ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: Công nghệ phần mền

ĐỀ TÀI:

XÂY DỰNG ỨNG DỤNG PHỤC CHẾ ẢNH MÀU SỬ DỤNG MÔ HÌNH HỌC SÂU GANs

Người hướng dẫn: **ThS. NGUYỄN CÔNG DANH**

Sinh viên thực hiện: **NGUYỄN KHẮC NHÂN TÂM**

Số thẻ sinh viên: 102210325

Lớp: 21TCLC_KHDL2

Đà Nẵng, 06/2025

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP

1. Thông tin chung:

1. Họ và tên sinh viên: NGUYỄN KHẮC NHÂN TÂM
2. Lớp: 21TCLC_KHDL2 Số thẻ SV: 102210325
3. Tên đề tài: Xây dựng ứng dụng phục chế ảnh màu sử dụng mô hình học sâu GANs
4. Người hướng dẫn: ThS.NGUYỄN CÔNG DANH Học hàm/ học vị: Thạc Sĩ

II. Nhận xét, đánh giá đồ án tốt nghiệp:

1. Về tính cấp thiết, tính mới, khả năng ứng dụng của đề tài: (điểm tối đa là 2đ)

……..

……..

2. Về kết quả giải quyết các nội dung nhiệm vụ yêu cầu của đồ án: (điểm tối đa là 4đ)

……..

……..

3. Về hình thức, cấu trúc, bố cục của đồ án tốt nghiệp: (điểm tối đa là 2đ)

……..

……..

4. Đề tài có giá trị khoa học/ có bài báo/ giải quyết vấn đề đặt ra của doanh nghiệp hoặc nhà trường: (điểm tối đa là 1đ)

……..

……..

5. Các tồn tại, thiếu sót cần bổ sung, chỉnh sửa:

……..

……..

III. Tinh thần, thái độ làm việc của sinh viên: (điểm tối đa 1đ)

……..

IV. Đánh giá:

1. Điểm đánh giá: ……../10 (lấy đến 1 số lẻ thập phân)
2. Đề nghị: ☐ Được bảo vệ đồ án ☐ Bổ sung để bảo vệ ☐ Không được bảo vệ

Đà Nẵng, ngày tháng năm 2022

Người hướng dẫn

NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Đà Nẵng, ngày tháng năm 2025

Người phản biện

NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Đà Nẵng, ngày tháng năm 2025

Người hướng dẫn

TÓM TẮT

Tên đề tài: Xây dựng ứng dụng phục chế ảnh màu sử dụng mô hình học sâu GANs

Sinh viên thực hiện: Nguyễn Khắc Nhân Tâm

Số thẻ SV: 102210325 Lớp : 21TCLC_KHDL2

Trong bối cảnh hiện nay, nhu cầu phục chế và tái hiện lại các bức ảnh cũ, ảnh đen trắng lịch sử, hay ảnh từ các thiết bị ghi hình đơn sắc ngày càng trở nên quan trọng trong nhiều lĩnh vực như: lưu trữ văn hóa, gia đình, y tế, báo chí và nghệ thuật. Việc tô màu tự động cho ảnh trắng đen không chỉ giúp nâng cao giá trị thị giác, mà còn hỗ trợ con người hiểu rõ hơn về bối cảnh và nội dung của hình ảnh một cách sinh động.

Trong đề tài này, được triển khai mô hình Pix2Pix GAN, một mô hình học sâu dựa trên mạng Generative Adversarial Network (GAN), để giải quyết bài toán Image Colorization – tô màu cho ảnh trắng đen một cách tự động và có giám sát.

Hệ thống được huấn luyện trên một tập dữ liệu ảnh phong cảnh, trong đó mỗi cặp dữ liệu gồm:

- Ảnh trắng đen đầu vào (grayscale).
- Ảnh màu thật tương ứng (ground truth).

Mô hình gồm hai thành phần chính:

- Generator với kiến trúc U-Net có khả năng học và phục hồi màu sắc từ cấu trúc ảnh trắng đen.
- Discriminator sử dụng PatchGAN giúp đánh giá mức độ chân thực của từng vùng nhỏ (patch) trong ảnh màu sinh ra.

Được xây dựng giao diện người dùng bằng Gradio, cho phép người dùng tải ảnh trắng đen và nhận lại ảnh màu ngay lập tức trên trình duyệt, phục vụ nhu cầu kiểm thử và demo mô hình dễ dàng.

Thông qua quá trình huấn luyện và đánh giá, mô hình đã cho thấy khả năng phục hồi màu sắc tự nhiên, chi tiết rõ nét, và hoạt động hiệu quả với đa dạng ảnh phong cảnh.

Bằng cách này, đề tài không chỉ chứng minh được hiệu quả của các mô hình GAN trong xử lý ảnh, mà còn mở ra hướng ứng dụng thực tiễn trong phục chế hình ảnh, tạo ảnh nghệ thuật, và hỗ trợ truyền thông – sáng tạo nội dung số.

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Khắc Nhân Tâm Số thẻ sinh viên: 102210325

Lớp: 21TCLC_KHDL2 Khoa: Công nghệ thông tin Ngành: CNTT

1. *Tên đề tài đồ án:*

   Xây dựng ứng dụng phục chế ảnh màu sử dụng mô hình học sâu GANs

2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

3. *Các số liệu và dữ liệu ban đầu:*

   (Không có)

4. *Nội dung các phần thuyết minh và tính toán:*

   Nội dung các phần thuyết minh bao gồm:

   - **Mở đầu:** Phần mở đầu, giới thiệu về như cầu thực tế và lý do thực hiện đề tài, đồng thời giới thiệu tổng quan về đề tài và mục tiêu đạt được, các tính năng và đối tượng
   - **Chương 1 – Cơ sở lý thuyết:** Trình bày các kiến thức, lý thuyết được áp dụng vào trong đề tài
   - **Chương 2 – Phân tích và thiết kế hệ thống:** Xác định yêu cầu hệ thống và chức năng chính , nguồn dữ liệu,luồng xử lý tổng quát ,kiến trúc hệ thống
   - **Chương 3 – Triển khai và đánh giá:** Mô tả quá trình thực hiện hóa và các kết quả đạt được
   - **Chương 4 – Kết luận và hướng phát triển:** Đánh giá kết quả đạt được, nêu những điểm hạn chế và đề xuất các hướng cải tiến trong tương lai

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

   (Không có)

6. *Họ tên người hướng dẫn:* ThS. Nguyễn Công Danh

7. *Ngày giao nhiệm vụ đồ án: / / 2025*

8. *Ngày hoàn thành đồ án: / / 2025*

Đà Nẵng, ngày tháng năm 2025

Trưởng Bộ môn …….. **Người hướng dẫn**

LỜI NÓI ĐẦU

Đề tài "Phục chế ảnh màu sử dụng mô hình Pix2Pix GAN" là kết quả của quá trình học tập, nghiên cứu và vận dụng kiến thức chuyên ngành trong suốt thời gian học tại Khoa Công nghệ Thông tin – Trường Đại học Bách khoa – Đại học Đà Nẵng. Đây cũng là đồ án tốt nghiệp, đánh dấu bước trưởng thành trong tư duy, kỹ năng lập trình, nghiên cứu và triển khai ứng dụng trí tuệ nhân tạo vào thực tiễn.

Trong thời đại công nghệ số hiện nay, nhu cầu phục chế ảnh đen trắng thành ảnh màu ngày càng được quan tâm, không chỉ trong lưu trữ tư liệu, bảo tồn văn hóa – lịch sử, mà còn trong nghệ thuật, truyền thông và thị giác máy tính. Trên tinh thần đó, em đã lựa chọn triển khai mô hình Pix2Pix GAN – một kiến trúc mạnh mẽ dựa trên Generative Adversarial Network, nhằm tạo ra một hệ thống có khả năng tự động tô màu cho ảnh trắng đen một cách tự nhiên và hiệu quả.

Em xin gửi lời cảm ơn chân thành đến TS. Nguyễn Công Danh – người đã tận tâm hướng dẫn, chỉ bảo và hỗ trợ em xuyên suốt quá trình thực hiện đồ án. Những góp ý quý báu và sự động viên của thầy là nguồn động lực lớn giúp em hoàn thành đề tài này.

Em cũng xin chân thành cảm ơn Ban Giám hiệu, các thầy cô Khoa Công nghệ Thông tin đã truyền đạt kiến thức, tạo điều kiện học tập và nghiên cứu tốt nhất để em có thể phát triển bản thân.

Mặc dù đã nỗ lực hoàn thiện đề tài với tinh thần trách nhiệm cao nhất, nhưng do thời gian và kinh nghiệm nghiên cứu còn hạn chế, đề tài chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến từ quý thầy cô và các bạn để hoàn thiện và phát triển đề tài tốt hơn trong tương lai.

Cuối cùng, em xin kính chúc quý thầy cô và các anh chị, các bạn luôn dồi dào sức khỏe, thành công trong công việc và cuộc sống. Em hy vọng Khoa Công nghệ Thông tin – Trường Đại học Bách khoa – Đại học Đà Nẵng sẽ ngày càng phát triển và tiếp tục là môi trường học tập lý tưởng cho các thế hệ sinh viên sau này.

Em xin trân trọng cảm ơn!

CAM ĐOAN

Em xin cam đoan:

1. Báo cáo đồ án tốt nghiệp Tên đề tài: Xây dựng ứng dụng phục chế ảnh màu sử dụng mô hình học sâu GANs là công trình nghiên cứu của chính cá nhân em dưới sự hướng dẫn trực tiếp của giảng viên **Ths. Nguyễn Công Danh**
2. Tất cả tài liệu tham khảo đều được trích dẫn đầy đủ và cụ thể
3. Nếu có những sao chép không hợp lệ, vi phạm, em xin chịu hoàn toàn trách nhiệm

Sinh viên thực hiện

Nguyễn Khắc Nhân Tâm

MỤC LỤC

DANH SÁCH CÁC HÌNH VẼ

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Từ viết tắt	Diễn giải
AI	Artificial Intelligence
GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
U-Net	U-shaped Network – Mạng có kiến trúc hình chữ U (dùng trong segmentation/generator)
RGB	Red Green Blue
GPU	Graphics Processing Unit
LR	Learning Rate
L1 Loss	Mean Absolute Error (MAE)
API	Application Programming Interface
I/O	Input/Output
TF	TensorFlow
PNG/JPG	Portable Network Graphics / Joint Photographic Experts Group

MỞ ĐẦU

1. Giới thiệu

Với sự phát triển nhanh chóng và vượt bậc của trí tuệ nhân tạo (AI) trong những năm gần đây, nhiều bài toán thị giác máy tính (Computer Vision) đã đạt được những bước tiến đáng kể, trong đó có bài toán tô màu ảnh đen trắng (Image Colorization). Trước đây, việc khôi phục màu sắc cho hình ảnh chỉ có thể thực hiện bằng phương pháp thủ công hoặc các kỹ thuật truyền thống, vốn tốn nhiều thời gian và đòi hỏi sự can thiệp từ con người. Tuy nhiên, những phương pháp này thường không mang lại kết quả thực tế cao do giới hạn về khả năng hiểu ngữ cảnh và chi tiết trong ảnh.

Ngày nay, với sự ra đời của các mô hình học sâu như mạng nơ-ron tích chập (CNN) và đặc biệt là các mô hình đối kháng sinh (GAN), khả năng tự động hóa và nâng cao chất lượng tô màu ảnh đã được cải thiện đáng kể. Một trong những kiến trúc nổi bật là Pix2Pix GAN, một mô hình học có giám sát sử dụng kiến trúc encoder-decoder kết hợp với discriminator để học ánh xạ giữa ảnh đen trắng và ảnh màu tương ứng. Mô hình này đã cho thấy hiệu quả rõ rệt trong nhiều nghiên cứu và ứng dụng thực tế.

Tuy nhiên, khi tìm hiểu và nghiên cứu sâu về bài toán này, nhận thấy tài liệu tiếng Việt về tô màu ảnh sử dụng mô hình học sâu, đặc biệt là Pix2Pix còn khá hạn chế và rời rạc. Do đó trong quá trình thực hiện đồ án, tác giả không chỉ đặt kỳ vọng xây dựng thành công mô hình phục chế ảnh màu với kết quả như kỳ vọng, mà còn mong muốn đóng góp một phần tài liệu tham khảo hữu ích cho những bạn sinh viên quan tâm đến lĩnh vực này trong tương lai.

2. Mục đích và ý nghĩa của đề tài

2.1. Mục đích

Mục tiêu của nghiên cứu này là phát triển một hệ thống tự động tô màu ảnh đen trắng (Image Colorization) dựa trên mô hình học sâu, cụ thể là mô hình Pix2Pix GAN, với các mục tiêu cụ thể như sau:

- Xây dựng một mô hình có khả năng tự động chuyển đổi ảnh đen trắng thành ảnh màu, đảm bảo độ chân thực, tự nhiên và giữ được các chi tiết quan trọng của ảnh gốc.
- Cải thiện chất lượng ảnh tô màu bằng cách ứng dụng kiến trúc GAN kết hợp giữa generator và discriminator, đồng thời khai thác sức mạnh từ mạng CNN trong việc trích xuất và học đặc trưng ảnh.

Tối ưu hóa hiệu suất và độ tổng quát của mô hình, nhằm giúp hệ thống có thể áp dụng trong thực tế ở các lĩnh vực như phục hồi ảnh cũ, số hóa tư liệu lịch sử, chỉnh sửa ảnh nghệ thuật hoặc hỗ trợ các ứng dụng sáng tạo hình ảnh trong truyền thông và quảng cáo.

