**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**PHÂN HIỆU**



**ĐỀ TÀI:**

**NGHIÊN CỨU MÔ HÌNH HỌC SÂU ỨNG DỤNG**

**TRONG PHÁT HIỆN DEEPFAKE**

**Sinh viên thực hiện:** Lê Thị Yến Nhi

Mai Văn Tiền

**Lớp:** S25-64CNTT

**Bộ môn:** Công Nghệ Thông Tin

**Giáo viên hướng dẫn:** ThS. Vũ Thị Hạnh

***Hồ Chí Minh, 2025***

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 2](#_Toc714)

[I. MỞ ĐẦU 3](#_Toc8541)

[1. Tổng quan nghiên cứu tình hình thuộc lĩnh vực đề tài 3](#_Toc7555)

[2. Mục tiêu đề tài 3](#_Toc3229)

[3. Phương pháp nghiên cứu, đối tượng và phạm vi nghiên cứu 3](#_Toc4890)

[3.1. Phương pháp nghiên cứu 3](#_Toc18680)

[3.2. Đối tượng nghiên cứu 4](#_Toc15697)

[3.3. Phạm vi nghiên cứu 4](#_Toc2199)

[II. NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU 5](#_Toc12097)

[Chương 1: Giới thiệu tổng quan về mạng ResNet 5](#_Toc23937)

[1.1. Deep Learning 5](#_Toc30966)

[1.2. Mô hình GAN 5](#_Toc19755)

[1.3. Mạng Mesonet 6](#_Toc20804)

[1.4. Mạng ResNet 8](#_Toc9597)

[1.4.1. Mạng ResNet là gì? 8](#_Toc7987)

[1.4.2. Kiến trúc ResNet 10](#_Toc18207)

[Chương 2: Google Colab 15](#_Toc26526)

[2.1. Google Colab là gì? 15](#_Toc30474)

[Chương 3: Thu thập và tiền xử lý dữ liệu 16](#_Toc2303)

[3.1. Thu thập dữ liệu 16](#_Toc6568)

[3.2. Tiền xử lý dữ liệu 16](#_Toc16005)

[Chương 4: Xây dựng và huấn luyện mô hình 19](#_Toc3092)

[4.1. Mô hình Mesonet 19](#_Toc27873)

[4.2. Mô hình ResNet 19](#_Toc13368)

[4.3. Huấn luyện mô hình 21](#_Toc3694)

[4.4. Đánh giá mô hình 22](#_Toc26316)

[4.5. Kết quả hai mô hình 31](#_Toc414)

[Chương 5: Xây dựng Web App phát hiện ảnh Deepfake 33](#_Toc31383)

[5.1. Xây dựng FrontEnd và BackEnd 33](#_Toc21459)

[5.2. Xử lý nhận diện khuôn mặt 34](#_Toc23126)

[5.3. Cấu hình để load mô hình Deepfake 34](#_Toc13332)

[III. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 35](#_Toc13109)

[1. Kết luận chung về đề tài 35](#_Toc21556)

[2. Hướng phát triển của đề tài 35](#_Toc3010)

[TÀI LIỆU THAM KHẢO 37](#_Toc29798)

[PHỤ LỤC 38](#_Toc12223)

[(1) Code lưu mô hình vào thư mục save models: 38](#_Toc29788)

[(2) Code xử lý nhận diện khuôn mặt: 38](#_Toc32574)

[(3) Định nghĩa mô hình Meso4: 40](#_Toc24153)

[(4) Định nghĩa mô hình ResNet50: 41](#_Toc25584)

[(5) Tải lên trọng số bias, định nghĩa transforms: 41](#_Toc17381)

[(6) Định nghĩa route dự đoán: 42](#_Toc1418)

**DANH MỤC HÌNH ẢNH**

[Hình 1 : Kiến trúc Meso4 7](#_Toc1178)

[Hình 2 : Cấu trúc cơ bản của Meso4 8](#_Toc17610)

[Hình 3 : Tỉ lệ Training Error và Test Error 9](#_Toc23814)

[Hình 4 : Kỹ thuật Skip Connection 10](#_Toc27812)

[Hình 5 : Cấu trúc mạng nơ-ron tích chập 11](#_Toc22861)

[Hình 6 : Kiến trúc mạng VGG-19, PlainNet và ResNet34 12](#_Toc32439)

[Hình 7 : Thông số lớp tích chập theo từng mô hình ResNet 12](#_Toc17708)

[Hình 8 : Dữ liệu gốc trên Kaggle 16](#_Toc7492)

[Hình 9 : Lọc dữ liệu không hợp lệ trong Dataset 16](#_Toc4734)

[Hình 10 : Chia đều dữ liệu cho 2 nhãn Real và Fake 17](#_Toc29602)

[Hình 11 : Dữ liệu tập Test 17](#_Toc18179)

[Hình 12 : Kiến trúc tổng quát mô hình Meso4 19](#_Toc18405)

[Hình 13 : Kiến trúc tổng quát mô hình ResNet50 20](#_Toc21635)

[Hình 14 : Minh họa tầng tích chập đầu tiên của ResNet50 21](#_Toc22924)

[Hình 15 : Huấn luyện ở mức learning rate 1e-03 mô hình ResNet50 23](#_Toc17080)

[Hình 16 : Huấn luyện ở mức learning rate 1e-04 mô hình ResNet50 23](#_Toc109)

[Hình 17 : Huấn luyện ở mức learning rate 1e-05 mô hình ResNet50 23](#_Toc1250)

[Hình 18 : Biểu đồ Loss và Accuracy của ResNet50 24](#_Toc26407)

[Hình 19 : Dự đoán kết quả trên tập test với ResNet50 24](#_Toc17333)

[Hình 20 : Huấn luyện ở mức learning rate 1e-03 mô hình Meso4 25](#_Toc28209)

[Hình 21 : Huấn luyện ở mức learning rate 1e-04 mô hình Meso4 25](#_Toc26241)

[Hình 22 : Huấn luyện ở mức learning rate 1e-05 mô hình Meso4 26](#_Toc21358)

[Hình 23 : Biểu đồ Loss và Accuracy của Meso4 26](#_Toc5985)

[Hình 24 : Dự đoán kết quả trên tập test với Meso4 27](#_Toc30721)

[Hình 25 : Classification Report của ResNet50 28](#_Toc22454)

[Hình 26 : Classification Report của Meso4 29](#_Toc14434)

[Hình 27 : ROC Curve ResNet50 30](#_Toc14547)

[Hình 28 : ROC Curve Meso4 30](#_Toc2737)

[Hình 29 : Biểu đồ ma trận nhầm lẫn so sánh 2 mô hình ResNet50 và Meso4 32](#_Toc3322)

[Hình 30 : Giao diện trang web 33](#_Toc22347)

1. **MỞ ĐẦU**
2. **Tổng quan nghiên cứu tình hình thuộc lĩnh vực đề tài**

Trước tình hình cách mạng công nghiệp 4.0 hiện nay, việc chạy đua về lĩnh vực trí tuệ nhân tạo đã không còn quá xa lạ khi các ông lớn trong ngành công nghệ liên tục cho ra các sản phẩm về AI điển hình gần đây nhất với ChatGPT của OpenAI hay DeepSeek. Điều này làm khuấy lên những làn sóng dư luận khi mà AI dần phát triển thay thế cũng như hỗ trợ con người trong nhiều lĩnh vực và ngành nghề. Tiêu biểu với việc ứng dụng công nghệ AI hỗ trợ trong việc xử lí ảnh như làm rõ nét hình ảnh, ứng dụng AI tạo ra những dáng đứng mới từ bức ảnh tạo dáng có sẵn hay hơn thế khi sử dụng mô hình GAN để tạo ra những ảnh trông có vẻ giống như thật nhưng lại không phải ảnh thật. Trước những lợi ích mà nó đem lại đồng nghĩa với việc cũng tồn tại những mặt trái, khi mà nhiều kẻ xấu lợi dụng việc tạo ra những hình ảnh giả mạo người thật để lừa đảo hay tận dụng chúng với mục đích xấu.

Từ thực trạng trên, việc tạo ra những ứng dụng cũng như công nghệ nhận diện và phân biệt ảnh thật hay ảnh giả do AI tạo ra càng cần thiết hơn. Mô hình sẽ giúp nhận diện đâu là ảnh thật, đâu là ảnh giả với tỉ lệ bao nhiêu, hỗ trợ người sử dụng phân biệt ảnh thật giả.

1. **Mục tiêu đề tài**

Trước sự phát triển nhanh chóng của công nghệ Deepfake, việc nhận diện và phân biệt nội dung thật – giả trở nên cấp thiết, đặc biệt trong bối cảnh thông tin sai lệch có thể gây ảnh hưởng tiêu cực đến cá nhân, tổ chức và xã hội. Đề tài này nhằm nghiên cứu các kỹ thuật phát hiện Deepfake, xây dựng và đánh giá mô hình nhận diện nội dung bị làm giả dựa trên các phương pháp học máy (Machine Learning) và học sâu (Deep Learning). Qua đó, nghiên cứu hướng đến việc nâng cao nhận thức của người dùng, đồng thời cung cấp một công cụ hỗ trợ phát hiện Deepfake hiệu quả, góp phần hạn chế tác động tiêu cực của công nghệ này trong thực tế.

1. **Phương pháp nghiên cứu, đối tượng và phạm vi nghiên cứu**
   1. **Phương pháp nghiên cứu**

* Nghiên cứu tài liệu: Nghiên cứu các công nghệ Deepfake, phương pháp phát hiện hình ảnh giả mạo, các mô hình học sâu.
* Phân tích dữ liệu: Thu thập dữ liệu trên các trang website cung cấp dữ liệu miễn phí về deepfake và tiền xử lý dữ liệu hình ảnh để đưa vào mô hình.
* Xây dựng các mô hình học sâu, triển khai và so sánh.
* Huấn luyện và đánh giá hiệu suất của mô hình bằng thông số
* Thực nghiệm và tối ưu hóa: Điều chỉnh siêu tham số
  1. **Đối tượng nghiên cứu**
* Công nghệ deepfake và các kỹ thuật tạo ảnh giả
* Thuật toán, mô hình học sâu trong phát hiện Deepfake
* Phương pháp xử lý ảnh và trích xuất đặc trưng nhận diện deepfake
  1. **Phạm vi nghiên cứu**
* Phạm vi công nghệ: Tập trung phát hiện hình ảnh giả mạo, không áp dụng cho video
* Phạm vi mô hình: Xây dựng, đánh giá các mô hình học sâu phát hiện deepfake
* Phạm vi ứng dụng: Đề xuất mô hình có thể áp dụng vào các hệ thống kiểm duyệt nội dung, bảo vệ danh tính cá nhân

1. **NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU**

**Chương 1: Giới thiệu tổng quan về mạng ResNet**

* 1. **Deep Learning**

Deep Learning có thể nói là 1 nhánh con nữa của học máy, tuy nhiên chúng có thể xử lí những dữ liệu phức tạp hơn và ít dựa vào con người hơn khi mà học sâu dựa trên mạng nơ-ron nhân tạo. Chúng có thể tự học dựa trên những lỗi trước đó trong khi học máy không thể làm được điều này khi không có sự can thiệp của con người. Dù vậy, điểm yếu của học sâu là tiêu hao quá nhiều dữ liệu do nó cần sức mạnh tính toán nhiều hơn học máy, do vậy việc sử dụng đôi khi rất tốn kém với những mô hình lớn.

* 1. **Mô hình GAN**

Trước sự phát triển bùng nổ của AI, với trình độ vượt bậc của con người thì việc biến những việc tưởng chừng như không thể lại trở thành có thể, cũng như việc tạo ra hình ảnh con người, cảnh vật hay động vật chân thật đến mức mắt người nhìn vào cũng không thể nghĩ được nó là ảnh chỉ được tạo từ trí tuệ nhân tạo.

Mô hình mạng đối kháng tạo sinh - Generative Adversarial Network (GAN) [1] từ những ý tưởng đó mà được tạo ra. Mô hình GAN được giới thiệu lần đầu tiên vào năm 2014 bởi J. Goodfellow với mục đích tạo ra dữ liệu mới giống với dữ liệu thực. Là một kiến trúc mạng nơ-ron đặc biệt gồm 2 mạng nơ-ron chính: mạng tạo sinh (generator) và mạng phân biệt (discriminator).

* **Mạng tạo sinh - Generator:** mang nhiệm vụ tạo ra các loại dự liệu giả giống như thật.
* **Mạng phân biệt - Discriminator:** ngược lại với mạng tạo sinh, mạng phân biệt giữ nhiệm vụ như một người giám sát với mục đích là phát hiện dữ liệu do mạng tạo sinh tạo ra là giả hay thật.

Mô hình GAN hoạt động dựa trên nguyên lí mạng tạo sinh sẽ liên tục tạo ra dữ liệu giả và mạng phân biệt sẽ phân biệt chúng. Mạng tạo sinh sẽ tiếp tục tạo dữ liệu giả mà đến khi nào mạng phân biệt không còn phân biệt được đâu là dữ liệu thật và đâu là dữ liệu giả thì mô hình GAN lúc này thành công. Ngày nay, mô hình GAN được ứng dụng rất nhiều trong việc tạo hình ảnh hay nâng cao chất lượng hình ảnh, tuy nhiên, một số thành phần lại lạm dụng tính chất và kẽ hở của mô hình để tạo ra dữ liệu giả dùng với mục đích xấu. Do vậy, việc không ngừng phát triển và ứng dụng công nghệ để phòng ngừa, phát hiện dữ liệu giả vô cùng cần thiết.

