ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
MÔN PHÁT TRIỂN ỨNG DỤNG WEB

Báo cáo bài tập môn Thị giác máy tính
Segment and Restore image

Sinh viên: Nguyễn Nhật Tùng - 20002177
Đỗ Minh Tuấn - 20002175
Lã Anh Trúc - 20002169
Chuyên ngành: K65 Kỹ thuật điện tử và tin học
Giảng viên giảng dạy: Phạm Tiến Lâm

Hà Nội, 2023

Lời cảm ơn

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Trường Đại học Khoa học Tự nhiên đã đưa môn học Thị giác máy tính này vào chương trình giảng dạy. Môn học này tạo ra định hướng nghiêc cứu khoa học, vô cùng bổ ích và tạo các kiến thức đối với Khoa và đặc biệt là ngành của sinh viên Kỹ thuật điện tử và tin học. Đồng thời, môn học này cũng cung cấp đủ kiến thức hữu ích và gắn với nhu cầu thực tiễn của sinh viên.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên giảng dạy là thầy Phạm Tiến Lâm và thầy Đặng Văn Báu-trợ giảng bộ môn đã truyền đạt những kiến thức quý báu, giúp đỡ em hoàn thành báo cáo cho nghiên cứu tiểu luận này. Trong thời gian tham gia môn học vừa qua mà nhận được sự giúp đỡ của thầy đã giúp em có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc. Đối với em, đây chắc chắn là kiến thức quý báu, là hành trang hữu dụng kể cả sau bậc đại học và trong cuộc sống.

Do chưa có nhiều kinh nghiệm về thiết kế Web cũng như thiếu xót về kiến thức, bài tập lớn cuối kỳ này không tránh được việc còn những sai sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ phía các Thầy để đề tài tiểu luận hoàn thiện hơn. Em xin chân thành cảm ơn.

MỤC LỤC

Lời mở đầu

Hiện nay, Công nghệ thị giác máy tính (Computer Vision) rất quan trọng trong nhiều lĩnh vực và có ảnh hưởng lớn đến nhiều khía cạnh của cuộc sống hiện đại. Thị giác máy tính đi sâu vào khả năng giúp máy tính có khả năng nhận biết, phân tích và tìm hiểu các đối tượng có trong hình ảnh hoặc video. Dễ dàng lấy ví dụ về nhận dạng và phân loại hình ảnh trong xác định khuôn mặt, hoặc nhận diện vật thể trợ giúp ngành công nghiệp tự động lái xe hay xử lý hình ảnh y khoa,…

Dự án Segment and Restore Image của nhóm chúng tôi thực hiện nhắm tới khả năng nhận diện được chính xác vị trí vật thể, cụ thể là nhận diện người từ đó tách riêng bộ phần này để khôi phục lại ảnh chỉ gồm ngoại cảnh xung quanh. Từ những hiểu biết về các mô hình học máy và học sâu, bài báo cáo sẽ đi sâu vào cách thức hoạt động của các model học máy có trong dự án. Dự án được thực hiện trên Google Colab và có thể áp dụng trên Web cho các bài toán đa dạng hơn.

Chú ý rằng dự án này vẫn chưa đầy đủ các chức năng cần có, nên vẫn tồn tại nhiều sai sót trong thiết kế thực và kết quả sẽ được cập nhật tiếp. Bên cạnh đó, vì điều kiện không cho phép, tài nguyên chạy dự án cũng không nhiều, nên các mô hình chưa được tối ưu hoàn toàn, mong thầy và các bạn đóng góp ý kiến.

Chương 1. Tổng quan

Thị giác máy tính (computer vision) là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh và, nói chung là dữ liệu đa chiều từ thế giới thực để cho ra các thông tin số hoặc biểu tượng, ví dụ trong các dạng quyết định. Việc phát triển lĩnh vực này có bối cảnh từ việc sao chép các khả năng thị giác con người bởi sự nhận diện và hiểu biết một hình ảnh mang tính điện tử. Thị giác máy tính là môn khoa học liên quan đến các hệ thống nhân tạo có trích xuất các thông tin từ các hình ảnh. Dữ liệu hình ảnh có thể nhiều dạng khác nhữ chuỗi hình ảnh, video, các cảnh từ đa camera, ảnh đa chiều từ máy quét y học,…

Lịch sử của Thị giác máy tính (Computer Vision) bắt đầu từ những năm 1960 và đã trải qua một quá trình phát triển đáng kể từ đó. Dưới đây là một tóm tắt về lịch sử của Thị giác máy tính:

- Những năm 1960-1970: Thị giác máy tính bắt đầu với việc nghiên cứu về xử lý ảnh đầu tiên. Trong giai đoạn này, các nhà nghiên cứu tập trung vào xử lý ảnh số và cơ bản, như phát hiện biên, lọc nhiễu và trích xuất đặc trưng đơn giản.
- Những năm 1980-1990: Các phương pháp phân tích và nhận dạng đối tượng bắt đầu phát triển trong giai đoạn này. Các mô hình tổng quát được áp dụng, bao gồm mô hình học máy và mạng nơ-ron nhân tạo, nhằm giải quyết những vấn đề phức tạp hơn như nhận dạng khuôn mặt và đối tượng.
- Những năm 2000-2010: Sự phát triển của công nghệ máy tính và các phương pháp học máy đã mở ra nhiều cánh cửa mới cho Thị giác máy tính. Các thuật toán học sâu, đặc biệt là các mô hình mạng nơ-ron tích chập (CNN), trở thành xu hướng chính và đã đạt được nhiều thành công đáng kể trong các bài toán như phân loại ảnh, nhận dạng đối tượng và phân tích hình ảnh y khoa.
- Những năm 2010-đến nay: Trong giai đoạn này, Thị giác máy tính đã tiếp tục phát triển với các công nghệ mới, như RNN (Recurrent Neural Network), GAN (Generative Adversarial Network) và đặc biệt là việc kết hợp thị giác máy tính với các lĩnh vực khác như xử lý ngôn ngữ tự nhiên, tự động lái xe, thị giác máy tính trong thời gian thực và nhiều ứng dụng khác.
- Trong những năm gần đây, Thị giác máy tính đã có sự phát triển đáng kể đến mức có thể nhận dạng và phân loại hình ảnh với độ chính xác tương đương hoặc vượt qua khả năng của con người. Việc ứng dụng Thị giác máy tính đã lan rộng vào nhiều lĩnh vực cho dù đó là trong công nghiệp, y tế, an ninh, xe tự động hoặc giải trí[1].

