HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO THỰC TẬP CƠ SỞ

ĐỀ TÀI:TỔNG HỢP VÀ CHỈNH SỬA ẢNH DỰA TRÊN MÔ HÌNH ANYCOST GAN

Giảng viên hướng dẫn : **Ths. Bùi Văn Kiên**
Sinh viên : **Nguyễn Thanh Tùng**
MSV : **B21DCCN771**

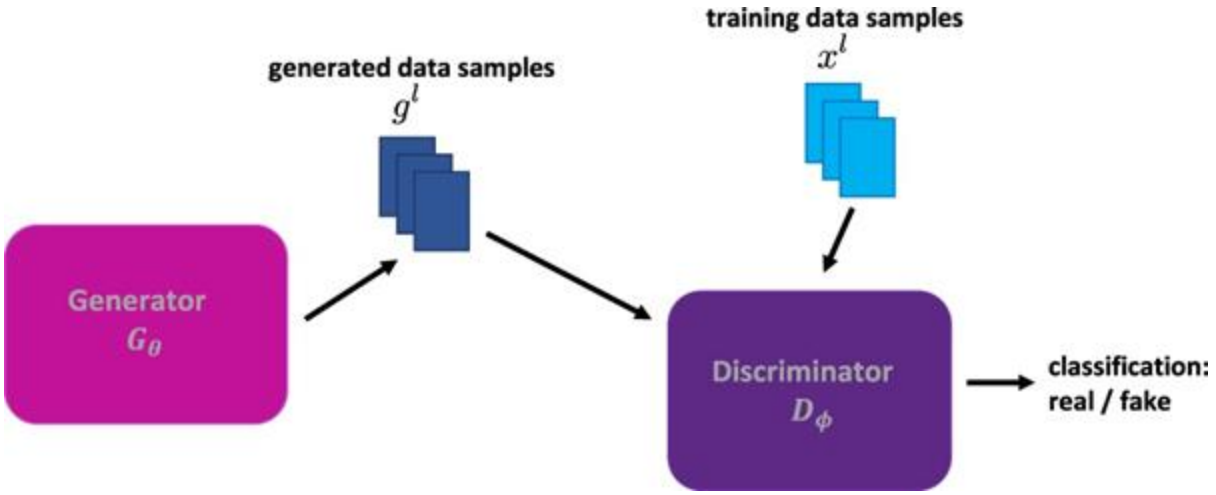
Hà Nội – 2025

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Trong những năm gần đây, sự phát triển của trí tuệ nhân tạo (AI) và học sâu (deep learning) đã mang đến những bước tiến vượt bậc trong nhiều lĩnh vực, đặc biệt là trong xử lý và tạo dựng hình ảnh. Một trong những xu hướng nổi bật là việc sử dụng Mạng Sinh Generative Adversarial Networks (GANs) để tạo ra các hình ảnh mới hoặc chỉnh sửa các bức ảnh hiện có một cách tự động. Trong số các mô hình GAN tiên tiến, Anycost GAN đã trở thành một trong những giải pháp tiên phong, nổi bật với khả năng tổng hợp và chỉnh sửa ảnh hiệu quả ngay cả khi có tài nguyên tính toán hạn chế. Dự án này chúng em sẽ đi sâu vào việc áp dụng mô hình Anycost GAN để tổng hợp và chỉnh sửa ảnh cũng như là khám phá những lợi ích mà mô hình này mang lại

1.1. Tổng quan về GAN và Anycost GAN

Mạng đối nghịch tạo sinh (Generative Adversarial Networks -GAN) là một mô hình học sâu nổi bật, hoạt động dựa trên nguyên tắc đối kháng giữa 2 mạng nơ ron nhân tạo:



- Mạng tạo sinh (Generator - G): Có nhiệm vụ tạo ra dữ liệu giả nhưng trông giống dữ liệu thật.
- Mạng phân biệt (Discriminator - D): Có nhiệm vụ phân biệt dữ liệu thật từ dữ liệu giả do Generator tạo ra.

Hai mạng này được huấn luyện đồng thời trong một quá trình liên tục, Generator cố gắng đánh lừa Discriminator, trong khi Discriminator cố gắng phân biệt chính xác dữ liệu thật và dữ liệu giả. Qua thời gian, Generator sẽ cải thiện khả năng tạo dữ liệu sao cho ngày càng chân thực hơn.

GAN có thể tạo ra các hình ảnh cực kỳ chân thực từ dữ liệu đầu vào, điều này đã mở ra nhiều cơ hội mới trong các ứng dụng như tổng hợp ảnh, cải tiến chất lượng hình ảnh, và các tác vụ sáng tạo khác. Tuy nhiên, một trong những thách thức lớn khi áp dụng GAN trong thực tế là yêu cầu tài nguyên tính toán rất lớn, đặc biệt khi tạo ra các hình ảnh có độ phân giải cao.

Anycost GAN là một mô hình cải tiến từ GAN truyền thống, với mục tiêu tối ưu hóa chi phí tính toán trong quá trình huấn luyện và suy luận mà vẫn duy trì chất lượng hình ảnh ở mức cao. Bằng cách điều chỉnh và tối ưu hóa các tham số tính toán, Anycost GAN có thể hoạt động hiệu quả ngay cả trên các thiết bị với tài nguyên phần cứng hạn chế, từ đó mở rộng khả năng ứng dụng mô hình trong nhiều môi trường thực tế, như thiết bị di động hoặc các ứng dụng đám mây với chi phí tính toán tiết kiệm.

1.2. Cơ chế hoạt động

Anycost GAN sử dụng hai cơ chế chính để tối ưu hóa quá trình huấn luyện và tạo ảnh:

- **Huấn luyện đa độ phân giải (multi-resolution training)**: Cơ chế này cho phép mô hình học cách tạo ảnh ở nhiều mức độ phân giải khác nhau. Thay vì huấn luyện mô hình chỉ với một độ phân giải cố định, Anycost GAN có thể huấn luyện trên nhiều mức phân giải và linh hoạt thay đổi giữa các mức phân giải khi cần thiết. Điều này giúp giảm thời gian huấn luyện và tăng tốc độ tạo ảnh mà không làm giảm chất lượng hình ảnh.
- **Điều chỉnh kênh thích ứng (adaptive channel pruning):** Một trong những cải tiến quan trọng của Anycost GAN là khả năng tự động điều chỉnh số lượng kênh tính toán trong mạng dựa trên tài nguyên phần cứng hiện có. Điều này giúp mô hình hoạt động hiệu quả hơn trên các thiết bị có phần cứng hạn chế, đồng thời vẫn duy trì được chất lượng hình ảnh đầu ra ở mức chấp nhận được.

  Các cơ chế này không chỉ giúp tối ưu hóa tài nguyên tính toán mà còn làm cho mô hình linh hoạt hơn khi triển khai trên các thiết bị có phần cứng khác nhau, từ các máy tính để bàn mạnh mẽ cho đến các thiết bị di động có cấu hình thấp.