**Điểm khác biệt của việc trích xuất đặc trưng ảnh thật và ảnh giả trong mô hình GAN:**

Ngay từ ban đầu, GAN đã hướng tới mục tiêu tạo ra dữ liệu giả giống thật nhưng không phải là ảnh thật, do vậy việc trích xuất đặc trưng ảnh sẽ khác rất nhiều. GAN dùng mạng phân biệt và mạng tạo sinh để trích xuất dữ liệu theo hướng đánh giá tổng thể từ ảnh, với một bên tạo dữ liệu giả, một bên kiểm tra dữ liệu giả đó có phải thật hay không:

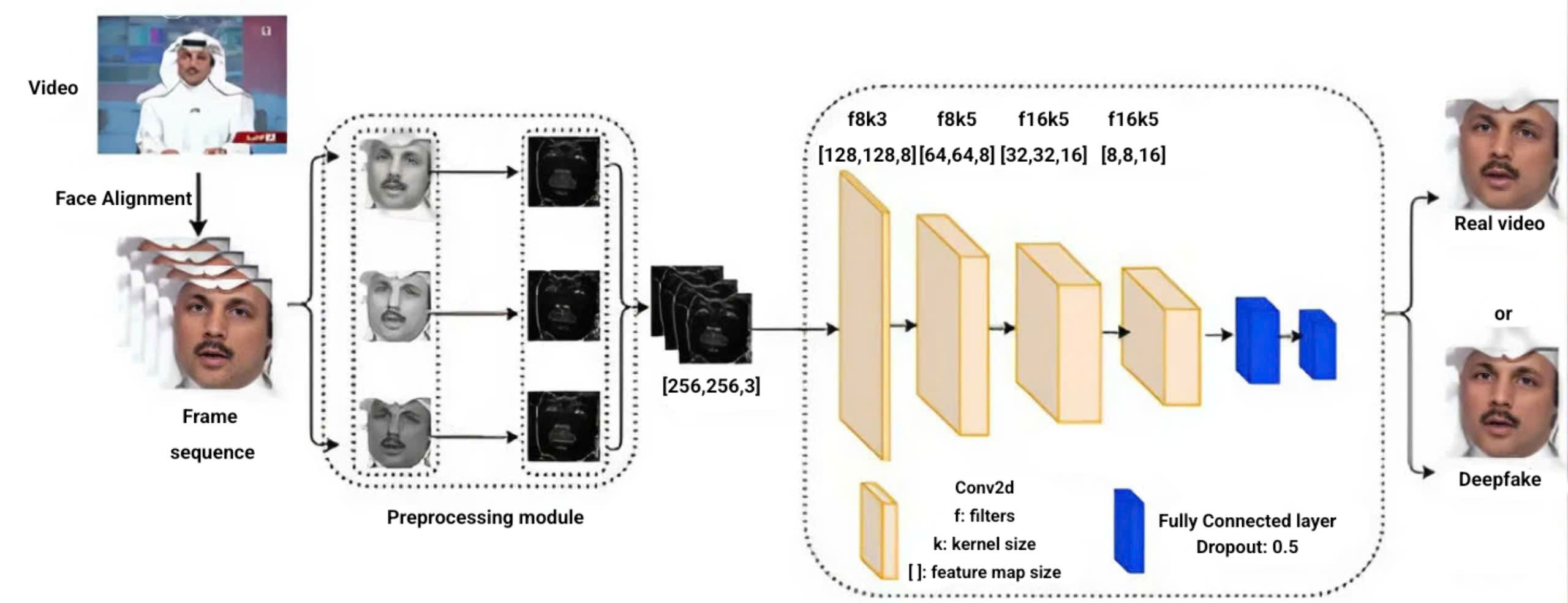
+ Khi trích xuất đặc trưng từ ảnh thật, GAN sử dụng mạng phân biệt (Discriminator) dùng các tầng tích chập để tìm các đặc trưng quan trọng của ảnh như: đổ bóng, phân phối ánh sáng, kết cấu tự nhiên của ảnh, cấu trúc vật thể, khuôn mặt,.. để từ những đặc trưng này đưa ra một chuẩn mực cho ảnh thật.

+ Khi trích xuất đặc trưng từ ảnh giả, GAN sử dụng mạng tạo sinh (Generator), tuy nhiên ban đầu GAN sẽ không thể nào cho ra kết quả tốt nhất, nó sẽ không đủ chi tiết để tạo ra hình ảnh có độ chi tiết giống ảnh thật, mạng phân biệt sẽ dùng tầng tích chập để phát hiện nhiễu (ví dụ: độ méo mó của khuôn mặt, lỗi đổ bóng,..) trong ảnh của mạng tạo sinh, tìm sự khác biệt giữa chuẩn mực ảnh thật so với ảnh giả, lúc này mạng phân biệt sẽ phản hồi lại cho mạng sinh. Từ đó, mạng tạo sinh sẽ rút kinh nghiệm rồi khắc phục nhược điểm của lần tạo dữ liệu trước để cho ra kết quả tốt hơn ở những lần sau.

* 1. **Mạng Mesonet**

Mesonet được đề xuất bởi D. Afchar và cộng sự của mình trong bài báo "Mesonet: a Compact Facial Video Forgery Detection Network" vào năm 2018. Với mục đích chính của mô hình là sử dụng chuyên biệt cho Deepfake - nhận diện hình ảnh giả mạo. Mô hình vẫn dựa theo những nguyên lí và kiến trúc cơ bản của mạng nơ-ron tích chập, nhưng thay vì tập trung vào việc cố gắng học các đặc trưng chi tiết của hình ảnh như mô hình khác, đồng nghĩa với việc những mô hình đó có kiến trúc phức tạp và đa tầng hơn. Mesonet không như vậy, mô hình được cấu tạo khá đơn giản và nông, phù hợp cho việc nhận diện các đặc trưng trung bình chẳng hạn như phát hiện độ sáng khác nhau của bức ảnh hay cử chỉ khác thường trên mặt, tối ưu và có ít tham số hơn các mô hình khác. Do vậy, mô hình huấn luyện có tốc độ nhanh hơn những mô hình khác, đối với mặt lợi là vậy tuy nhiên để so về thế mạnh nhận diện với các mô hình học sâu khác trên thị trường hiện nay thì Mesonet không bằng các mô hình khác như VGG, PlainNet,.. Đổi lại mô hình khá nhẹ và phù hợp cho những dự án nhỏ không cần nhiều tài nguyên.

Mô hình Meso4 là một trong những mô hình của mạng Mesonet với cấu trúc khá đơn giản. Mô hình lấy đầu vào là hình ảnh, video màu biểu diễn dạng RGB với 4 tầng tích chập mà trong mỗi tầng tích chập sử dụng Batch Normalization và hàm kích hoạt giúp cho mô hình hội tụ nhanh hơn cũng như giữ ổn định, giảm tình trạng overfitting trong quá trình huấn luyện. Sau cùng là hai lớp kết nối cuối cùng (fully-connected layer) trước khi qua lớp đầu ra với dự đoán hình ảnh, video là thật hay deepfake.



Hình 1: Kiến trúc Meso4

Theo kiến trúc mô hình Meso4 cơ bản sẽ có cấu hình như sau:

A screenshot of a computer

Description automatically generated

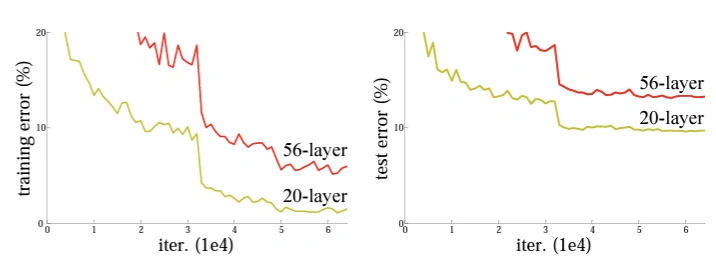
Hình 2: Cấu trúc cơ bản của Meso4

Với đầu vào là frame ảnh RGB, trong tầng tích chập thứ nhất sử dụng kernel 3x3 còn 3 tầng còn lại dùng kernel 5x5, nhằm mục đích để mô hình học được các đặc trưng chi tiết của dữ liệu. Số lượng filters đều là 8 ở 2 tầng đầu tiên và 16 ở 2 tầng cuối nhằm để cho mô hình ban đầu sẽ học những đặc trưng đơn giản, cơ bản càng về sau tăng số lượng filter để cho mô hình học sâu hơn vào tập dữ liệu, giúp học thêm nhiều thông tin hơn. Padding và stride giúp kiểm soát kích thước đầu ra cũng như hiệu suất của mô hình, trong tầng thứ nhất thông số của cả hai đều là 1 giúp giữ nguyên kích thước của ảnh, từ tầng thứ hai trở đi, tăng lên 2 khiến kích thước giảm dần nhưng sẽ luôn đảm bảo kernel quét hết toàn bộ ảnh chứ không bỏ sót chi tiết nào. Kèm theo đó, việc sử dụng hàm kích hoạt Leaky ReLU giúp mô hình học tốt hơn trong các tình huống mà dữ liệu có sự biến động mạnh tránh mất gradient trong quá trình huấn luyện, cuối cùng, sử dụng hàm kích hoạt Sigmoid cho bài toán nhận diện Deepfake.

* 1. **Mạng ResNet**
     1. **Mạng ResNet là gì?**

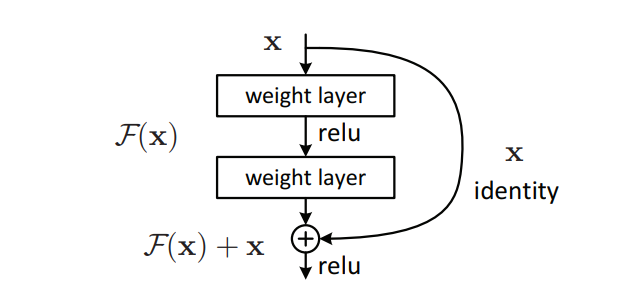
Mạng ResNet hay còn được gọi là mạng nơ-ron dư thừa, là một kiến trúc học sâu mà trong đó các lớp học các hàm dư (residual functions) liên quan đến đầu vào của lớp. Được cho ra mắt và phát triển bởi một nhóm nghiên cứu thuộc Microsoft gồm Kaiming He, Xiangyu Zhang, Shaoqing Ren và Jian Sun với mục đích nhằm để nhận diện hình ảnh và dành chiến thắng trong cuộc thi ILSVRC năm 2015.

Trước khi ResNet được nghiên cứu và cho ra đời, cũng đã có những mô hình khác xuất hiện như: Plain Network, LeNet hay VGG. Tuy nhiên, những mô hình này xuất hiện vấn đề về Vanishing Gradient (Tiêu Biến Độ Dốc) hay Exploding Gradient (Bùng Nổ Độ Dốc), là hiện tượng khi mà mô hình được huấn luyện và học tập từ dữ liệu, khi tính toán hàm mất mát (loss function) trong quá trình lan truyền ngược (backpropagation) độ dốc (gradient) của hàm quá nhỏ hoặc quá lớn khiến cho các trọng số không được cập nhật hay trọng số quá lớn làm cho mô hình huấn luyện trở nên chậm đi hoặc không hội tụ được khiến hiệu suất của mô hình giảm đi rất nhiều, với đầu vào là dữ liệu mới có thể dẫn tới việc dự đoán không chính xác gây ra sai lệch lớn. Có thể thấy được thông qua ảnh dưới, khi huấn luyện dữ liệu ta tăng số lớp lên từ 20 lên 56 lớp thì tỉ lệ lỗi khi huấn luyện lại tăng cao với mô hình dùng 56 lớp [2], lẽ ra với số lớp càng nhiều thì trọng số phải càng được tối ưu và cho ra kết quả tốt hơn mô hình có số lớp nhỏ, ấy vậy mà chúng lại đi ngược chứng tỏ rằng mô hình giảm hiệu quả rất nhiều do 2 vấn đề ta đã đề cập trước đó.



Hình 3: Tỉ lệ Training Error và Test Error

Mạng ResNet ra đời nhóm nghiên cứu đã đưa ra khái niệm “Khối dư” (residual block) và sử dụng kỹ thuật Skip Connection (bỏ qua kết nối) mục đích là để bỏ qua kết nối các hoạt động của một lớp với các lớp tiếp theo bằng việc bỏ qua một số lớp ở giữa, điều này tạo thành Khối Dư dẫn đến các mạng được tạo ra bằng cách xếp chồng các khối dư này lại tạo thành kiến trúc mạng nơ-ron dư thừa. Việc đưa ra khái niệm mới và kỹ thuật Skip Connection nhằm để giải quyết vấn đề về hiện tượng Tiêu Biến/Bùng Nổ độ dốc - vấn đề mà những mô hình trước đó chưa làm được [3].