2.2. Ý nghĩa

Đề tài **"Phục chế ảnh màu sử dụng mô hình Pix2Pix GAN"** mang ý nghĩa quan trọng trong cả lĩnh vực công nghệ và xã hội, đặc biệt trong bối cảnh nhu cầu số hóa, bảo tồn và nâng cao chất lượng hình ảnh ngày càng cao trong nhiều ngành như lưu trữ văn hóa, truyền thông, lịch sử và giải trí.

- **Về mặt công nghệ:**

Việc áp dụng mô hình Pix2Pix GAN – một trong những kiến trúc tiêu biểu của mạng đối kháng sinh (GAN) **–** cho thấy tiềm năng to lớn của trí tuệ nhân tạo trong lĩnh vực xử lý ảnh**.** Bằng cách huấn luyện mô hình với các cặp ảnh đen trắng và ảnh màu tương ứng, hệ thống có thể học cách khôi phục màu sắc một cách tự nhiên và thuyết phục, giúp tái tạo lại hình ảnh với độ chân thực cao.

Mô hình này không chỉ dừng lại ở bài toán tô màu mà còn mở ra hướng ứng dụng rộng rãi cho các bài toán như phục chế ảnh cũ, ảnh lịch sử, chuyển đổi phong cách ảnh, hoặc thậm chí dùng trong ngành công nghiệp điện ảnh và nghệ thuật số.

- **Về mặt xã hội:**

Đề tài góp phần tạo ra một công cụ hữu ích giúp phục hồi di sản hình ảnh**,** đặc biệt là các tài liệu, ảnh lịch sử quý hiếm vốn chỉ có phiên bản đen trắng. Việc tô màu các bức ảnh này không chỉ mang lại giá trị thị giác cao hơn mà còn giúp thế hệ sau tiếp cận tư liệu một cách sinh động, gần gũi và dễ hiểu hơn.

Ngoài ra, mô hình còn có thể hỗ trợ người dùng cá nhân – như phục hồi ảnh kỷ niệm gia đình, ảnh thẻ cũ – mà không cần phải sử dụng phần mềm chỉnh sửa chuyên sâu.

- **Tính thực tiễn và phát triển**:

Việc kết hợp AI với giao diện người dùng thông qua thư viện Gradio cho phép triển khai mô hình nhanh chóng thành một ứng dụng demo trên nền web, phục vụ việc kiểm thử, trình bày hoặc sử dụng thực tế một cách trực quan. Điều này thể hiện sự kết hợp giữa nghiên cứu học thuật và tính ứng dụng thực tế trong phát triển phần mềm AI hiện đại.

Tóm lại, đề tài không chỉ dừng lại ở việc nghiên cứu mô hình GAN mà còn thể hiện tính ứng dụng cao của trí tuệ nhân tạo trong việc nâng cao chất lượng ảnh số, đồng thời mở ra tiềm năng cho nhiều hướng phát triển mới trong lĩnh vực xử lý ảnh và thị giác máy tính trong tương lai

2.3. Đối tượng nghiên cứu

- Dữ liệu:
    - Bộ dữ liệu gồm các ảnh màu được chuyển đổi thành ảnh đen trắng để làm đầu vào cho mô hình huấn luyện.

  - Các ảnh có độ phân giải đồng đều (ví dụ: 150x150), đảm bảo chất lượng hình ảnh cho quá trình huấn luyện và đánh giá.
  - Dữ liệu được chia thành ba phần: tập huấn luyện, tập kiểm tra và tập đánh giá.
- Các kỹ thuật và mô hình học sâu:
  - Mạng GAN, đặc biệt là mô hình Pix2Pix GAN, kết hợp kiến trúc U-Net cho Generator và CNN phân biệt ảnh thật/giả cho Discriminator.
  - Các kỹ thuật hỗ trợ như Batch Normalization, Dropout, và hàm mất mát kết hợp (Adversarial loss và L1 loss).
- Công cụ và thư viện: TensorFlow, OpenCV, Keras, PIL(Pillow),Numpy, Matplotlib

2.4. Phương pháp nghiên cứu

- **Thu thập và chuẩn bị dữ liệu**:
  - Sử dụng bộ dữ liệu ảnh màu, sau đó chuyển đổi thành ảnh đen trắng để làm đầu vào cho mô hình.
  - Ảnh được chuẩn hóa về kích thước (ví dụ: 150x150) và định dạng phù hợp cho huấn luyện mạng nơ-ron.
  - Tách tập dữ liệu thành tập huấn luyện, kiểm tra và đánh giá để đảm bảo quá trình học được kiểm soát chặt chẽ.
- **Tiền xử lý dữ liệu**:
  - Biến đổi ảnh màu sang không gian màu L*a*b* hoặc giữ định dạng RGB tùy theo yêu cầu kiến trúc mô hình.
  - Ảnh đầu vào (đen trắng) sẽ chỉ chứa kênh L (độ sáng), trong khi ảnh đầu ra là kênh a* và b* (thông tin màu).
- **Xây dựng mô hình**:
  - Áp dụng mô hình Pix2Pix GAN, bao gồm hai thành phần chính:
    - Generator: Mô hình U-Net học ánh xạ từ ảnh đen trắng sang ảnh màu.
    - Discriminator: Mạng CNN đánh giá mức độ chân thực của ảnh màu được sinh ra.
  - Huấn luyện mô hình theo cơ chế đối kháng (adversarial learning), kết hợp với hàm mất mát L1 để đảm bảo ảnh đầu ra sát với ảnh thực tế.
- **Đánh giá mô hình**:
  - L1 Loss (Mean Absolute Error): Đo sai số trung bình tuyệt đối giữa ảnh màu gốc và ảnh do Generator tạo ra.
  - Adversarial Loss: Đánh giá khả năng đánh lừa Discriminator của Generator.

  - Discriminator Loss: Phản ánh khả năng phân biệt giữa ảnh thật và ảnh sinh.
  - Quan sát đánh giá trực quan để xác định độ chân thực và tính tự nhiên của ảnh màu.
- **Thử nghiệm và cải tiến**:
  - Thử nghiệm với các biến thể của mạng U-Net, thay đổi số lượng tầng, bộ lọc và hàm kích hoạt.
  - Tinh chỉnh các siêu tham số như learning rate, batch size, epoch, kết hợp các kỹ thuật như Dropout, Batch Normalization để tăng độ ổn định và tránh overfitting.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 Tổng quan về công nghệ sử dụng

1.1.1 Gradio

Hình 1.1 Logo của Gradio

Gradio là một thư viện Python mã nguồn mở dùng để xây dựng giao diện người dùng (UI) cho các mô hình học máy và ứng dụng AI một cách nhanh chóng, đơn giản và trực quan. Với Gradio, bạn có thể tạo ra một giao diện web tương tác chỉ với vài dòng code, giúp việc kiểm thử mô hình hoặc trình bày demo trở nên dễ dàng hơn bao giờ hết

Tính năng nổi bật của Gradio:

- **Tạo giao diện web nhanh chóng và trực quan**
  - Cho phép tạo giao diện dạng kéo – thả, nhập văn bản, tải ảnh, chọn checkbox… mà không cần viết HTML, CSS hay JavaScript.
  - Dễ dàng tích hợp trực tiếp vào notebook (Jupyter, Google Colab) hoặc chạy cục bộ trên trình duyệt.
- **Hỗ trợ đa dạng kiểu dữ liệu**
  - Có thể xử lý nhiều loại đầu vào và đầu ra như:
    - Văn bản (Text)
    - Ảnh (Image)
    - Âm thanh (Audio)
    - Video
    - File, bảng biểu, số liệu v.v.
- **Tương thích tốt với các mô hình học máy**
  - Dễ dàng kết hợp với TensorFlow, PyTorch, Keras, HuggingFace Transformers, scikit-learn, OpenCV,...
  - Chỉ cần định nghĩa hàm Python xử lý đầu vào và trả về đầu ra, Gradio sẽ tự động xây dựng giao diện phù hợp.
- **Dễ triển khai và chia sẻ**

- Giao diện có thể chia sẻ công khai qua liên kết Gradio hoặc tích hợp vào web/app thực tế thông qua API hoặc export sang HTML.
- Không cần cấu hình server phức tạp như Flask/Django.

1.1.2 Ngôn ngữ Python

Là một ngôn ngữ lập trình bậc cao, đa năng và hướng đối tượng, ra đời từ năm 1991. Mặc dù đã có tuổi đời, Python vẫn duy trì vị thế hàng đầu nhờ cú pháp rõ ràng, ngắn gọn, dễ đọc và dễ nhớ.

Ngôn ngữ này tương thích trên nhiều nền tảng như Windows, macOS và Linux, đồng thời có kho thư viện chuẩn và bên thứ ba phong phú, hỗ trợ mạnh mẽ cho các lĩnh vực phát triển web, khoa học dữ liệu, trí tuệ nhân tạo, tự động hóa và scripting.

Hình 1.2 Logo của Python

Đặc điểm nổi bật của Python:

- **Ngữ pháp đơn giản, dễ đọc:** Giảm thiểu boilerplate, giúp tăng tốc độ phát triển.
- **Hỗ trợ đa mô hình lập trình:** Vừa thủ tục, vừa hướng đối tượng
- **Xử lý lỗi bằng ngoại lệ:** Cơ chế try/except giúp quản lý tình huống bất thường rõ ràng
- **Khả năng nhúng:** Có thể đóng gói làm ngôn ngữ kịch bản (Scripting) trong các ứng dụng khác
- **Chạy đa nền tảng:** Không cần thay đổi code khi chuyển giữa Windows, macOS và Linux

1.1.3 Google Colab

Hình 1.3 Logo của Google Colab

Google Colab (Colaboratory) là một dịch vụ miễn phí do Google cung cấp, cho phép người dùng viết và thực thi mã Python trực tiếp trên trình duyệt, với khả năng sử dụng GPU hoặc TPU để tăng tốc tính toán. Đây là một công cụ đặc biệt hữu ích trong các dự án học máy và học sâu nhờ sự tiện lợi, dễ truy cập và không yêu cầu cấu hình máy tính cá nhân cao.

Một số ưu điểm chính của Google Colab:

- **Miễn phí sử dụng GPU/TPU**: Cho phép truy cập phần cứng tăng tốc xử lý như GPU (NVIDIA Tesla T4/K80) và TPU giúp huấn luyện mô hình nhanh hơn đáng kể.
- **Dễ dàng chia sẻ và cộng tác**: Colab hỗ trợ lưu trữ và chia sẻ tài liệu qua Google Drive, giúp làm việc nhóm hiệu quả.
- **Không cần cài đặt môi trường**: Các thư viện phổ biến như TensorFlow, PyTorch, OpenCV, scikit-learn... đã được cài đặt sẵn, tiết kiệm thời gian thiết lập ban đầu.
- **Tích hợp với Google Drive**: Dễ dàng truy cập và lưu trữ dữ liệu, mô hình, hoặc kết quả huấn luyện trên Drive.
- **Giao diện giống Jupyter Notebook**: Dễ sử dụng và trực quan, phù hợp cho cả người mới và các nhà nghiên cứu chuyên sâu.

Google Colab đã được sử dụng làm môi trường huấn luyện và thử nghiệm trong đồ án, giúp nhóm triển khai và huấn luyện mô hình phục chế ảnh màu với Pix2Pix GAN mà không cần cấu hình phần cứng chuyên biệt.

1.1.4 Hugging face

Hình 1.4 Logo của huggingface

Hugging Face là một nền tảng và thư viện mã nguồn mở nổi tiếng trong lĩnh vực trí tuệ nhân tạo, đặc biệt là xử lý ngôn ngữ tự nhiên (NLP) và mô hình học sâu hiện đại. Với thư viện chủ đạo là Transformers, Hugging Face cung cấp hàng nghìn mô hình pretrained mạnh mẽ có thể dễ dàng áp dụng vào các bài toán như dịch máy, phân tích cảm xúc, sinh văn bản, tổng hợp ngôn ngữ, thị giác máy tính và nhiều lĩnh vực khác.