1.3. Ưu điểm và ứng dụng

- **Xem trước nhanh và hiệu quả**:Người dùng có thể xem trước hình ảnh ở độ phân giải thấp mà không tốn nhiều tài nguyên tính toán. Sau đó, mô hình sẽ tạo phiên bản cuối cùng với độ phân giải cao nhất, giúp tiết kiệm thời gian và tài nguyên trong quá trình tạo ảnh.

- **Tích hợp dễ dàng vào các ứng dụng thực tế**:Mô hình này hỗ trợ nhiều kiến trúc huấn luyện sẵn, cho phép triển khai nhanh chóng vào các ứng dụng thực tế như chỉnh sửa ảnh, tạo ảnh từ mô tả, và nâng cấp chất lượng ảnh. Điều này làm

cho Anycost GAN trở thành một công cụ hữu ích cho các nhà phát triển trong việc tạo ra các sản phẩm ứng dụng mạnh mẽ.

- **Ứng dụng đa dạng**:
  Chỉnh sửa ảnh chân dung: Anycost GAN có thể sử dụng để chỉnh sửa các bức ảnh chân dung, ví dụ như thay đổi các đặc điểm như nụ cười, màu tóc, hoặc thêm các chi tiết trang điểm.

  Tạo ảnh từ dữ liệu đầu vào: Mô hình có thể sinh ra các bức ảnh từ dữ liệu đầu vào có sẵn, ví dụ như từ các mô tả văn bản hoặc dữ liệu đầu vào hình ảnh.

  Nâng cấp chất lượng ảnh: Anycost GAN giúp nâng cấp độ phân giải và chi tiết của các bức ảnh có độ phân giải thấp, cải thiện chất lượng ảnh trong các ứng dụng đòi hỏi hình ảnh rõ nét.

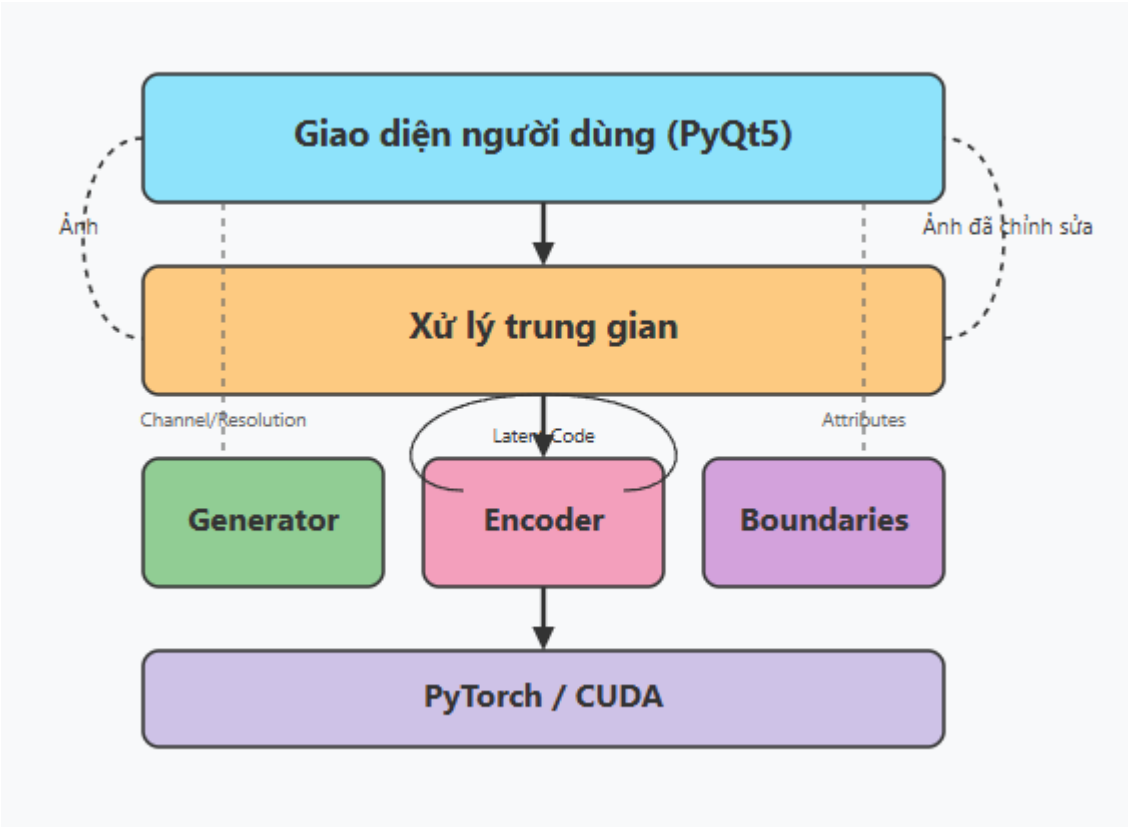
1.4. Đánh giá chất lượng

Để đánh giá chất lượng hình ảnh tạo ra từ Anycost GAN, hệ thống tích hợp các chỉ số đánh giá phổ biến như:

- **Fréchet Inception Distance (FID):** Đánh giá độ chân thực của hình ảnh.
- **Perceptual Path Length (PPL**): Kiểm tra mức độ ổn định và khả năng điều khiển mô hình

CHƯƠNG 2. KIẾN TRÚC HỆ THỐNG

2.1. Kiến trúc tổng quan



2.2. Giao diện người dùng (PyQt5)

- Hiển thị ảnh gốc và ảnh sau chỉnh sửa.
- Cung cấp thanh trượt để điều chỉnh các thuộc tính khuôn mặt như biểu cảm, màu tóc, độ rộng mắt,...
- Gồm các nút chức năng như Reset, Finalize, giúp người dùng kiểm soát quá trình chỉnh sửa.

2.3. Lớp xử lý trung gian (Middleware)

- Điều phối toàn bộ luồng xử lý giữa giao diện và mô hình GAN.
- Sử dụng các lớp như FaceEditor và Worker để mã hóa ảnh đầu vào, cập nhật latent code theo điều chỉnh của người dùng.

- Quản lý luồng đa nhiệm thông qua QThreadPool, đảm bảo ứng dụng luôn phản hồi nhanh.

