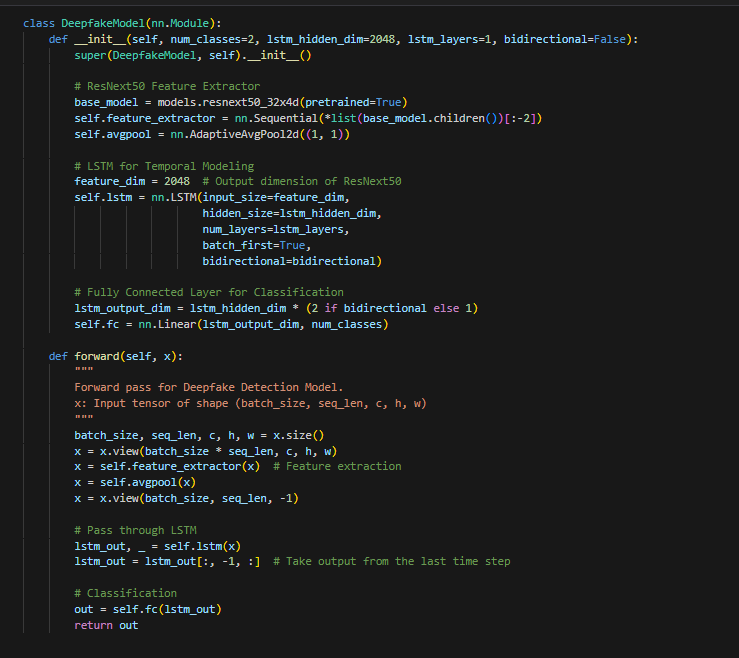
Hàm tách frame và trich xuất mặt.

Sau khi hoàn tất tiền xử lý, dữ liệu được chia thành ba tập: huấn luyện (train), kiểm định (validation), và kiểm thử (test). Phân chia này được thực hiện bằng phương pháp lấy mẫu phân tầng (stratified sampling) để đảm bảo tỷ lệ cân bằng giữa hai lớp real và fake. Cụ thể, tập dữ liệu được chia theo tỷ lệ 64%-16%-20% cho các tập train, val, và test. Việc phân chia hợp lý đảm bảo rằng mô hình có thể học tập trên dữ liệu đa dạng và được đánh giá một cách chính xác trên dữ liệu chưa từng thấy.



Chia tập dữ liệu

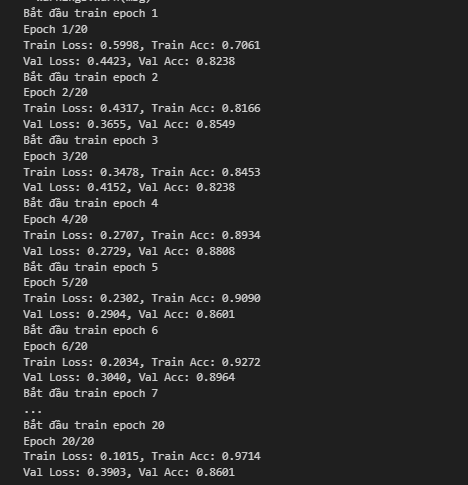
Trong bước xây dựng mô hình, kiến trúc ResNeXt-50 được sử dụng làm bộ trích xuất đặc trưng không gian từ các khung hình. Sau đó, một tầng LSTM với 2048 chiều tiềm ẩn và tỷ lệ dropout 0.4 được sử dụng để học mối quan hệ thời gian giữa các khung hình. Tiếp theo, tầng pooling thích ứng và tầng fully connected được sử dụng để dự đoán đầu ra REAL hoặc FAKE. Mô hình được tối ưu hóa bằng thuật toán Adam với learning rate 1×10−41 \times 10^{-4} và weight decay 1×10−51 \times 10^{-5}, trong khi CrossEntropyLoss được sử dụng làm hàm mất mát.



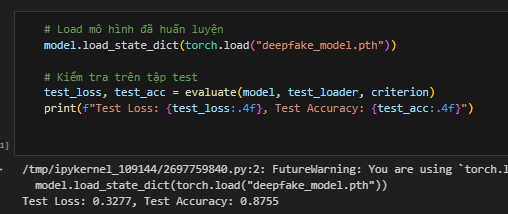
Kiến trúc mô hình

Thí nghiệm được thực hiện trong 20 epoch, với các đánh giá được thực hiện trên tập kiểm định sau mỗi epoch để đảm bảo mô hình không bị quá khớp. Cuối cùng, mô hình được lưu lại dưới dạng file deepfake\_model.pth và được nạp lại để kiểm tra trên tập kiểm thử. Các kết quả bao gồm độ chính xác, ma trận nhầm lẫn, và báo cáo phân loại, giúp đánh giá hiệu quả mô hình trong việc phân biệt video thật và giả. Quy trình này cung cấp một cách tiếp cận toàn diện để xử lý dữ liệu và huấn luyện mô hình phát hiện deepfake.