Tính năng nổi bật của Hugging Face:

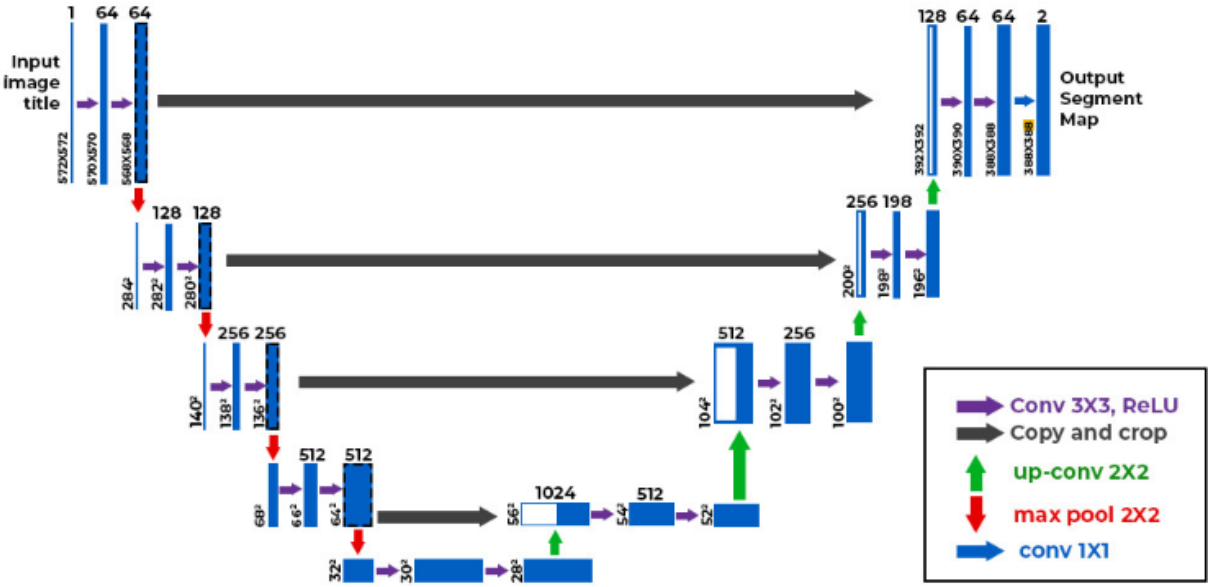
- **Thư viện mô hình Transformers mạnh mẽ và sẵn sàng sử dụng**
  - Bao gồm hàng ngàn mô hình được huấn luyện trước (pretrained) từ các tổ chức lớn như Google, Facebook, OpenAI, DeepMind…
  - Hỗ trợ các kiến trúc nổi tiếng như BERT, GPT-2/3, T5, RoBERTa, DistilBERT, ViT, CLIP, Stable Diffusion, v.v.
- **Hỗ trợ đa lĩnh vực AI**
  - Không chỉ NLP, Hugging Face còn hỗ trợ các lĩnh vực như:
    - Thị giác máy tính (Computer Vision): phân loại ảnh, phát hiện vật thể, colorization,...
    - Xử lý âm thanh: chuyển giọng nói thành văn bản, tổng hợp giọng nói.
    - Học tăng cường (Reinforcement Learning): thông qua thư viện transformers + accelerate.
- **Tương thích dễ dàng với TensorFlow và PyTorch**
  - Tất cả các mô hình đều hỗ trợ cả hai backend chính: TensorFlow và PyTorch.
  - Cấu trúc mô hình thống nhất, dễ tinh chỉnh (fine-tune) và tích hợp vào pipeline thực tế.
- **Model Hub – Thư viện mô hình AI lớn nhất thế giới**
  - Hugging Face cung cấp Hugging Face Hub – một kho lưu trữ trực tuyến các mô hình, datasets và không gian demo.
  - Người dùng có thể tìm kiếm – thử – tải – triển khai các mô hình một cách đơn giản thông qua 1 dòng code.
- **Dễ tích hợp vào ứng dụng thực tế**
  - Hugging Face hỗ trợ API RESTful để gọi mô hình trực tiếp từ server.

- Kết hợp cực tốt với Gradio để tạo giao diện tương tác nhanh chóng

1.2 Mạng nơron tích chập

Mạng nơ-ron tích chập (Convolutional Neural Network – CNN) đóng vai trò cốt lõi trong cả hai thành phần chính của mô hình Pix2Pix GAN: Generator và Discriminator. Thay vì sử dụng CNN cho các tác vụ phân loại như truyền thống, Pix2Pix khai thác khả năng học đặc trưng không gian của CNN để thực hiện chuyển đổi hình ảnh từ ảnh đen trắng sang ảnh màu

- **Generator (U-Net CNN)**

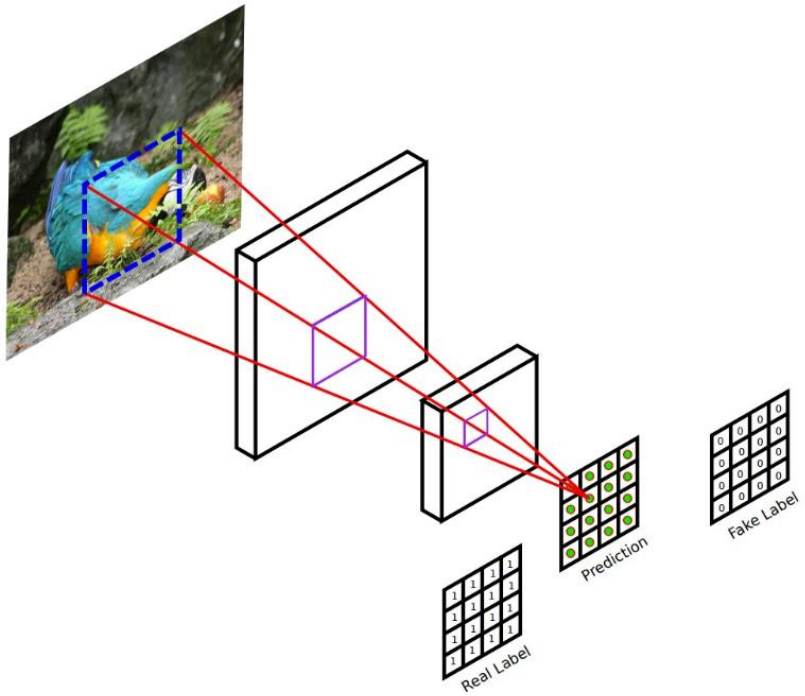


Hình 1.5 Ảnh mô tả kiến trúc Unet

Generator trong Pix2Pix sử dụng kiến trúc U-Net – một dạng CNN đối xứng gồm encoder và decoder:

- **Encoder** là chuỗi các lớp tích chập (convolutional layers) kết hợp với chuẩn hóa batch (batch normalization) và hàm kích hoạt Leaky ReLU, giúp trích xuất các đặc trưng từ ảnh đầu vào đen trắng.
- **Decoder** thực hiện quá trình phục hồi ảnh bằng cách sử dụng các lớp tích chập chuyển vị (transposed convolutional layers), kết hợp với skip connections từ encoder, giúp bảo toàn chi tiết ảnh gốc.
- Việc dùng U-Net giúp mô hình học được cả thông tin toàn cục và chi tiết cục bộ, từ đó tái tạo màu sắc chính xác cho từng vùng ảnh.

- **Discriminator (PatchGAN CNN)**



Hình 1.6 Ảnh mô tả kiến trúc PatchGAN

Discriminator có kiến trúc CNN gọi là PatchGAN, thay vì phân biệt toàn bộ ảnh thật/giả, nó phân biệt các vùng nhỏ (patches) trên ảnh:

- Gồm nhiều lớp tích chập liên tiếp để đánh giá từng vùng ảnh có chân thực hay không.
- PatchGAN giúp mô hình tập trung vào các đặc trưng cục bộ như biên cạnh, họa tiết và kết cấu màu sắc, điều này rất quan trọng trong phục chế ảnh.

Các thành phần chính của CNN trong Pix2Pix

- **Lớp tích chập (Convolutional Layer)**: Trích xuất đặc trưng từ ảnh, được sử dụng trong cả generator và discriminator.
- **Lớp chuẩn hóa (Batch Normalization)**: Ổn định và tăng tốc quá trình huấn luyện.
- **Hàm kích hoạt**: Leaky ReLU dùng trong discriminator, ReLU trong generator giúp mô hình học phi tuyến tính tốt hơn.
- **Lớp tích chập chuyển vị (Transposed Convolution)**: Giúp mở rộng kích thước ảnh đầu ra trong decoder của Generator.

Ưu điểm của CNN:

- **Tự động trích xuất đặc trưng**: CNN tự động học các đặc trưng hình ảnh quan trọng (cạnh, đường viền, họa tiết) mà không cần thiết kế thủ công, giúp phục chế ảnh hiệu quả hơn.
- **Xử lý tốt dữ liệu hình ảnh**: Với khả năng tận dụng mối quan hệ không gian giữa các pixel, CNN đặc biệt phù hợp với tác vụ chuyển đổi ảnh đen trắng sang ảnh màu.
- **Tiết kiệm tham số**: Nhờ sử dụng bộ lọc chia sẻ tham số, CNN có ít trọng số hơn các mạng fully connected, giúp giảm nguy cơ quá khớp và cải thiện hiệu năng.
- **Giảm chiều thông minh**: Pooling giúp giữ lại đặc trưng quan trọng, giảm kích thước dữ liệu, tăng tốc độ huấn luyện và tăng tính khái quát.
- **Làm việc tốt với dữ liệu nhiều kênh**: CNN dễ dàng xử lý ảnh màu nhiều kênh như RGB, rất phù hợp với nhiệm vụ tái tạo màu sắc.
- **Khả năng học sâu**: CNN có thể xây dựng nhiều lớp, cho phép học đặc trưng từ đơn giản đến phức tạp – rất cần thiết trong các tác vụ sinh ảnh.
- **Tính bất biến theo vị trí (Translation Invariance)**: Giúp phát hiện đặc trưng tại nhiều vị trí khác nhau trong ảnh, đảm bảo kết quả phục chế ổn định ngay cả khi đối tượng trong ảnh bị dịch chuyển.
- **Ứng dụng rộng rãi**: Không chỉ trong phục chế ảnh, CNN còn được dùng hiệu quả trong phân loại, phát hiện đối tượng, xử lý video, và nhiều lĩnh vực khác của thị giác máy tính.

1.3 Mạng Pix2Pix

Pix2Pix là một kiến trúc học sâu được thiết kế để chuyển đổi một hình ảnh đầu vào thành một hình ảnh đầu ra tương ứng, với nhiệm vụ cụ thể có thể khác nhau như: chuyển ảnh phác thảo thành ảnh thật, ảnh ban đêm thành ban ngày, hoặc trong trường hợp này là tô màu ảnh đen trắng.

Pix2Pix là một mô hình thuộc họ Generative Adversarial Networks (GANs) – mạng sinh đối kháng, trong đó hai mạng nơ-ron Generator (Bộ sinh) và Discriminator (Bộ phân biệt) cùng học và cạnh tranh với nhau để tạo ra hình ảnh đầu ra có chất lượng cao, gần giống với dữ liệu thật.

Cấu trúc mô hình Pix2Pix cũng bao gồm hai thành phần chính:

- **Generator-Unet (Bộ sinh ảnh):**

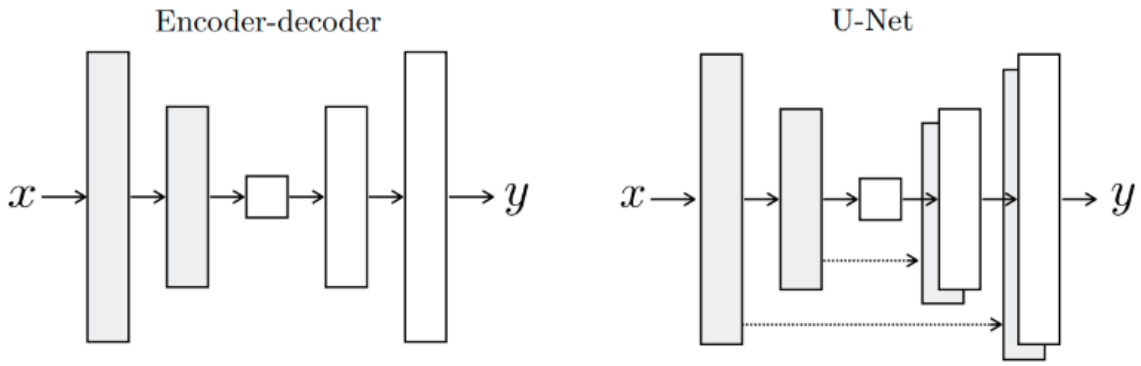
Generator trong Pix2Pix chịu trách nhiệm chuyển ảnh đầu vào (ảnh đen trắng) thành ảnh màu tương ứng. Trong quá trình huấn luyện, Generator học cách trích xuất

các đặc trưng từ ảnh đen trắng và sinh ra các chi tiết màu sắc phù hợp với ngữ cảnh hình ảnh.

Trong mô hình Pix2Pix GAN áp dụng cho bài toán tô màu ảnh đen trắng, **Generator** đóng vai trò quan trọng trong việc học ánh xạ từ ảnh đen trắng sang ảnh màu tương ứng. Để đảm bảo giữ được cả thông tin toàn cục và chi tiết cục bộ của ảnh, mô hình Generator sử dụng kiến trúc **U-Net**, một dạng mạng nơ-ron tích chập đặc biệt mạnh mẽ trong các bài toán sinh ảnh.

Cấu trúc Generator thường sử dụng mô hình **U-Net**, gồm hai phần:

- **Encoder**: mã hóa ảnh đầu vào thành các đặc trưng trừu tượng.
- **Decoder**: giải mã các đặc trưng này thành ảnh màu đầu ra.
- Kết hợp với skip connections (nối tắt) để giữ lại thông tin chi tiết ở từng cấp độ ảnh.



Hình 1.7 Kiến trúc Unet sử dụng trong Generator

Generator: Pix2Pix thì được dựa trên kiến trúc Unet, một kiến trúc mạng nổi tiếng trong xử lý ảnh y tế (biomedical image segmentation), để khởi tạo generator. Các layers ở nhánh bên trái (phần encoder) sẽ được concatenate trực tiếp vào các layers ở nhánh bên phải (phần decoder) có cùng kích thước. Kết nối này được gọi là kết nối tắt (skip connection) nhằm bổ sung thêm thông tin và gia tăng độ chính xác.

- **Discriminator PatchGAN (Bộ phân biệt ảnh):**

Discriminator có nhiệm vụ phân biệt ảnh nào là ảnh thật (từ dữ liệu huấn luyện) và ảnh nào là ảnh giả (do Generator tạo ra). Trong Pix2Pix, Discriminator thường được thiết kế dưới dạng PatchGAN, không đánh giá toàn bộ ảnh một cách tổng thể mà đánh giá các vùng nhỏ (patches) trong ảnh đầu ra, từ đó đảm bảo rằng từng vùng ảnh nhỏ cũng mang đặc điểm thật.