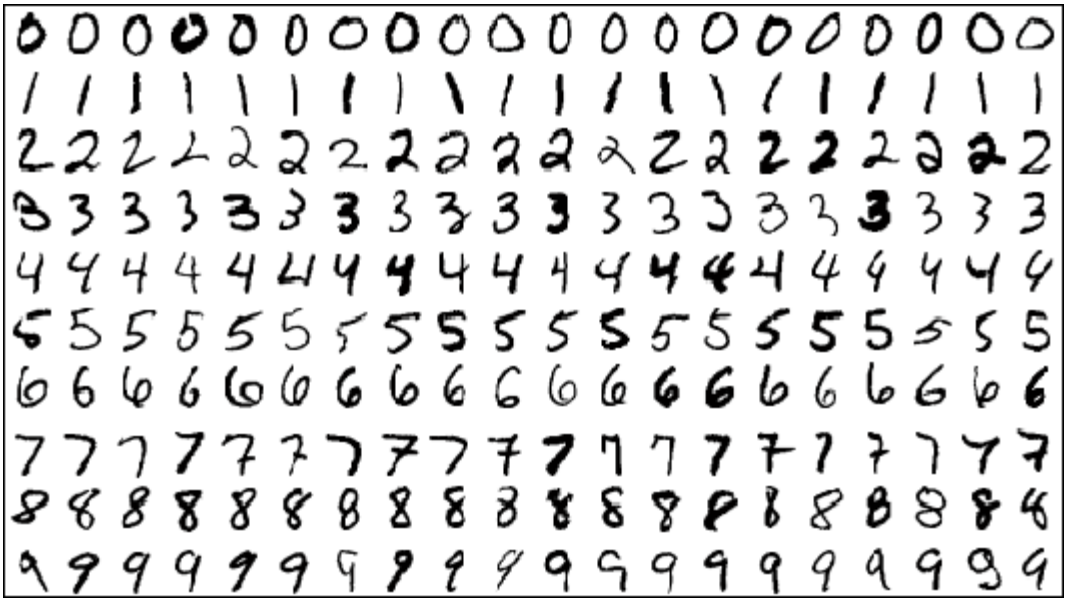
❖ Mô hình học sâu

1. Cấu trúc mạng

➢ **Generator**

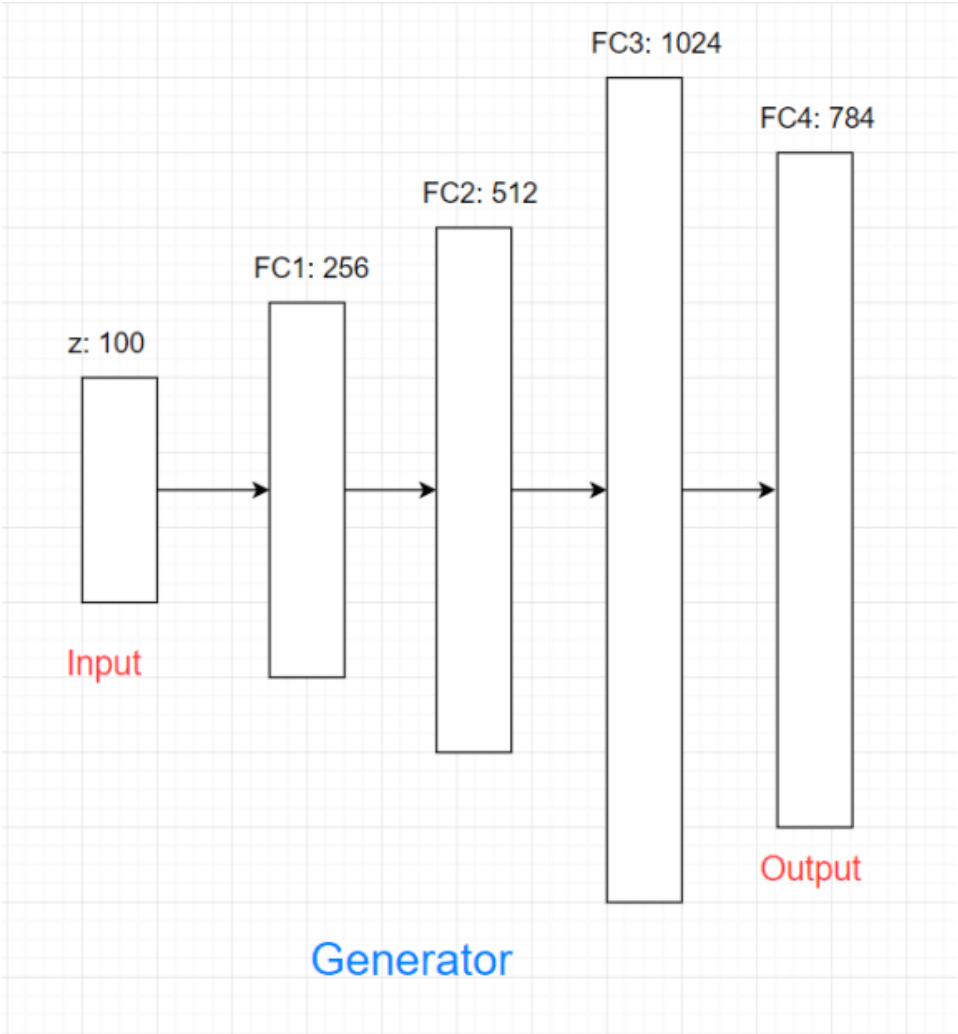
Generator là mạng sinh ra dữ liệu, tức là sinh ra các chữ số giống với dữ liệu trong MNIST dataset. Generator có input là noise (random vector) là output là chữ số.

Input của Generator là noise để khi ta thay đổi noise ngẫu nhiên thì Generator có thể sinh ra một biến dạng khác của chữ viết tay. Noise cho Generator thường được sinh ra từ normal distribution hoặc uniform distribution.



Input của Generator là noise vector 100 chiều.

Sau đấy mô hình neural network được áp dụng với số node trong hidden layer lần lượt là 256, 512, 1024.`