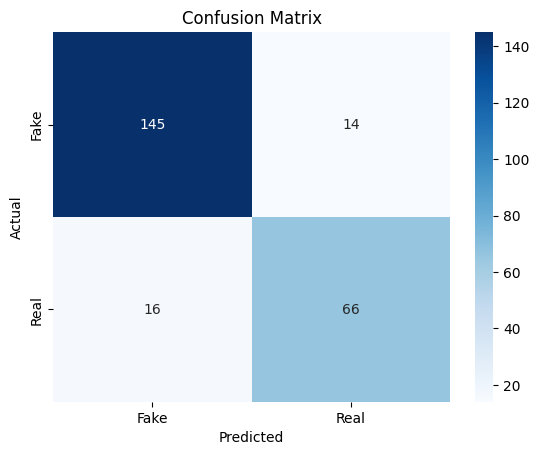
## 4.3 Kết quả và đánh giá



Kết quả huấn luyện và đánh giá mô hình phát hiện deepfake từ các hình ảnh cho thấy sự tiến bộ rõ ràng qua các epoch cũng như hiệu suất tổng thể khá tốt trên tập kiểm thử. Đầu tiên, trong giai đoạn huấn luyện, mô hình dần cải thiện qua từng epoch, thể hiện qua việc giảm đáng kể giá trị loss và tăng độ chính xác trên cả tập huấn luyện và tập kiểm định. Cụ thể, ở epoch 1, mô hình đạt train loss là 0.5998 với train accuracy là 70.61%, và val loss là 0.4423 với val accuracy là 82.38%. Đến epoch 20, train loss giảm mạnh xuống còn 0.1015 với train accuracy đạt 97.14%, trong khi val loss là 0.3903 và val accuracy đạt 86.01%. Điều này cho thấy mô hình không chỉ học tốt trên tập huấn luyện mà còn duy trì khả năng tổng quát trên tập kiểm định, giảm nguy cơ overfitting.



Khi tải mô hình đã huấn luyện để kiểm tra trên tập test, kết quả cho thấy mô hình đạt test loss là 0.3277 và test accuracy là 87.55%. Đây là một kết quả tốt, cho thấy mô hình hoạt động ổn định trên tập dữ liệu chưa từng thấy. Phân tích chi tiết bằng ma trận nhầm lẫn (confusion matrix) cho thấy rằng trong số 159 video fake, mô hình dự đoán chính xác 145 video (true positive), chỉ nhầm lẫn 14 video là real (false negative). Đối với video real, trong tổng số 82 video, mô hình dự đoán chính xác 66 video (true negative) và nhầm lẫn 16 video là fake (false positive). Điều này cho thấy mô hình có xu hướng hoạt động tốt hơn khi phân loại video fake so với video real.



precision recall f1-score support

Fake 0.90 0.91 0.91 159

Real 0.82 0.80 0.81 82

accuracy 0.88 241

macro avg 0.86 0.86 0.86 241

weighted avg 0.87 0.88 0.88 241

Báo cáo phân loại (classification report) cung cấp thêm các thước đo chi tiết như precision, recall và f1-score. Đối với lớp fake, precision đạt 0.90, recall đạt 0.91, và f1-score đạt 0.91, thể hiện rằng mô hình rất hiệu quả trong việc nhận diện video giả với tỷ lệ dương tính chính xác cao. Đối với lớp real, precision là 0.82, recall là 0.80, và f1-score là 0.81, thấp hơn một chút so với lớp fake. Điều này có thể giải thích bằng sự mất cân bằng dữ liệu giữa hai lớp, do số lượng video fake lớn hơn video real.

Về độ chính xác tổng thể, mô hình đạt accuracy là 88% trên tập test. Trung bình macro và weighted các chỉ số precision, recall và f1-score đều đạt 0.86-0.88, cho thấy hiệu suất tốt trên toàn bộ dữ liệu kiểm thử. Tuy nhiên, sự khác biệt nhỏ giữa hai lớp chỉ số f1-score gợi ý rằng mô hình có thể được cải thiện thêm bằng cách cân bằng lại dữ liệu hoặc tinh chỉnh thêm các siêu tham số.