Hình 4: Kỹ thuật Skip Connection

Với:

* x là tập hợp các lớp tích chập
* F(x) là hàm phần dư (residual function)
* H(x) là hàm ánh xạ từ đầu vào đến đầu ra

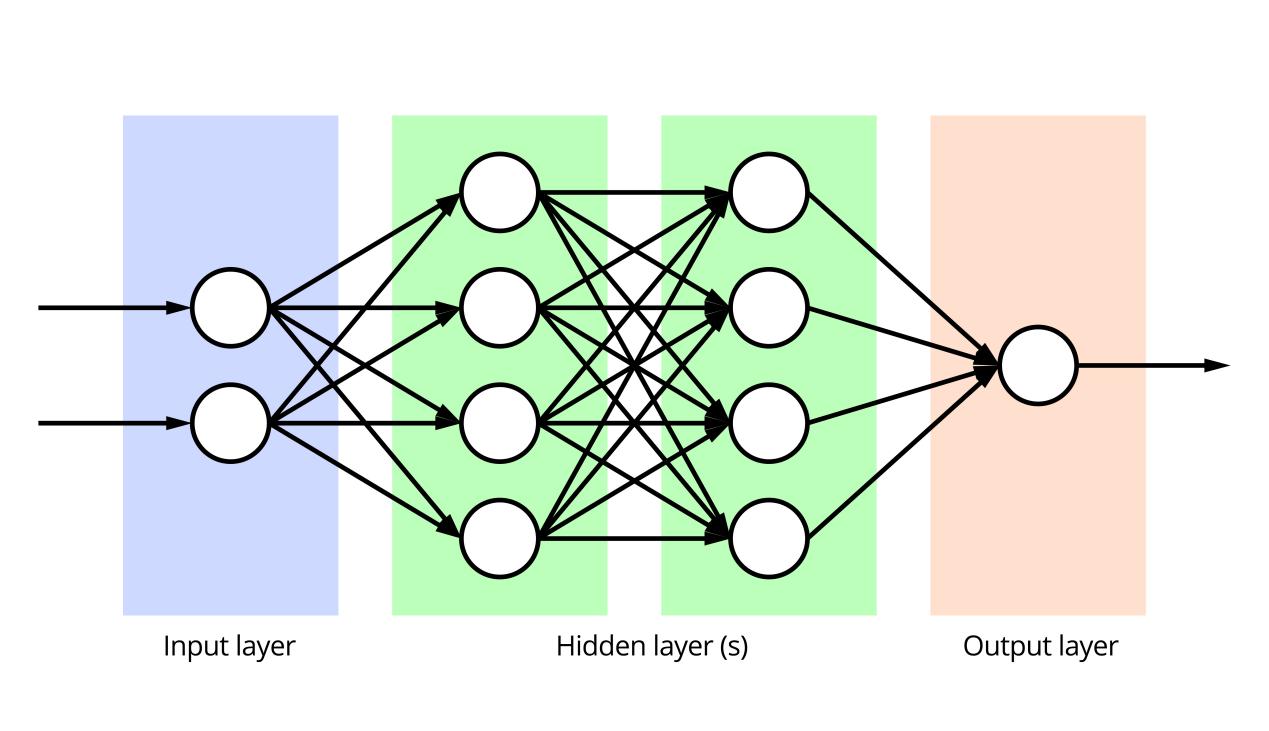
Thay vì học một ánh xạ trực tiếp từ H(x) như những mạng khác, mạng ResNet sẽ học một ánh xạ phần dư F(x) = H(x) - x và đầu ra cuối cùng là F(x) + x. Điều này giúp cải thiện và hạn chế được những vấn đề về Tiêu Biến Độ Dốc và Bùng Nổ Độ Dốc.

* + 1. **Kiến trúc ResNet**
       1. **Tính năng**

Tiến hành tìm hiểu sâu hơn, kiến trúc của mạng ResNet có các tính năng chính sau [4]:

* Kết nối phần dư (Residual Connections): kết hợp các kết nối dư, cho phép huấn luyện mạng nơ-ron rất sâu và khắc phục vấn đề tiêu biến độ dốc.
* Ánh xạ đồng nhất (Identity Mapping): với hàm đầu ra cuối là F(x) + x giúp quá trình huấn luyện dễ dàng hơn thay vì ánh xạ thực tế trực tiếp thông qua H(x).
* Độ sâu (Depth): cho phép tạo ra các mạng nơ-ron rất sâu, từ đó cải thiện hiệu suất trong các tác vụ nhận diện hình ảnh.
* Giải quyết vấn đề về tham số: thông qua kiến trúc ResNet giúp mô hình làm việc hiệu quả hơn bằng việc giảm thiểu tham số.
  + - 1. **Kiến trúc**

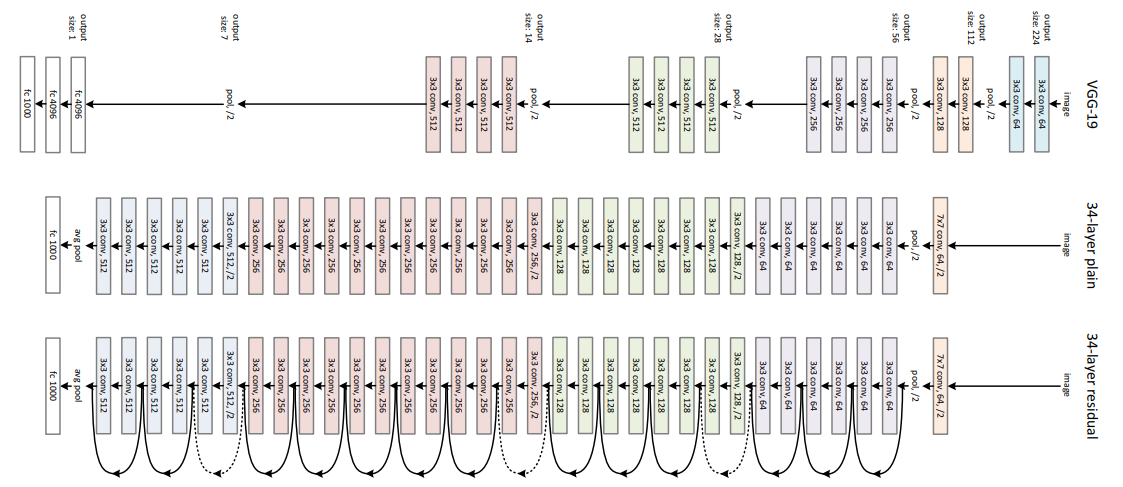
Mạng ResNet là một trong nhiều mô hình khác của mạng nơ-ron tích chập (CNN), có thể thấy về cấu trúc cơ bản mạng ResNet vẫn được cấu tạo bởi 3 phần chính: đầu vào (input), các lớp ẩn (hidden layers) và sau cùng là đầu ra (output) [5].



Hình 5: Cấu trúc mạng nơ-ron tích chập

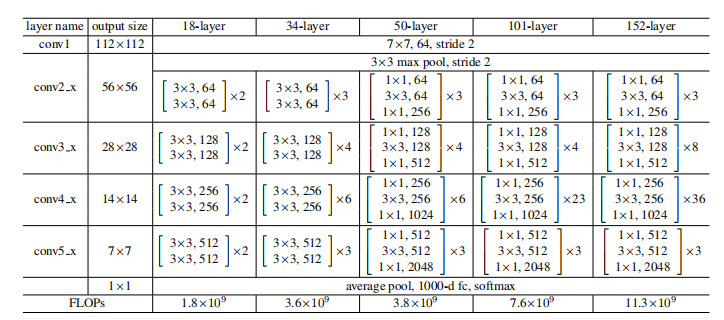
Với việc sử dụng mạng ResNet để nhận diện hình ảnh thì lớp đầu vào lúc này sẽ là ảnh, có thể là ảnh xám với số lượng màu đơn điệu hay ảnh màu RGB với số màu đa dạng hơn, được biểu diễn phức tạp hơn dưới dạng ma trận kích thước theo tùy kích thước của bức ảnh và chồng nhiều ma trận màu lên nhau. Sau đó, trải qua các lớp tích chập khác nhau để đến lớp tích chập đầy đủ cuối cùng trước khi đến lớp đầu ra [1].

Bằng việc sử dụng kiến trúc mạng Plain được lấy cảm hứng từ mô hình VGG-19, sau đó cải tiến để thêm Skip Connections vào mô hình từ đó kiến trúc mạng ResNet ra đời. Dưới đây là mô hình kiến trúc của VGG-19, Plain và ResNet:



Hình 6: Kiến trúc mạng VGG-19, PlainNet và ResNet34

Mạng ResNet có các phiên bản: ResNet18, ResNet34, ResNet50, ResNet101, ResNet152,.. với các số tương ứng với số lớp tích chập được sử dụng của mô hình, càng về sau số lớp càng cao thì mô hình càng mạnh hơn cho hiệu suất nhận diện cao hơn nhưng cũng đồng nghĩa với việc tốn tài nguyên cũng như lượng lớn dữ liệu cần để huấn luyện mô hình. Dưới đây là bảng tổng hợp so sánh các lớp tích chập và thông số theo từng mô hình ResNet [2].



Hình 7: Thông số lớp tích chập theo từng mô hình ResNet

Có thể thấy với 5 mô hình thì mặc định luôn có 5 tầng tích chập: conv1, conv2\_x, conv3\_x, conv4\_x, conv5\_x. Dữ liệu sẽ được trải qua lớp trích xuất đặc trưng (max-pooling) cùng các tầng tích chập mà ở đó các lớp tích chập được tổng hợp và gói gọn trong từng khối cơ bản (basic block) hay khối cổ chai (bottleneck block), tương ứng theo mỗi mô hình sẽ có số lượng khối cổ chai với các lớp tích chập khác nhau, sau cùng trải qua lớp kết nối cùng (fully-connected layer) với hàm kích hoạt (activation function) trước khi qua lớp đầu ra. Trong đó:

* **Khối cơ bản - Basic Block:** khối cơ bản, mỗi khối đều sử dụng bộ lọc (kernel) 3x3 để trích xuất đặc trưng nên đầu vào và đầu ra của khối đều có số lượng kênh (channels) như nhau. Sử dụng cho mô hình ResNet18/34 vì 2 mô hình này có độ sâu khá tương đối, việc sử dụng khối cơ bản cho mô hình này đã đủ để có thể trích xuất các đặc trưng cần thiết cho mô hình học và huấn luyện.
* **Khối cổ chai - Bottleneck Block:** khác với khối cơ bản, khối cổ chai giúp thu hẹp số lượng kênh sau đó mở rộng trở lại nhằm để giảm độ phức tạp tính toán và số lượng tham số trong khi vẫn duy trì khả năng học các đặc trưng phức tạp. Từ ResNet50 trở đi thì độ sâu của mô hình bắt đầu sâu hơn nên việc giảm và tăng chiều, trích xuất đặc trưng giúp giảm độ phức tạp tính toán, tăng hiệu suất mô hình hơn. Trong mỗi khối cổ chai, chia các lớp tích chập thành conv1: sử dụng bộ lọc 1x1, conv2: sử dụng bộ lọc 3x3, conv3: sử dụng bộ lọc 1x1 tương ứng với tính năng của chúng khi conv1 dùng để giảm số lượng kênh (channels) đầu vào, conv2 dùng để trích xuất đặc trưng từ dữ liệu sau khi được giảm chiều và cuối cùng là conv3 giúp tăng chiều khi dùng bộ lọc 1x1 để tăng số lượng kênh trở lại.
* **Bộ lọc - Kernel:** là một ma trận nhỏ và ma trận này luôn là ma trận lẻ, ví dụ 1x1 hay 3x3,… được sử dụng trong các lớp tích chập, ma trận này sẽ trượt qua ma trận dữ liệu - ở bài toán của chúng ta là dữ liệu ảnh màu thực hiện phép tính tích chập tạo ra bản đồ đặc trưng từ dữ liệu đầu vào, từ đó giúp giảm số lượng tham số và phát hiện được các đặc trưng quan trọng của ảnh.
* **Lớp trích xuất đặc trưng - Pooling Layer:** mục đích là để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng.
* **Lớp kết nối cuối cùng - Fully-Connected Layer:** là lớp cuối cùng của mô hình trước khi đưa ra đầu ra, sau khi ảnh được truyền qua nhiều lớp tích chập cũng như lớp trích xuất đặc trưng thì mô hình đã học được những đặc điểm khác nhau của hình ảnh, lúc này ảnh sẽ được làm phẳng tức là chuyển ma trận từ chiềucao\*chiều rộng\*chiều sâu (H\*W\*D) thành 1 vector, hay hiểu đơn giản hơn ví dụ từ ma trận 3x3, có 3 hàng thì những hàng ấy chuyển về dạng cột và nối đuôi nhau tạo thành một vector dài vector này đi qua lớp kết nối để kết hợp các đặc điểm của ảnh từ đó được đầu ra của mô hình.
* **Hàm kích hoạt - Activation Function:** được sử dụng sau mỗi lớp tích chập và lớp kết nối cuối cùng giúp mô hình học được các mối quan hệ phi tuyến phức tạp, cải thiện khả năng biểu diễn của mô hình, quyết định đầu ra dựa trên đầu vào. Có các loại hàm kích hoạt khác nhau, tùy theo mục đích sử dụng khác nhau chẳng hạn như hàm softmax dành cho các bài toán có đầu ra với dự đoán là tỉ lệ phần trăm, hàm tanh trả về giá trị từ [-1,1] cho các bài toán dự đoán cảm xúc,..