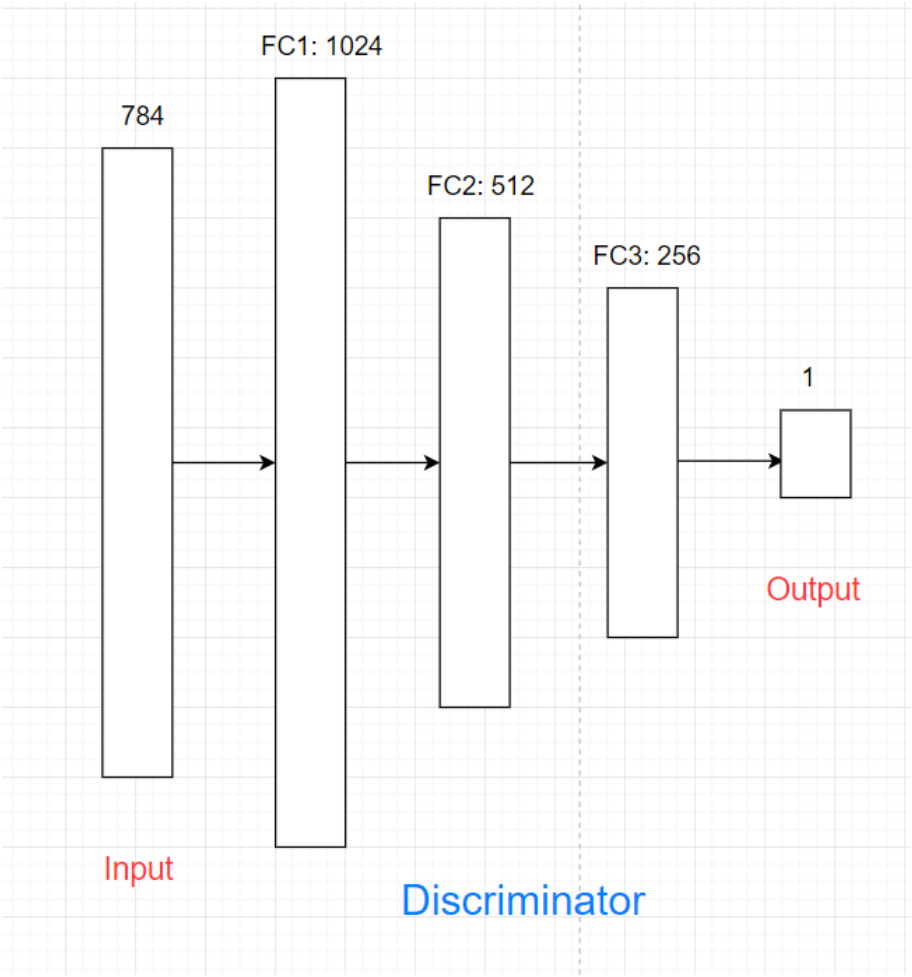


Output layer có số chiều là 784, vì output đầu ra là ảnh giống với dữ liệu MNIST, ảnh xám kích thước 28*28 (784 pixel).

Output là vector kích thước 784*1 sẽ được reshape về 28*28 đúng định dạng của dữ liệu MNIST.

- **Discriminator**

  Discriminator là mạng để phân biệt xem dữ liệu là thật (dữ liệu từ dataset) hay giả (dữ liệu sinh ra từ Generator). Trong bài toán này thì discriminator dùng để phân biệt chữ số từ bộ MNIST và dữ liệu sinh ra từ Generator. Discriminator có input là ảnh biểu diễn bằng 784 chiều, output là ảnh thật hay ảnh giả.



Input của Discriminator là ảnh kích thước 784 chiều.

Sau đấy mô hình neural network được áp dụng với số node trong hidden layer lần lượt là 1024, 512, 256. Mô hình đối xứng lại với Generator.

Output là 1 node thể hiện xác suất ảnh input là ảnh thật, hàm sigmoid được sử dụng.

2. Encoder

ResNet (Residual Network) giải quyết vấn đề mô hình học sâu càng nhiều lớp thì càng khó huấn luyện, bằng cách đưa vào "kết nối tắt" (skip connections) hoặc residual connections.

Thay vì học một ánh xạ trực tiếp H(x), ResNet học phần hiệu F(x) = H(x) - x, nên đầu ra sẽ là:

$$H(x) = F(x) + x$$

*** Cấu trúc ResNet-50**

- 50 lớp sâu, chia thành:
  - 1 lớp conv đầu (7×7)
  - 4 khối chính (residual blocks), mỗi khối gồm nhiều "bottleneck blocks":
    - Block1: 3 bottlenecks
    - Block2: 4 bottlenecks
    - Block3: 6 bottlenecks
    - Block4: 3 bottlenecks
- Tổng số lớp trọng số: 49 conv + 1 fully-connected = 50 lớp

Mỗi **bottleneck block** có 3 lớp:

- Conv 1×1 (giảm chiều)
- Conv 3×3 (xử lý chính)
- Conv 1×1 (tăng chiều)
  → Cộng thêm kết nối tắt: output = x + F(x)

3. Boundaries

Trong không gian này, các chiều không gian không chỉ là dữ liệu vô nghĩa – chúng có thể liên quan đến các thuộc tính cụ thể như:

- Tuổi (age)
- Giới tính (gender)
- Đeo kính / không (eyeglasses
- Biểu cảm khuôn mặt (smile, angry,...)
- Kiểu tóc, góc mặt, ánh sáng, v.v.

Boundary là một mặt phẳng hoặc hyperplane trong không gian latent, chia tách không gian thành 2 phần:

- Một bên là ảnh có thuộc tính A (ví dụ: có kính)
- Bên còn lại là ảnh không có A (ví dụ: không kính)

Khi bạn có latent vector w (đại diện cho 1 khuôn mặt), và 1 boundary vector b (ví dụ ranh giới giữa “già” và “trẻ”), bạn có thể điều chỉnh thuộc tính ảnh bằng cách dịch latent vector dọc theo boundary đó:

$$w' = w + \alpha \cdot b$$

- alpha > 0: tăng độ "già"
- alpha < 0: tăng độ "trẻ"

Sau đó đưa w' vào mô hình sinh ảnh → kết quả là khuôn mặt mới đã thay đổi thuộc tính.

2.3. Mô hình hoạt động tổng quát

- Chứa các mô hình chính: Generator, Encoder và Boundaries
- Thực hiện các phép tính toán chuyên sâu để tạo và chỉnh sửa ảnh

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

3.1.Ngôn ngữ lập trình Python

Hệ thống sử dụng ngôn ngữ lập trình Python

3.2. PyTorch

PyTorch là thư viện mã nguồn mở dùng cho Machine Learning và Deep Learning, được phát triển bởi Facebook AI Research (FAIR). Trong Anycost GAN, PyTorch được sử dụng để xây dựng và huấn luyện mô hình GAN, đồng thời cung cấp các API giúp quản lý tensor, huấn luyện mô hình trên GPU và tối ưu quá trình cập nhật tham số một cách hiệu quả. Nhờ PyTorch, hệ thống có thể tính toán linh hoạt và tăng tốc huấn luyện nhờ CUDA.

3.3. CUDA/cuDNN

CUDA (Compute Unified Device Architecture) là nền tảng tính toán song song của NVIDIA, cho phép xử lý nhanh chóng trên GPU. CUDA giúp tăng tốc huấn luyện bằng cách chạy nhiều lớp neuron song song trên GPU, giúp quá trình sinh ảnh trực quan nhanh hơn. CUDA cũng được sử dụng để giảm tải CPU và xử lý tensor một cách hiệu quả.

3.4. Horovod

Horovod là một thư viện giúp tối ưu hóa huấn luyện mô hình deep learning trên nhiều GPU, giúp giảm thời gian huấn luyện bằng cách phân tán dữ liệu. Horovod giúp hỗ trợ huấn luyện phân tán bằng cách chạy các script với nhiều GPU và giảm thời gian tính toán bằng cách tối ưu hóa giao tiếp giữa các GPU.

3.5. NumPy

NumPy là một thư viện hỗ trợ tính toán số học hiệu suất cao với mảng đa chiều. Numpy dùng để xử lý dữ liệu đầu vào trước khi đưa vào mô hình và hỗ trợ các phép toán ma trận và tensor trong huấn luyện mạng nơ-ron.

3.6. Torchvision

Torchvision là một thư viện đi kèm với PyTorch, cung cấp các công cụ và dataset hỗ trợ xử lý ảnh. Torchvision dùng để tải và tiền xử lý các tập dữ liệu như FFHQ, CelebA-HQ, LSUN Car và cung cấp các phương pháp biến đổi ảnh cần thiết cho huấn luyện mô hình.