Nhìn chung, các kết quả này khẳng định rằng kiến trúc ResNeXt-50 kết hợp với LSTM, cùng với pipeline tiền xử lý mạnh mẽ tập trung vào khuôn mặt, đã giúp mô hình phát hiện deepfake hiệu quả.

## 4.4 Triển khai giao diện

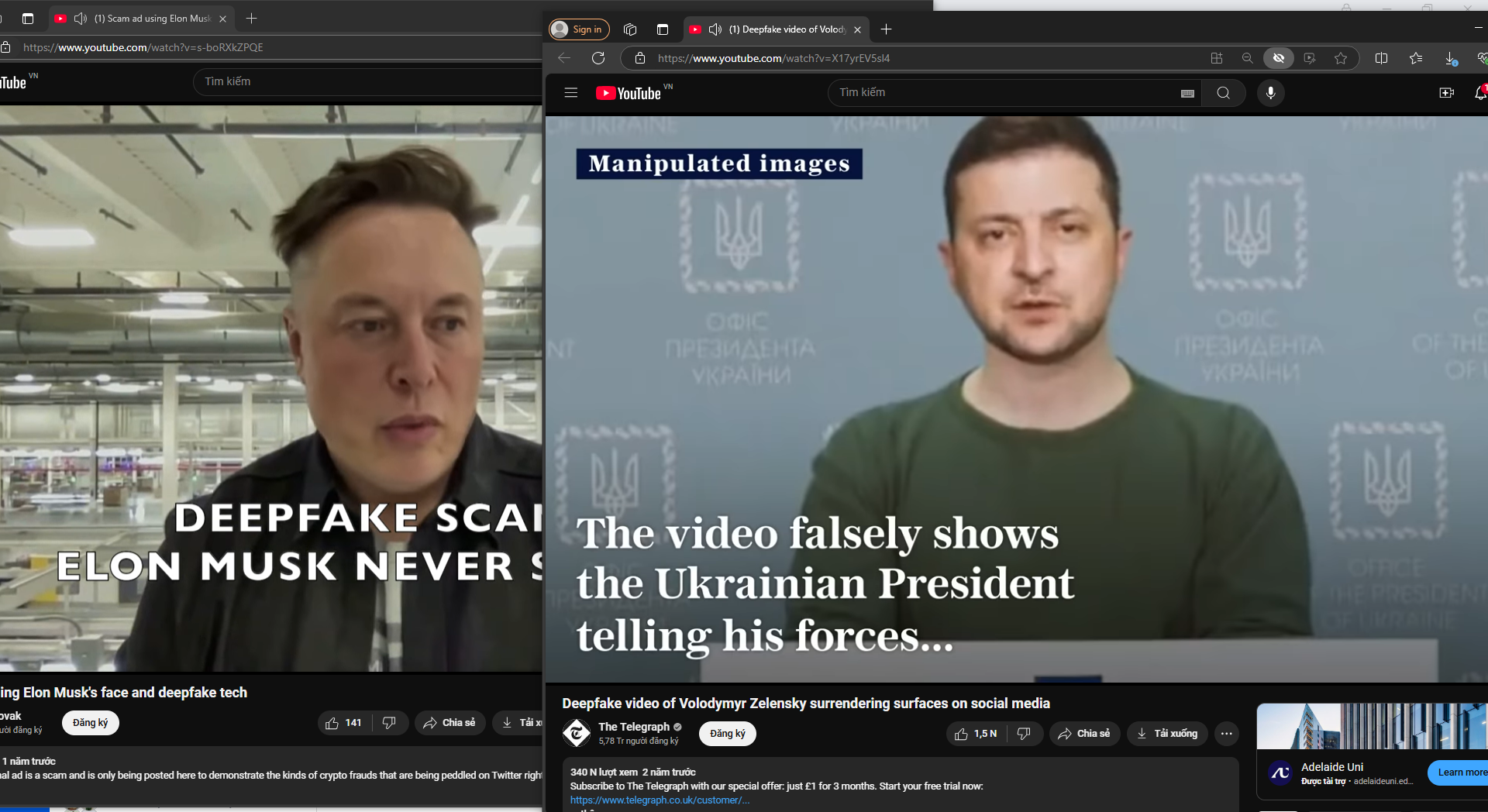
**Giao diện và thực nghiệm hệ thống phát hiện deepfake**