**Chương 2: Google Colab**

**2.1. Google Colab là gì?**

Google Colab là một sản phẩm của Google Research, ra mắt vào năm 2022 với mục đích tạo ra một môi trường làm việc, cài đặt và lưu trữ trên đám mây mà không cần phải lưu hay sử dụng trên phần cứng của máy tính. Khi làm việc với các dự án về khoa học dữ liệu, nhất là các dự án về học máy hay học sâu, việc đòi hỏi một nguồn tài nguyên lớn của máy tính là điều cần thiết nhưng những vấn đề về chi phí khiến cho người học, người làm dự án đau đầu. Google Colab ra đời và khắc phục điều đó khi nền tảng này cung cấp cho người dùng tài nguyên cả về CPU lẫn GPU.

Hiện nay, Google Colab có 2 lựa chọn cho người dùng, sử dụng miễn phí với gói tài nguyên GPU giới hạn 15GB một ngày hoặc sử dụng bản nâng cấp được cung cấp tài nguyên tính toán và dung lượng GPU lớn hơn phục vụ cho mô hình cần hoạt động với tài nguyên lớn hỗ trợ hiệu suất cao.

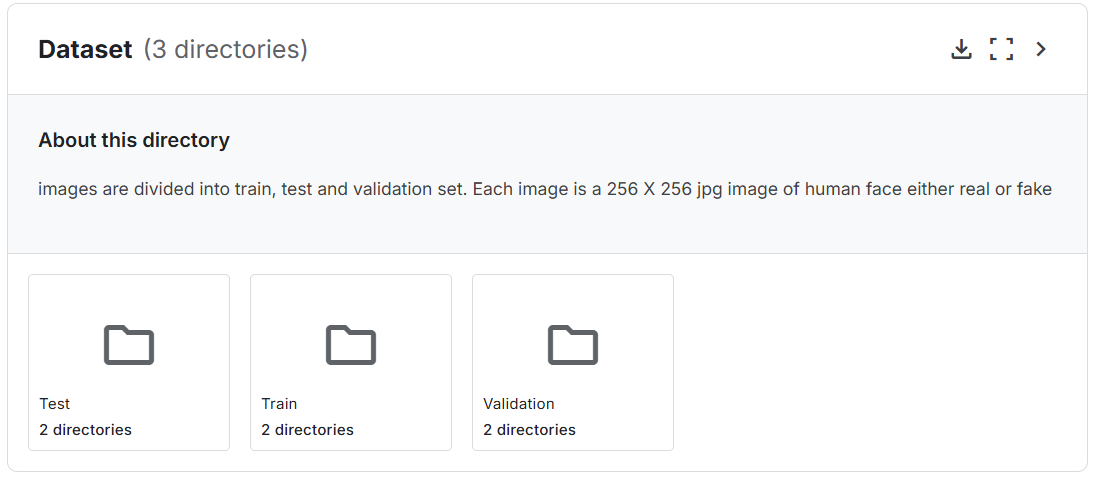
Google Colab cũng hỗ trợ nhiều thư viện phổ biến bao gồm TensorFlow, PyTorch, OpenCV, Scikit-learn, Pandas, NumPy và nhiều thư viện khác mà không cần cài đặt thủ công. Điều này giúp người dùng có thể nhanh chóng triển khai mô hình, thử nghiệm thuật toán mà không gặp trở ngại về cấu hình. Ngoài ra, Google Colab hỗ trợ kết nối với Google Drive, giúp người dùng lưu trữ và truy xuất dữ liệu một cách thuận tiện. Các tệp dữ liệu có thể được tải trực tiếp từ Drive vào môi trường Colab, hỗ trợ tốt cho các bài toán xử lý dữ liệu lớn hoặc huấn luyện mô hình. Bên cạnh đó, với phiên bản trả phí Colab Pro và Colab Pro+, người dùng có thể truy cập vào tài nguyên mạnh mẽ hơn như GPU Tesla T4, V100 hoặc A100 giúp tăng tốc quá trình huấn luyện mô hình đáng kể. Phiên bản nâng cấp này cũng cung cấp thời gian sử dụng dài hơn giảm thiểu tình trạng ngắt kết nối do giới hạn thời gian so với bản miễn phí.

Nhìn chung, Google Colab là một công cụ hữu ích và mạnh mẽ, đặc biệt phù hợp với sinh viên, nhà nghiên cứu và lập trình viên trong lĩnh vực trí tuệ nhân tạo và khoa học dữ liệu, giúp họ tiếp cận với tài nguyên tính toán mạnh mẽ mà không cần đầu tư vào phần cứng đắt đỏ.

**Chương 3: Thu thập và tiền xử lý dữ liệu**

**3.1. Thu thập dữ liệu**

Do tính riêng tư của ảnh người thật nên việc thu thập dữ liệu được chúng em lựa chọn và xáo trộn với nhau từ những tập dữ liệu hình ảnh giả và thật có sẵn miễn phí trên trang web kaggle với lượng dữ liệu gốc là 190 mươi nghìn hình ảnh với đuôi dạng jpg được chia sẵn dữ liệu ra 3 thư mục Train, Validation, Test:



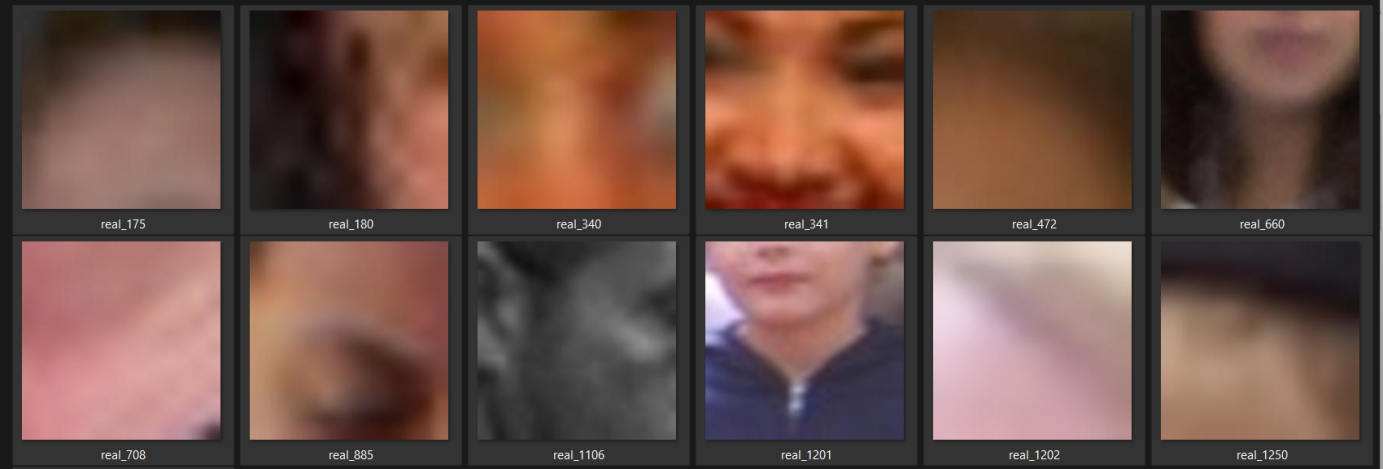
Hình 8: Dữ liệu gốc trên Kaggle

Nguồn dữ liệu hình ảnh thật/giả được lấy từ trang kaggle:

<https://www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images/data>

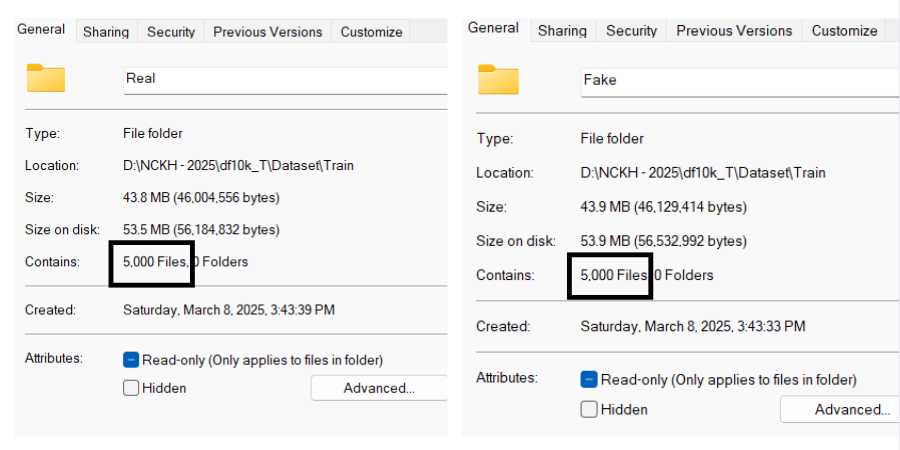
**3.2. Tiền xử lý dữ liệu**

Sau khi tải về, do lượng dữ liệu quá lớn nếu sử dụng để huấn luyện với môi trường đám mây miễn phí của Colab sẽ không đủ tài nguyên. Chính vì vậy chúng em lọc bớt dữ liệu và chỉ lấy 10 nghìn ảnh để huấn luyện. Lọc đi những dữ liệu không hợp lệ như ảnh quá mờ, hoặc không rõ hình dạng:



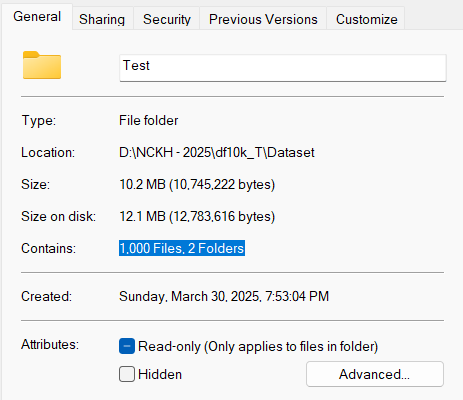
Hình 9: Lọc dữ liệu không hợp lệ trong Dataset

Ngoài ra, để trong lúc huấn luyện mô hình không thiên vị khi tập dữ liệu Fake/Real chênh lệch nhau gây mất cân bằng, ta tiến hành cân bằng tập dữ liệu giữa nhãn Real và nhãn Fake cho đều nhau:



Hình 10: Chia đều dữ liệu cho 2 nhãn Real và Fake

Với tập Test, tương tự chúng em cũng lọc dữ liệu như thế với folder Fake 500 ảnh và Real 500 ảnh, lúc này ta có được tập dữ liệu hoàn chỉnh để huấn luyện cho mô hình.



Hình 11: Dữ liệu tập Test

Khi huấn luyện dữ liệu với mô hình ResNet50, kích thước ảnh đầu vào phù hợp và cho ra kết quả tốt nhất với kích thước ảnh là 224x224 pixels, với Meso4 thì kích thước là 256x256 pixels. Nên cần điều chỉnh lại tất cả kích thước của ảnh, tổng cộng là 10 nghìn hình (RandomResizedCrop). Ngoài ra, để cho mô hình có thể học được hình ảnh ở những góc độ khác nhau, ta lật (RandomHorizontalFlip), xoay hình ảnh ở nhiều góc (RandomRotation) và điều chỉnh ảnh ở chế độ ảnh màu (Normalize). Bên cạnh đó do sử dụng model ResNet50 của Pytorch nên ta cũng đổi dạng ảnh từ PIL Image sang Pytorch Tensor (To Tensor).

**Đối với ResNet50:**

|  |
| --- |
| transform = transforms.Compose([  transforms.RandomHorizontalFlip(),  transforms.RandomRotation(10),  transforms.RandomResizedCrop(224, scale=(0.8, 1.0)), #224x224 phù hợp với mạng resnet50  transforms.ToTensor(),  transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])  ]) |

**Đối với Meso4:**

|  |
| --- |
| transform\_meso4 = transforms.Compose([  transforms.RandomHorizontalFlip(),  transforms.RandomRotation(10),  transforms.RandomResizedCrop(256, scale=(0.8, 1.0)), #256x256  transforms.ToTensor(),  transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])  ]) |

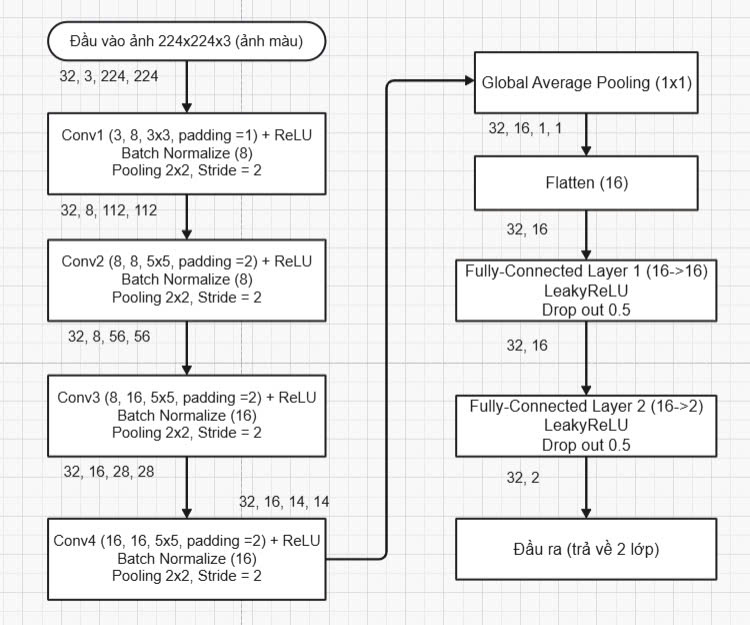
**Chương 4: Xây dựng và huấn luyện mô hình**

**4.1. Mô hình Mesonet**

Sử dụng mô hình Meso4 với 4 tầng tích chập, kiến trúc khá đơn giản và là một kiến trúc nhẹ, ứng dụng ổn trong các mô hình phát hiện Deepfake, tối ưu phát hiện Deepfake ảnh tĩnh.