CHƯƠNG 4. TRIỂN KHAI MÔ HÌNH

4.1. Triển Khai Mô Hình GAN

4.1.1. Bộ Sinh Ảnh (Generator)

- Bộ sinh ảnh (Generator) đóng vai trò trung tâm trong mô hình GAN, có nhiệm vụ tạo ra hình ảnh mới từ vector ngẫu nhiên (latent vector). Trong Anycost GAN, Generator có khả năng tùy biến cao, hỗ trợ nhiều cấu hình để đáp ứng nhu cầu sinh ảnh với độ phân giải và chi tiết khác nhau
    - Hỗ trợ nhiều cấu hình khác nhau:
        - anycost-ffhq-config-f: Sinh ảnh khuôn mặt người dựa trên tập dữ liệu FFHQ.
        - anycost-ffhq-config-f-flexible: Phiên bản linh hoạt hơn, cho phép thay đổi kiến trúc trong lúc huấn luyện hoặc sinh ảnh
        - stylegan2-ffhq-config-f: Cấu hình gốc từ StyleGAN2, dùng làm tham chiếu
        - anycost-car-config-f: Dùng để sinh ảnh ô tô từ tập dữ liệu LSUN Car
    - Các trọng số (pre-trained weights) được cung cấp dưới dạng URL, sau đó được tải về và load trực tiếp vào mô hình thông qua PyTorch.

❖ **Các tham số chính trong Genenerator:**

- resolution: Độ phân giải của hình ảnh sinh ra.
- style_dim: Kích thước vector phong cách (thường là 512).
- n_mlp: Số lượng fully connected layer trong mạng MLP phong cách.
- channel_multiplier: Hằng số nhân kênh, quản lý số kênh tại các tầng khác nhau.
- blur_kernel: Nhân lọc mà Generator sử dụng.
- act_func: Hàm kích hoạt sử dụng trong các lớp.

❖ **Các bước hoạt động:**

1. Chuẩn bị style code: Sử dụng MLP nhằm chuyển đổi latent vector (z) sang phong cách (w).

2. Sinh noise ngẫu nhiên: Sinh noise để bổ sung các chi tiết tinh tế vào hình ảnh.

3. Khởi tạo constant input: Ban đầu, Generator dùng tập constant input (thay vì random noise như các GAN truyền thống).

4. Xây dựng các tầng conv: Gồm các lớp StyledConv, ToRGB, và Upsample để tạo hình ảnh đạt độ phân giải mong muốn.

5. Sinh hình ảnh: Qua dàn layer conv, cuối cùng sinh ra hình ảnh RGB.

4.1.2. Bộ phân biệt (Discriminator)

Discriminator nhận ảnh đầu vào (có thể là ảnh thật hoặc ảnh sinh ra bởi Generator), và đưa ra dự đoán xem ảnh đó là "thật" hay "giả".

❖ **Các tham số chính:**

- resolution: Kích thước hình ảnh nhận vào.
- channel_multiplier: Hằng số nhân kênh.
- blur_kernel: Nhân lọc.
- act_func: Hàm kích hoạt.
- n_res: Số tầng resblock.
- modulate: Tùy chọn bật/tắt g_arch modulation.

❖ **Các bước hoạt động:**

1. Nhận ảnh đầu vào: Discriminator nhận hình ảnh và tầng convolution đầu tiên xử lý tín hiệu.

2. Xây dựng các ResBlock: Gồm nhiều tầng ResBlock để giảm kích thước hình ảnh và tăng chi tiết quan trọng.

3. Thêm minibatch discrimination: Kỹ thuật giúp phân biệt được tạo ra bởi Generator hay hình thật.

4. Fully Connected Layer: Cuối cùng, lớp fully connected sắp xếp dữ liệu và phân loại.

4.1.3. Bộ Mã Hóa (Encoder)

- Dựa trên kiến trúc ResNet50.
- Được thiết kế để trích xuất mã đặc trưng từ ảnh đầu vào.
- Huấn luyện với VGG LPIPS loss để đảm bảo chất lượng mã hóa phù hợp với StyleGAN2.

❖ **Phương Thức forward**

- Đầu Vào: Hình ảnh đầu vào với kích thước (batch_size, channels, height, width), yêu cầu height và width phải là 256.
- Quy Trình Xử Lý:

  1. Convolution và Batch Normalization: Hình ảnh đầu vào được đưa qua lớp conv1, bn1, relu, và maxpool để trích xuất các đặc trưng cơ bản.

  2. Các Khối Residual: Đặc trưng được đưa qua các khối layer1, layer2, layer3, và layer4 để trích xuất các đặc trưng sâu hơn.

  3. Pooling và Flatten: Đặc trưng được đưa qua lớp avgpool để giảm kích thước không gian xuống 1x1, sau đó được làm phẳng (flatten) thành vector.

  4. Fully Connected Layer: Vector đặc trưng được đưa qua lớp fc để ánh xạ thành latent code.

  5. Cộng Giá Trị Trung Bình: Latent code được cộng thêm giá trị trung bình mean_latent để tạo ra style code cuối cùng.

- Đầu Ra: Tensor có kích thước (batch_size, n_style, style_dim), chứa các style code.

❖ **Cách Thức Hoạt Động**

- Mã Hóa Hình Ảnh:

● Hình ảnh đầu vào được mã hóa thành các đặc trưng trung gian thông qua các lớp convolution và residual blocks của ResNet50.

- Các đặc trưng này sau đó được ánh xạ thành latent code thông qua lớp fully connected.
- Tạo Style Code: Latent code được điều chỉnh bằng cách cộng thêm giá trị trung bình mean latent, tạo ra các style code có thể được sử dụng trong các mô hình sinh.

4.1.4. Các lớp xử lý ảnh trong Mạng Nơ-ron Nhân Tạo

❖ **PixelNorm**

- Mục Đích: Chuẩn hóa các giá trị pixel trong tensor đầu vào.
- Phương Thức: forward(x): Chuẩn hóa tensor x bằng cách chia cho căn bậc hai của trung bình bình phương các giá trị pixel.

❖ **ConstantInput**

- Mục Đích: Tạo một đầu vào cố định cho mô hình.
- Phương Thức: forward(batch): Tạo một tensor cố định và lặp lại nó theo kích thước batch.

❖ **NoiseInjection**

- Mục Đích: Thêm noise ngẫu nhiên vào tensor đầu vào.
- Phương Thức: forward(image, noise=None): Thêm noise vào tensor image. Nếu noise không được cung cấp, nó sẽ được tạo ngẫu nhiên.

❖ **Upsample**

- Mục Đích: Thực hiện upsampling trên tensor đầu vào.
- Phương Thức: forward(x): Upsample tensor x bằng cách sử dụng kernel đã được đăng ký.

❖ **Blur**

- Mục Đích: Áp dụng phép làm mờ (blur) lên tensor đầu vào.
- Phương Thức: forward(x): Làm mờ tensor x bằng cách sử dụng kernel đã được đăng ký.