**1. Giao diện người dùng**

Hệ thống phát hiện deepfake được thiết kế với giao diện trực quan, bao gồm các phần chính sau:

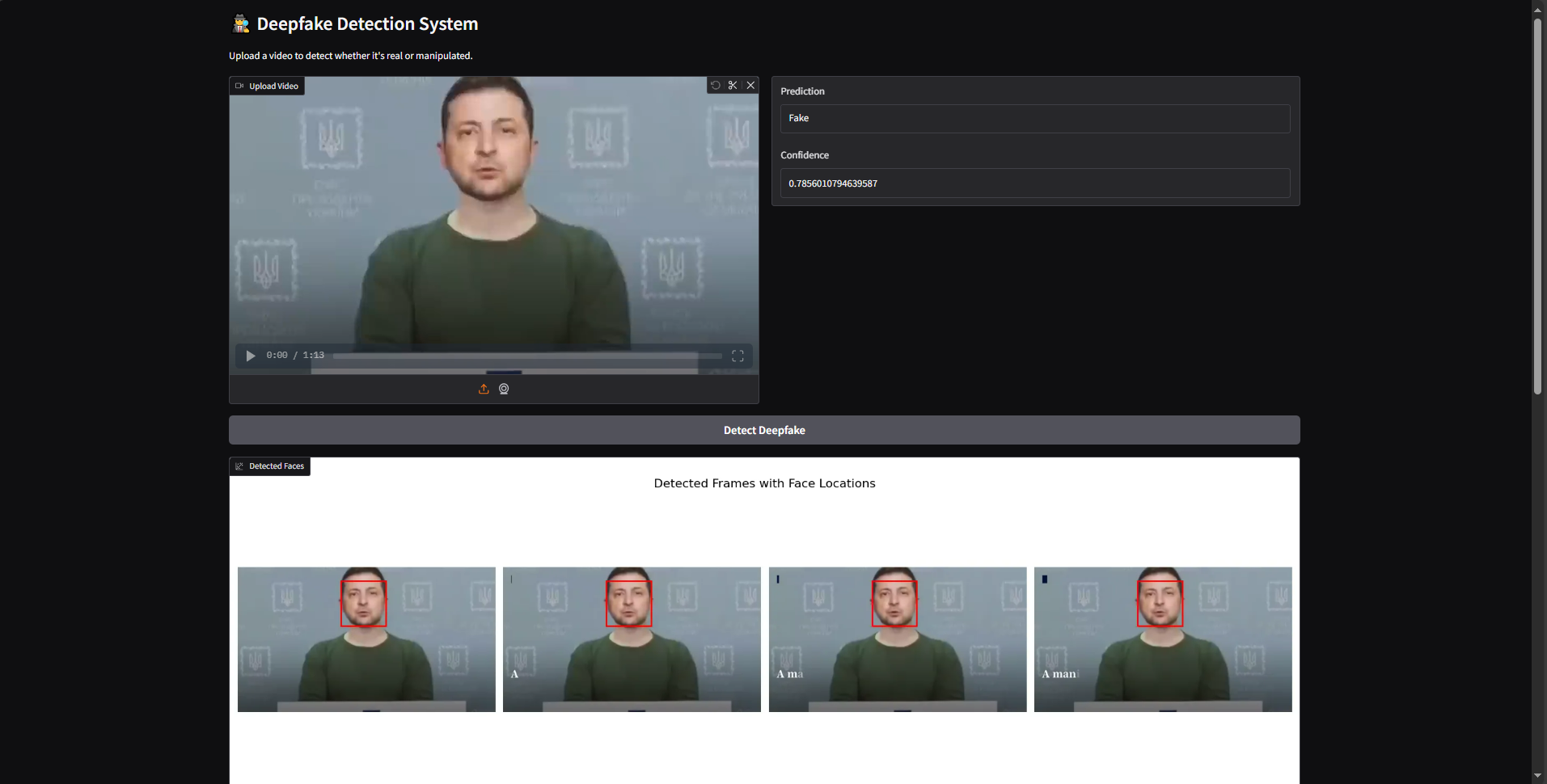
* **Khu vực tải lên video hoặc sử dụng camera**: Người dùng có thể tải một video bất kỳ từ thiết bị cá nhân để kiểm tra xem nó là thật hay giả. Khu vực này hiển thị video đã tải lên và cung cấp các nút điều khiển phát/tạm dừng video.
* **Thông tin dự đoán**: Sau khi xử lý, hệ thống hiển thị kết quả dự đoán, bao gồm:
  + **Kết luận (Prediction)**: Thật (**Real**) hoặc giả (**Fake**).
  + **Độ tin cậy (Confidence)**: Một giá trị xác suất (0 đến 1) thể hiện mức độ tin cậy của mô hình trong việc đưa ra dự đoán.
* **Phát hiện khuôn mặt**: Hệ thống tự động phát hiện các khung hình chứa khuôn mặt từ video, hiển thị từng khung hình với vị trí khuôn mặt được đánh dấu bằng các hộp giới hạn màu đỏ.