Như đã đề cập ở phần trước, mô hình Pix2Pix GAN hoạt động dựa trên cơ chế đối kháng giữa hai mạng nơ-ron: Generator và Discriminator. Trong đó, Discriminator đóng vai trò như một "người giám định" có nhiệm vụ phân biệt ảnh màu thật và ảnh màu được sinh ra từ ảnh đen trắng.

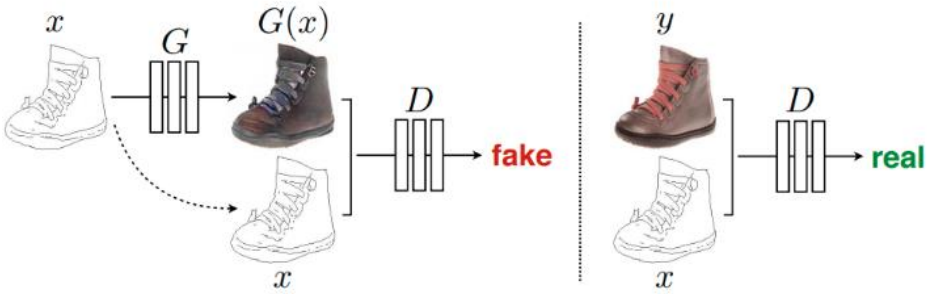
Cơ chế hoạt động của PatchGAN:

- Ảnh đầu vào của Discriminator là một cặp ảnh: ảnh đen trắng (input) + ảnh màu (thật hoặc sinh ra từ Generator).
- Discriminator sẽ chia ảnh đầu ra thành các vùng nhỏ (ví dụ: 70×70 pixel) và đánh giá từng vùng đó là "thật" hay "giả".
- Việc này giúp Discriminator tập trung vào chi tiết cục bộ như cạnh vật thể, vùng màu, họa tiết – những yếu tố rất quan trọng trong phục chế ảnh.

Ưu điểm của Discriminator PatchGAN:

- Giúp giữ cho ảnh màu đầu ra sắc nét, không bị mờ như khi dùng chỉ L1 loss.
- Tăng cường khả năng phát hiện các vùng màu "giả", từ đó ép Generator học tốt hơn.

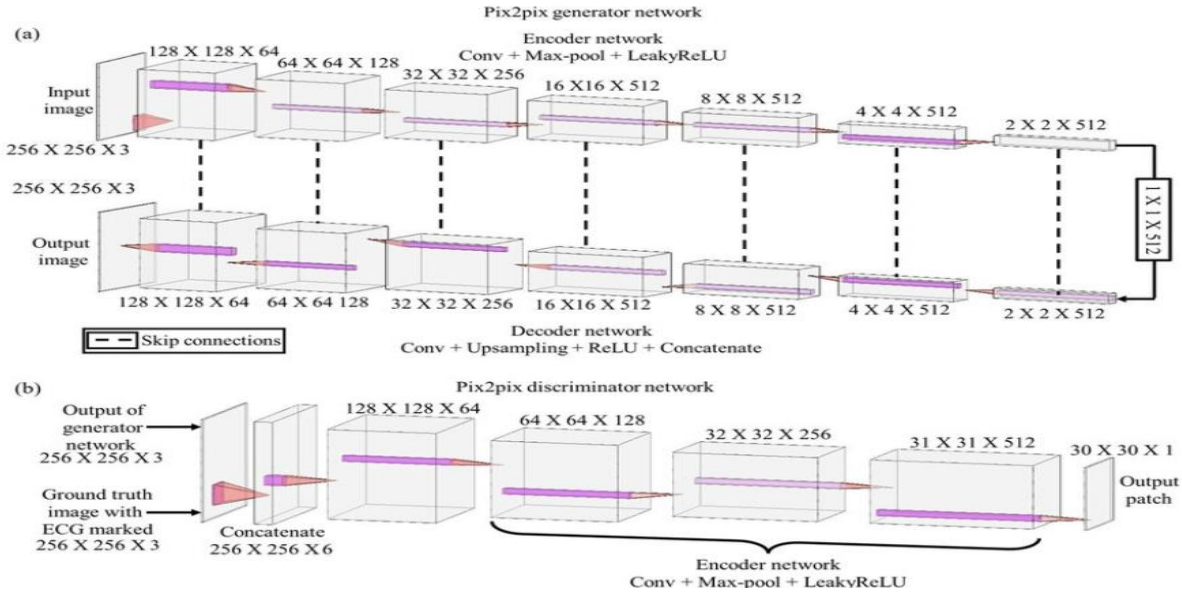
Tiết kiệm tài nguyên so với việc đánh giá toàn bộ ảnh lớn.



Hình 1.8 Kiến trúc Discriminator

Discriminator: Đầu vào của discriminator là một cặp ảnh x,y. Trong đó x là ảnh source đầu vào và y là ảnh target được sinh ra từ generator hoặc được lấy từ tập train. Hai ảnh x,y có cùng kích thước và sẽ được concatenate với nhau trước khi truyền vào mô hình discriminator. Cặp ảnh sẽ có nhãn là real nếu ảnh y được lấy từ tập train và trái lại khi y được sinh ra từ generator thì cặp ảnh x,y nó sẽ mang nhãn fake.

Dưới đây là ảnh minh họa kiến trúc tổng quát của mạng Pix2Pix:



Hình 1.9 Kiến trúc tổng quát mạng Pix2Pix

Ưu điểm mạng Pix2Pix	**Nhược điểm mạng Pix2Pix**
Tái tạo màu sắc chân thực: Kết hợp giữa mạng GAN và hàm mất mát L1 giúp ảnh màu sinh ra vừa có màu sắc mượt mà, vừa gần giống ảnh thật.	Phụ thuộc vào dữ liệu huấn luyện: Nếu dữ liệu không đủ đa dạng về đối tượng và màu sắc, mô hình dễ sinh màu sai lệch hoặc không thực tế.
Học được ngữ cảnh hình ảnh: Nhờ kiến trúc U-Net với skip connection, mô hình có thể giữ lại các chi tiết hình ảnh gốc, giúp tô màu chính xác hơn.	Huấn luyện khó ổn định: Vì là mô hình GAN nên dễ gặp vấn đề như mode collapse hoặc mất cân bằng giữa Generator và Discriminator.
Thực thi nhanh khi đã huấn luyện: Sau khi train xong, mô hình có thể tô màu ảnh đen trắng chỉ trong vài mili-giây, phù hợp với ứng dụng thời gian thực.	Yêu cầu phần cứng mạnh: Huấn luyện mô hình trên tập ảnh lớn hoặc ảnh độ phân giải cao cần GPU mạnh và bộ nhớ lớn.
Áp dụng linh hoạt cho nhiều phong cách ảnh: Có thể áp dụng cho ảnh chân dung, phong cảnh, vật thể,... chỉ cần đủ dữ liệu huấn luyện tương ứng.	Không có "cảm nhận màu sắc thực tế": Mô hình học theo thống kê, nên đôi khi màu tô ra khác với kỳ vọng của con người nếu ảnh nhập mơ hồ.

Bảng 1.1 Ưu điểm và nhược điểm mạng Pix2Pix

1.4 Kết chương

Trong chương này, đã trình bày tổng quan các công nghệ và công cụ chính được sử dụng trong quá trình xây dựng và phát triển mô hình phục chế ảnh màu sử dụng Pix2Pix GAN. Cụ thể, thư viện Gradio giúp tạo giao diện tương tác trực quan, Google Colab hỗ trợ huấn luyện mô hình với phần cứng mạnh mẽ, Python là nền tảng ngôn ngữ lập trình linh hoạt, và Hugging Face cung cấp kho tài nguyên mô hình AI phong phú phục vụ mở rộng và thử nghiệm. Bên cạnh đó, các kiến thức về mạng nơ-ron tích chập (CNN) **và** Pix2Pix GAN đã được trình bày rõ ràng, từ kiến trúc U-Net trong Generator đến cơ chế hoạt động của PatchGAN trong Discriminator.

Tổng quan này không chỉ cung cấp nền tảng lý thuyết vững chắc cho đồ án, mà còn làm rõ lý do lựa chọn công nghệ, cũng như cách các công cụ trên hỗ trợ lẫn nhau trong quá trình triển khai hệ thống. Những nội dung này là cơ sở quan trọng để bước sang các chương tiếp theo – nơi mô hình được hiện thực hóa và áp dụng vào bài toán tô màu ảnh đen trắng một cách hiệu quả.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Yêu cầu chức năng

Hệ thống hỗ trợ phục chế ảnh màu từ ảnh đen trắng sử dụng mô hình học sâu Pix2Pix GAN, bao gồm các chức năng chính sau:

2.1.1 Tải ảnh đầu vào

- Cho phép người dùng chọn hoặc tải lên ảnh đen trắng từ thiết bị để làm đầu vào cho hệ thống.

2.1.2 Tiền xử lý ảnh

- Tự động resize ảnh về kích thước chuẩn (256×256 pixel).
- Chuẩn hóa giá trị pixel về khoảng [0, 1] để phù hợp với yêu cầu đầu vào của mô hình.

2.1.3 Phục chế ảnh màu bằng mô hình Pix2Pix

- Ảnh đen trắng sau khi tiền xử lý sẽ được đưa vào Generator để tạo ra ảnh màu.
- Mô hình đã được huấn luyện với cặp ảnh grayscale-color để học chuyển đổi.

2.1.4 Hiển thị ảnh kết quả

- Giao diện trực quan hiển thị ảnh đen trắng gốc và ảnh đã phục chế màu.
- Cho phép người dùng quan sát kết quả trước và sau khi phục chế.

2.1.5 Tải ảnh kết quả về máy

- Cung cấp chức năng tải ảnh màu sau khi phục chế về thiết bị cá nhân.

2.1.6 Ví dụ minh họa

- Tích hợp thư mục examples/ chứa một số ảnh mẫu để người dùng thử nghiệm nhanh.

2.1.7 Giao diện trực quan bằng Gradio

- Giao diện web được xây dựng bằng thư viện Gradio.

2.2 Chức năng chính của hệ thống

stt	Tên chức năng	Mô tả
1	Tiền xử lý dữ liệu	Đọc ảnh từ thư mục gray/ và color/, resize về 256x256, chuẩn hóa

2	Tạo cặp ảnh huấn luyện	Ghép ảnh đen trắng với ảnh màu tương ứng để huấn luyện mô hình
3	Huấn luyện mô hình Pix2Pix	Áp dụng mạng đối kháng Pix2Pix với kiến trúc Generator (U-Net) và Discriminator (PatchGAN)
4	Theo dõi và lưu trọng số	Theo dõi loss, dừng sớm, giảm learning rate và lưu trọng số tốt nhất
5	Dự đoán ảnh màu hóa	Dự đoán ảnh màu từ ảnh grayscale đầu vào
6	Triển khai giao diện người dùng	Tạo giao diện web với Gradio cho phép người dùng upload ảnh và xem kết quả

Bảng 2.1 Chức năng chính của hệ thống

2.3 Nguồn dữ liệu

Trong đồ án này, được sử dụng tập dữ liệu Landscape Image Colorization, một bộ dữ liệu gồm các hình ảnh phong cảnh tự nhiên (landscape) đã được xử lý thành ảnh màu và ảnh đen trắng, nhằm phục vụ cho bài toán tô màu ảnh (Image Colorization) – một ứng dụng thuộc lĩnh vực thị giác máy tính (Computer Vision).

Thông tin chi tiết về tập dữ liệu:

- **Nguồn dữ liệu**: Được trích xuất từ một kho ảnh phong cảnh tổng hợp, trong đó ảnh màu là ảnh gốc, còn ảnh đen trắng được chuyển đổi bằng phương pháp chuyển không gian màu từ RGB sang thang độ xám (grayscale).
- **Số lượng ảnh**:
    - 7.129 ảnh màu (color)
    - 7.129 ảnh đen trắng tương ứng (gray)
- **Kích thước ảnh**: Mỗi ảnh được resize về kích thước cố định **150x150 pixel**, đảm bảo thống nhất đầu vào cho mô hình học sâu.
- **Định dạng**: Tất cả ảnh đều có định dạng .jpg hoặc .png, được tổ chức trong các thư mục riêng biệt cho tập huấn luyện (train) và tập xác thực (validation).
- **Tiền xử lý dữ liệu**:
    - Chuẩn hóa ảnh về dải giá trị [0,1]
    - Chuyển đổi ảnh màu sang không gian màu Lab (hoặc RGB tùy theo mô hình)

- Ghép nối ảnh đen trắng làm đầu vào và ảnh màu làm nhãn để huấn luyện mô hình Pix2Pix

Ưu điểm của dataset:

- **Dữ liệu trực quan và đồng nhất**: Các ảnh trong bộ dữ liệu là ảnh phong cảnh với bố cục rõ ràng, màu sắc hài hòa, giúp mô hình học được mối quan hệ giữa kết cấu hình ảnh và màu sắc tự nhiên một cách hiệu quả.
- **Chất lượng hình ảnh cao**: Các ảnh có độ phân giải tốt, giúp quá trình trích xuất đặc trưng chính xác hơn, hỗ trợ mô hình tái tạo màu sắc mượt mà và rõ nét.
- **Tính phù hợp với bài toán tô màu ảnh**: Vì ảnh gốc là ảnh màu và được chuyển thành đen trắng để làm dữ liệu huấn luyện, nên bộ dữ liệu này rất phù hợp với các mô hình học có giám sát trong bài toán Image Colorization.