Để phù hợp với quá trình huấn luyện và dung lượng giới hạn trên notebook Colab, chúng em đã cấu hình và chỉnh sửa lại kiến trúc và đầu ra của mô hình để tiến trình ổn định và nhanh hơn. Dưới đây là mô hình kiến trúc tổng quát dùng để huấn luyện trên Colab:

* Với Meso4 thì mô hình sẽ có 4 lớp tích chập, mà ở mỗi lớp đều sẽ dùng hàm kích hoạt ReLU và lớp Pooling để làm giảm số lượng kênh

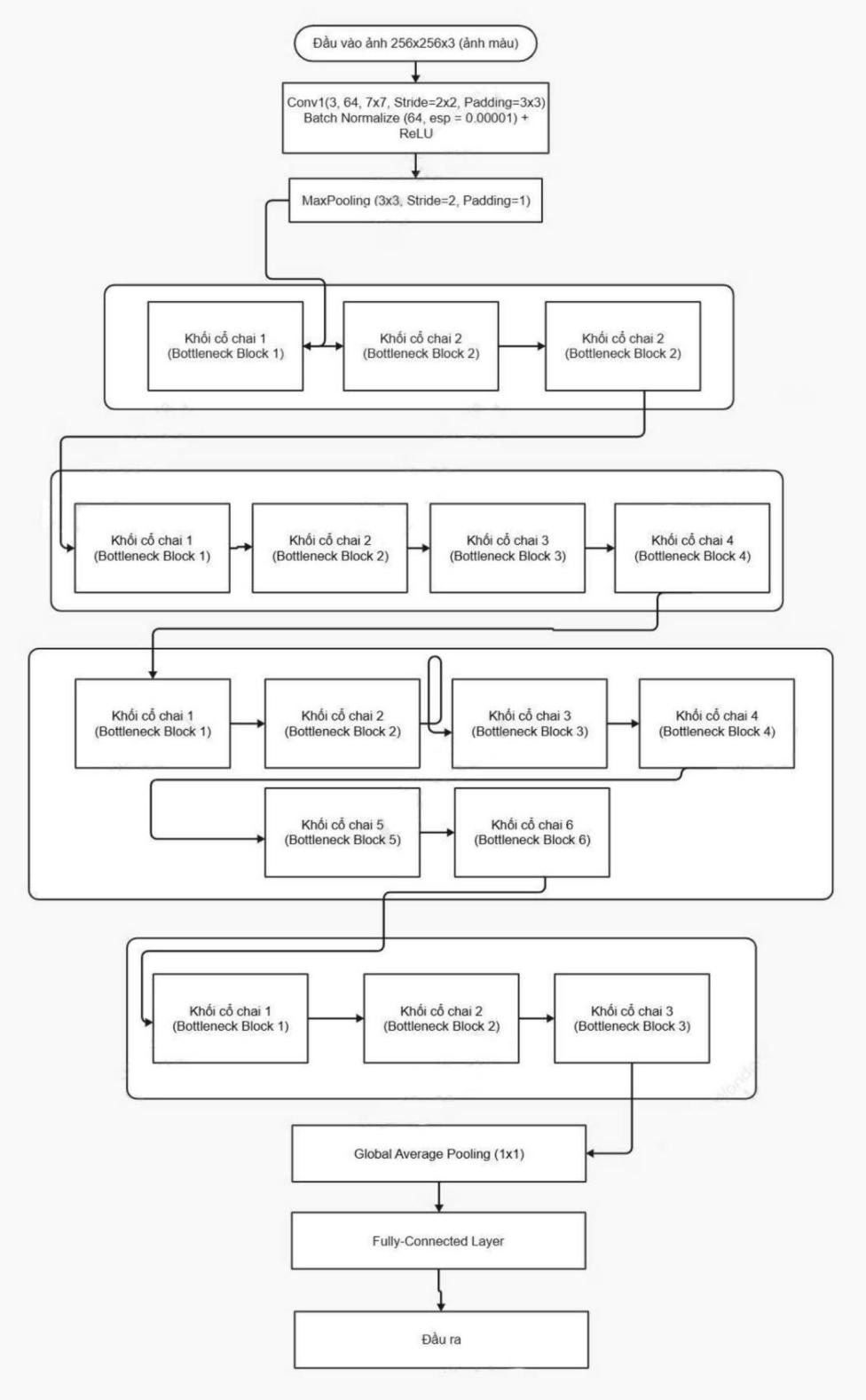


Hình 12: Kiến trúc tổng quát mô hình Meso4

**4.2. Mô hình ResNet**

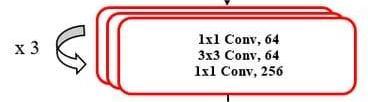
Với ResNet, chúng em sử dụng mô hình ResNet50 với số tầng sâu hơn phù hợp để học chi tiết các đặc trưng của dữ liệu, với kĩ thuật Skip Connection cũng như các khối Residual Blocks giúp giảm mất mát thông tin đáng kể. Mô hình có khả năng phát hiện Deepfake tốt hơn nhưng đồng thời cũng yêu cầu tài nguyên lớn.

ResNet50 sẽ có 4 tầng tích chập chính mà mỗi tầng sẽ có số khối cổ chai khác nhau, mỗi khối cổ chai sẽ có 3 lớp tích chập, lớp tích chập đầu tiên mang bộ lọc 1x1 mang nhiệm vụ giảm số lượng kênh, tiếp tục đến lớp tích chập số 2 sẽ là 3x3 để trích xuất đặc trưng và sau cùng là lớp có bộ lọc 1x1 lần nữa mà lần này sẽ mang nhiệm vụ tăng số lượng kênh trở lại để đến khối cổ chai tiếp theo.



Hình 13: Kiến trúc tổng quát mô hình ResNet50

Đây là ảnh minh họa cho tầng tích chập đầu tiên với 3 khối cổ chai:



Hình 14: Minh họa tầng tích chập đầu tiên của ResNet50

**4.3. Huấn luyện mô hình**

Để mô hình sau huấn luyện cho kết quả khách quan hơn thì chúng em cho huấn luyện cả 2 mô hình Meso4 của Mesonet và Resnet50 để đưa ra và so sánh kết quả về hiệu suất cũng như độ nhận diện một cách khách quan nhất.

* Cấu hình cho đầu ra của mô hình Meso4 sử dụng hàm kích hoạt LeakyReLU và ở lớp kết nối cuối cùng với đầu ra 2 lớp là Fake và Real:

|  |
| --- |
| self.pool = nn.MaxPool2d(2, stride=2, padding=0)  self.global\_pool = nn.AdaptiveAvgPool2d((1, 1))  self.fc1 = nn.Linear(16, 16)  self.fc2 = nn.Linear(16, num\_classes)  self.dropout = nn.Dropout(0.5)  self.leakyrelu = nn.LeakyReLU(negative\_slope=0.1) |

Để mô hình huấn luyện tốt hơn thì chúng em có sử dụng quá trình xác thực chéo, chia dữ liệu huấn luyện 10 nghìn ảnh thành 5 phần bằng nhau mà mỗi phần chứa 2 nghìn ảnh, dữ liệu sẽ luân phiên được huấn luyện với 5 phiên làm việc, mỗi phiên huấn luyện 4 epochs và batch size là 32, cứ dùng 4 phần dữ liệu cho huấn luyện thì 1 phần còn lại cho kiểm tra và đưa ra kết quả. Đồng thời tuning tham số learning rate (tỉ lệ học) của mô hình để xem mô hình hoạt động ổn định và cho kết quả tốt nhất ở tỉ lệ learning rate nào:

|  |
| --- |
| k\_folds = 5  kf = KFold(n\_splits=k\_folds, shuffle=True)  learning\_rates = [1e-3, 1e-4, 1e-5] #tuning learning rate |

* Đối với ResNet50, ta tái sử dụng mô hình ResNet50 có sẵn của Pytorch và cấu hình đầu ra cho mô hình trong quá trình huấn luyện, với đầu ra sử dụng hàm kích hoạt Linear và số lớp là 2 tương ứng với lớp Fake (ảnh giả) và Real (ảnh thật):

|  |
| --- |
| model = models.resnet50(pretrained=True)  model.fc = nn.Sequential(  nn.Linear(model.fc.in\_features, 512),  nn.ReLU(),  nn.Dropout(0.5),  nn.Linear(512, num\_classes)  )  model = model.to(device) |

Cũng tương tự như mô hình Meso4, ResNet50 cũng dùng xác thực chéo để huấn luyện mô hình và tuning tham số learning rate để kiểm tra xem mô hình hoạt động tốt ở mức learning rate nào trong 3 tham số [1e-3, 1e-4, 1e-5].

|  |
| --- |
| k\_folds = 5  kf = KFold(n\_splits=k\_folds, shuffle=True)  learning\_rates = [1e-3, 1e-4, 1e-5] |

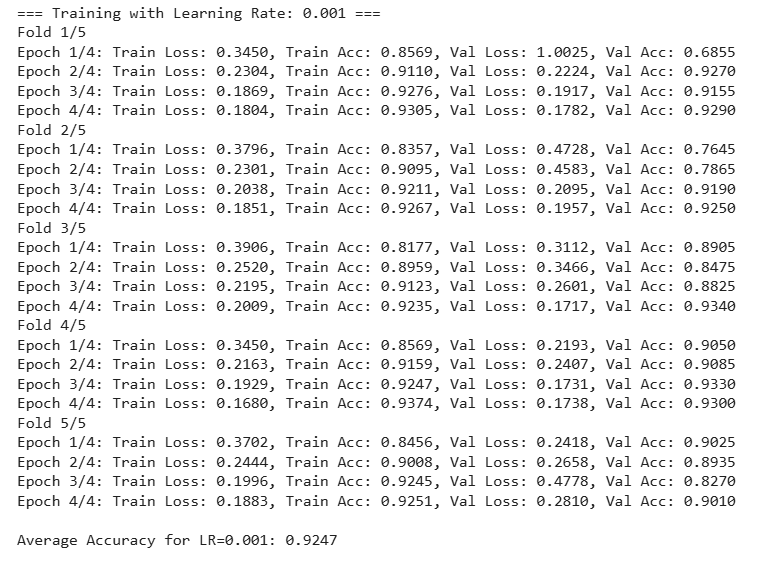
**4.4. Đánh giá mô hình**

Sau khi huấn luyện, mô hình đã cho những kết quả khá khả quan. Tuy nhiên với số lượng 10 nghìn ảnh và tốc độ GPU của phiên bản notebook miễn phí của Colab thì việc huấn luyện khá mất thời gian với thời gian huấn luyện ResNet50 gần 2 giờ đồng hồ và Meso4 nhanh hơn với gần 1 giờ đồng hồ.

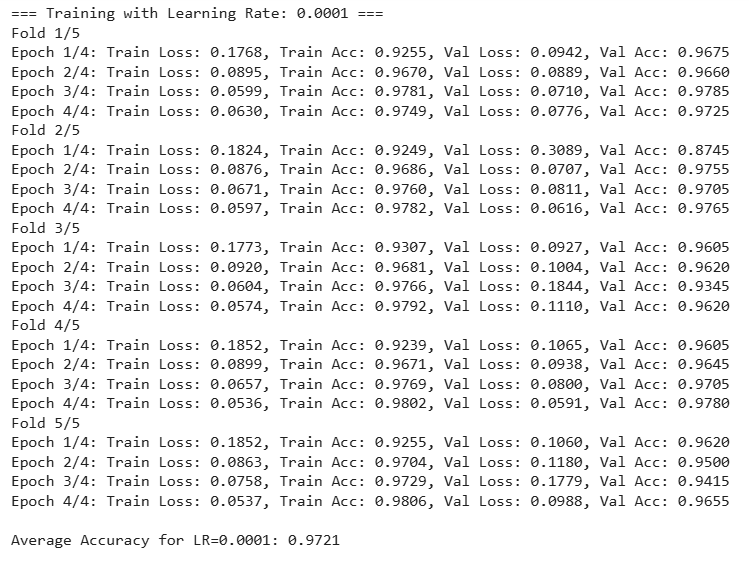
* **Biểu đồ Loss và Val:**

**+ Với ResNet50:**

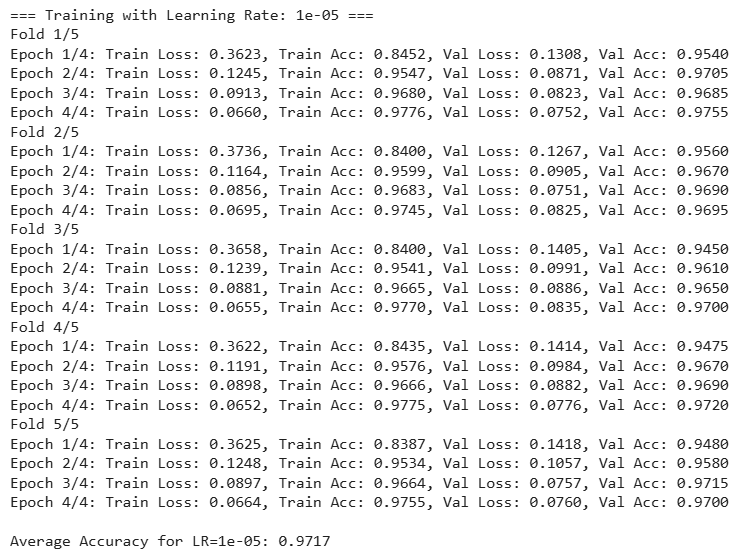
Sau gần 2 giờ đồng hồ cho việc huấn luyện dữ liệu, mô hình cho kết quả huấn luyện tốt nhất ở mức learning rate 1e-04 với độ chính xác khoảng 0.9721, trong khi đó với mức learning rate 1e-03 chỉ có 0.92 và 0.917 khá suýt soát với learning rate 1e-05.