Hình 1. Cách AI nhận biết các vật thể như thế nào

Phân đoạn (segmentation) là một trong những nhiệm vụ quan trọng trong lĩnh vực thị giác máy tính. Nhiệm vụ của phân đoạn là chia ảnh thành các vùng (hoặc đối tượng) khác nhau trong hình ảnh để định vị và phân loại chúng. Mục tiêu của phân đoạn ảnh là đơn giản hóa hoặc thay đổi cách thể hiện ảnh thành thứ gì đó có thể dễ dàng phân tích hơn. Segmentation được đánh giá quan trọng trong thị giác máy tính vì giúp máy tính có khả năng nhận biết và phân loại các vùng riêng biệt trong hình ảnh, từ đó hỗ trợ nhận dạng, phân tích và xử lý hình ảnh một cách chi tiết và chính xác. Phân đoạn có ứng dụng rộng rãi trong nhiều lĩnh vực. Ví dụ, trong học tập, phân tích giúp xác định cấu trúc cấu trúc anh, phân loại tế bào hoặc phát hiện và đo kích thước tế bào ung thư. Trong xe tự lái, phân đoạn giúp nhận dạng và định vị các vật thể, làn sóng đường hay biển báo giao thông. Ngoài ra, phân đoạn cũng được sử dụng trong xử lý hình ảnh, tổ chức dữ liệu, truy tìm hình ảnh và nhiều lĩnh vực khác[2].

Tuy nhiên, phân đoạn vẫn tồn tại nhiều thách thức như có nhiều công thức khác nhau. Vấn đề phức tạp nhất là đảm bảo độ chính xác của phân đoạn, đặc biệt là khi xử lý các hình ảnh có sự bảo đảm, biến dạng không đồng nhất hoặc điều kiện ánh sáng khác nhau. Các kỹ thuật phân tích hiện đại sử dụng các mạng nơ-ron thần tốc (CNN) và các thuật toán học sâu để giải quyết các công thức này. Trong những năm gần đây, phân tích đã có những điều đáng kể, đặc biệt là sự phát triển của các mạng nơ-ron tích chớp. Các mô hình như U-Net, Mask R-CNN và DeepLab đã mang lại những kết quả ấn tượng trong phân đoạn, với khả năng đạt được độ chính xác cao hơn và có thể áp dụng trong thời gian thực.

Khôi phục hình ảnh (Restore image) là hoạt động lấy một hình ảnh bị hỏng/nhiễu và ước tính hình ảnh gốc, sạch. Các yếu tố có thể ảnh hưởng tới bức ảnh cần khôi phục như bị làm mờ (blur), nhiễu (noise) hoặc tiêu cự của máy ảnh. Việc khôi phục các bức ảnh cũ trở nên sắc nét hơn, phục hồi gần như nguyên vẹn so với ban đầu đã không còn hiếm gặp đối với các chức năng photoshop hoặc thậm chí áp dụng các mô hình học sâu.

Theo đó mục đích của dự án này là sự dụng các mô hình học sâu để xử lý các bức ảnh selfie, nhận diện được đối tượng là con người có trong ảnh và khôi phục các ngoại cảnh của bức ảnh. Để nhận diện người có trong ảnh, mô hình unet được áp dụng để nhằm phân đoạn hình ảnh và mô hình deepfillv2 của Pytorch để khôi phục ảnh. Ứng dụng của dự án này có thể sử dụng để nâng cao chất lượng của các mô hình học máy xác định vị trí, nhận diện người,…

Hướng đi của bài toán là xử lý dữ liệu ảnh đầu vào, xác định phần vùng đối tượng cần xóa (cụ thể là người). Bước tiếp theo là xóa đối tượng và lấp đầy ảnh bằng việc sử dụng model học sâu. Bài toán có thể phát triển xa hơn và ap dụng cho các bài toán xóa vật thể mong muốn và khôi phục ảnh

Hình 2. Minh họa cho kết quả dự án.

Chương 2. Các mô hình sử dụng

2.1. Unet model

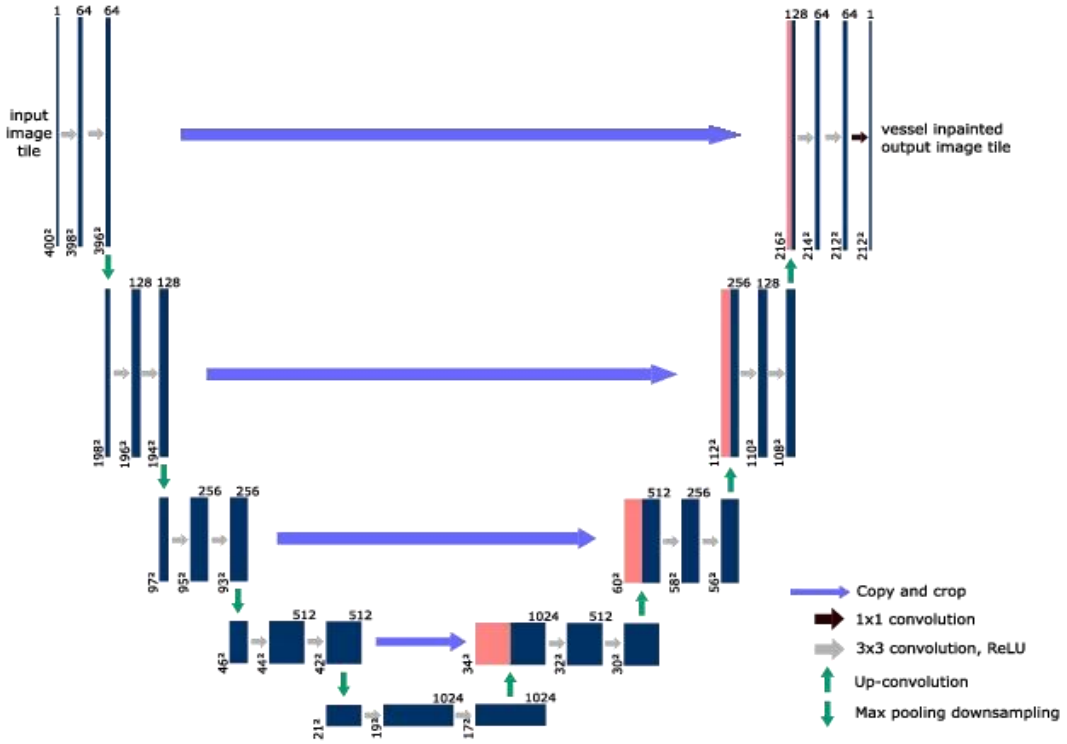
Mô hình U-Net là một kiến trúc học sâu thường được sử dụng cho các nhiệm vụ phân đoạn hình ảnh, đặc biệt là trong phân tích hình ảnh y tế. Nó được đề xuất lần đầu tiên bởi Olaf Ronneberger, Philipp Fischer và Thomas Brox vào năm 2015. Cái tên "U-Net" xuất phát từ kiến trúc hình chữ U của mạng, giống với cấu trúc bộ mã hóa-giải mã. Mô hình U-Net được thiết kế đặc biệt cho các nhiệm vụ trong đó cả thông tin theo ngữ cảnh cục bộ và toàn cầu đều quan trọng, chẳng hạn như phân đoạn các đối tượng trong hình ảnh[3].

Kiến trúc U-Net bao gồm đường dẫn bộ mã hóa và đường dẫn giải mã tương ứng. Đường dẫn bộ mã hóa ghi lại bối cảnh và trích xuất các đặc điểm từ hình ảnh đầu vào bằng cách áp dụng các phép toán tích chập và gộp, giảm độ phân giải không gian trong khi tăng số lượng kênh. Đường dẫn bộ giải mã lấy các đặc điểm được trích xuất bởi đường dẫn bộ mã hóa và thực hiện các thao tác tích chập lên (còn được gọi là tích chập chuyển vị) để khôi phục dần độ phân giải không gian của hình ảnh đầu vào. Ở mỗi giai đoạn của bộ giải mã, các đặc điểm từ giai đoạn tương ứng của đường dẫn bộ mã hóa cũng được kết nối để thu thập cả thông tin cục bộ và toàn cầu.