Hạn chế của dataset:

- **Độ đa dạng nội dung còn hạn chế**: Do chủ yếu tập trung vào ảnh phong cảnh, bộ dữ liệu có thể thiếu sự đa dạng về đối tượng như con người, đồ vật hay cảnh vật trong nhà, làm giảm khả năng tổng quát hóa của mô hình.
- **Phụ thuộc vào ánh sáng tự nhiên**: Một số ảnh có điều kiện ánh sáng không đồng đều, dễ gây nhiễu khi huấn luyện mô hình, đặc biệt với các vùng tối hoặc chói sáng.

Dữ liệu được tổ chức rất đơn giản với 2 thư mục chính:

dataset/

color/ : chứa ảnh màu (ground truth)

gray/ : chứa ảnh đen trắng tương ứng (input)

- Mỗi ảnh trong gray/ có tên trùng với ảnh tương ứng trong color/, đảm bảo có thể ghép chính xác từng cặp ảnh đen trắng - ảnh màu để huấn luyện mô hình.

2.4 Ứng dụng học máy trong chương trình

Xử lý dữ liệu đầu vào:

- **Tiền xử lý ảnh màu và ảnh đen trắng**:
  Dữ liệu bao gồm các cặp ảnh: ảnh gốc (màu) và ảnh đã được chuyển sang đen trắng (grayscale). Mỗi cặp được resize về kích thước chuẩn 256×256 pixels, chuẩn hóa giá trị pixel về khoảng [0, 1], và chuyển thành định dạng tensor phù hợp với đầu vào mạng nơ-ron.

- **Ghép cặp dữ liệu**:
  Ảnh màu và ảnh đen trắng được ghép thành các cặp (gray_image, color_image) dựa trên tên file trùng nhau. Dữ liệu sau đó được chia thành 3 phần: train (80%), validation (10%) và test (10%).
- **Pipeline huấn luyện**:
  TensorFlow tf.data.Dataset được sử dụng để xây dựng pipeline đọc dữ liệu, tiền xử lý, batching và tiền xử lý song song, giúp tối ưu tốc độ nạp dữ liệu trong quá trình huấn luyện.

Xây dựng mô hình học máy:

- **Generator (U-Net):**
  Generator được xây dựng theo kiến trúc U-Net với 8 tầng downsampling và 7 tầng upsampling. Các skip connection được sử dụng để giữ lại chi tiết không gian trong ảnh gốc. Mô hình nhận đầu vào là ảnh đen trắng (1 kênh) và sinh ra ảnh RGB (3 kênh).
- **Discriminator (PatchGAN):**
  Discriminator sử dụng kiến trúc PatchGAN để đánh giá tính chân thực của từng vùng ảnh nhỏ (patch). Mô hình nhận đầu vào là một cặp ảnh (ảnh grayscale + ảnh màu thực/tạo) và phân biệt ảnh sinh ra có "thật" hay không.

Huấn luyện mô hình:

- Chiến lược huấn luyện:
  Cả Generator và Discriminator được huấn luyện đồng thời theo cơ chế GAN. Với mỗi batch:
  - Generator cố gắng tạo ảnh màu càng giống ảnh thật càng tốt (tối ưu GAN Loss + λ * L1 Loss).
  - Discriminator cố gắng phân biệt giữa ảnh thật và ảnh do Generator sinh ra.
- Tối ưu hóa:
  - Sử dụng optimizer Adam với learning rate ban đầu là 2e-4 và beta_1 = 0.4.
  - Áp dụng early stopping nếu độ lỗi trên tập validation không cải thiện sau patience số epoch.
  - Áp dụng giảm learning rate tự động nếu mô hình dừng cải thiện, giúp huấn luyện ổn định hơn.
- Theo dõi và checkpoint:
  - Trọng số mô hình tốt nhất được lưu tại thư mục pix2pix_checkpoints_minimal.
  - Mỗi epoch được theo dõi chỉ số gen_adv_loss, gen_l1_loss, và disc_loss.

- Chỉ số val_gen_l1_loss được dùng làm tiêu chí chính để lưu mô hình tốt nhất.

Triển khai mô hình:

- Sau khi huấn luyện, mô hình Generator tốt nhất được nạp lại từ file best_generator.weights.h5
- Một giao diện người dùng được xây dựng bằng thư viện Gradio, cho phép người dùng tải ảnh đen trắng và nhận ảnh màu tương ứng do mô hình sinh ra.
- Chức năng inference bao gồm:
  - Tiền xử lý ảnh người dùng tải lên.
  - Dự đoán ảnh màu bằng Generator.
  - Hiển thị kết quả trên giao diện.

2.5 Luồng xử lý tổng quát

- Dữ liệu đầu vào:
  - Ảnh đen trắng từ thư mục gray/
  - Ảnh màu gốc từ thư mục color/
- Tiền xử lý:
  - Resize ảnh về (256, 256)
  - Chuẩn hóa giá trị pixel về [0, 1]
  - Ghép cặp ảnh gray và color theo tên ảnh
- Huấn luyện mô hình:
  - Generator học cách biến đổi ảnh đen trắng thành ảnh màu
  - Discriminator học cách phân biệt ảnh màu thật và ảnh màu do generator tạo
  - Sử dụng loss function kết hợp giữa BinaryCrossEntropy và L1 Loss
- Lưu mô hình tốt nhất:
  - Dựa trên chỉ số val_gen_l1_loss trong quá trình huấn luyện
- Giao diện người dùng:
  - Dùng thư viện Gradio để tạo giao diện kéo-thả ảnh
  - Khi người dùng tải ảnh grayscale lên, mô hình sẽ tô màu và trả kết quả ngay trên web

2.6 Kiến trúc hệ thống

Thành phần	Vai trò chính
Tiền xử lý dữ liệu	Đọc, resize, chuẩn hóa và ghép cặp ảnh từ thư mục gray/ và color/
Huấn luyện mô hình	Xây dựng kiến trúc Pix2Pix GAN, tính toán loss, cập nhật trọng số, lưu model
Triển khai web	Tích hợp mô hình đã huấn luyện vào giao diện Gradio phục vụ inferencing

Bảng 2.2 Kiến trúc hệ thống

2.7 Kết chương

Chương này đã phân tích và thiết kế toàn bộ hệ thống phục chế ảnh màu từ ảnh đen trắng sử dụng mô hình Pix2Pix GAN. Các yêu cầu chức năng và phi chức năng được xác định rõ ràng, từ việc xử lý dữ liệu, huấn luyện mô hình đến triển khai giao diện người dùng.

Luồng xử lý tổng quát đã mô tả cách hệ thống tiếp nhận ảnh đầu vào, xử lý, sinh ảnh màu và hiển thị kết quả cho người dùng thông qua giao diện Gradio. Đồng thời, các thành phần chính như tiền xử lý, kiến trúc mô hình và giao diện cũng được thiết kế mạch lạc và dễ bảo trì.

Những phân tích và thiết kế trong chương này là cơ sở để thực hiện **việc** triển khai, huấn luyện mô hình và kiểm thử hệ thống trong các chương tiếp theo của đồ án.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1 Cấu trúc hệ thống

3.1.1. Chia tập dữ liệu

Để đảm bảo mô hình được huấn luyện hiệu quả và có thể đánh giá được khả năng tổng quát hóa, chúng tôi đã tiến hành chia tập dữ liệu ảnh thành ba phần: tập huấn luyện(train), tập xác thực (validation) và tập kiểm tra (test) theo tỷ lệ phổ biến trong học máy

```
train_ds = dataset.take(train_count).batch(batch_size).prefetch(tf.data.AUTOTUNE)
remaining = dataset.skip(train_count)
val_size = (total_images - train_count) // 2
val_ds = remaining.take(val_size).batch(batch_size).prefetch(tf.data.AUTOTUNE)
test_ds = remaining.skip(val_size).batch(batch_size).prefetch(tf.data.AUTOTUNE)

return train_ds, val_ds, test_ds
```

Cụ thể:

- 80% dữ liệu được sử dụng để huấn luyện mô hình.
- 10% dữ liệu được dành cho quá trình xác thực trong quá trình huấn luyện, giúp tinh chỉnh siêu tham số và ngăn hiện tượng quá khớp.
- 10% còn lại được sử dụng để đánh giá khách quan hiệu suất của mô hình sau khi huấn luyện.

Toàn bộ dữ liệu đầu vào bao gồm các cặp ảnh:

- Một ảnh đen trắng (đầu vào của Generator).
- Một ảnh màu tương ứng (ground truth để tính hàm mất mát). Các cặp ảnh này được trích xuất từ hai thư mục gray/ và color/, trong đó mỗi ảnh ở thư mục gray/ có tên trùng khớp với ảnh tương ứng trong thư mục color/.

Sau khi nạp dữ liệu và ghép cặp, chúng tôi thực hiện xáo trộn toàn bộ tập ảnh (shuffle) để tránh hiện tượng mô hình học theo thứ tự dữ liệu, đồng thời sử dụng kỹ thuật mini-batch trong huấn luyện (batch size = 64) để tăng tốc quá trình học và ổn định hơn trong việc tính gradient.

Quá trình chia dữ liệu được hiện thực thông qua thư viện TensorFlow, cụ thể bằng việc sử dụng tf.data.Dataset để xử lý song song, ghép nối ảnh và tạo pipeline huấn luyện hiệu quả, tiết kiệm bộ nhớ và tối ưu hóa hiệu suất huấn luyện

3.1.2. Trực quan hóa cặp ảnh

Sau khi xử lý và chia tập dữ liệu thành các cặp ảnh (ảnh đen trắng, ảnh màu), chúng tôi thực hiện trực quan hóa dữ liệu để kiểm tra chất lượng ảnh và đảm bảo các cặp ảnh được ghép đúng.

Chúng tôi xây dựng một hàm có tên plot_image_pairs(...) để hiển thị một số lượng cặp ảnh bất kỳ từ tập huấn luyện.

```python
def plot_image_pairs(dataset, num_pairs=5):
    # Unbatch the dataset to get individual image pairs
    unbatched_ds = dataset.unbatch()

    # Take the specified number of pairs
    data_iterator = unbatched_ds.take(num_pairs).as_numpy_iterator()

    plt.figure(figsize=(15, 5 * (num_pairs // 5 + 1)))  # Adjust figure size

    for i, (gray, color) in enumerate(data_iterator):
        plt.subplot(num_pairs // 5 + 1, 5, i + 1)
        plt.imshow(gray.squeeze(), cmap='gray')  # Squeeze to remove single channel dimension
        plt.title(f'Grayscale Input {i+1}')
        plt.axis('off')

        plt.subplot(num_pairs // 5 + 1, 5, i + 1 + num_pairs)
        plt.imshow(color)
        plt.title(f'Color Output {i+1}')
        plt.axis('off')

    plt.tight_layout()
    plt.show()
```

Cách thức hoạt động:

- Hàm nhận vào một tf.data.Dataset (đã được batch và ghép cặp trước đó).
- Dataset được unbatch để lấy từng cặp ảnh riêng lẻ.
- Duyệt qua num_pairs cặp ảnh (mặc định là 5).
- Với mỗi cặp:
    - Ảnh đen trắng được hiển thị ở trên, dùng cmap='gray'.
    - Ảnh màu thật (ground truth) được hiển thị ở dưới.

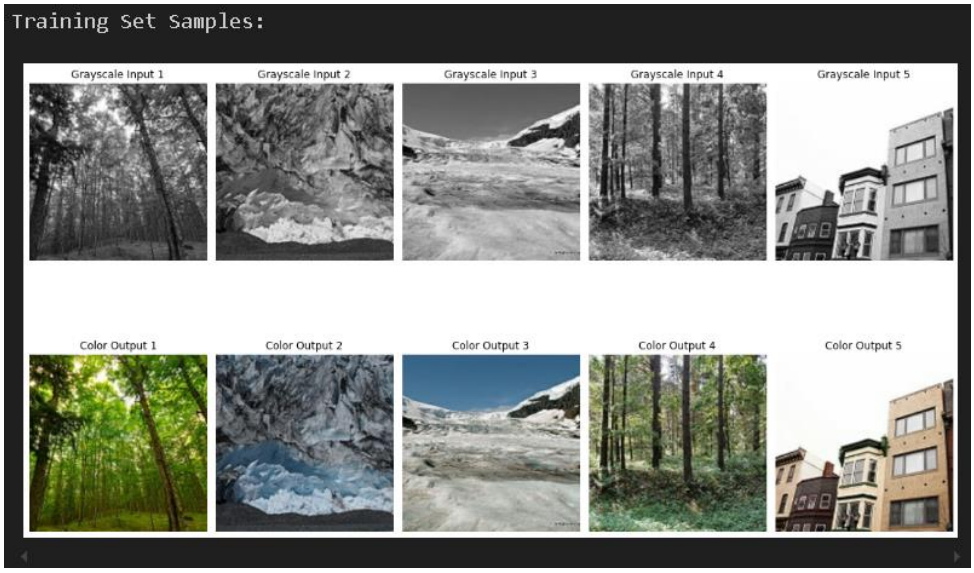
Mục đích:

- Xác nhận quá trình đọc và xử lý ảnh không xảy ra lỗi (như ảnh bị méo, vỡ, sai định dạng…).
- Đảm bảo rằng ảnh đầu vào và ảnh nhãn khớp nội dung, cùng kích thước và đúng thứ tự.
- Kiểm tra sơ bộ chất lượng dữ liệu bằng trực quan – đây là một bước quan trọng trước khi huấn luyện mạng GAN.