Hình 15: Huấn luyện ở mức learning rate 1e-03 mô hình ResNet50

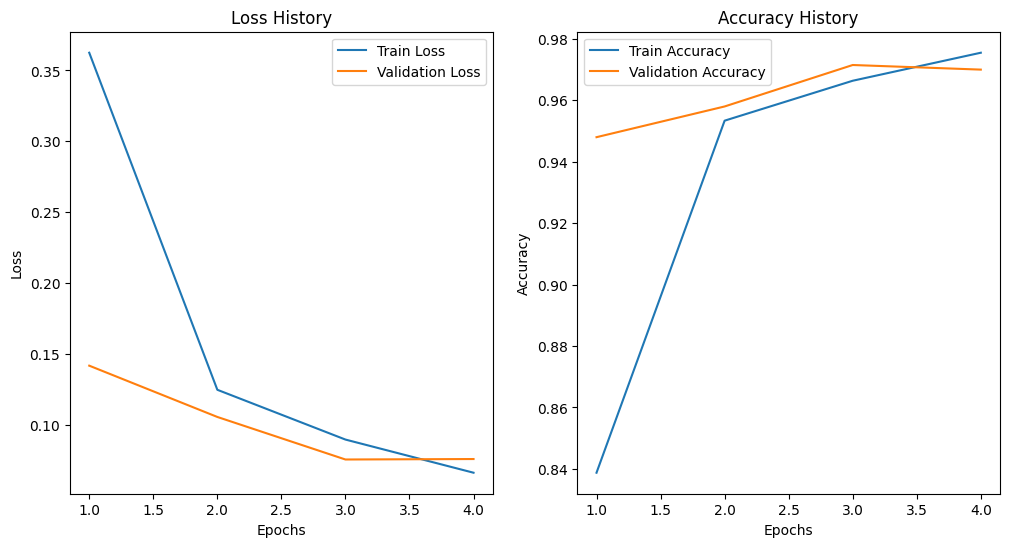


Hình 16: Huấn luyện ở mức learning rate 1e-04 mô hình ResNet50



Hình 17: Huấn luyện ở mức learning rate 1e-05 mô hình ResNet50

Sau khi huấn luyện, trực quan hóa biểu đồ loss và accuracy để ta có thể nhận xét một cách tổng quan nhất. Nhìn vào biểu đồ có thể thấy biểu đồ loss giảm dần và dần hội tụ gần nhau cho thấy không có dấu hiệu mô hình bị overfitting (đường train giảm trong khi val tăng) chứng tỏ mô hình học tập rất tốt. Sang biểu đồ accuracy, có thể thấy độ chính xác tăng qua từng epoch và duy trì mức độ tăng ổn định với độ chính xác cao, biểu đồ accuracy của train và val nằm gần nhau, tuy đường val tăng chậm hơn nhưng cũng cho thấy mô hình hoạt động và cho kết quả rất khả quan:



Hình 18: Biểu đồ Loss và Accuracy của ResNet50

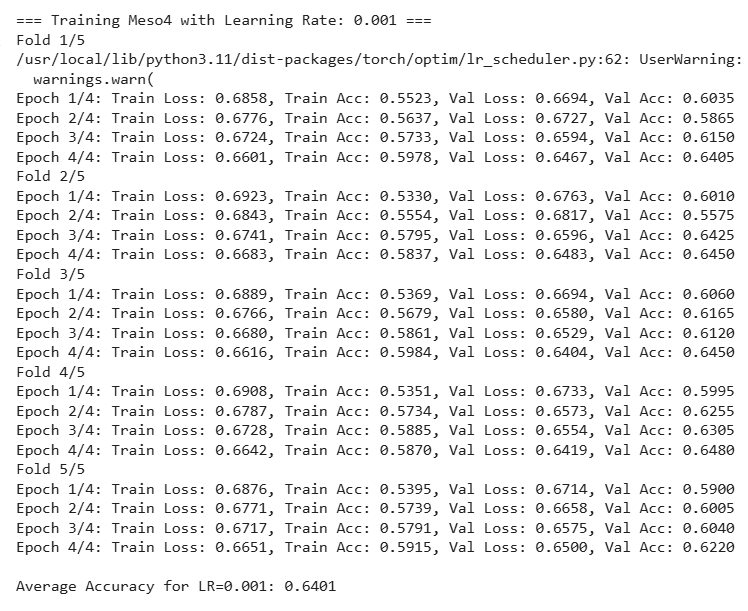
Kết quả kiểm thử trên tập test, cho kết quả dự đoán chính xác với 2 ảnh giả:



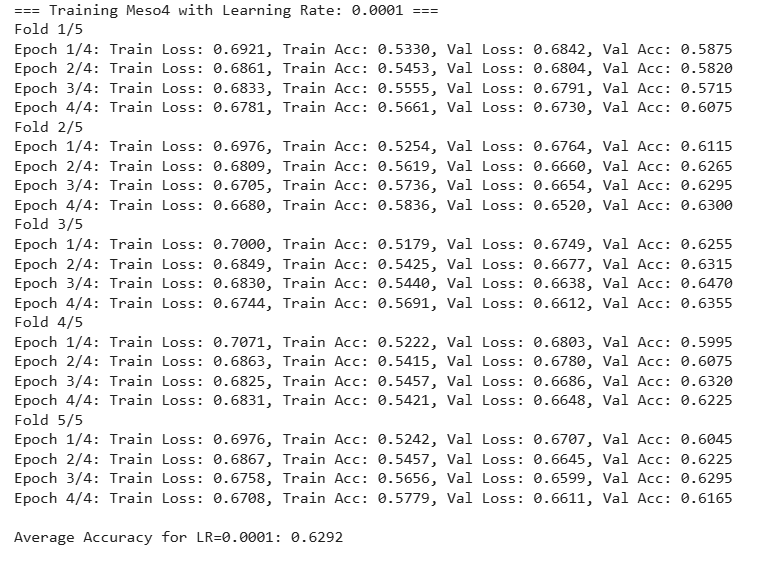
Hình 19: Dự đoán kết quả trên tập test với ResNet50

**+ Với Meso4:**

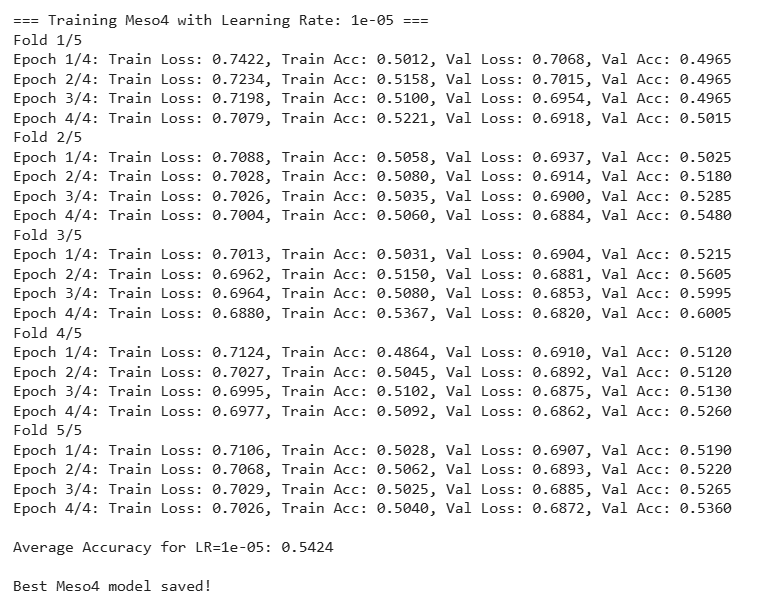
Đối lập với mô hình ResNet50, Meso4 tuy cho tốc độ huấn luyện nhanh hơn nhưng đồng thời độ chính xác lại không cao và dự đoán sai khá nhiều so với mô hình ResNet50. Cũng tuning ở 3 tham số learning rate [1e-03, 1e-04, 1e-05] nhưng kết quả tốt nhất mô hình nhận được ở mức learning rate 1e-03 với độ chính xác khoảng 0.64.

****

Hình 20: Huấn luyện ở mức learning rate 1e-03 mô hình Meso4

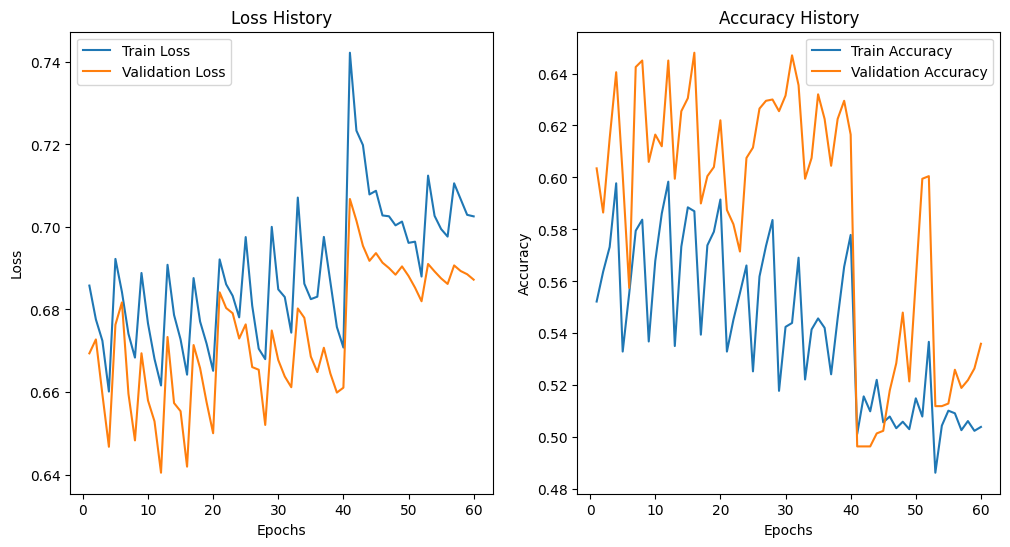
****

Hình 21: Huấn luyện ở mức learning rate 1e-04 mô hình Meso4

****

Hình 22: Huấn luyện ở mức learning rate 1e-05 mô hình Meso4

Biểu đồ loss và accuracy của Meso4 cũng bất thường và cho kết quả không ổn định, sự giao động quá lớn của cả biểu đồ loss và accuracy chứng tỏ mô hình Meso4 học chưa tốt, kém hơn rất nhiều so với ResNet50, có thể thấy trong quá trình tuning biểu đồ giao động mạnh và không đều. Điều này có thể một phần do vấn đề hạn chế về GPU, chúng em đã tinh chỉnh thông số để phù hợp với mức GPU miễn phí mà nền tảng cho phép nên có lẽ mô hình đã bị giới hạn khả năng hoạt động 1 phần.