Mô hình U-Net có các kết nối bỏ qua, cho phép bộ giải mã truy cập các tính năng từ các lớp trước đó trong bộ mã hóa. Các kết nối bỏ qua này cho phép bản địa hóa chính xác trong quá trình phân đoạn bằng cách cung cấp các chi tiết chi tiết từ lớp đầu đến lớp sau

Khác với các mô hình Mask RCNN, Deeplab v3, Unet có những khác biệt sau:

- Thứ nhất, toàn bộ kiến trúc không hề sử dụng một lớp fully connected nào. Nếu các bạn từng làm việc với deep learning, với các mô hình end-to-end thông thường, lớp kế cuối của mạng sẽ là các lớp fully connected để kết nối các đặc trưng đã phân tích được nhằm đưa ra kết quả dự đoán. Tuy nhiên, ở kiến trúc U-net, việc kết nối các đặc trưng sẽ do nửa thứ 2 của "chữ U" đảm nhận, điều này giúp mạng không cần mạng fully connected, do đó có thể chấp nhận input với kích thước bất kì
- Thứ hai: U-net sử dụng Phương pháp đệm (Padding method), điều này giúp kiến trúc có thể phân đoạn hình ảnh được hoàn toàn. Phương pháp này đặc biệt quan trọng khi segment cho các hình ảnh, nếu không, độ phân giải có thể bị hạn chế bởi dung lượng của bộ nhớ GPU.



Hình 3. Cấu trúc U-net

2.2. Deepfillv2 model

Inpainting image (hay còn gọi là hoàn thiện hình ảnh) là nhiệm vụ tổng hợp các nội dung thay thế trong các vùng bị thiếu sao cho việc sửa đổi trở nên thực tế một cách trực quan và đúng về mặt ngữ nghĩa. Nó cho phép loại bỏ sự phân tâm các đối tượng hoặc chỉnh sửa các vùng không mong muốn trong ảnh. Nó cũng có thể được mở rộng cho các tác vụ bao gồm không cắt xén hình ảnh/video, quay lại bố trí, ghép, nhắm mục tiêu lại, bố cục lại, nén, siêu phân giải, hài hòa và nhiều thứ khác.

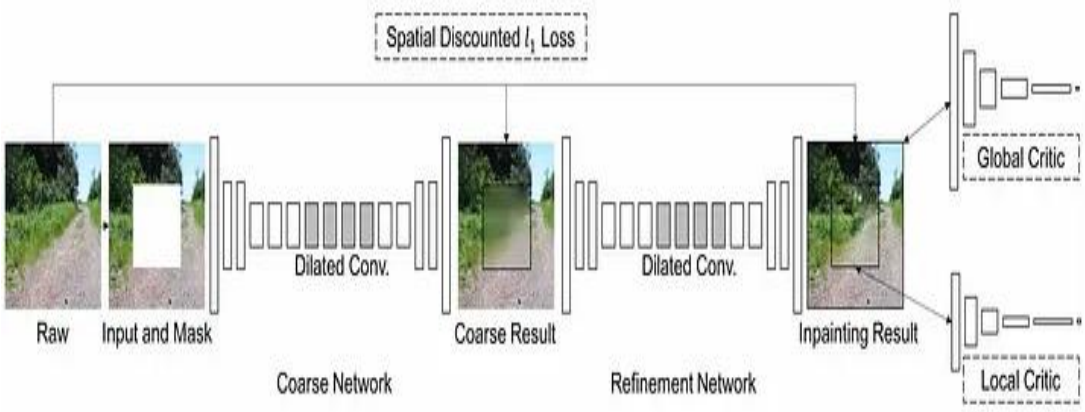
Deepfillv2[4] model được cải tiến từ Deepfillv1, Partial Convolution, EdgeConnect.

Deepfillv1 model được mô tả qua bài báo Generative Image Inpainting with Contextual Attention[5] gồm 2 thành phần: kiến trúc khung Image Inpainting và chú ý ngữ cảnh (Contextual attention).

Kiến trúc khung bao gồm hai mạng tạo và hai mạng phân biệt đối xử:

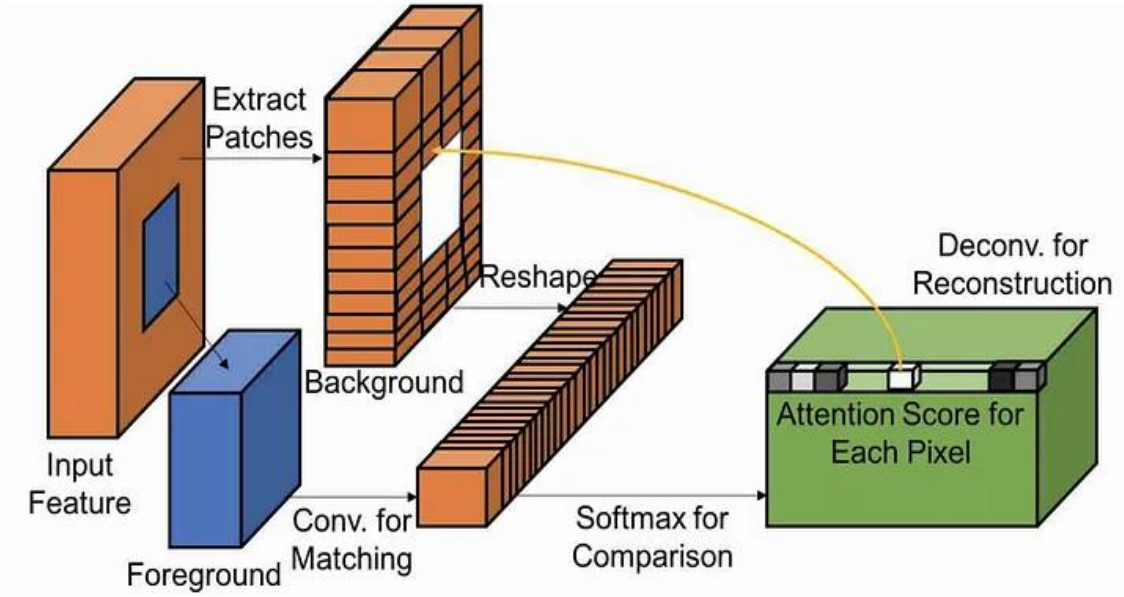
- Hai bộ tạo tuân theo các mạng tích chập đầy đủ với các tích chập giãn nở (dilated Convolution). Một generator dành cho việc tái tạo và một generator khác dành cho phân loại. Được gọi là cấu trúc mạng từ thô đến mịn tiêu chuẩn