Sau khi đã hoàn tất quá trình chia dữ liệu và tiền xử lý, chúng tôi tiến hành kiểm tra trực quan các cặp ảnh (ảnh đen trắng và ảnh màu tương ứng) trên cả ba tập: train, validation và test.

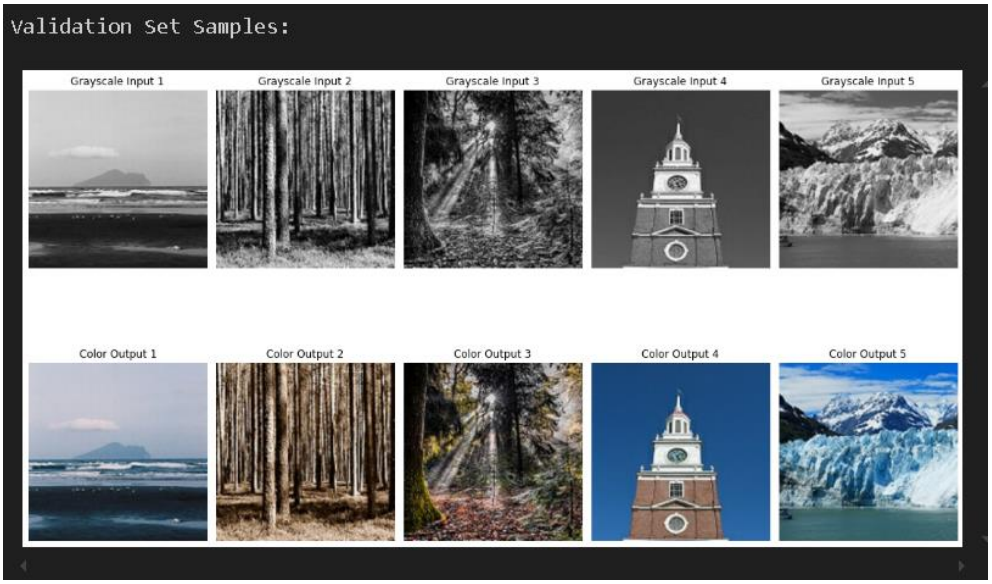
Cụ thể, chúng tôi sử dụng hàm plot_image_pairs(...) để hiển thị 5 cặp ảnh từ mỗi tập:

- **Tập huấn luyện (Training set):** Dùng để mô hình học mối quan hệ giữa ảnh đen trắng và ảnh màu



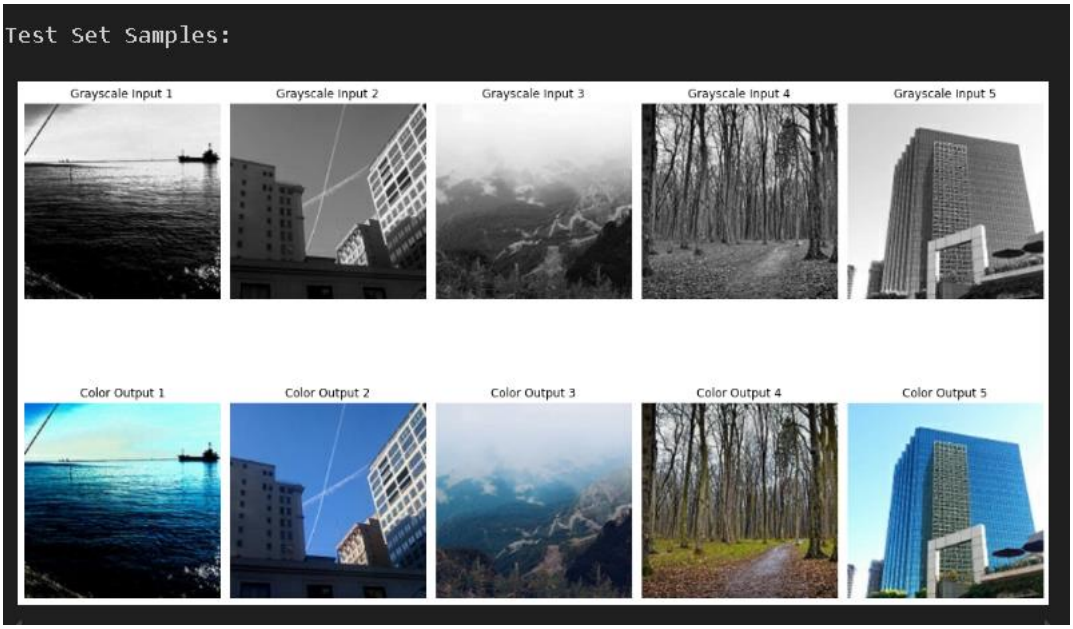
Hình 3.1 Cặp ảnh ví dụ của tập train

- **Tập xác thực (Validation set)**: Giúp đánh giá mô hình trong quá trình huấn luyện



Hình 3.2 Cặp ảnh ví dụ của tập val

- **Tập kiểm tra (Test set)**: Đánh giá khả năng tổng quát hóa của mô hình sau khi huấn luyện xong.



Hình 3.3 Cặp ảnh ví dụ của tập test

3.2 Thuật toán triển khai với mô hình Pix2Pix

3.2.1. Xây dựng Generator

Trong mô hình Pix2Pix GAN, Generator đóng vai trò sinh ảnh màu từ ảnh đen trắng đầu vào. Để đảm bảo mô hình vừa học được thông tin toàn cục, vừa giữ được chi tiết cục bộ, chúng tôi sử dụng kiến trúc U-Net cho Generator.

Kiến trúc U-Net gồm hai phần chính:

- **Encoder** : có nhiệm vụ trích xuất đặc trưng từ ảnh đen trắng.
- **Decoder** : tái tạo lại ảnh màu từ các đặc trưng đã mã hóa.
- Giữa các lớp tương ứng của encoder và decoder được kết nối bằng các **skip connections**, nhằm giúp phục hồi chính xác các chi tiết của ảnh gốc.

Các thành phần chính được xây dựng như sau:

Khối Encoder – downsample()

Đây là các lớp nén đặc trưng (convolutional blocks) trong encoder.

```
def downsample(filters, size, apply_batchnorm=True):
  result = tf.keras.Sequential()
  result.add(tf.keras.layers.Conv2D(filters, size, strides=2, padding='same',
                                    kernel_initializer='he_normal', use_bias=False))

  if apply_batchnorm:
    result.add(tf.keras.layers.BatchNormalization())
  result.add(tf.keras.layers.LeakyReLU())

  return result
```

- Sử dụng lớp Conv2D với stride = 2 để giảm kích thước ảnh.
- Áp dụng Batch Normalization để tăng ổn định khi huấn luyện (tùy chọn).
- Dùng hàm kích hoạt LeakyReLU để duy trì gradient khi đầu vào nhỏ.

Mỗi khối downsample thực hiện:

- Giảm kích thước không gian ảnh (height, width).
- Tăng số lượng filters (chiều sâu đặc trưng).

Khối Decoder – upsample()

Đây là các lớp giải mã đặc trưng (deconvolutional blocks) trong decoder

```
def upsample(filters, size, apply_dropout=False):
  result = tf.keras.Sequential()
  result.add(tf.keras.layers.Conv2DTranspose(filters, size, strides=2,padding='same',
                                             kernel_initializer='he_normal',use_bias=False))
  result.add(tf.keras.layers.BatchNormalization())

  if apply_dropout:
      result.add(tf.keras.layers.Dropout(0.5))

  result.add(tf.keras.layers.ReLU())

  return result
```

- Sử dụng Conv2DTranspose để phóng to ảnh (upsample).
- Dùng Batch Normalization và ReLU để tăng độ phi tuyến và ổn định.
- Có thể thêm Dropout để giảm overfitting ở một số lớp decoder.

Các khối upsample giúp mô hình phục hồi lại kích thước ban đầu của ảnh, đồng thời sử dụng các skip connections từ encoder để giữ chi tiết.

Sau khi định nghĩa các khối downsample() và upsample() trong kiến trúc U-Net, chúng tôi tiến hành xây dựng mô hình Generator hoàn chỉnh bằng cách kết hợp các khối này trong một mô hình tuyến tính có skip connections.

Kiến trúc tổng thể:

Generator được xây dựng theo kiến trúc **U-Net 8 tầng**, bao gồm:

- **8 khối downsampling** liên tiếp (Encoder), mỗi khối giảm kích thước không gian và tăng số lượng kênh đặc trưng.
- **7 khối upsampling** (Decoder), mỗi khối phục hồi dần kích thước ảnh và kết hợp thông tin từ encoder thông qua các skip connections.

- Một lớp Conv2DTranspose cuối cùng (tầng output) để tái tạo ảnh màu 3 kênh, sử dụng hàm kích hoạt tanh.

Các thành phần cụ thể:

- **Đầu vào (Input)**:
  Kích thước linh hoạt [None, None, 1] – ảnh đen trắng có 1 kênh.
- **Encoder (down_stack)**:
  - Gồm 8 khối downsample với các số lượng filters: 64, 128, 256, 512 × 4.
  - Lớp đầu tiên không sử dụng Batch Normalization (apply_batchnorm=False).
  - Kích thước ảnh giảm dần sau mỗi khối.
- **Decoder (up_stack)**:
  - Gồm 7 khối upsample, với các filters: 512 × 3, sau đó 256, 128, 64.
  - 3 lớp đầu tiên sử dụng Dropout (apply_dropout=True) để chống overfitting.
  - Sau mỗi khối upsample, đặc trưng được kết hợp (concatenate) với đặc trưng từ encoder tương ứng (skip connection).
- **Tầng cuối (last)**:
  - Lớp Conv2DTranspose với 3 filters để tái tạo ảnh màu RGB.
  - Sử dụng hàm kích hoạt tanh để chuẩn hóa đầu ra về dải [-1, 1], tương ứng với ảnh đã được chuẩn hóa khi đầu vào.

Luồng kết nối:

- Đầu vào được truyền qua các khối downsample, kết quả của mỗi tầng được lưu vào danh sách skips.
- Trong phần upsample, lần lượt lấy kết quả từ up_stack và kết hợp với tầng skip tương ứng từ encoder (trừ tầng cuối).
- Sau cùng, đầu ra được xử lý qua tầng last để tạo ảnh màu hoàn chỉnh.

3.2.2. Xây dựng Discriminator

Trong kiến trúc Pix2Pix GAN, Discriminator đóng vai trò là bộ phân biệt, giúp đánh giá ảnh màu được tạo ra bởi Generator là ảnh thật hay giả. Khác với các mô hình GAN truyền thống chỉ phân biệt toàn bộ ảnh, mô hình Pix2Pix sử dụng kiến trúc PatchGAN, trong đó Discriminator chỉ đánh giá các vùng nhỏ (patches) trong ảnh đầu ra.

Kiến trúc tổng thể:

- Discriminator nhận vào một cặp ảnh gồm:
  - Ảnh đen trắng (input của Generator).
  - Ảnh màu (có thể là ảnh thật hoặc ảnh được sinh ra bởi Generator).
- Hai ảnh này được ghép lại theo chiều kênh và đưa vào mô hình.

Dưới đây mã code của Discriminator:

```
def build_discriminator():

    initializer = tf.random_normal_initializer(0., 0.02)

    inp = tf.keras.layers.Input(shape=[None, None, 1], name='input_image')
    tar = tf.keras.layers.Input(shape=[None, None, 3], name='target_image')

    x = tf.keras.layers.concatenate([inp, tar])

    down1 = downsample(64, 4, False)(x)
    down2 = downsample(128, 4)(down1)
    down3 = downsample(256, 4)(down2)

    zero_pad1 = tf.keras.layers.ZeroPadding2D()(down3)
    conv = tf.keras.layers.Conv2D(512, 4, strides=1,
                                  kernel_initializer=initializer,
                                  use_bias=False)(zero_pad1)

    batchnorm1 = tf.keras.layers.BatchNormalization()(conv)

    leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)

    zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)

    last = tf.keras.layers.Conv2D(1, 4, strides=1,
                                  kernel_initializer=initializer)(zero_pad2)

    return tf.keras.Model(inputs=[inp, tar], outputs=last, name='PatchGAN_Discriminator')
```

- **Input**:
  - inp: ảnh đen trắng có shape [None, None, 1].
  - tar: ảnh màu (ground truth hoặc sinh ra) có shape [None, None, 3].
  - Hai ảnh này được concatenate lại tạo thành tensor đầu vào có 4 kênh.
- **Các lớp convolution**:
  - 3 khối downsample liên tiếp:
    - downsample(64, 4, False)
    - downsample(128, 4)
    - downsample(256, 4)

    - Các lớp này giúp giảm dần kích thước ảnh và trích xuất đặc trưng quan trọng từ ảnh kết hợp.

- **Tầng convolution đặc biệt**:
    - Sử dụng ZeroPadding2D để mở rộng biên ảnh trước khi áp dụng Conv2D với 512 filters.
    - Sau đó là BatchNormalization và LeakyReLU.
- **Tầng cuối (Output layer)**:
    - Một lớp Conv2D cuối cùng với 1 filter và stride = 1.
    - Đầu ra là một ma trận 2D chứa xác suất của từng vùng nhỏ (patch) là ảnh thật.

Tức là mỗi giá trị trong đầu ra đại diện cho tính chân thực của một vùng ảnh (patch) thay vì cả bức ảnh

3.2.3. Hàm mất mát (Loss Function)

Trong mô hình Pix2Pix GAN, quá trình huấn luyện được điều khiển bởi hai hàm mất mát riêng biệt: một cho Generator và một cho Discriminator. Ngoài ra, mô hình còn sử dụng tham số điều chỉnh λ (LAMBDA) để cân bằng giữa tính chân thực của ảnh và độ chính xác về màu sắc.