❖ **EqualConv2d**

- Mục Đích: Lớp convolution với trọng số được chuẩn hóa.
- Phương Thức: forward(x): Áp dụng phép convolution lên tensor x với trọng số đã được chuẩn hóa.

❖ **EqualLinear**

- Mục Đích: Lớp fully connected với trọng số được chuẩn hóa.

- Phương Thức: forward(x): Áp dụng phép linear transformation lên tensor x với trọng số đã được chuẩn hóa.

❖ **ModulatedConv2d**

- Mục Đích: Lớp convolution có điều chế (modulated convolution), sử dụng style code để điều chỉnh trọng số.
- Phương Thức: forward(x, style): Áp dụng phép convolution có điều chế lên tensor x với style code style.

❖ **StyledConv**

- Mục Đích: Kết hợp ModulatedConv2d và NoiseInjection để tạo ra các đặc trưng có style.
- Phương Thức: forward(x, style, noise=None): Áp dụng convolution có điều chế, thêm noise, và kích hoạt bằng hàm kích hoạt (activation function).

❖ **ToRGB**

- Mục Đích: Chuyển đổi đặc trưng thành ảnh RGB.
- Phương Thức: forward(x, style, skip=None): Chuyển đổi tensor x thành ảnh RGB và cộng thêm tensor skip nếu có.

❖ **AdaptiveModulate**

- Mục Đích: Điều chế tensor đầu vào dựa trên kiến trúc mạng (g_arch).
- Phương Thức: forward(x, g_arch): Điều chế tensor x dựa trên g_arch.

❖ **ConvLayer**

- Mục Đích: Lớp convolution kết hợp các phép toán như convolution, blur, và activation.
- Phương Thức: forward(x, g_arch=None): Áp dụng các phép toán convolution, blur, và activation lên tensor x.

❖ **ResBlock**

- Mục Đích: Khối residual kết hợp hai lớp convolution và một skip connection.
- Phương Thức: forward(x, g_arch=None): Áp dụng hai lớp convolution và cộng thêm skip connection.

❖ **Cách Thức Hoạt Động**

1. Chuẩn Hóa và Điều Chế: Các lớp như PixelNorm và ModulatedConv2d giúp chuẩn hóa và điều chế các giá trị trong tensor đầu vào, giúp mô hình ổn định hơn trong quá trình huấn luyện.

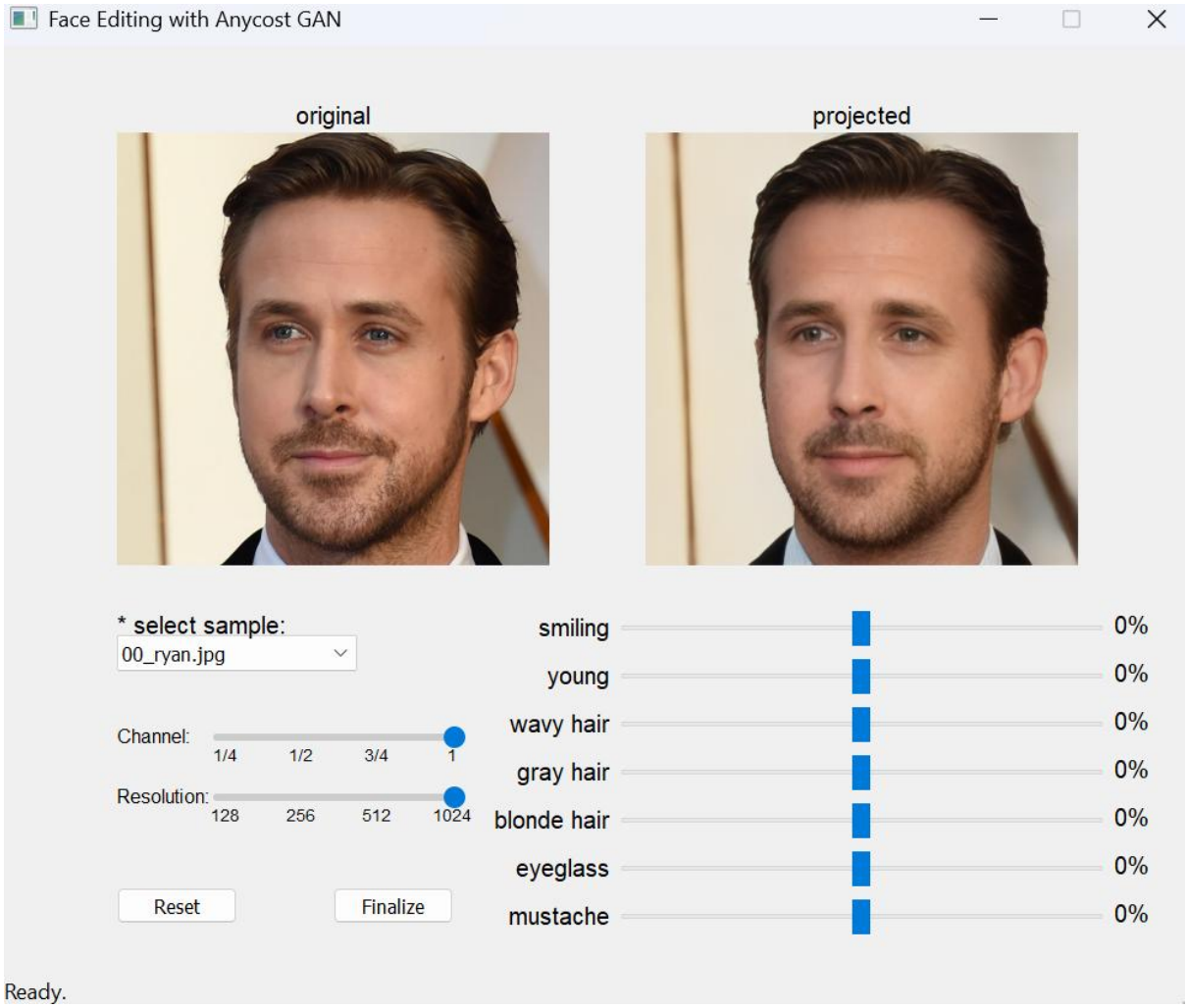
2. Thêm Noise: Lớp NoiseInjection thêm noise ngẫu nhiên vào tensor, giúp tạo ra sự đa dạng trong kết quả đầu ra.

3. Upsampling và Blur: Các lớp Upsample và Blur giúp tăng kích thước và làm mờ tensor, giúp tạo ra các hình ảnh có độ phân giải cao và mượt mà hơn.

4. Residual Blocks: Các khối residual giúp mô hình học các đặc trưng sâu hơn mà không bị mất thông tin.

4.2. Triển Khai Ứng Dụng Chỉnh Sửa Khuôn Mặt Sử Dụng Anycost GAN

DEMO



1. Ảnh khuôn mặt

- Bên trái (original): Ảnh gốc được tải lên
- Bên phải (projected): Ảnh đã được "chiếu" (projected) vào không gian latent của StyleGAN. Đây là kết quả của việc encoder chuyển đổi ảnh gốc thành latent code và generator tạo lại ảnh từ latent code đó.