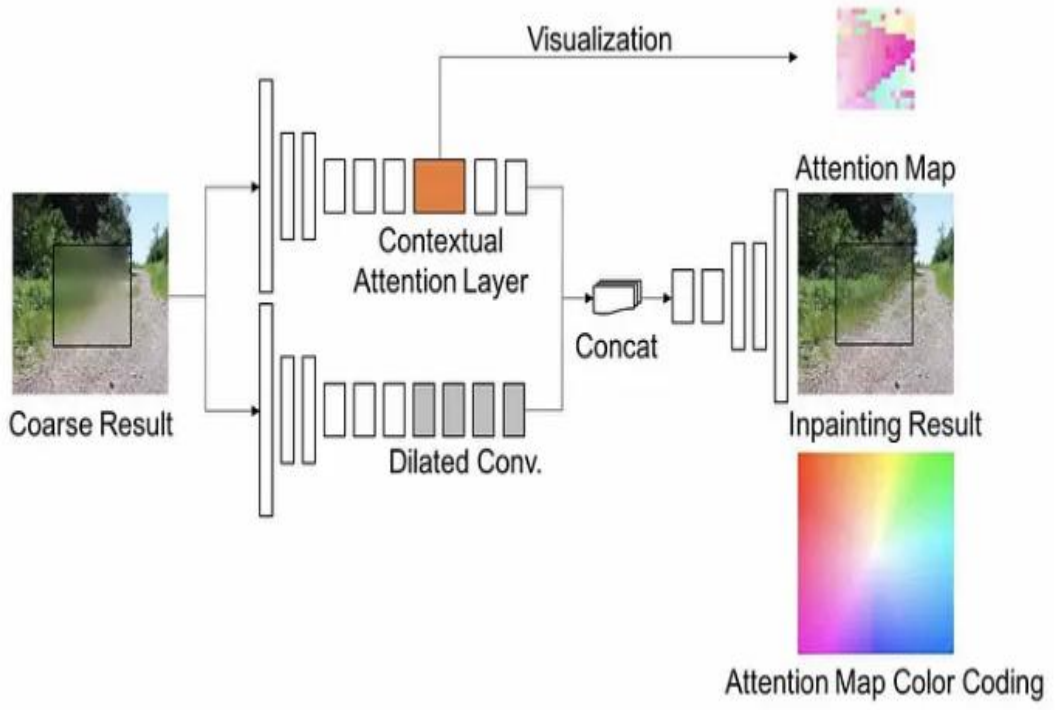
- Hai bộ phân biệt đối xử cũng xem xét các hình ảnh hoàn chỉnh cả trên toàn cầu và cục bộ. Bộ phân biệt đối xử toàn cục lấy toàn bộ hình ảnh làm đầu vào trong khi bộ phân biệt cục bộ lấy vùng được lấp đầy làm đầu vào



Hình 4. Cấu trúc mô hình khôi phục ảnh của Deepfillv1

Cơ chế chú ý theo ngữ cảnh được đề xuất để mượn thông tin theo ngữ cảnh một cách hiệu quả từ các vị trí không gian ở xa để tái tạo lại các pixel bị thiếu. Sự chú ý theo ngữ cảnh được áp dụng cho mạng sàng lọc thứ hai. Mạng tái thiết thô đầu tiên chịu trách nhiệm ước tính sơ bộ các vùng bị thiếu. Tương tự như trước, các bộ phân biệt đối xử toàn cục và cục bộ được sử dụng để khuyến khích các chi tiết kết cấu cục bộ tốt hơn của các pixel được tạo





Hình 5-6. Cấu trúc chú ý ngữ cảnh của Deepfillv1

Partial Convolution[6] (tích chập một phần) hoạt động theo chức năng cập nhật mask đầu vào như Lớp chập một phần. Gọi W là trọng số của bộ lọc tích chập đối với bộ lọc tích chập và b là độ lệch tương ứng. X là đặc trưng giá trị (độ sáng của pixel) cho cửa sổ tích chập (trượt) hiện tại và M là giá trị nhị phân trên ma trận mask tương ứng. Sự tích chập một phần ở mọi vị trí, được biểu thị như sau:

$$x' = \begin{cases} W^T(X \cdot M)\dfrac{sum(1)}{sum(M)} + b, & if\ sum(M) > 0 \\ 0 & otherwise \end{cases}$$

Sau mỗi lần thực hiện quá trình tích chập một phần, cần cập nhật lại giá trị trên ma trận mask nếu trường hợp tích chập một phần điều chỉnh đầu ra của nó trên ít nhất một đầu vào hợp lệ:

$$m' = \begin{cases} 1, & if\ sum(M) > 0 \\ 0, & otherwise \end{cases}$$

Phần cuối cùng đầu vào của lớp tích chập sẽ chứa phần nối của đầu vào ban đầu hình ảnh có lỗ và mặt nạ gốc, giúp người mẫu có thể sao chép pixel không lỗ. Chi tiết mạng được tìm thấy trong tập tin bổ

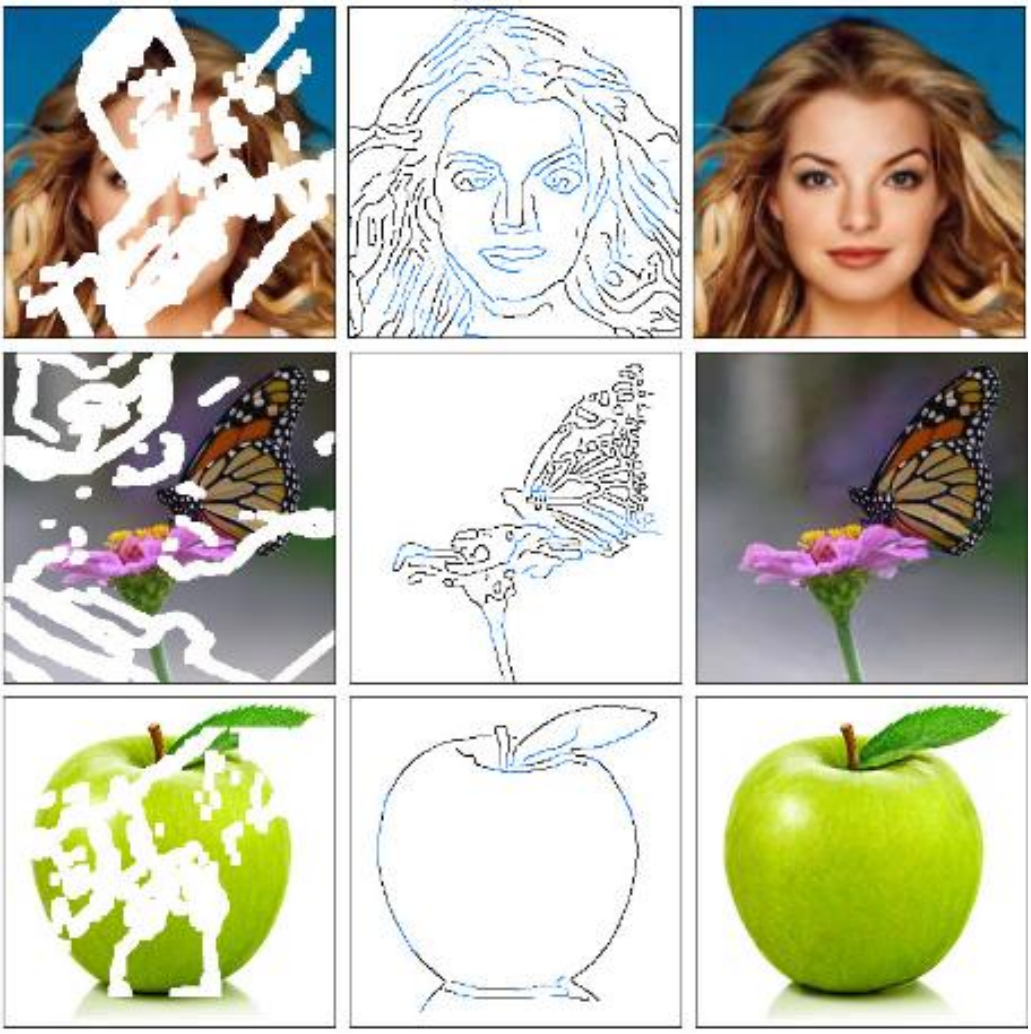
sung. Tích chập một phần như phần đệm. Mô hình sử dụng tích chập từng phần với che chắn thích hợp ở ranh giới hình ảnh thay cho phần đệm thông thường. Điều này đảm bảo nội dung không được tô màu ở viền ảnh sẽ không bị ảnh hưởng bởi các nội dung không hợp lệ các giá trị bên ngoài hình ảnh - có thể được hiểu là một khoảng trống khác khác