Hình 23: Biểu đồ Loss và Accuracy của Meso4

Kết quả kiểm thử trên tập test cho thấy mô hình Meso4 nhầm lẫn nhiều, khi ResNet50 dự đoán đúng 2 ảnh với nhãn giả thì Meso4 lại nhầm chúng là ảnh thật:



Hình 24: Dự đoán kết quả trên tập test với Meso4

* **Classification Report:**

Ta sẽ đánh giá hiệu suất của mô hình dựa trên các chỉ số hiệu xuất sau:

+ Accuracy: độ chính xác trên toàn tập

+ Precision: độ chính xác với dự đoán

+ Recall: tỷ lệ phát hiện đúng

+F1-Score: trung bình hài hòa của Precision và Recall

**Label Fake:**

- Precision: TP/TP+FP

- Recall: TP/TP+FN

- F1 Score: 2.(Presion.Recall)/(Presion + Recall)

**Label Real:**

- Precision: TN/TN+FN

- Recall: TN/TN+FP

- F1 Score: 2.(Presion.Recall)/(Presion + Recall)

**Với:**

- TP (True Positive) - Dự đoán đúng với nhãn dương (Fake)

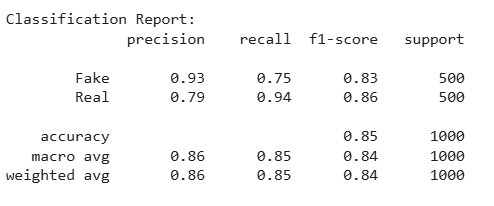
- TN (True Negative) - Dự đoán đúng với nhãn âm (Real)

- FP (False Positive) - Dự đoán sai, nhưng thực tế là âm (nhầm lẫn nhãn Real thành Fake)

- FN (False Negative) - Dự đoán sai, nhưng thực tế là dương (nhầm lẫn nhãn Fake thành Real)

**+ Classification Report của ResNet50:**

Sau huấn luyện, ta sử dụng model sau huấn luyện in ra bảng Classification Report để xem các chỉ số Precision, Recall, F1-Score xem đánh giá trên tập Test:

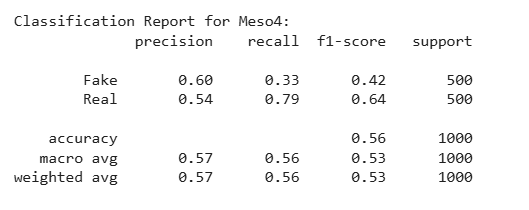


Hình 25: Classification Report của ResNet50

Nhìn vào số liệu, ta thấy mô hình ResNet50 có chỉ số Precision của lớp Fake là 0.93, điều này cho thấy khả năng dự đoán ảnh Fake cực kỳ chính xác. Tuy nhiên, ở lớp Real, Precision chỉ đạt 0.79, nghĩa là trong số các ảnh được dự đoán là Real, có tới 0.21 thực chất là Fake. Đối với chỉ số Recall, mô hình phát hiện được 75% ảnh Fake thực tế, đồng nghĩa với việc bỏ sót 25% ảnh giả. Trong khi đó, Recall của lớp Real lên tới 0.94 chứng tỏ mô hình nhận diện ảnh thật rất tốt. Điều này cho thấy mô hình gặp khó khăn với việc nhận diện ảnh giả mạo chất lượng cao hơn, trong khi dễ dàng nhận biết đặc điểm của ảnh thật. Cuối cùng với F1-score - chỉ số cân bằng giữa Precision và Recall - đạt 0.83 cho lớp Fake và 0.86 cho lớp Real cho một kết quả khá cao, đặc biệt là ở khả năng phát hiện ảnh Fake. Độ chính xác tổng thể của mô hình với accuracy là 0.85 cho thấy mô hình này đáng tin cậy để triển khai thực tế tuy nhiên vẫn chưa phải quá cao khi mô hình vẫn còn nhận diện sai ảnh thật thành giả do đó vẫn cần điều chỉnh để giảm thiểu các trường hợp này.

**+ Classification Report của Meso4:**

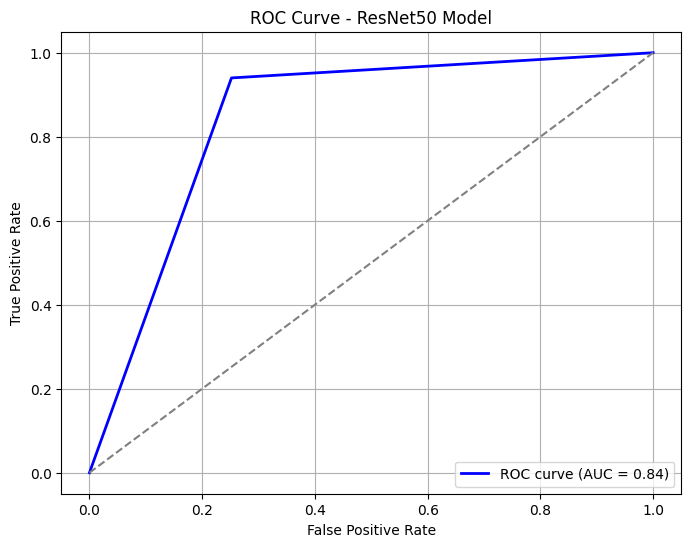
Tương tự như ResNet50, với Meso4 ta cũng in bảng Report đánh giá trên tập Test:



Hình 26: Classification Report của Meso4

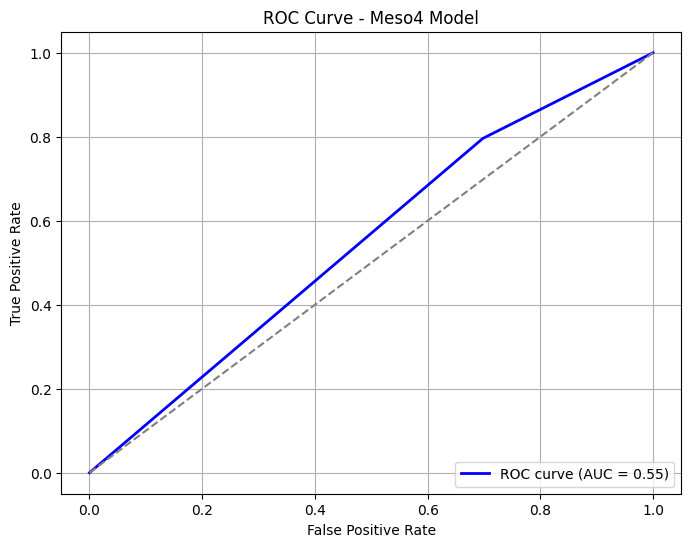
Mô hình Meso4 thể hiện sự chênh lệch rõ rệch giữa các lớp và cho số liệu đánh giá quá thấp. Với Precision của lớp Fake đạt 0.6 chứng tỏ việc nhầm lẫn cao khi tới 0.4 còn lại nhầm giữa Fake thành Real. Ở lớp Real, Precision còn thấp hơn chỉ đạt ở mức 0.54 cho thấy mô hình gần như không thể phân biệt rõ ràng giữa hai lớp với gần một nửa số ảnh được dự đoán là Real thì thực chất nhãn gốc lại là Fake cho thấy mô hình còn nhầm lẫn rất nhiều giữa 2 lớp. Chỉ số Recall của lớp Fake chỉ đạt 0.33 nghĩa là mô hình bỏ sót tới 67% ảnh Fake trong thực tế, một tỉ lệ quá cao đối với một mô hình nhận diện. Ngược lại, Recall của lớp Real lên tới 0.79 cho thấy mô hình có xu hướng nghiêng về dự đoán Real. Sự chênh lệch lớn này cho thấy mô hình gặp khó khăn nghiêm trọng trong việc nhận diện các đặc trưng của ảnh Fake, việc nhầm lẫn quá nhiều như vậy nếu sử dụng mô hình vào thực tế sẽ mang lại hiệu quả cực kém. F1-score chỉ đạt 0.42 cho lớp Fake và 0.64 cho lớp Real cho thấy hiệu suất tổng thể của mô hình thấp, thêm vào đó độ chính xác của mô hình chỉ ở mức 0.56, chỉ số chỉ nằm xấp xỉ ở ngưỡng này cho thấy mô hình Meso4 trong huấn luyện lần này chưa đủ khả năng để triển khai trong thực tế. Mô hình cần phải được cải thiện, điều chỉnh thông số cũng như cho mô hình huấn luyện với lượng dữ liệu lớn hơn cũng như nâng cấp nền tảng đám mây Colab để mang lại hiệu quả tốt nhất thay vì ngắt quãng gián đoạn làm giảm hiệu suất mô hình.

* **Đồ thị ROC-AUC:**



Hình 27: ROC Curve ResNet50

* **ResNet50:** AUC đạt 0.84 cho thấy mô hình này có khả năng phân biệt tốt giữa ảnh thật và ảnh giả. Cụ thể, khi chấp nhận khoảng 20% ảnh thật bị nhầm thành ảnh giả (FPR=0.2), mô hình có thể phát hiện được tới 80% ảnh giả thực sự (TPR=0.8). Điều này chứng tỏ ResNet50 đạt được sự cân bằng tốt giữa độ chính xác và độ bao phủ, phù hợp để triển khai trong thực tế. Hiệu suất cao của ResNet50 đến từ kiến trúc mạng sâu với các khối residual, cùng khả năng trích xuất đặc trưng mạnh mẽ nhờ được pre-train trên tập dữ liệu lớn ImageNet.



Hình 28: ROC Curve Meso4

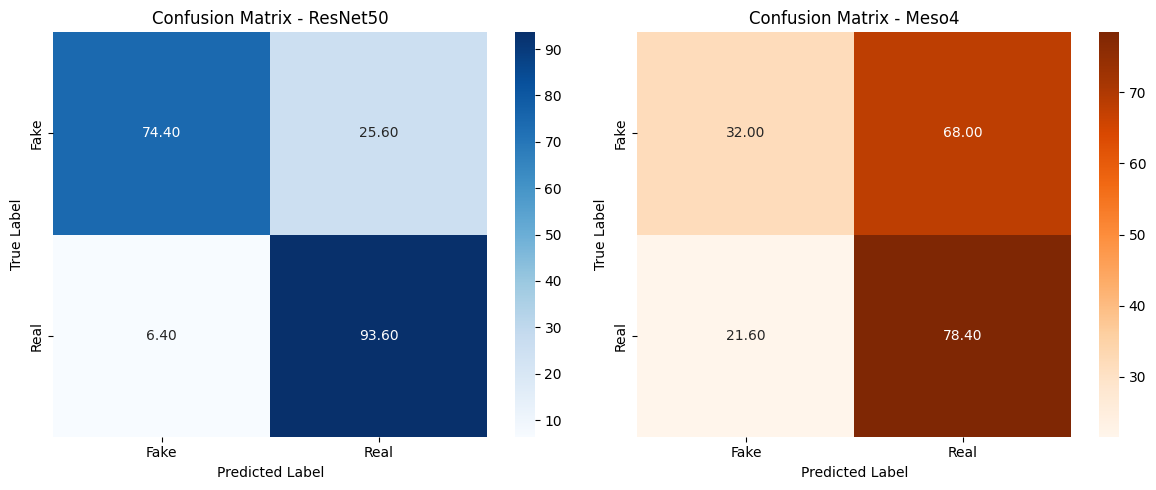
* **Meso4:** Trái ngược hoàn toàn, Meso4 chỉ đạt AUC=0.55 - một con số gần với ngưỡng đoán ngẫu nhiên (0.5), cho thấy mô hình này gần như không có khả năng phân biệt ảnh thật và ảnh giả. Để phát hiện được 60% ảnh giả (TPR=0.6), mô hình phải chấp nhận tới 50% ảnh thật bị phân loại nhầm (FPR=0.5). Hiệu suất kém này bắt nguồn từ kiến trúc quá nông của Meso4 với chỉ 4 lớp tích chập, cùng việc sử dụng stride lớn khiến mất nhiều thông tin quan trọng. Mô hình không có khả năng học các đặc trưng phức tạp cần thiết để nhận biết ảnh giả chất lượng cao. Kết quả này cho thấy Meso4 trong trường hợp hiện tại chưa đủ tin cậy để ứng dụng thực tế và cần được cải tiến đáng kể về cả kiến trúc lẫn phương pháp huấn luyện.

**4.5. Kết quả hai mô hình**

Thông qua biểu đồ Confusion Matrix có thể thấy được:

* Đối với mô hình ResNet50, tỷ lệ dự đoán nhầm từ ảnh giả sang ảnh thật trên tập test ở mức 25.6%, trong khi đó tỷ lệ dự đoán chính xác ảnh giả là 74.4%. Đồng thời, mô hình nhận diện ảnh thật với độ chính xác cao, chỉ có 6.4% bị nhầm thành ảnh giả, còn lại 93.6% được nhận diện đúng. Qua đó thấy được, tỉ lệ dự đoán đúng ảnh thật trên tập Test cao hơn nhiều so với tỉ lệ dự đoán đúng ảnh giả, kết quả này có thể nói là tạm chấp nhận được khi mà với bài toán thực tế nhận diện Deepfake, mô hình nên chú trọng vào việc nhận diện đúng ảnh giả hơn.
* Ngược lại, mô hình Meso4 có hiệu suất thấp hơn rất nhiều, nhầm lẫn tới 68% ảnh giả thành ảnh thật, chỉ nhận diện đúng 32% ảnh giả. Đối với ảnh thật, tỷ lệ nhầm lẫn cũng cao hơn, 21.6% bị nhận diện nhầm thành ảnh giả, chỉ có 78.4% được dự đoán đúng. Chứng tỏ rằng cùng một số lượng epochs và fold chia như nhau, hiệu suất của mô hình Meso4 thấp hơn rất nhiều so với ResNet50, việc chỉ nhận diện đúng 32% ảnh giả cho thấy mô hình này không thể nào sử dụng với việc nhận diện Deepfake trong thực tế, chỉ số nhầm lẫn Fake thành Real, Real thành Fake quá cao cho thấy mô hình vẫn chưa được tối ưu.

Từ kết quả trên, có thể thấy rằng ResNet50 hoạt động hiệu quả hơn rõ rệt trong việc nhận diện hình ảnh đã qua Deepfake so với Meso4.



