**TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN TỐT NGHIỆP**

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

Người hướng dẫn**: TS. Nguyễn Công Danh**

Sinh viên thực hiện**:**

**Nguyễn Khắc Nhân Tâm LỚP: 21TCLC\_KHDL2 NHÓM: 0**

**Đà Nẵng, 12/2024**

**MỤC LỤC**

[DANH MỤC HÌNH VẼ 3](#_Toc200149581)

[MỞ ĐẦU 1](#_Toc200149582)

[1. TỔNG QUAN ĐỀ TÀI. 1](#_Toc200149583)

[1.1. Thực trạng hiện nay 1](#_Toc200149584)

[1.1.1 Mục tiêu nghiên cứu 1](#_Toc200149585)

[1.1.2 Đối tượng nghiên cứu 2](#_Toc200149586)

[1.1.3 Phương pháp nghiên cứu 2](#_Toc200149587)

[1.2. Tổng quan đề tài 3](#_Toc200149588)

[2. THU NHẬP DỮ LIỆU 4](#_Toc200149589)

[2.1. Nguồn dữ liệu 4](#_Toc200149590)

[2.2. Chi tiết dữ liệu 5](#_Toc200149591)

[2.3. Tiền xử lý dữ liệu 5](#_Toc200149592)

[3. CƠ SỞ LÝ THUYẾT 6](#_Toc200149593)

[3.1. Ý tưởng 6](#_Toc200149594)

[3.2 Cơ sở lý thuyết 6](#_Toc200149595)

[3.2.1. Mạng nơron tích chập 6](#_Toc200149596)

[3.2.3. Mạng Pix2Pix 8](#_Toc200149597)

[3.2.4. Generator-Unet 11](#_Toc200149598)

[3.2.5. Discriminator 12](#_Toc200149599)

[3.2.5. Loss Funtion 13](#_Toc200149600)

[4. THUẬT TOÁN TRIỂN KHAI 15](#_Toc200149601)

[4.1. Xử lí dữ liệu 15](#_Toc200149602)

[4.1.1. Chia tập dữ liệu 15](#_Toc200149603)

[4.1.2 Trực quan hóa cặp ảnh 15](#_Toc200149604)

[4.2 Thuật toán triển khai với mô hình Pix2Pix 18](#_Toc200149605)

[4.2.1 Xây dựng Generator 18](#_Toc200149606)

[4.2.2 Xây dựng Discriminator 20](#_Toc200149607)

[4.2.3 Hàm mất mát (Loss Funtion) 22](#_Toc200149608)

[5. CHƯƠNG TRÌNH VÀ KẾT QUẢ 24](#_Toc200149609)

[5.1. Tổ chức chương trình 24](#_Toc200149610)

[5.2. Ứng dụng học máy trong chương trình 24](#_Toc200149611)

[5.3. Ứng dụng kiểm thử 26](#_Toc200149612)

[5.4 Ngôn ngữ cài đặt 27](#_Toc200149613)

[5.5 Kết quả thực thi chương trình 27](#_Toc200149614)

[5.5.1 Giao diện chính của chương trình 27](#_Toc200149615)

[5.5.2 Kết quả thực thi của chương trình 27](#_Toc200149616)

[5.5.3. Nhận xét và đánh giá 29](#_Toc200149617)

[6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 31](#_Toc200149618)

[6.1. Kết luận 31](#_Toc200149619)

[TÀI LIỆU THAM KHẢO 33](#_Toc200149620)

# DANH MỤC HÌNH VẼ

[Hình 3.2.1. Kiến trúc mạng nơron tích chập 8](#_Toc200149572)

[Hình 3.2.3.3. Kiến trúc tổng quát mạng Pix2Pix 10](#_Toc200149573)

[Hình 3.2.4.1. Kiến trúc Unet trong Generator 12](#_Toc200149574)

[Hình 3.2.4.1. Kiến trúc Discriminator 13](#_Toc200149575)

[Hình 5.5.1. Giao diện chính của chương trình 27](#_Toc200149576)

[Hình 5.5.2.3. Kết quả phục chế ảnh 1 28](#_Toc200149577)

[Hình 5.5.2.4. Kết quả phục chế ảnh 2 29](#_Toc200149578)

[29](#_Toc200149579)

[Hình 5.5.2.4. Kết quả phục chế ảnh 3 29](#_Toc200149580)

MỞ ĐẦU

Hiện nay, nhu cầu khôi phục và cải thiện chất lượng hình ảnh ngày càng tăng cao, đặc biệt là trong các lĩnh vực như lưu trữ di sản hình ảnh, phim ảnh cổ, và nâng cao trải nghiệm thị giác trong các ứng dụng truyền thông. Trong bối cảnh đó, bài toán tô màu ảnh đen trắng – *image colorization* – đang nhận được sự quan tâm lớn từ cộng đồng nghiên cứu và ứng dụng. Đây là một bài toán khó vì nó yêu cầu mô hình không chỉ phục hồi màu sắc mà còn hiểu ngữ cảnh và nội dung ảnh một cách sâu sắc.