2. Thanh chọn mẫu

- "select sample": Dropdown menu để chọn ảnh mẫu có sẵn
- "Tải ảnh": Nút đã được thêm vào cho phép người dùng tải ảnh riêng của họ lên thay vì sử dụng ảnh mẫu.

3. Các thanh điều chỉnh

Các thuộc tính khuôn mặt

- smiling: Điều chỉnh độ cười
- young: Điều chỉnh độ trẻ/già

- wavy hair: Điều chỉnh độ gợn sóng của tóc
- gray hair: Điều chỉnh độ bạc của tóc
- blonde hair: Điều chỉnh độ vàng của tóc
- eyeglass: Điều chỉnh kính mắt
- mustache: Điều chỉnh râu mép

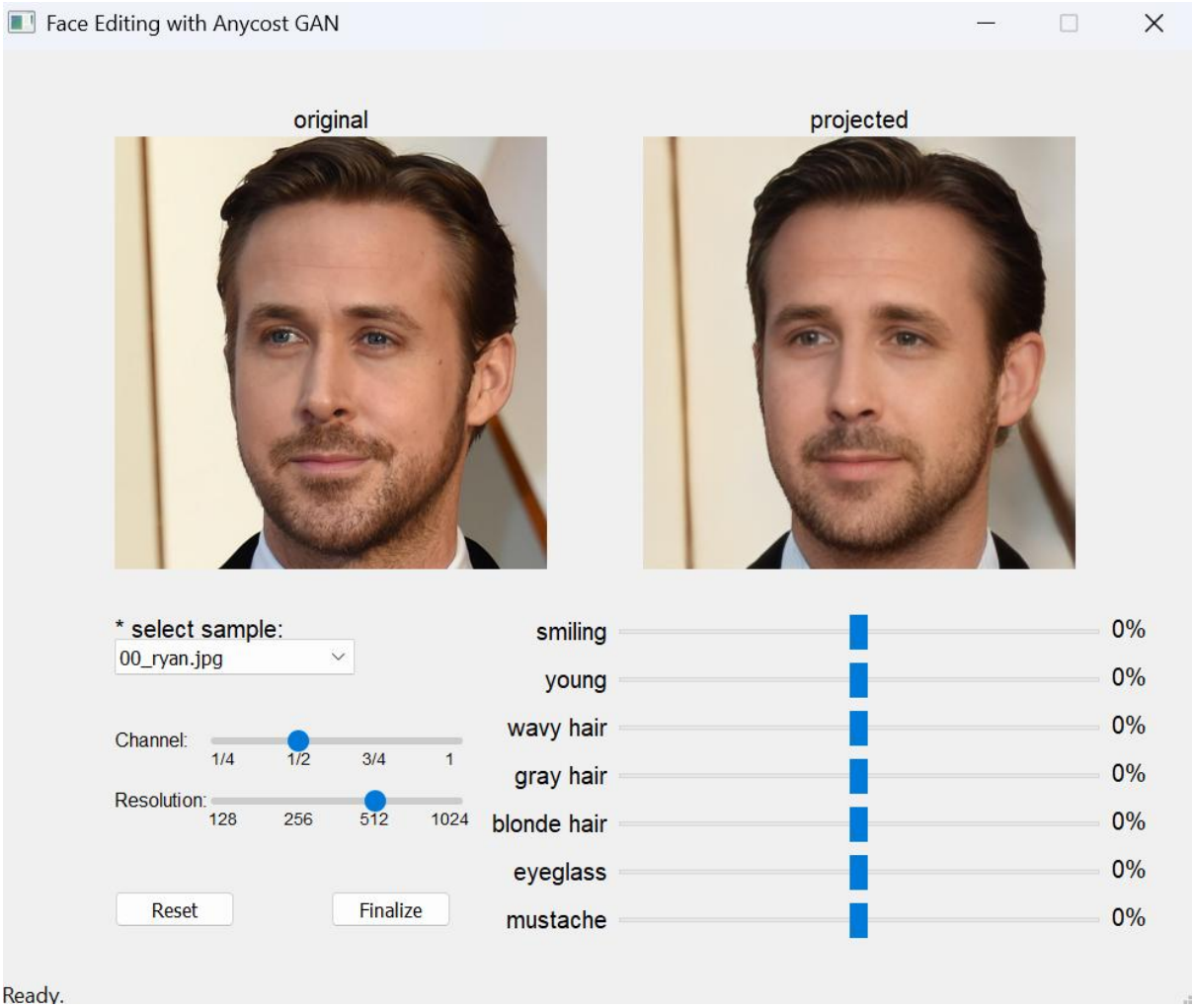
Các thanh này đều đang ở vị trí gần về phía bên phải, cho thấy các thuộc tính đang được điều chỉnh ở mức độ dương (tăng) nhẹ.

Tham số hiệu suất

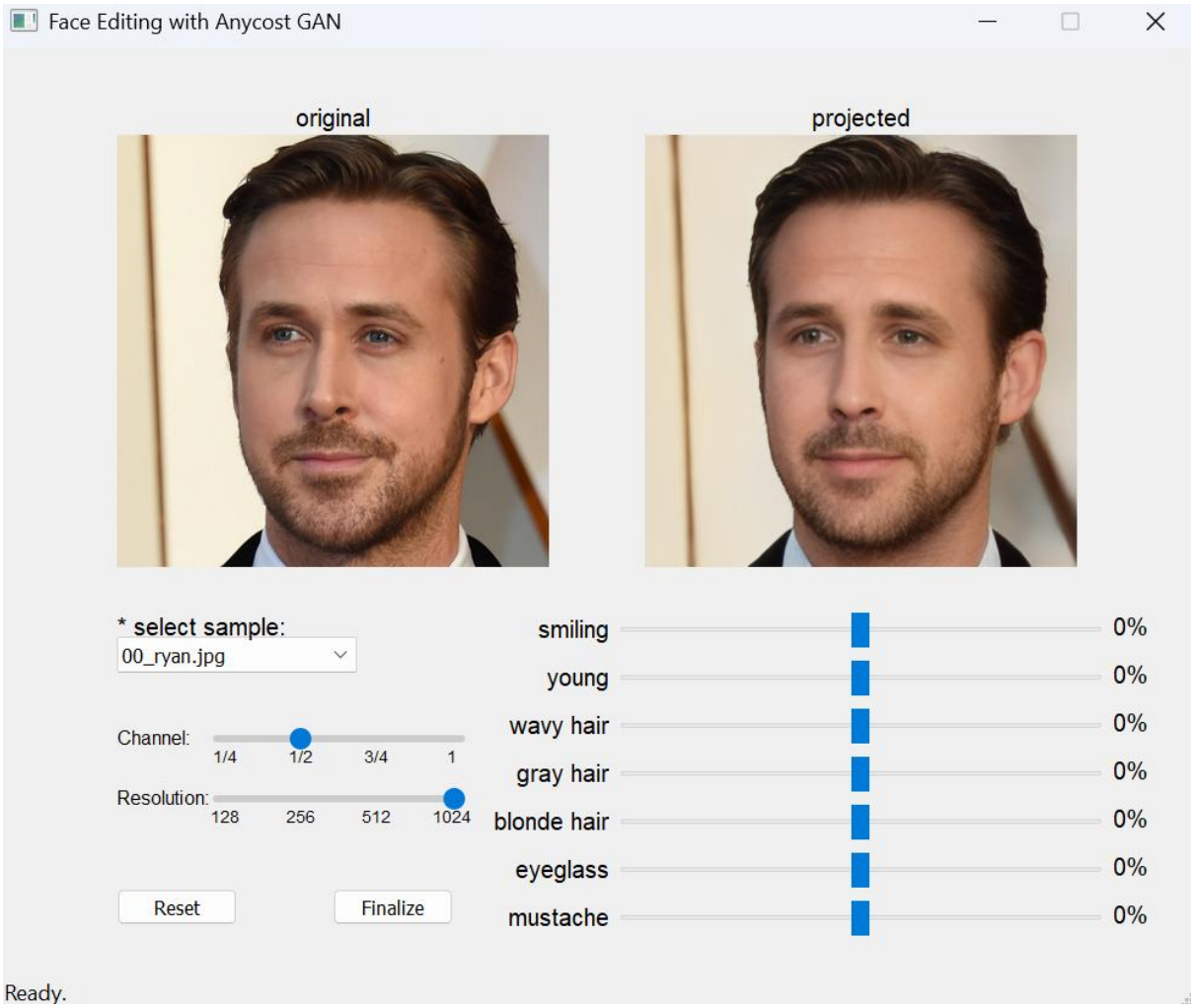
- channel: Điều chỉnh tỷ lệ kênh từ 1/4 đến 1
- resolution: Độ phân giải từ 128px đến 1024px