Hình 29: Biểu đồ ma trận nhầm lẫn so sánh 2 mô hình ResNet50 và Meso4

Thêm vào đó, về chỉ số Accuracy giữa 2 mô hình trên tập Test:

+ ResNet50 cho độ chính xác 0.85 trong khi đó Meso4 chỉ cho ra 0.56, điều này chứng tỏ mô hình ResNet điều này chứng tỏ mô hình ResNet50 có độ chính xác vượt trội so với Meso4 khi nhận diện hình ảnh Deepfake. Với độ chính xác 85%, ResNet50 cho thấy khả năng phân loại ổn định hơn và đáng tin cậy hơn trong môi trường thực tế. Trong khi đó, với độ chính xác chỉ 56%, mô hình Meso4 không đạt yêu cầu cho bài toán này, khi tỷ lệ nhầm lẫn quá cao có thể dẫn đến những sai sót nghiêm trọng trong ứng dụng thực tiễn. Ngoài ra, từ kết quả confusion matrix, có thể nhận thấy rằng ResNet50 có độ nhạy (recall) cao hơn đối với ảnh thật, điều này giúp giảm thiểu việc nhận diện nhầm các ảnh không bị Deepfake. Tuy nhiên, việc dự đoán đúng ảnh giả vẫn chưa đạt mức lý tưởng, cho thấy cần có những cải tiến để nâng cao khả năng phát hiện Deepfake hơn nữa.

Tóm lại, ResNet50 thể hiện khả năng vượt trội hơn Meso4 trong bài toán nhận diện Deepfake cả về độ chính xác tổng thể lẫn khả năng phân biệt ảnh thật và giả.

**Chương 5: Xây dựng Web App phát hiện ảnh Deepfake**

**5.1. Xây dựng FrontEnd và BackEnd**

Hệ thống gồm 2 phần:

* **Backend** : Nơi xử lý các yêu cầu, xử lý logic, nhiệp vụ.
* **Frontend** : Nơi hiển thị dữ liệu và trình bày hóa cho người dùng.

Công nghệ sử dụng:

* **Backend:**

Flask – là một framework nhẹ cho phát triển ứng dụng web phía backend.

OpenCV – là một thư viện được thiết kế để giải quyết các vấn đề thị giác máy tính.

Pytorch – là một framework học máy, được tạo bởi Facebook.

* **Frontend :**

ReactJS – là một framework thông dụng phát triển ứng dụng web phía client.

ReactBits – là một thư viện mã nguồn mở cung cấp các componnet tiện lợi.

Hình ảnh Giao diện mẫu:



Hình 30: Giao diện trang web

**5.2. Xử lý nhận diện khuôn mặt**

Sử dụng mô hình được huấn luyện sẵn: Haar Cascade Frontalface, sử dụng để nhận diện khuôn mặt theo thời gian thực, nhanh và tiện dụng. (Phụ lục 2)

Các tham số :

* scaleFactor : Xác định tỉ lệ thu nhỏ ảnh mỗi lần quét để tìm khuôn mặt ở các kích thước khác nhau.Giá trị lớn thì tăng tốc nhưng có thể bỏ sót một số khuôn mặt, nhỏ hơn thì sẽ phát hiện chính xác hơn nhưng chậm hơn
* minNeighbors : Số lượng hàng xóm tối thiểu để một vùng được xác định là khuôn mặt.

Cách sử dụng:

* Tải mô hình về máy, định cấu hình cho nó.
* Đầu vào là hình ảnh
* Đầu ra là một mảng một chiều chứa thông tin về boudingBox (x,y,w,h).Cho phép chúng ta cắt các ảnh nhận diện và vẽ đánh dấu khuôn mặt đã được nhận dạng.

**5.3. Cấu hình để load mô hình Deepfake**

Để có thể sử dụng mô hình trên Web App, ta cần làm theo những bước sau:

* Định nghĩa mô hình Meso4 (Phụ lục 3)
* Định nghĩa mô hình ResNet50 (Phụ lục 4)
* Xử lí chính và dự đoán: ta định nghĩa mô hình và tải lên các trọng số bias, của mô hình và định nghĩa transform để bắt đầu cho việc dự đoán (Phụ lục 5) rồi định nghĩa route dự đoán và thực hiện dự đoán và đưa ra kết quả (Phụ lục 6)

1. **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**
2. **Kết luận chung về đề tài**

Trong quá trình làm và phát triển đề tài, chúng em nhận thấy có những ưu và nhược điểm khi làm về nhận diện hình ảnh deepfake.

* Tổng kết kết quả: Hai mô hình khi huấn luyện hoàn tất với kết quả, Resnet50 – mô hình cho kết quả tốt hơn Meso4 rất nhiều theo độ chính xác 85% cho Resnet50 và 56% cho Meso4. Về các thông số khác Meso4 quá thấp với f1-score chỉ ở mức 0.42 dưới mức 0.5.Về Meso4 với kiến trúc khá nông không phù hợp với bài toán deepfake, cho thấy Resnet50 với kiến trúc sâu và có nhiều tính năng đặc biệt có thể phù hợp với bài toán nhận diện deepfake.
* Về nhược điểm: do tài nguyên hạn chế, việc sử dụng nền tảng đám mây Colab mà không mua phiên bản nâng cấp khiến quá trình huấn luyện gặp nhiều khó khăn khi Colab chỉ cho miễn phí sử dụng GPU 15GB mỗi ngày và đặt lại trạng thái chạy sau mỗi 12 giờ, điều đó gây khó khăn khi chạy với số lượng batch và epoch lớn hay sử dụng mô hình ResNet cao cấp hơn làm tiêu hao GPU quá nhiều, hết dung lượng trước khi thấy được thành quả. Tập dữ liệu sử dụng để huấn luyện cũng không lớn, chỉ khoảng 10 nghìn ảnh nên các đặc trưng mà mô hình học được chưa đủ lớn, tuy cho kết quả tốt nhưng khi thử nghiệm với những hình ảnh giả mạo tiên tiến hiện nay thì vẫn cho ra kết quả nhầm lẫn nhiều. Đồng thời, dữ liệu ảnh về mặt người thật cũng mang tính bảo mật cao nên chỉ được sử dụng những nguồn ảnh có sẵn trên các nền tảng được cho phép khiến việc tìm kiếm ảnh chất lượng tốt hơn trở nên khó khăn.Tóm lại tài nguyên vận hành còn hạn chế, tập dữ liệu còn chưa bao quanh hoàn toàn dữ liệu deepfake, chưa thực nghiệm để đưa ra dự đoán theo khung hình cho video.

1. **Hướng phát triển của đề tài**

Cần thu thập dữ liệu có tính chọn lọc cao, phù hợp và có tính theo thời gian, cập nhật và nâng cấp giúp mô hình phát triển và học được những đặc điểm của deepfake một cách thực tế nhất và chính xác cao, nâng hiệu suất cao và phù hợp với thực tế.

Chặng đường tiếp theo hướng đến phát hiện deepfake trên video mà người dùng cung cấp, phân tách từng khung hình và đưa vào mô hình học máy để dự đoán theo thời gian của video, giúp người dùng có thể theo dõi và hình dung và trực quan hơn. Để làm được điều đó cần phải tìm hiểu kỹ lưỡng về mô hình học máy để tăng hiệu suất tốc độ dự đoán và độ chính xác.

Ngoài ra, đề tài này chỉ dừng lại ở việc nhận diện mặt người, chúng ta có thể tiến xa hơn với việc thu thập dữ liệu khác từ phong cảnh hay là động vật để việc nhận diện dữ liệu đa dạng hơn. Ứng dụng mô hình cho các tổ chức hay công ty để nhận diện giả mạo, chống lừa đảo.

**TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | N. T. Tuấn, Deep Learning cơ bản, 2020. |
| [2] | X. Z. S. R. J. S. Kaiming He, Deep Residual Learning for Image Recognition, 2015. |
| [3] | geeksforgeeks, "Residual Networks (ResNet) – Deep Learning," [Online]. Available: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/. |
| [4] | B. Banjara, "Deep Residual Learning for Image Recognition (ResNet Explained)," January 2025. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/02/deep-residual-learning-for-image-recognition-resnet-explained/#h-how-resnet-works. |
| [5] | D. D. i. D. Learning, "Mạng phần dư (ResNet)," [Online]. Available: http://d2l.aivivn.com/chapter\_convolutional-modern/resnet\_vn.html. |

**PHỤ LỤC**

1. Code lưu mô hình vào thư mục save models:

|  |
| --- |
| save\_path = "/content/drive/MyDrive/Colab Notebooks/save models/best\_resnet50\_T1.pth"  torch.save(model.state\_dict(), save\_path)  print(f"Model saved at {save\_path}") |

1. Code xử lý nhận diện khuôn mặt:

|  |
| --- |
| import cv2  from util import generateUnique as gn  def getFaceIMG(filePath):  face\_detect = cv2.CascadeClassifier("model/haarcascade\_frontalface\_default.xml")  img = cv2.imread(filePath)  faces = face\_detect.detectMultiScale(img, scaleFactor=1.3, minNeighbors=6)  detectList = []  try:  i = 0  for x, y, w, h in faces:  detect\_face = img[y : y + h, x : x + w]  ext = filePath.split(".")[-1]  detectFiles = f"face\_{i}\_{gn.generateUniqueTimestamp()}.{ext}"  detect\_filePath = f"store\\{detectFiles}"  cv2.imwrite(detect\_filePath, detect\_face)  detectList.append(detectFiles)  i += 1    for x, y, w, h in faces:  cv2.rectangle(img, (x, y, w, h), (255, 0, 0), 3)  org\_file = f"org\_{gn.generateUniqueTimestamp()}.{ext}"  org\_filePath = f"store\\{org\_file}"  cv2.imwrite(org\_filePath, img)  return org\_file, detectList  except:  return filePath.split("\\")[-1], [] |

1. Định nghĩa mô hình Meso4:

|  |
| --- |
| import torch.nn.functional as F  class Meso4(nn.Module):  def \_\_init\_\_(self, num\_classes=2):  super(Meso4, self).\_\_init\_\_()  self.conv1 = nn.Conv2d(3, 8, kernel\_size=3, padding=1)  self.conv2 = nn.Conv2d(8, 8, kernel\_size=5, padding=2)  self.conv3 = nn.Conv2d(8, 16, kernel\_size=5, padding=2)  self.conv4 = nn.Conv2d(16, 16, kernel\_size=5, padding=2)  self.batchnorm1 = nn.BatchNorm2d(8)  self.batchnorm2 = nn.BatchNorm2d(8)  self.batchnorm3 = nn.BatchNorm2d(16)  self.batchnorm4 = nn.BatchNorm2d(16)  self.pool = nn.MaxPool2d(2, stride=2, padding=0)  self.global\_pool = nn.AdaptiveAvgPool2d((1, 1))  self.fc1 = nn.Linear(16, 16)  self.fc2 = nn.Linear(16, num\_classes)  self.dropout = nn.Dropout(0.5)  self.leakyrelu = nn.LeakyReLU(negative\_slope=0.1)  def forward(self, x):  x = self.pool(F.relu(self.batchnorm1(self.conv1(x))))  x = self.pool(F.relu(self.batchnorm2(self.conv2(x))))  x = self.pool(F.relu(self.batchnorm3(self.conv3(x))))  x = self.pool(F.relu(self.batchnorm4(self.conv4(x))))  x = self.global\_pool(x)  x = torch.flatten(x, 1)  x = self.dropout(self.leakyrelu(self.fc1(x)))  x = self.dropout(self.fc2(x))  return x |

1. Định nghĩa mô hình ResNet50:

|  |
| --- |
| from torchvision import models  import torch.nn as nn  resnet50 = models.resnet50(pretrained=True)  resnet50.fc = nn.Sequential(  nn.Linear(resnet50.fc.in\_features, 512),  nn.ReLU(),  nn.Dropout(0.5),  nn.Linear(512, 2), # 2 nhãn fake và real  ) |

1. Tải lên trọng số bias, định nghĩa transforms:

|  |
| --- |
| from flask import Blueprint, jsonify, request  from handler.upload import uploadHanlder  from util.getFace import getFaceIMG  import torch  import PIL.Image as Image  import torchvision.transforms as transforms  from model.meso4 import Meso4  from model.resnet50 import resnet50  PATHFILEMODELMESO = "model\\best\_meso4.pth"  PATHFILEMODELRESNET = "model\\best\_resnet50.pth"  predict\_bp = Blueprint("upload", \_\_name\_\_)  # model = Meso4()  # model.load\_state\_dict(torch.load(PATHFILEMODELMESO))  # model.eval()  # img\_transformer = transforms.Compose(  # [  # transforms.Resize((256, 256)),  # transforms.ToTensor(),  # transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]),  # ]  # )  # Resnet50  model = resnet50  model.load\_state\_dict(torch.load(PATHFILEMODELRESNET))  model.eval()  img\_transformer = transform = transforms.Compose(  [  transforms.Resize((224, 224)),  transforms.ToTensor(),  transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]),  ]  ) |

1. Định nghĩa route dự đoán:

|  |
| --- |
| @predict\_bp.route("/predict", methods=["POST"])  def predictHanler():  file = uploadHanler()  if file == -1 or file == -2:  return jsonify({"msg": "", "status": file})    faceFile = getFaceIMG(file)  org, detectFaceList = faceFile  detectList = []    for item in detectFaceList:  path = f"store\\{item}"  image = Image.open(path)  image = img\_transformer(image).float()  image = image.unsqueeze(0)    with torch.no\_grad():  output = model(image)  output = torch.softmax(output, 1)  result = output.reshape(-1).tolist()  detectList.append(result)  return jsonify(  {  "msg": "",  "status": 0,  "org": org,  "detectFace": detectFaceList,  "detectList": detectList,  }  ) |