**4.4.2 Quy trình thực nghiệm**

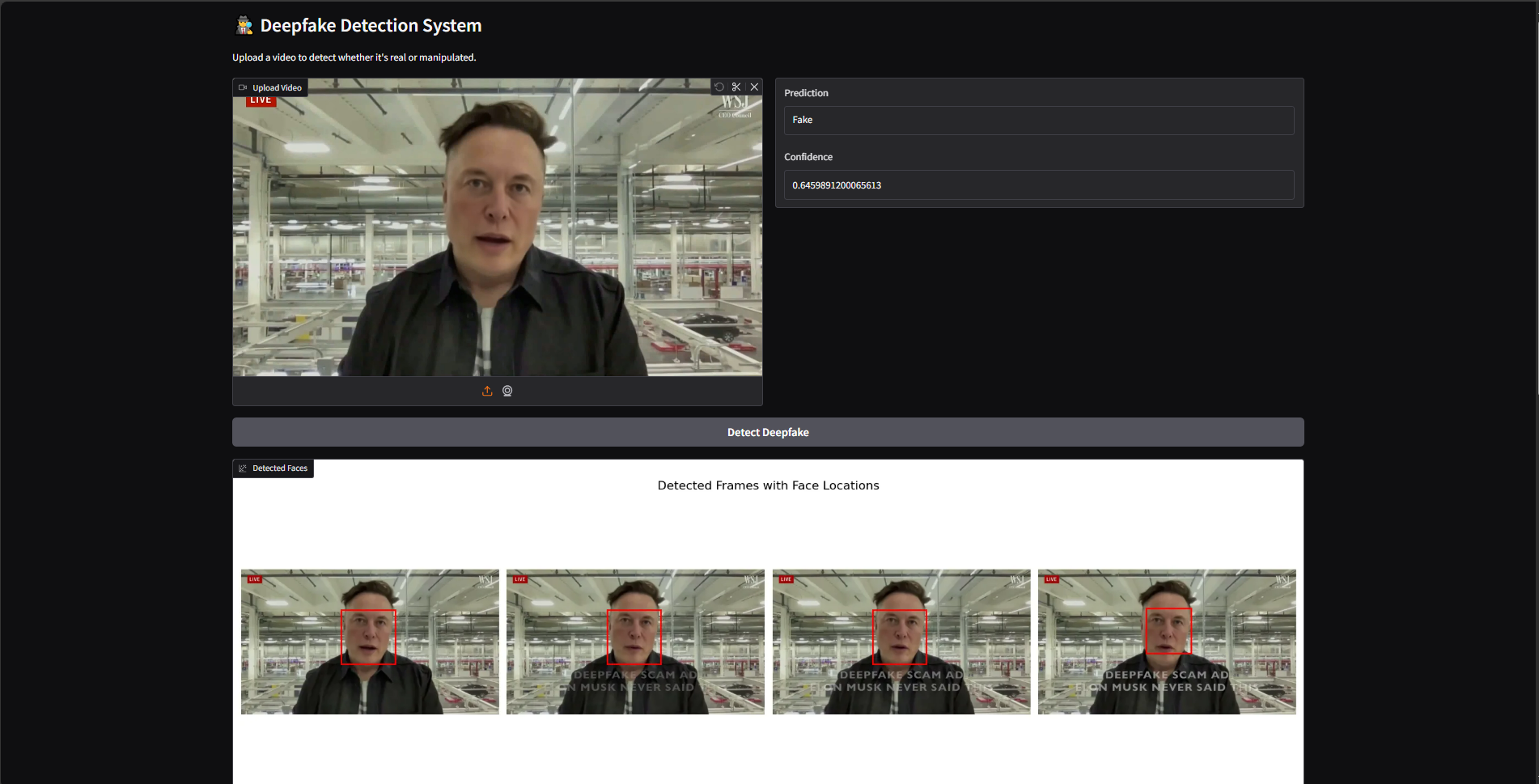


**Bước 1**: Tải video mẫu lên hệ thống. Hai video được sử dụng trong thực nghiệm:

Một video giả mạo của Tổng thống Ukraine Volodymyr Zelensky, được lan truyền trên mạng với nội dung giả tạo rằng ông đầu hàng trong xung đột với Nga.