4. Các nút chức năng

- Reset: Đặt lại tất cả các thanh trượt về vị trí ban đầu
- Finalize: Tạo hình ảnh cuối cùng với chất lượng tốt nhất



Điều chỉnh kênh sẽ giúp tốc độ tính toán nhanh hơn và đỡ tốn tài nguyên hơn, có thể phù hợp với một số máy cấu hình yếu khi điều chỉnh kênh thấp



Thay đổi độ phân giải sẽ quyết định về độ nét của ảnh

Độ phân giải cao:

- Yêu cầu nhiều tài nguyên GPU/CPU
- Xử lý chậm hơn
- Cần nhiều bộ nhớ hơn

Độ phân giải thấp:

- Yêu cầu ít tài nguyên
- Xử lý nhanh hơn
- Sử dụng ít bộ nhớ hơn

4.2.1. Các Thư Viện và Module

- PyTorch: Sử dụng để tải và chạy mô hình GAN.

- NumPy: Xử lý dữ liệu hình ảnh.
- PIL (Pillow): Đọc và hiển thị hình ảnh.
- PyQt5: Tạo giao diện người dùng với các thành phần như cửa sổ, nút bấm, thanh trượt, v.v.
- os: Quản lý đường dẫn và tệp tin.
- time: Đo thời gian thực thi các tác vụ.

4.2.2 Các Lớp Chính

- WorkerSignals và Worker: Quản lý các tác vụ chạy trong background (đa luồng) để tránh đóng băng giao diện người dùng khi thực hiện các tác vụ tốn thời gian như sinh ảnh.
- FaceEditor: Lớp chính của ứng dụng, quản lý giao diện người dùng và các chức năng chính.

4.2.3. Các Chức Năng Chính

- Tải và Hiển Thị Ảnh Gốc: Ứng dụng tải ảnh gốc từ thư mục assets/demo/input_images và hiển thị lên giao diện.
- Chỉnh Sửa Ảnh: Người dùng có thể chỉnh sửa ảnh thông qua các thanh trượt tương ứng với các thuộc tính như nụ cười, màu tóc, v.v.
- Thay Đổi Độ Phân Giải và Kênh Màu: Người dùng có thể điều chỉnh độ phân giải và tỷ lệ kênh màu của ảnh đầu ra.
- Reset và Finalize: Cho phép người dùng reset các thay đổi hoặc áp dụng các thay đổi cuối cùng.

4.2.4. Cách Thức Hoạt Động

1. **Khởi Tạo Ứng Dụng**

- Khi ứng dụng khởi chạy, lớp FaceEditor được khởi tạo, tải các tài nguyên cần thiết như mô hình GAN, các ảnh gốc, và các hướng chỉnh sửa (directions).
- Giao diện người dùng được thiết lập với các thành phần như ảnh gốc, ảnh đã chỉnh sửa, các thanh trượt, và nút bấm.

2. **Chỉnh Sửa Ảnh**
- Người dùng có thể chọn ảnh từ danh sách và chỉnh sửa ảnh bằng cách kéo các thanh trượt. Mỗi thanh trượt tương ứng với một thuộc tính khuôn mặt cụ thể.
- Khi người dùng thả thanh trượt, ứng dụng sẽ tính toán lại latent code dựa trên giá trị của thanh trượt và sinh ra ảnh mới.

3. **Thay Đổi Độ Phân Giải và Kênh Màu**
- Người dùng có thể điều chỉnh độ phân giải và tỷ lệ kênh màu của ảnh đầu ra thông qua các thanh trượt tương ứng. Điều này cho phép tạo ra các ảnh với chất lượng và kích thước khác nhau.

4. **Reset và Finalize**
- Nút Reset cho phép người dùng đưa các thanh trượt về giá trị ban đầu và hiển thị lại ảnh gốc.
- Nút Finalize áp dụng các thay đổi cuối cùng và sinh ra ảnh với độ phân giải và kênh màu đầy đủ.

4.2.5. Các Thành Phần Giao Diện

- Ảnh Gốc và Ảnh Đã Chỉnh Sửa: Hiển thị ảnh gốc và ảnh đã chỉnh sửa.
- Thanh Trượt: Cho phép người dùng điều chỉnh các thuộc tính khuôn mặt.
- Nút Reset và Finalize: Cho phép người dùng reset hoặc áp dụng các thay đổi.
- Thanh Trạng Thái: Hiển thị thông báo và thời gian thực hiện các tác vụ.

4.2.6. Các Thành Phần Backend

- Mô Hình GAN: Sử dụng mô hình Anycost GAN để sinh ảnh dựa trên latent code.
- Latent Code: Đại diện cho các đặc trưng của ảnh, được điều chỉnh thông qua các thanh trượt.
- Đa Luồng: Sử dụng QThreadPool để chạy các tác vụ sinh ảnh trong background, tránh đóng băng giao diện người dùng.

TÀI LIỆU THAM KHẢO

(Ji Lin, Anycost GANs for Interactive Image Synthesis and Editing, 2021)

(Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, Alexei A. Efros, Generative Visual Manipulation on the Natural Image Manifold, 2016)

(Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila, Analyzing and Improving the Image Quality of StyleGAN, *2019*)