Dựa trên nhu cầu đó, chúng em quyết định thực hiện nghiên cứu và ứng dụng mô hình học sâu hiện đại, cụ thể là mô hình **Pix2Pix GAN**, một biến thể của Generative Adversarial Network, để giải quyết bài toán tô màu ảnh đen trắng. Thông qua mô hình này, chúng em đánh giá hiệu quả phục hồi màu sắc dựa trên độ chân thực, độ sắc nét cũng như khả năng tổng quát hóa trên các tập dữ liệu mới. Đồng thời, chúng em cũng phân tích chi phí tính toán và tiềm năng cải tiến mô hình trong tương lai.

Qua đồ án này, chúng em kỳ vọng không chỉ góp phần vào việc phát triển các ứng dụng phục hồi và nâng cao chất lượng hình ảnh, mà còn mở rộng hiểu biết về khả năng áp dụng các mô hình học sâu trong xử lý ảnh và thị giác máy tính hiện đại.

# TỔNG QUAN ĐỀ TÀI.

## 1.1. Thực trạng hiện nay

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, chúng em 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, qua đồ án này, chúng em mong muốn không chỉ ứng dụng thành công mô hình vào bài toán cụ thể, mà còn đóng góp một phần tài liệu tham khảo hữu ích cho những bạn sinh viên, nhà nghiên cứu quan tâm đến lĩnh vực này trong tương lai.

### 1.1.1 Mục tiêu nghiên cứu

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.

### 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.

### 1.1.3 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**:
  + Sử dụng các chỉ số đánh giá chất lượng ảnh như **PSNR (Peak Signal-to-Noise Ratio)** và **SSIM (Structural Similarity Index)** để đo lường độ tương đồng giữa ảnh gốc và ảnh được tô màu.
  + 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.

## 1.2. Tổng quan đề tài

Để giải quyết bài toán chúng em đề xuất hai mô hình học sâu tiên tiến nhất hiện nay, bao gồm: mô hình sequence-to-sequence dựa trên RNN, mô hình sequence-to-sequence có cơ chế chú ý kết hợp với xử lý ngôn ngữ để tạo ra các chú thích video có ý nghĩa. Qua hai mô hình này, chúng em đánh giá độ chính xác, chi phí tính toán và hiệu quả của mô hình, từ đó đề xuất các hướng phát triển trong tương lai khi tập dữ liệu lớn hơn và các hành động trong video được mô tả đa dạng hơn.

# 2. THU NHẬP DỮ LIỆU

## 2.1. Nguồn dữ liệu

Trong đồ án này, chúng em 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.

## 2.2. Chi tiết dữ liệu

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.

## 2.3. Tiền xử lý dữ liệu

* Resize ảnh về 150×150 pixels để phù hợp với kích thước đầu vào của mô hình.
* Chuẩn hóa giá trị pixel về [0,1] hoặc [-1,1] (tùy theo framework sử dụng).
* Ảnh đen trắng được nạp làm đầu vào của Generator, còn ảnh màu dùng làm ground truth để so sánh và tính loss.

# 3. CƠ SỞ LÝ THUYẾT

## 3.1. Ý tưởng

Phục chế ảnh đen trắng là quá trình tô màu tự động cho ảnh nhằm khôi phục lại màu sắc tự nhiên, tăng tính chân thực và giá trị thẩm mỹ. Đề tài này đề xuất sử dụng mô hình học sâu Pix2Pix GAN để thực hiện nhiệm vụ phục chế ảnh. Mô hình học cách chuyển đổi từ ảnh đen trắng sang ảnh màu bằng cách kết hợp giữa Generator và Discriminator nhằm tạo ra ảnh màu có độ chân thực cao. Dữ liệu huấn luyện là ảnh phong cảnh, được chia thành ảnh đen trắng đầu vào và ảnh màu mục tiêu.

Mô hình Generator được thiết kế theo kiến trúc U-Net để bảo toàn các chi tiết không gian trong ảnh, trong khi Discriminator giúp phân biệt ảnh màu thật và ảnh màu được phục chế, từ đó cải thiện chất lượng đầu ra. Quá trình huấn luyện sử dụng hàm mất mát kết hợp giữa L1 loss và adversarial loss để đạt được ảnh đầu ra vừa chính xác về màu sắc, vừa tự nhiên về cấu trúc.