Một video giả mạo Elon Musk được sử dụng trong chiến dịch lừa đảo tiền mã hóa.



Hệ thống xử lý video bằng cách trích xuất từng khung hình, phát hiện khuôn mặt trong mỗi khung, sau đó sử dụng mô hình học sâu để phân tích các đặc trưng video khuôn mặt và dự đoán video là thật hay giả.

**Bước 3**: Kết quả được hiển thị trong giao diện, bao gồm kết luận và độ tin cậy cho từng video.

**4.4.3. Kết quả thực nghiệm**

* **Video giả mạo Volodymyr Zelensky**:
  + **Dự đoán**: Fake (giả).
  + **Độ tin cậy**: 0.7856 (78.56%).
  + Các khung hình được phát hiện với khuôn mặt đều được đánh dấu chính xác. Kết quả cho thấy hệ thống nhận diện tốt các đặc trưng giả mạo trong video.

**Video giả mạo Elon Musk**:

* + **Dự đoán**: Fake (giả).
  + **Độ tin cậy**: 0.6459 (64.59%).
  + Hệ thống phát hiện khuôn mặt và dự đoán chính xác video này là giả mạo. Tuy nhiên, độ tin cậy thấp hơn so với video của Zelensky, có thể do mức độ tinh vi trong việc tạo deepfake của video này.

# 6. Kết Luận (Conclusion)

Mô hình phát hiện deepfake đã đạt được kết quả ấn tượng với độ chính xác trên tập kiểm thử là 87.55%. Quá trình huấn luyện cho thấy sự tiến bộ ổn định, từ train accuracy 70.61% ở epoch đầu đến 97.14% ở epoch cuối, trong khi val accuracy đạt 86.01%. Phân tích trên tập kiểm thử chỉ ra rằng mô hình phân loại chính xác phần lớn video fake và real, với f1-score cho lớp fake đạt 0.91 và lớp real đạt 0.81. Những kết quả này khẳng định hiệu quả của pipeline tiền xử lý dữ liệu và kiến trúc kết hợp ResNeXt-50 và LSTM.

Những kết quả này cho thấy khả năng ứng dụng hiệu quả của mô hình trong việc phát hiện deepfake, một vấn đề đang ngày càng quan trọng trong bảo mật và kiểm duyệt nội dung. Việc tập trung vào khuôn mặt đã giúp tăng độ chính xác và giảm nhiễu không cần thiết, chứng minh vai trò quan trọng của tiền xử lý dữ liệu. Tuy nhiên, chênh lệch giữa các lớp chỉ ra cần cải thiện để giảm tỷ lệ nhầm lẫn, đặc biệt là với video thật. Nghiên cứu này mở ra hướng ứng dụng thực tế trong việc bảo vệ sự tin cậy và an toàn thông tin trong xã hội.

Trong quá trình thực hiện, khó khăn lớn nhất là sự mất cân bằng dữ liệu giữa video thật và giả, dẫn đến độ chính xác thấp hơn ở lớp video thật. Điều này được giải quyết một phần bằng cách sử dụng phương pháp lấy mẫu phân tầng và tăng cường dữ liệu (data augmentation). Ngoài ra, việc xử lý khung hình và trích xuất khuôn mặt đòi hỏi tài nguyên tính toán lớn, nhưng đã được tối ưu hóa bằng cách chạy song song và sử dụng GPU hiệu năng cao.

Trong tương lai, việc phát triển sử dụng tập data lớn hơn để cân là cần thiết. Hơn nữa, nghiên cứu nên mở rộng với các mô hình tiên tiến như Vision Transformers hoặc áp dụng thêm kỹ thuật multi-task learning để tăng độ chính xác. Việc tích hợp mô hình vào các hệ thống phát hiện real-time và kiểm chứng trên tập dữ liệu lớn, đa dạng hơn cũng là hướng phát triển tiềm năng.

# 7. Tài liệu tham khảo