EdgeConnect[7] là một mô hình vẽ hình ảnh nằm trong khuôn khổ DeepFillv2. Nó được thiết kế để lấp đầy các vùng bị thiếu hoặc bị hỏng trong hình ảnh dựa trên bối cảnh của các pixel xung quanh. Mục tiêu chính của EdgeConnect là hoàn thiện các phần còn thiếu của hình ảnh một cách liền mạch, sao cho các vùng không được tô màu nhất quán về mặt trực quan với phần còn lại của hình ảnh.

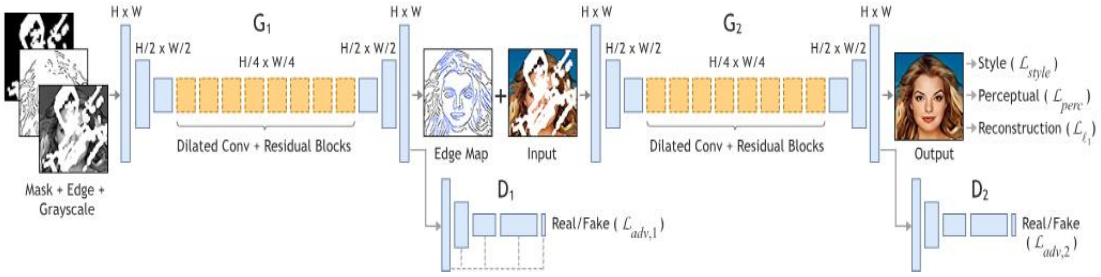
Cách tiếp cận được EdgeConnect sử dụng bao gồm hai bước chính: tạo cạnh và hoàn thiện hình ảnh:

- Tạo cạnh: Trong bước này, EdgeConnect trước tiên tạo bản đồ cạnh bằng cách sử dụng thông tin có sẵn trong hình ảnh. Bản đồ cạnh chứa thông tin về ranh giới và đường viền của các đối tượng trong ảnh. Điều này được thực hiện bằng cách sử dụng một mạng đã được huấn luyện để phát hiện các cạnh.
- Hoàn thiện hình ảnh: Sau khi tạo bản đồ cạnh, EdgeConnect sẽ sử dụng nó cùng với thông tin hình ảnh có sẵn để hoàn thiện các vùng còn thiếu. Nó sử dụng một mạng hoàn thiện được đào tạo để tái tạo lại các pixel bị thiếu dựa trên bối cảnh được cung cấp bởi bản đồ cạnh và nội dung hình ảnh đã biết.

Bằng cách kết hợp bản đồ cạnh được tạo với mạng hoàn chỉnh, EdgeConnect có thể khắc sâu các vùng bị thiếu theo cách phù hợp nhất với cấu trúc và hình thức tổng thể của hình ảnh.



Hình 7. Ví dụ minh họa cấu trúc EgdeConnect



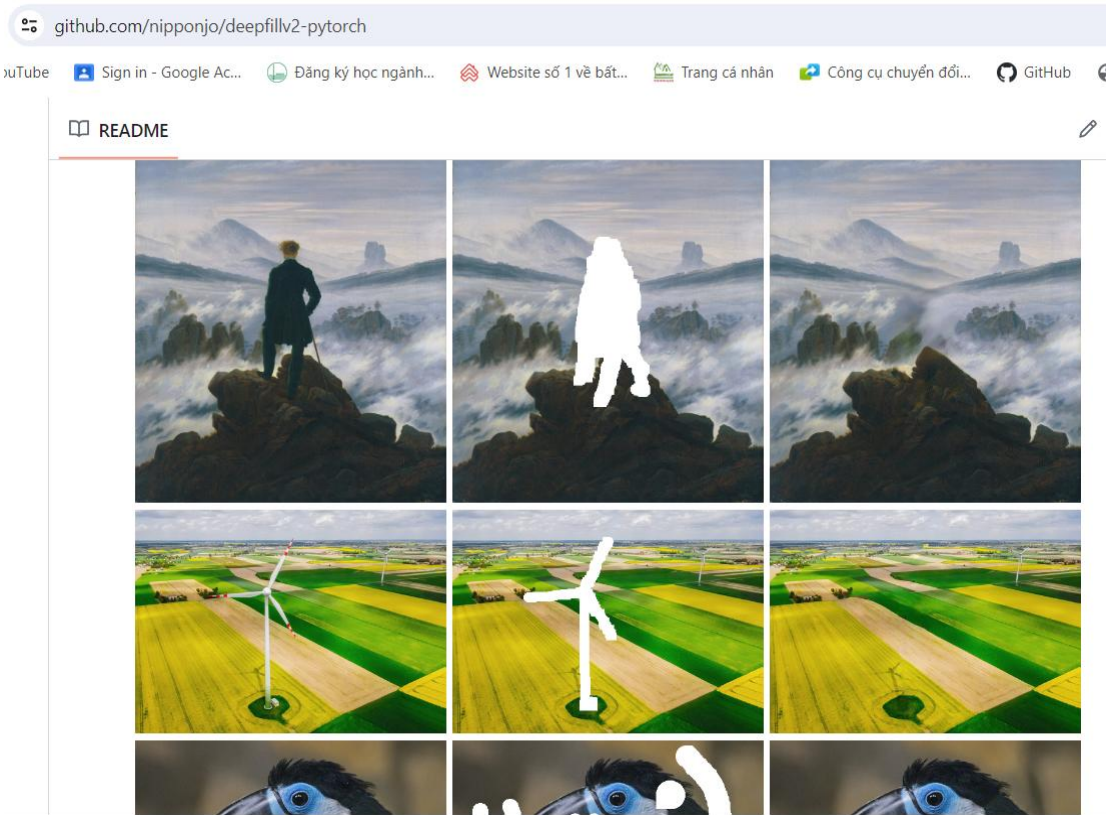
Hình 8. Mô hình cấu trúc EgdeConnect

Để kết nối DeepFillv1, Partial Convolution và EdgeConnect thành DeepFillv2, cần xem xét các chức năng và tính năng cụ thể của từng mô hình cũng như xác định cách chúng có thể bổ sung cho nhau trong quy trình inpainting:

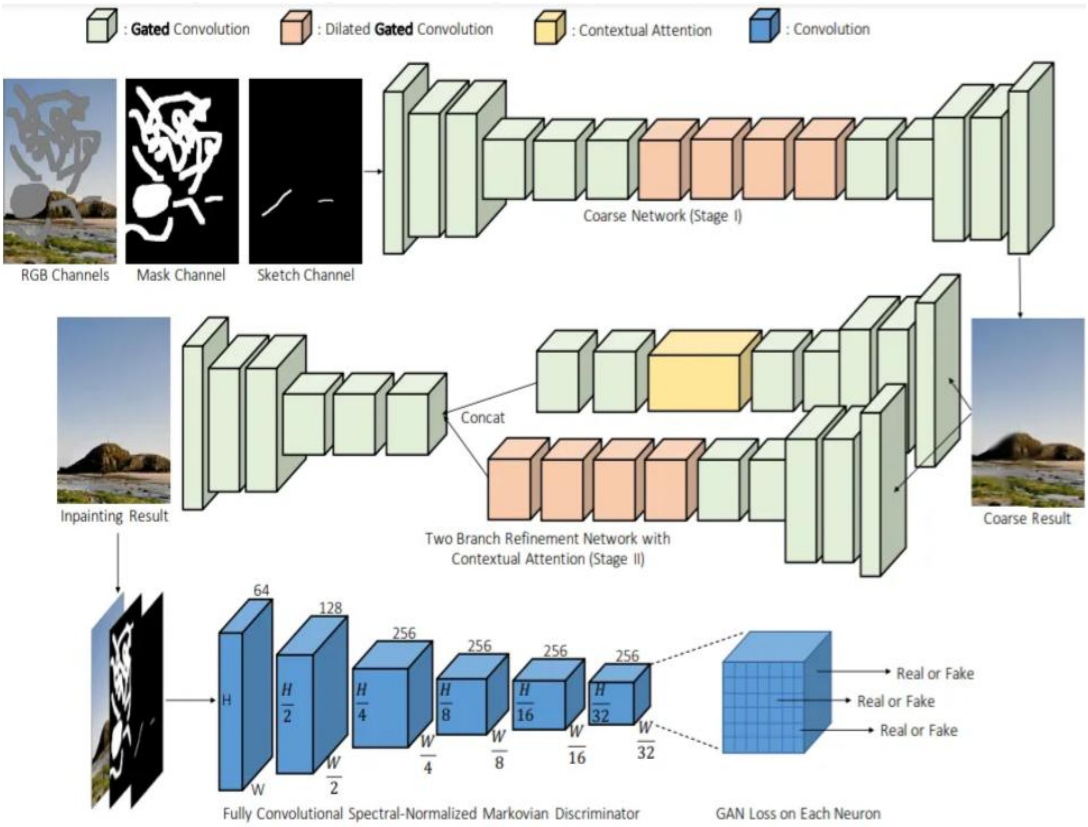
- Tích hợp tích chập một phần: Mạng tích chập một phần (PCN) được thiết kế đặc biệt để xử lý dữ liệu đầu vào một phần và bị hỏng bằng cách

cập nhật động các bộ lọc tích chập đã học dựa trên tính khả dụng của các pixel đầu vào hợp lệ. Vì DeepFillv2 cũng xử lý các hình ảnh chưa hoàn chỉnh nên bạn có thể kết hợp mô-đun Chuyển đổi từng phần vào bộ mã hóa hoặc bộ giải mã của DeepFillv2 model. Điều này sẽ nâng cao khả năng của DeepFillv2 trong việc xử lý các mặt nạ hoặc hình ảnh không đều với các vùng bị thiếu.

- EdgeConnect: tập trung vào việc tạo bản đồ cạnh chính xác trước khi thêm các vùng còn thiếu. Điều này có thể hữu ích trong việc cung cấp hướng dẫn và ngữ cảnh bổ sung cho DeepFillv2. Bạn có thể kết hợp đầu ra của bước tạo cạnh của EdgeConnect làm đầu vào bổ sung hoặc biểu diễn tính năng cho DeepFillv2. Bằng cách kết hợp các bản đồ biên với thông tin hình ảnh có sẵn, DeepFillv2 có thể hưởng lợi từ việc vẽ chính xác hơn ở những khu vực có ranh giới hoặc cấu trúc phức tạp.



Hình 9. Ví dụ minh họa của deepfillv2[8]



Hình 10. Cấu trúc của deepfillv2 model

Chương 3. Thực hiện dự án

3.1. Chuẩn bị dữ liệu

Dữ liệu là các bức ảnh selfie thu thập trên mạng, được hiển thị và sử dụng công khai trên các trang mạng. Đặc trưng của ảnh gồm các người đang chụp ảnh tại địa điểm nào đó. Đối với các bức ảnh selife thì có một vài vấn đề của ảnh như có thể ảnh hướng tới độ chính xác của mô hình trong Bảng 1. Các vấn đề này có thể bao gồm các vấn đề ngoại cảnh như thời tiết, thời gian chụp hay do người chụp tự chỉnh mờ đi.

Các vấn đề	Tỷ lệ
Ảnh chụp xa, người quá bé so với kích thước ảnh	3%
Ảnh chụp có nhiều người	8%
Người dùng làm mờ phông	6%
Khung cảnh địa điểm khác biệt theo thời gian	20%
Có vật thể ở trước người	12%

Bảng 1. Tỷ lệ ảnh có các yếu tố ảnh hướng tới phân đoạn người.

Hình 11 . Ví dụ hình ảnh làm mờ phông (A) và có nhiều người trong ảnh (B)

3.2. Xử lý dữ liệu

Roboflow[9] là một framework dành cho nhà phát triển Thị giác Máy tính để thu thập dữ liệu tốt hơn để xử lý trước và các kỹ thuật đào tạo mô hình . Roboflow có sẵn các tập dữ liệu công khai cho người dùng và cũng có quyền truy cập để người dùng tải lên dữ liệu tùy chỉnh của riêng họ. Roboflow chấp nhận các định dạng chú thích khác nhau. Trong quá trình xử lý trước dữ liệu, có các bước liên quan như định hướng hình ảnh, thay đổi kích thước, độ tương phản và tăng cường dữ liệu. Toàn bộ quy trình làm việc có thể được điều phối