Đề tài sẽ tiến hành thực nghiệm trên tập ảnh phong cảnh với nhiều khung cảnh đa dạng như biển, núi, rừng và thành phố. Mô hình được đánh giá bằng các chỉ số SSIM, PSNR và so sánh trực quan giữa ảnh gốc và ảnh được tô màu. Kết quả kỳ vọng sẽ cho thấy khả năng phục chế ảnh hiệu quả và ứng dụng được trong các lĩnh vực như lưu trữ di sản, nghệ thuật và phục hồi ảnh cũ.

## 3.2 Cơ sở lý thuyết

### 3.2.1. 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

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

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.

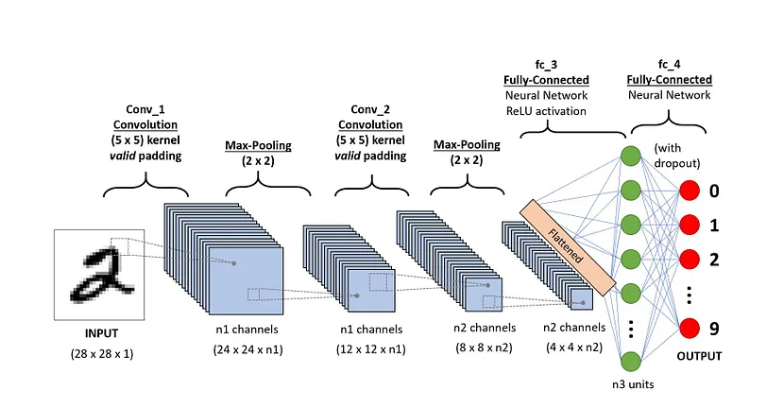
**2. Discriminator (PatchGAN CNN)**

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.



Hình 3.2.1. Kiến trúc mạng nơron tích chập

### 3.2.3. 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:

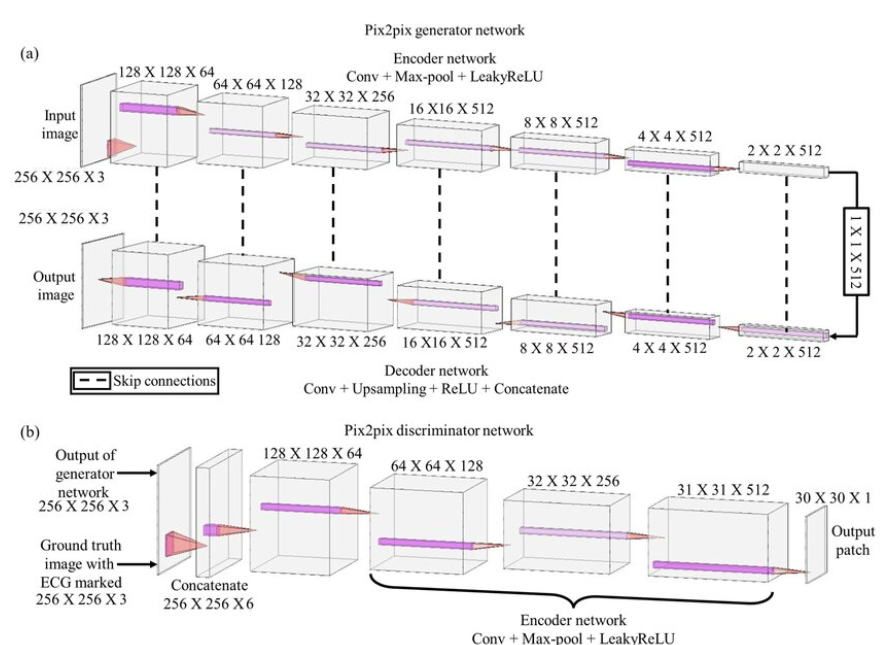
* **Generator (Bộ sinh ảnh):**

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.

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.
* **Discriminator (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.



Hình 3.2.3.3. 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. Ưu và nhược điểm mô hình Pix2Pix*

### 3.2.4. Generator-Unet

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.