- **Hàm mất mát cho Generator**

```
def generator_loss(disc_generated_output, gen_output, target):
  adv_loss = loss_object(tf.ones_like(disc_generated_output), disc_generated_output)
  l1_loss = tf.reduce_mean(tf.abs(target - gen_output))
  total_gen_loss = adv_loss + (LAMBDA * l1_loss)
  return total_gen_loss, adv_loss, l1_loss
```

Hàm mất mát của Generator gồm 2 thành phần chính:

- **Adversarial Loss (adv_loss):**
    - Được tính bằng Binary Cross Entropy giữa đầu ra của Discriminator và nhãn 1 (ý nghĩa: ảnh sinh ra là thật).
    - Mục tiêu: khiến Discriminator "tin rằng" ảnh màu do Generator tạo ra là ảnh thật.
    - Tăng tính chân thực của ảnh sinh.
- **L1 Loss (l1_loss):**
    - Là sai số trung bình tuyệt đối giữa ảnh sinh ra (gen_output) và ảnh thật (target).

- Mục tiêu: đảm bảo kết quả đầu ra giống thực tế về mặt nội dung.
- Giúp ảnh tái tạo ra ít bị nhiễu và giữ cấu trúc ổn định.

- **Tổng Loss của Generator (total_gen_loss):**

Kết hợp hai thành phần trên, tổng loss của Generator:

$$\text{loss}_G = \text{adv\_loss} + \lambda \cdot \text{l1\_loss}$$

Trong đó, **λ (LAMBDA)** được đặt là 100 để ưu tiên độ chính xác màu sắc.

- **Hàm mất mát cho Discriminator**

```
def discriminator_loss(disc_real_output, disc_generated_output):
  real_loss = loss_object(tf.ones_like(disc_real_output), disc_real_output)
  generated_loss = loss_object(tf.zeros_like(disc_generated_output), disc_generated_output)
  total_disc_loss = real_loss + generated_loss
  return total_disc_loss
```

- **Real Loss:** Phân biệt ảnh thật (ground truth) đúng là ảnh thật → nhãn 1.
- **Generated Loss:** Phân biệt ảnh sinh là ảnh giả → nhãn 0.
- **Tổng Discriminator Loss:** Cộng cả 2 thành phần lại, giúp Discriminator học cách phân biệt tốt hơn giữa ảnh thật và ảnh sinh từ Generator.

Tổng Loss của Discriminator:

$$\text{loss}_D = \text{real\_loss} + \text{generated\_loss}$$

3.3 Quá trình huấn luyện mô hình

3.3.1. Các tham số

```
GENERATOR_LR_INIT = 2e-4
DISCRIMINATOR_LR_INIT = 2e-4
ADAM_BETA_1 = 0.5
SAVE_DIR = 'gan_checkpoints_minimal'
EARLY_STOPPING_MONITOR = 'val_gen_l1_loss'
```

- GENERATOR_LR_INIT = 2e-4: tốc độ học ban đầu của Generator.
- DISCRIMINATOR_LR_INIT = 2e-4: tốc độ học ban đầu của Discriminator.
- ADAM_BETA_1 = 0.5: hệ số beta1 cho optimizer Adam.
- SAVE_DIR: thư mục lưu trọng số mô hình tốt nhất.
- EARLY_STOPPING_MONITOR = 'val_gen_l1_loss': chỉ số để theo dõi và dừng sớm khi không cải thiện

3.3.2. Vòng lặp huấn luyện

Mỗi epoch gồm các bước:

```python
@tf.function
def train_step_internal(input_image, target):
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        gen_output = generator(input_image, training=True)
        disc_real_output = discriminator([input_image, target], training=True)
        disc_generated_output = discriminator([input_image, gen_output], training=True)
        gen_total_loss, gen_adv_loss, gen_l1_loss = generator_loss(
            disc_generated_output, gen_output, target
        )
        disc_loss = discriminator_loss(disc_real_output, disc_generated_output)
    generator_gradients = gen_tape.gradient(gen_total_loss, generator.trainable_variables)
    discriminator_gradients = disc_tape.gradient(disc_loss, discriminator.trainable_variables)
    generator_optimizer.apply_gradients(zip(generator_gradients, generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(discriminator_gradients, discriminator.trainable_variables))
    return gen_adv_loss, gen_l1_loss, disc_loss
```

- Huấn luyện trên từng batch (train_step_internal)
  - Generator tạo ảnh màu từ ảnh đen trắng.
  - Discriminator đánh giá ảnh thật và ảnh sinh ra.
  - Tính tổng loss gồm:
    - gen_adv_loss: mất mát phân biệt (GAN loss).
    - gen_l1_loss: mất mát tái tạo (L1 loss giữa ảnh sinh và ảnh gốc).
    - disc_loss: mất mát của Discriminator.
  - Cập nhật trọng số cả hai mô hình thông qua gradient.

```python
@tf.function
def validation_step_internal(input_image, target):
    gen_output = generator(input_image, training=False)
    disc_generated_output = discriminator([input_image, gen_output], training=False)
    _, gen_adv_loss, gen_l1_loss = generator_loss(
        disc_generated_output, gen_output, target
    )
    return gen_adv_loss, gen_l1_loss
```

- Huấn luyện tập xác thực (validation_step_internal)
  - Tính loss trên tập validation để theo dõi tiến trình học.
  - So sánh chỉ số theo dõi (val_gen_l1_loss) với chỉ số tốt nhất.
  - Nếu tốt hơn thì lưu trọng số mô hình.

```python
if current_metric_value < best_metric_value:
    print(f"\n  >> Metric ({metric_key_to_monitor}) improved: {best_metric_value:.4f} -> {current_metric_value:.4f}. Saving models...")
    best_metric_value = current_metric_value
    generator.save_weights(best_gen_weights_path)
    discriminator.save_weights(best_disc_weights_path)
    epochs_no_improve = 0
    best_epoch = epoch + 1
    print()
```

- Cập nhật model tốt nhất và theo dõi: Lưu mô hình nếu **val_gen_l1_loss** cải thiện

```python
if perform_lr_decay and (epochs_no_improve % lr_decay_patience == 0) and epochs_no_improve > 0 :
    current_gen_lr = generator_optimizer.learning_rate.numpy()
    current_disc_lr = discriminator_optimizer.learning_rate.numpy()

    new_gen_lr = max(current_gen_lr * lr_decay_factor, min_lr)
    new_disc_lr = max(current_disc_lr * lr_decay_factor, min_lr)

    lr_updated = False
    if new_gen_lr < current_gen_lr:
        generator_optimizer.learning_rate.assign(new_gen_lr)
        lr_updated = True
    if new_disc_lr < current_disc_lr:
        discriminator_optimizer.learning_rate.assign(new_disc_lr)
        lr_updated = True

    if lr_updated:
        print(f"  >> Decaying learning rate after {epochs_no_improve} epochs of no improvement.")
        print(f"     Gen LR: {current_gen_lr:.1e} -> {generator_optimizer.learning_rate.numpy():.1e}")
        print(f"     Disc LR: {current_disc_lr:.1e} -> {discriminator_optimizer.learning_rate.numpy():.1e}\n")
```

- Điều chỉnh learning_rate: Giảm learning rate nếu không có cải thiện sau số epoch xác định.

```python
if perform_early_stopping and epochs_no_improve >= early_stopping_patience:
    print(f"-- Early stopping triggered after epoch {epoch + 1} --")
    print(f"   Best metric ({metric_key_to_monitor}): {best_metric_value:.4f} achieved at epoch {best_epoch}.")
    break
```

- Dừng sớm nếu không có cải thiện: Kết thúc huấn luyện sớm nếu không cải thiện sau **early_stopping_patience** epoch.

3.4 Tổ chức chương trình

Cấu trúc thư mục

Chương trình được tổ chức theo các thư mục và tệp chính như sau:

- landscape Images/: Chứa toàn bộ dữ liệu huấn luyện. Bao gồm hai thư mục con (mặc định):
    - gray/: chứa ảnh đen trắng đầu vào.
    - color/: chứa ảnh màu tương ứng (ground truth).
- source/: Chứa mã nguồn chính, bao gồm:
    - **examples:** Một số ảnh trắng đen được đưa sẵn vào đây để làm ví dụ thử nghiệm cho giao diện Gradio. Người dùng có thể chọn ảnh từ thư mục này để xem kết quả mô hình.
    - **pix2pix_checkpoints_minimal:** chứa các mô hình được huấn luyện:
        - best_discriminator.weights.h5: Trọng số của Generator tại thời điểm mô hình đạt kết quả tốt nhất.
        - best_generator.weights.h5 : Trọng số của Discriminator tương ứng.
    - **train_3.ipynb:** File Jupyter Notebook chính để huấn luyện mô hình Pix2Pix từ đầu. Toàn bộ pipeline xử lý dữ liệu, xây dựng kiến trúc mạng,

định nghĩa hàm mất mát, vòng lặp huấn luyện, early stopping, và lưu mô hình đều được thực hiện trong file này.

- **demo2.py:** File triển khai giao diện người dùng sử dụng thư viện Gradio
  - Giao diện này cho phép:
    - Tải lên ảnh trắng đen từ máy người dùng.
    - Dự đoán ảnh màu hóa tương ứng bằng Generator đã huấn luyện.
    - Hiển thị và lưu kết quả sinh ra.

3.5 Ứng dụng kiểm thử

Kiểm thử	Cách kiểm thử
1. Đánh giá quá trình huấn luyện	Theo dõi các chỉ số gen_adv_loss, gen_l1_loss, disc_loss được ghi nhận trong từng epoch thông qua Progbar.
2. Kiểm thử mô hình trên tập xác thực	Sử dụng tập validation_ds để đánh giá chất lượng sinh ảnh trong từng epoch bằng val_gen_l1_loss.
3. Kiểm thử cơ chế dừng sớm (Early Stopping)	Mô hình dừng huấn luyện nếu chỉ số val_gen_l1_loss không được cải thiện sau số lượng epoch cố định.
4. Kiểm tra khả năng phục hồi mô hình tốt nhất	Sau khi huấn luyện, mô hình tự động tải lại trọng số tốt nhất từ file best_generator.weights.h5.
5. Kiểm thử giảm tốc độ học (Learning Rate Decay)	Nếu không cải thiện, learning rate sẽ tự động giảm theo hệ số trong lr_decay_factor.
6. Kiểm thử triển khai mô hình với Gradio	Sử dụng script demo2.py để tải ảnh trắng đen và kiểm tra mô hình có thể sinh ảnh màu hợp lý qua giao diện.

Bảng 3.1 Ứng dụng kiểm thử

3.6 Ngôn ngữ cài đặt

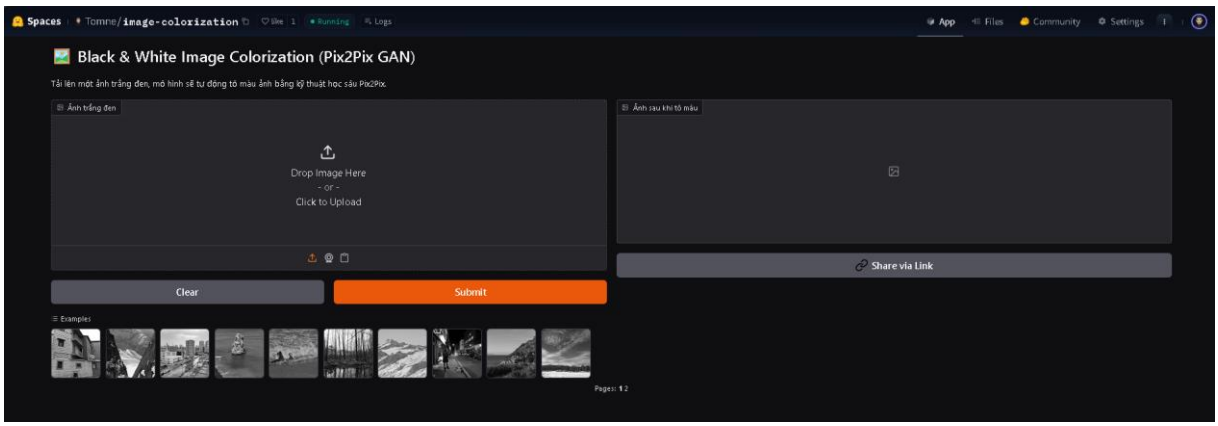
- **Ngôn ngữ cài đặt chính**: Python.
- **Ngôn ngữ tạo giao diện**: Giao diện người dùng được xây dựng bằng Gradio – một thư viện Python hỗ trợ tạo giao diện web tương tác nhanh chóng và đơn giản.

Các thư viện và phiên bản sử dụng:

- **TensorFlow**: Xây dựng và huấn luyện mô hình Pix2Pix GAN (Generator & Discriminator).
- **NumPy**: Xử lý dữ liệu ảnh dưới dạng mảng số học.
- **OpenCV (cv2)**: Tiền xử lý ảnh, đọc/ghi ảnh, thao tác với ảnh grayscale và RGB.
- **PIL (Python Imaging Library)**: Đọc, resize, hiển thị và lưu ảnh.
- **Matplotlib**: Vẽ biểu đồ, trực quan hóa các cặp ảnh input/output trong huấn luyện.
- **Gradio**: Tạo giao diện web để kiểm thử mô hình bằng ảnh thực tế

3.7 Kết quả thực thi chương trình

3.7.1. Giao diện chính chương trình



Hình 3.1 Giao diện chính

3.7.2. Kết quả thực thi chương trình

- **Ý nghĩa và cách đánh giá**

Trong đề tài Image Colorization bằng Pix2Pix, mô hình không sinh ra văn bản mà sinh ra ảnh màu từ ảnh trắng đen. Do đó, để đánh giá chất lượng đầu ra, chúng tôi sử dụng các chỉ số sau:

- **L1 Loss (Mean Absolute Error)**

Là sai số trung bình tuyệt đối giữa ảnh màu thực (ground truth) và ảnh màu sinh ra (output từ Generator). Chỉ số này được ghi nhận liên tục trong quá trình huấn luyện (cả training và validation).