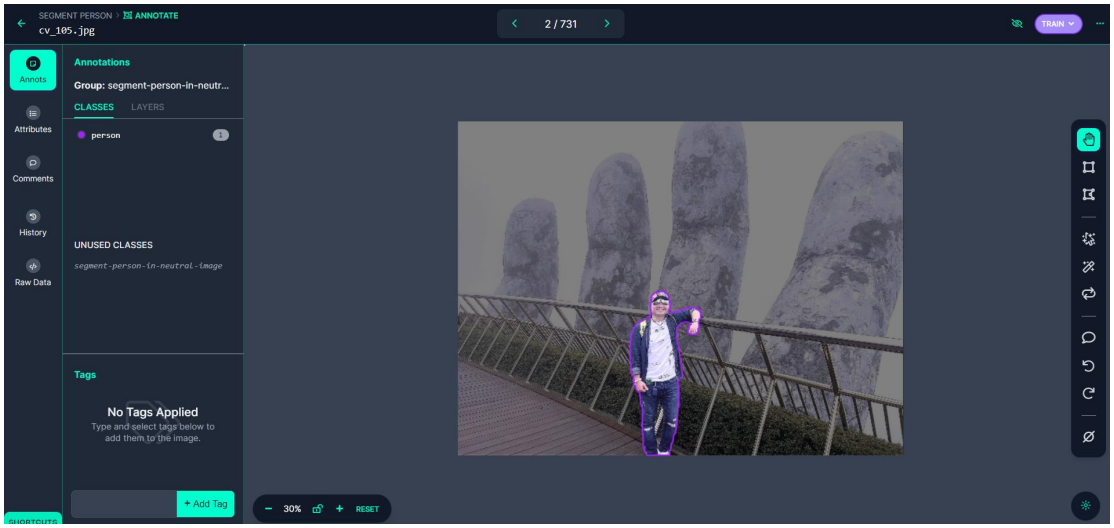
với các nhóm trong khuôn khổ. Đối với đào tạo mô hình, có một loạt thư viện mô hình đã có mặt như EfficientNet[10], MobileNet[11], Yolo[12], TensorFlow[13], PyTorch[14],…

Roboflow được sử dụng trong các ngành công nghiệp thị giác máy tính khác nhau cho các trường hợp sử dụng như – phát hiện rò rỉ khí đốt, phát hiện thực vật và cỏ dại, bảo trì máy bay, ước tính thiệt hại mái nhà, hình ảnh vệ tinh, ô tô tự lái, bộ đếm giao thông, dọn rác và nhiều hơn nữa.

Hình 12. Framework Roboflow

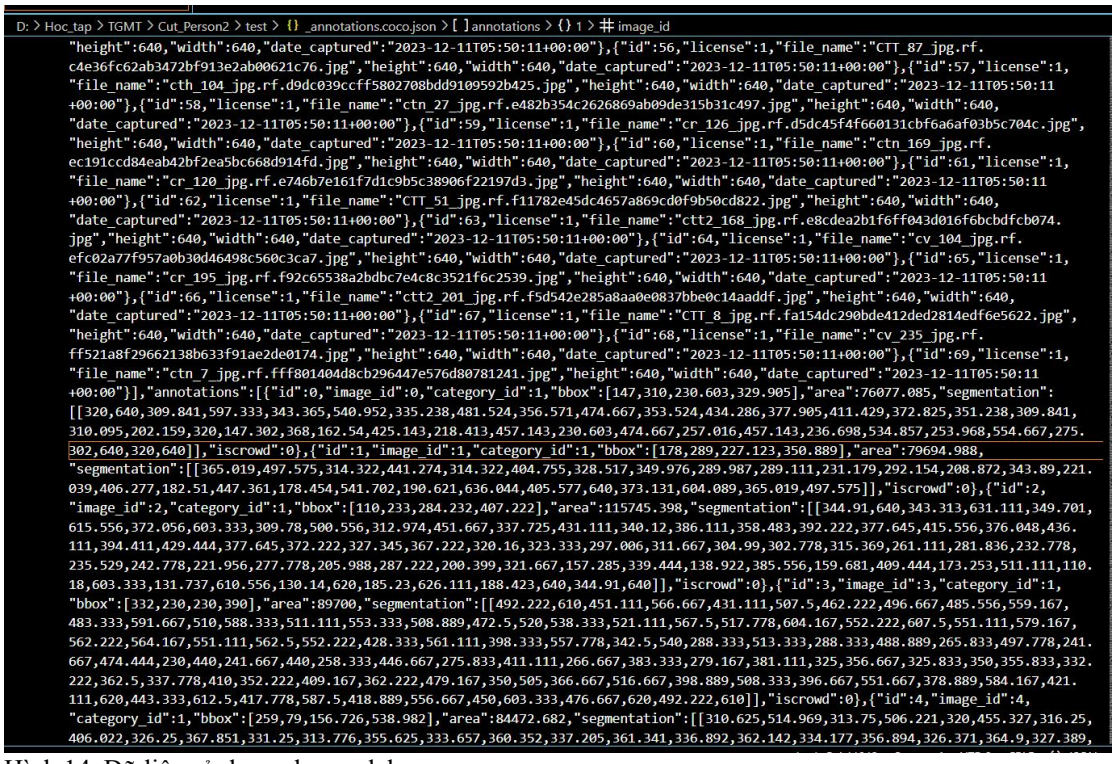
Phân đoạn phiên bản (Instance segmentation), còn được gọi là phân đoạn hình ảnh, là nhiệm vụ thị giác máy tính nhằm nhận dạng các đối tượng trong hình ảnh cùng với hình dạng liên quan của chúng. Đây là phần mở rộng của tính năng phát hiện đối tượng trong đó mỗi dự đoán cũng bao gồm một hình dạng thay vì chỉ một hộp giới hạn được xác định bởi điểm trung tâm, chiều rộng và chiều cao. Với tính năng phân đoạn phiên bản, ứng dụng của bạn có thể xác định số lượng đối tượng trong một hình ảnh, cách phân loại và đường viền của chúng. Nó hữu ích trong trường hợp bạn cần đo kích thước của các đối tượng được phát hiện (ví dụ: lá cà chua), cắt chúng ra khỏi nền của chúng (ví dụ để xóa nền) hoặc phát hiện chính xác hơn các đối tượng xoay thuôn dài (ví dụ: cây cầu trong ảnh vệ tinh).

Theo đó Roboflow cho phép người dùng tạo các dự án Instance segmentation đối với các dự liễu cần phân đoạn. Hình ảnh cho các vấn đề về phân đoạn mẫu phải được chú thích chính xác nhất có thể trong công cụ chú thích. Bạn nên vẽ đường viền xung quanh đối tượng muốn chú thích rồi gán cho đối tượng đó một lớp. Người dùng có thể gắn nhãn hình ảnh để phân đoạn ví dụ bằng cách vẽ đa giác trong Roboflow Annotate .



Hình 13 Thực hiện phân đoạn (segment) trên Roboflow

Để thuận tiện cho xây dựng mô hình U-net, gồm các ảnh và thông tin của chi tiết dữ liệu được tải về dưới định dạng .json. Theo đó các ảnh sẽ được điều chỉnh về chung kích thước 640x640. Trường thông tin ảnh trong "images" sử dụng "id" và "file_name" và vùng phân đoạn nằm trong "annotations" sử dụng "id", "image_id" và "segmentations" như trong Hình 15.



Hình 14. Dữ liệu sử dụng cho model

3.3. Xây dựng mô hình

Dữ liệu các file train, valid and test đều có 1 file .json để ghi lại thông tin về vùng segment của ảnh có id là "image_id". Dữ liệu segment này là một list các điểm x và y liện tục nhưng không chia ra thành từng list riêng để thuận tiện cho việc vẽ polygon của cv2. Trong Hình . list của vùng phân đoạn được

chia thành từng list nhỏ hơn gồm 2 điểm [xi, yi] để vẽ polygonfill trên ma trận 0 có cùng kích thước ảnh và được coi là đầu ra (output) của từng ảnh trong model segmentation.