Kiến trúc Unet gồm 2 phần chính:

* **Encoder (Bộ mã hóa)**: Gồm nhiều lớp tích chập (convolution) liên tiếp có stride > 1, nhằm giảm kích thước không gian của ảnh đầu vào và trích xuất các đặc trưng quan trọng. Mỗi lớp encoder bao gồm:
  + Lớp tích chập
  + Batch Normalization
  + Hàm kích hoạt LeakyReLU
* **Decoder (Bộ giải mã)**: Làm nhiệm vụ khôi phục lại ảnh từ các đặc trưng đã mã hóa bằng các lớp deconvolution (transposed convolution). Decoder cũng sử dụng batch normalization và hàm kích hoạt ReLU.

**Skip connections (nối tắt):**

* Một điểm đặc trưng của U-Net là các đường nối tắt giữa encoder và decoder tương ứng.
* Chúng giúp giữ lại thông tin chi tiết của ảnh gốc như cạnh, hình dạng, kết cấu… – rất quan trọng trong việc tô màu chính xác theo từng vùng ảnh.
* Việc truyền trực tiếp đặc trưng ở cấp độ thấp từ encoder sang decoder giúp ảnh đầu ra sắc nét hơn, tránh hiện tượng bị mờ hoặc sai lệch màu.

A diagram of a mathematical equation

AI-generated content may be incorrect.

Hình 3.2.4.1. Kiến trúc Unet trong Generator

* **Generator**: Đầu vào của generator là một bức ảnh nguồn (source image), như trong hình là ảnh chụp vệ tinh có màu. Sau đó bức ảnh được truyền vào một kiến trúc mạng tích chập để trích lọc đặc trưng thông qua quá trình down sampling. Ở quá trình này, output ở các layer sẽ có kích thước giảm dần thông qua tích chập 2 chiều. Trong khi đó mạng giải chập (Deconvolution) làm nhiệm vụ ngược lại là biến đổi đặc trưng thành ảnh đích (target image) thông qua quá trình up sampling bằng các layers chuyên biệt như transposed Convolution, Dilation Convolution, Upsampling 2D. Quá trình up sampling sẽ gia tăng dần kích thước output và kết quả trả ra là một bức ảnh đích mang nhãn fake.

### 3.2.5. Discriminator

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.

Khác với các mô hình phân loại truyền thống đánh giá toàn bộ ảnh, Pix2Pix sử dụng **PatchGAN** – một loại Discriminator được thiết kế để đánh giá tính chân thực của các vùng nhỏ (patches) trong ảnh.

**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.

A comparison of a pair of shoes

AI-generated content may be incorrect.

Hình 3.2.4.1. 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.

### 3.2.5. Loss Funtion

Trong mô hình Pix2Pix GAN, quá trình huấn luyện là sự kết hợp giữa một mạng sinh (Generator) và một mạng phân biệt (Discriminator), với mục tiêu giúp Generator tạo ra các ảnh màu có độ chân thực cao từ ảnh đen trắng. Để hướng dẫn mô hình học hiệu quả, Pix2Pix sử dụng hàm mất mát kết hợp, bao gồm:

**Adversarial Loss (GAN Loss)**

* Đây là hàm mất mát chính của mô hình GAN.
* Generator cố gắng sinh ảnh màu đủ "giống thật" để lừa được Discriminator.
* Discriminator cố gắng phân biệt giữa ảnh màu thật và ảnh màu giả do Generator sinh ra.
* Mục tiêu: Giúp ảnh đầu ra có tính chân thực, sống động như ảnh thật.

Loss function của Pix2Pix là một hàm dạng binary cross entropy có dạng như sau:\

A black text with a rectangular object on it

AI-generated content may be incorrect.

Trong đó:

* x: ảnh đen trắng (input)
* y: ảnh màu thật (ground truth)
* G(x): ảnh màu giả sinh ra từ Generator
* D(x,y): xác suất ảnh là thật do Discriminator dự đoán

**L1 Loss (Mean Absolute Error)**

* Là hàm mất mát đo khoảng cách trung bình tuyệt đối giữa ảnh màu sinh ra và ảnh màu gốc.
* Khác với Adversarial Loss (tập trung vào sự "thật"), L1 Loss đảm bảo độ chính xác về màu sắc và hình dạng.
* Sử dụng L1 thay vì L2 (MSE) giúp ảnh không bị mờ, giữ được màu sắc mượt mà hơn.

**Công thức:**

**A black text with a white background

AI-generated content may be incorrect.**

**Tổng Loss:**

Trong Pix2Pix, hai hàm mất mát trên được kết hợp lại để tối ưu hóa Generator:

A black text on a white background

AI-generated content may be incorrect.

Trong đó:

* λ là hệ số điều chỉnh, thường chọn giá trị như 100, để cân bằng giữa độ chân thực (GAN Loss) và độ chính xác (L1 Loss).