- **GAN Loss (Generator Adversarial Loss)**

Thể hiện khả năng đánh lừa Discriminator của Generator. Giá trị thấp cho thấy Generator tạo ảnh ngày càng "thật"

- **Discriminator Loss**

Thể hiện khả năng phân biệt giữa ảnh thật và ảnh sinh. Giá trị cân bằng giữa Discriminator và Generator là lý tưởng cho quá trình học ổn định.

- **Quá trình ghi nhận**

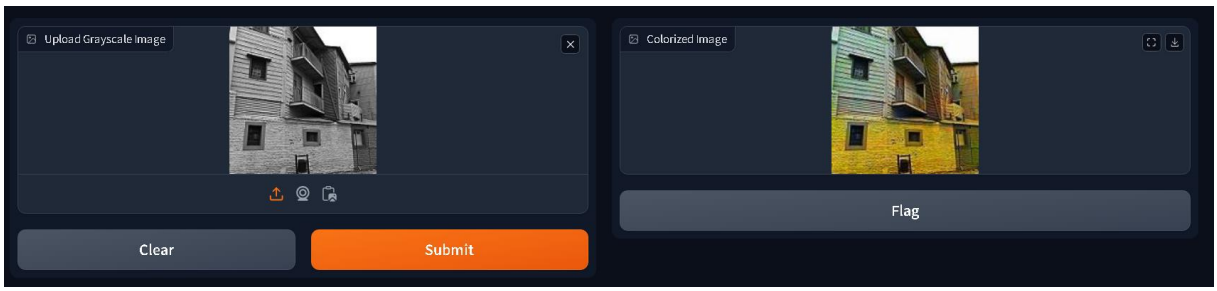
Trong quá trình huấn luyện, ba chỉ số gen_l1_loss, gen_adv_loss và disc_loss được hiển thị trực tiếp trong mỗi epoch bằng thanh tiến trình (Progbar). Dựa vào sự biến đổi của các giá trị này, chúng tôi đánh giá khả năng hội tụ và hiệu suất của mô hình.

- **Kết quả mô hình tốt nhất**

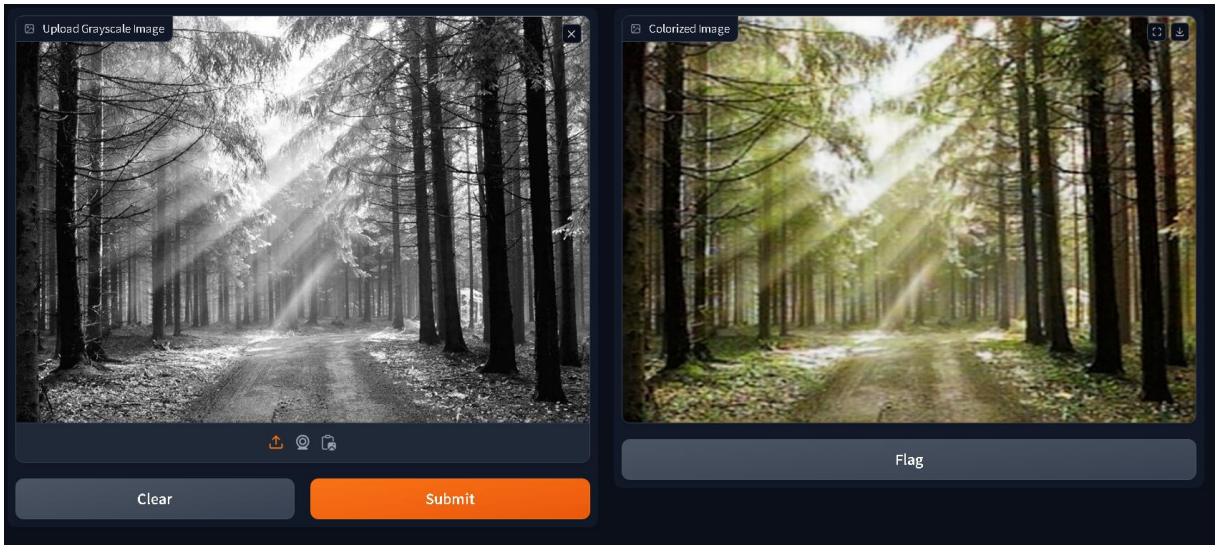
Mô hình Generator có trọng số tốt nhất (tức epoch có val_gen_l1_loss nhỏ nhất) được lưu lại tự động vào file best_generator.weights.h5, giúp đảm bảo mô hình được kiểm thử và triển khai với kết quả tối ưu nhất.

Phiên bản mô hình	Validation L1 Loss	Generator Adversarial Loss	Discriminator Loss	Số epoch đạt tốt nhất
Pix2Pix (kích thước ảnh 128×128)	0.1187	2.0021		96
Pix2Pix + Dropout (kích thước ảnh 256x256)	0.0227	0.7770	1.2789	133

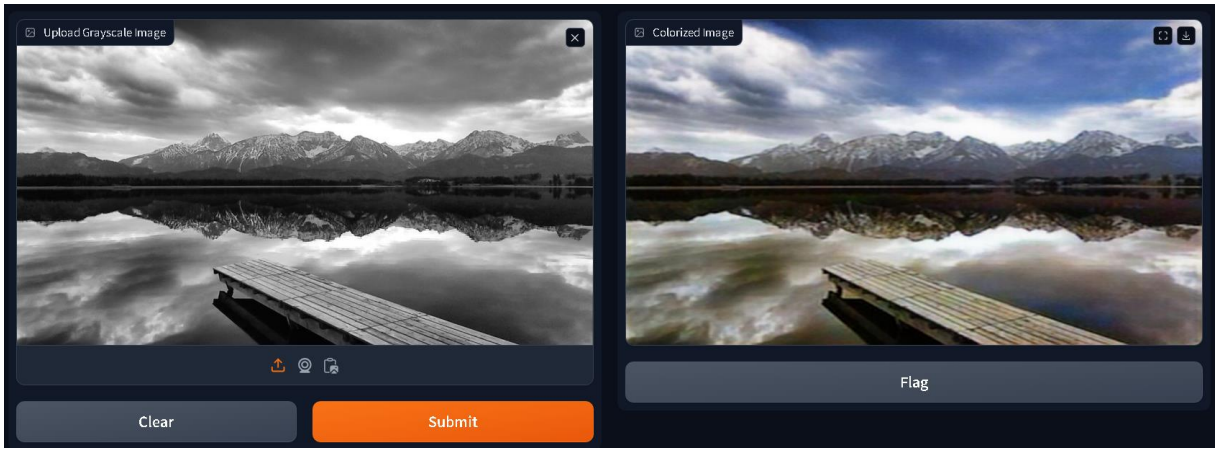
Bảng 3.2 Kiểm thử và phương pháp kiểm thử



Hình 3.2 Kết quả phục chế ảnh 1



Hình 3.3 Kết quả phục chế ảnh 2



Hình 3.4 Kết quả phục chế ảnh 3

3.8 Nhận xét và đánh giá

a. Nhận xét:

- Mô hình Pix2Pix GAN hoạt động ổn định và có khả năng phục chế màu cho ảnh trắng đen với chất lượng thị giác tốt.
- Loss (L1 loss và adversarial loss) giảm đều trong quá trình huấn luyện, cho thấy mô hình học được mối quan hệ giữa ảnh trắng đen và màu.
- Thời gian huấn luyện mô hình khá dài, đặc biệt khi làm việc với ảnh kích thước lớn (256x256), và cần GPU để rút ngắn thời gian huấn luyện.

- Trong một số trường hợp, màu sắc được phục chế vẫn chưa thật sự chính xác, đặc biệt với các vùng ảnh có ít thông tin hoặc các chi tiết phức tạp (bầu trời, vùng mờ nhòe).
- Giao diện người dùng được xây dựng bằng Gradio giúp kiểm thử mô hình dễ dàng, tạo trải nghiệm tốt cho người dùng cuối.

b. Đánh giá:

- Mô hình đã được triển khai thành công, thực hiện chuyển ảnh trắng đen thành ảnh màu có ý nghĩa thực tế cao, đặc biệt trong các bài toán phục chế ảnh lịch sử hoặc ảnh y tế.
- Mô hình đáp ứng đầy đủ yêu cầu kỹ thuật của đồ án môn học, ứng dụng kiến thức về mạng GAN, CNN, và pipeline xử lý ảnh.
- Kết hợp linh hoạt giữa các thành phần:
    - Thị giác máy tính: Xử lý ảnh đầu vào, normalize, resize, đánh giá loss.
    - Deep Learning: Xây dựng mô hình Pix2Pix GAN với Generator (U-Net) và Discriminator (PatchGAN).
- Kỹ năng triển khai giao diện người dùng, đánh giá mô hình và đóng gói mô hình để demo với Gradio được thực hiện tốt.
- Qua quá trình thực hiện đồ án, kỹ năng làm việc nhóm, giao tiếp với giảng viên, cũng như khả năng viết báo cáo và trình bày slide đã được nâng cao rõ rệt.

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết quả đạt được

Trong đồ án này, được đã triển khai thành công mô hình Pix2Pix GAN để thực hiện nhiệm vụ tô màu (colorization) cho ảnh trắng đen. Mô hình sử dụng kiến trúc Generator dạng U-Net và Discriminator dạng PatchGAN nhằm học được mối quan hệ giữa ảnh trắng đen và ảnh màu thực tế.

Được đã tiến hành xử lý dữ liệu từ tập ảnh phong cảnh, xây dựng pipeline huấn luyện bằng TensorFlow, theo dõi các chỉ số như L1 Loss, Adversarial Loss và Discriminator Loss, đồng thời lưu lại mô hình tốt nhất. Kết quả thử nghiệm cho thấy mô hình có khả năng phục chế ảnh màu với độ chi tiết và tương phản tốt trong đa số trường hợp.

Bên cạnh đó, được cũng triển khai giao diện người dùng bằng thư viện Gradio, giúp người dùng dễ dàng upload ảnh trắng đen và xem ảnh được tô màu ngay lập tức trên trình duyệt, phục vụ cho mục tiêu kiểm thử và trình diễn mô hình.

Tuy nhiên, mô hình vẫn tồn tại một số hạn chế, đặc biệt khi xử lý các ảnh có nội dung phức tạp, ánh sáng bất thường hoặc màu sắc khó dự đoán. Ngoài ra, quá trình huấn luyện mô hình GAN cũng yêu cầu phần cứng mạnh và thời gian huấn luyện dài để đạt được kết quả tối ưu.

Tổng kết lại, mô hình đã đáp ứng được các mục tiêu đề ra của đồ án, từ xử lý dữ liệu, xây dựng kiến trúc mô hình, huấn luyện, đánh giá đến triển khai ứng dụng thử nghiệm.

4.2 Hướng phát triển

1. Nâng cấp mô hình Generator bằng cách thay thế hoặc tinh chỉnh kiến trúc U-Net, áp dụng các kỹ thuật như Attention U-Net hoặc sử dụng GAN cải tiến như SPADE hoặc StyleGAN.
2. Tăng kích thước ảnh đầu vào, thử nghiệm huấn luyện trên ảnh có độ phân giải cao hơn để cải thiện độ sắc nét và tính chân thực.
3. Tối ưu tốc độ xử lý để ứng dụng mô hình vào môi trường thời gian thực (real-time inference), ví dụ như webcam phục chế ảnh tức thì.
4. Triển khai mô hình trên nền tảng web hoặc di động, tích hợp qua Flask hoặc Streamlit để mở rộng khả năng tiếp cận với người dùng.
5. Huấn luyện mô hình trên tập dữ liệu ảnh đa dạng hơn (ảnh người, vật thể, tranh vẽ…) nhằm cải thiện khả năng tổng quát hóa.

6. So sánh với các phương pháp khác như autoencoder, truyền thống (Histogram Matching), hoặc mô hình học sâu không sử dụng GAN để đánh giá toàn diện hơn.

TÀI LIỆU THAM KHẢO

[1] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.

[3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative Adversarial Nets. *NeurIPS*.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] TensorFlow Pix2Pix Tutorial. pix2pix: Image-to-image translation with a conditional GAN | TensorFlow Core.

[6] Pranav P. Jadhav, Tejas P. Kore, et al. (2023). *Image Colorization using Pix2Pix GAN*. In Proceedings of NCSTCS-11, *International Advanced Research Journal in Science, Engineering and Technology*. IARJSET-NCSTCS-11.pdf

[7] Kaggle Dataset: Landscape Image Colorization. Landscape color and grayscale images

[8] Gradio Documentation. https://www.gradio.app

[9] Phạm Đình Khánh. *Pix2Pix GAN - Biến ảnh trắng đen thành ảnh màu sử dụng GAN*.https://phamdinhkhanh.github.io/2020/11/13/pix2pixGAN.html.

[10] Hugging Face documents. Hugging Face – The AI community building the future.

PHỤ LỤC 1

PHỤ LỤC 2