# 4. THUẬT TOÁN TRIỂN KHAI

## 4.1. Xử lí dữ liệu

### 4.1.1. 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.

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.

### 4.1.2 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.

**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.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 3.2.4.1. Cặp ảnh ví dụ của tập train*

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

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 3.2.4.1. 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.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 3.2.4.1. Cặp ảnh ví dụ của tập test*

## 4.2 Thuật toán triển khai với mô hình Pix2Pix

### 4.2.1 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 (bên trái)**: có nhiệm vụ trích xuất đặc trưng từ ảnh đen trắng.
* **Decoder (bên phải)**: 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.

A screen shot of a computer code

AI-generated content may be incorrect.

* 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.

A screen shot of a computer program

AI-generated content may be incorrect.

* 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 × 5.
  + 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.

### 4.2.2 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:



* **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.

### 4.2.3 Hàm mất mát (Loss Funtion)

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**

Generator được huấn luyện với hai mục tiêu chính:

* Sinh ra ảnh màu có khả năng “đánh lừa” được Discriminator.
* Đảm bảo ảnh sinh ra càng giống ảnh thật càng tốt.

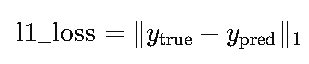
Do đó, hàm mất mát của Generator là tổng của hai thành phần:

* + **Adversarial Loss** (GAN Loss):  
    Sử dụng hàm **Binary Crossentropy** giữa đầu ra của Discriminator và nhãn "thật" (1) cho ảnh sinh ra.

A black text on a white background

AI-generated content may be incorrect.

* + **L1 Loss** (Mean Absolute Error):  
    Đo khoảng cách tuyệt đối trung bình giữa ảnh sinh ra và ảnh màu thật:



* + **Tổng loss** của Generator:  
    
* Trong đó, **λ (LAMBDA)** được đặt là 100 để ưu tiên độ chính xác màu sắc.
* **Hàm mất mát cho Discriminaor**

Discriminator được huấn luyện để phân biệt:

* Ảnh màu thật: nên dự đoán là 1.
* Ảnh màu giả (do Generator sinh ra): nên dự đoán là 0.

Gồm hai thành phần:

* **real\_loss**: BCE giữa ảnh thật và nhãn 1.
* **generated\_loss**: BCE giữa ảnh giả và nhãn 0.

**Tổng Loss**:

A black text with a plus

AI-generated content may be incorrect.

# 5. CHƯƠNG TRÌNH VÀ KẾT QUẢ

## 5.1. 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.

## 5.2. 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 = 0.5.
* 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.

## 5.3. 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 4.Kiểm thử và phương pháp kiểm thử*

## 5.4 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.

## 5.5 Kết quả thực thi chương trình

### 5.5.1 Giao diện chính của chương trình

**A screenshot of a black and white photo

AI-generated content may be incorrect.**

Hình 5.5.1. Giao diện chính của chương trình

### 5.5.2 Kết quả thực thi của 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 4.Kiểm thử và phương pháp kiểm thử*

*A screenshot of a computer

AI-generated content may be incorrect.*

Hình 5.5.2.3. Kết quả phục chế ảnh 1

*A screenshot of a screenshot of a forest

AI-generated content may be incorrect.*

Hình 5.5.2.4. Kết quả phục chế ảnh 2

A screenshot of a screenshot of a picture

AI-generated content may be incorrect.

Hình 5.5.2.4. Kết quả phục chế ảnh 3

### 5.5.3. 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.

# 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1. Kết luận

Trong đồ án này, chúng em 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.

Chúng em 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, chúng em 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.

## 6.2. 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](https://arxiv.org/abs/1611.07004). *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](https://arxiv.org/abs/1505.04597). *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](https://arxiv.org/abs/1512.03385). *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] TensorFlow Pix2Pix Tutorial. [pix2pix: Image-to-image translation with a conditional GAN  |  TensorFlow Core](https://www.tensorflow.org/tutorials/generative/pix2pix/).

[6] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[7] Kaggle Dataset: Landscape Image Colorization. https://www.kaggle.com/.

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

*.*

**Bảng phân công nhiệm vụ**

|  |  |  |
| --- | --- | --- |
| **Họ tên** | **Nhiệm vụ** | **Tình trạng** |
| 1.Nguyễn Khắc Nhân Tâm | + Thu thập dataset  + Tiền xử lý dữ liệu ảnh  + Xây dựng mô hình.  + Huấn luyện và tối ưu mô hình  + Triển khai mô hình.  + Viết báo cáo. | **Đã hoàn thành** |