Ngoài ra tên của ảnh không nằm cùng với vùng "anominations" nên cần append thêm id tương ứng với thuộc tính tên của ảnh "file_name" để nối ảnh nào với output của chúng. Thực hiện tượng tự với cả tập valid và test đã lấy dữ liệu đầu vào cho mô hình.

```
json_name = '_annotations.coco.json'
train_file = open(train_path+json_name)
data = json.load(train_file)

train_id_segment = []
#train_segment = []
train_segment_frame = []
train_file_frame = []
for i in data['annotations']:
    image_id = i['image_id']
    value = np.array(i['segmentation'])
    black = np.zeros((640, 640, 1))
    change = np.reshape(value[0], (int(len(value[0])/2), 2))
    change = np.array(change, np.int32)
    black = cv2.fillPoly(black, [change], (255, 255, 255))
    black[black >= 1] = 1
    black[black == 0] = 0

    # train_frame.append((id, name_img, value))
    train_segment_frame.append([image_id, black])

for j in data['images']:
    id = j['id']
    name_img = j['file_name']
    train_file_frame.append([id, name_img])
train_file.close()

train_data = []
for i in range(len(train_segment_frame)):
    for j in range(len(train_file_frame)):
        if train_segment_frame[i][0] == train_file_frame[j][0]:
            train_data.append([i, train_file_frame[j][1], train_segment_frame[i][1]])
```

Hình 15. Vẽ polygon thành ma trận mask đầu ra cho Unet model

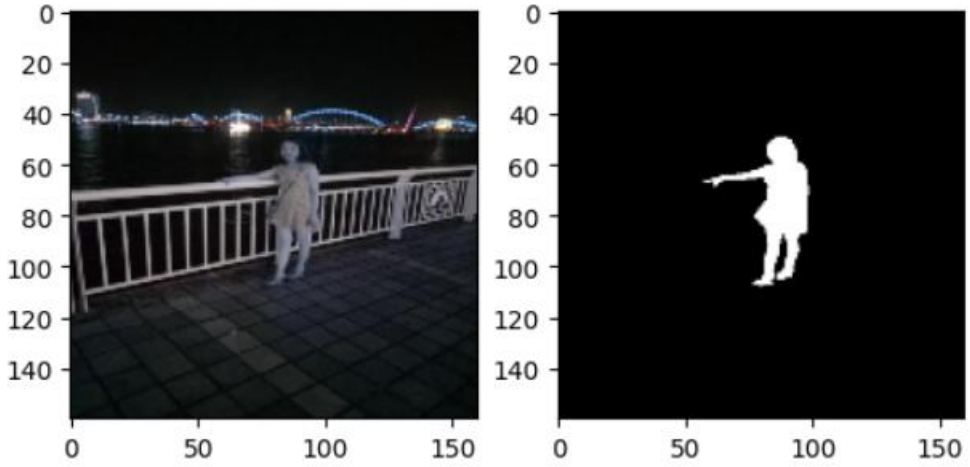
```
[ ] x_train = []
    y_train = []
    x_valid = []
    y_valid = []
    x_test = []
    y_test = []
    for i in range(int(3*len(train_data)/4)):
        img = cv2.imread(train_path+train_data[i][1])
        img = cv2.resize(img, (160, 160))/255.
        img = np.array(img, dtype=np.float32)
        x_train.append(img)

        mask = train_data[i][2]
        mask = cv2.resize(mask, (160, 160))
        y_train.append(mask)
```

Hình 16. Điều chỉnh kích thước ảnh về 160x160

Hình 17. là ví dụ đầu vào và ra nằm trong tập valid gồm ảnh ban đầu và ma trận mask đầu ra.

```
[ ]  plt.subplot(1, 2, 1)
     plt.imshow(x_valid[10])
     plt.subplot(1, 2, 2)
     plt.imshow(y_valid[10], 'gray')
```

```
<matplotlib.image.AxesImage at 0x7b14e7170c70>
```



Hình 17. Ví dụ input và output

Mean_IoU (Intersection-Over-Union) là một thước đo đánh giá phổ biến được sử dụng trong các tác vụ phân đoạn hình ảnh và phát hiện đối tượng để đánh giá độ chính xác của các vùng hoặc mask được dự đoán so với thực tế. Nó đo lường sự chồng chéo giữa vùng được dự đoán và vùng thực tế cơ bản bằng cách tính tỷ lệ giữa giao điểm và sự kết hợp của hai vùng[15]. Số liệu IoU đặc biệt hữu ích khi đánh giá các tác vụ liên quan đến phân đoạn nhị phân hoặc nhiều lớp, trong đó mục tiêu là xác định chính xác ranh giới hoặc vùng của các đối tượng trong một hình ảnh. Nó cung cấp thước đo mức độ phù hợp của các vùng được dự đoán với các vùng thực tế trên mặt đất.
Để tính toán IoU, các bước sau thường được thực hiện:

- Tính toán giao điểm: Xác định giao điểm giữa vùng được dự đoán và vùng thực tế cơ bản, xem xét các điểm ảnh hoặc điểm ảnh ba chiều mà chúng có chung.
- Tính toán sự kết hợp: Xác định sự kết hợp của vùng được dự đoán và vùng thực tế cơ bản, xem xét tất cả các điểm ảnh hoặc điểm ảnh ba chiều là một phần của một trong hai vùng.
- Tính toán IoU: Chia giao điểm cho hợp và biểu thị nó dưới dạng phần trăm hoặc giá trị thập phân. Giá trị IoU cao hơn cho thấy sự chồng chéo tốt hơn giữa các vùng sự thật được dự đoán và thực tế.

```
[ ]  def mean_iou(y_true, y_pred):
         yt0 = y_true[:,:,:,0]
         yp0 = K.cast(y_pred[:,:,:,0] > 0.5, 'float32')
         inter = tf.compat.v1.count_nonzero(tf.logical_and(tf.equal(yt0, 1), tf.equal(yp0, 1)))
         union = tf.compat.v1.count_nonzero(tf.add(yt0, yp0))
         iou = tf.where(tf.equal(union, 0), 1., tf.cast(inter/union, 'float32'))
         return iou
```

Hình 18. Mean IoU metrics

Theo đó mô hình được xây dựng gồm phần encoder tương tự CNN thông thường (Conv, MaxPooling) và layer tăng và width and height của ma trận đặc trưng giảm gấp đôi sau từng bước trong down sampling và được lưu lại trong layers. Khi decorder trở lại mỗi bước đều nhân trở lại để nhân đặc trưng vùng cần tìm so với đặc trưng góc cạnh và vị trí của nó trên ma trận ban đầu.

```
def unet(sz = (160, 160, 3)):
  x = Input(sz)
  inputs = x

  #down sampling
  f = 12
  layers = []

  for i in range(0, 5):
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    layers.append(x)
    x = MaxPooling2D() (x)
    f = f*2
  ff2 = 64

  #bottleneck
  j = len(layers) - 1
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2DTranspose(ff2, 2, strides=(2, 2), padding='same') (x)
  x = Concatenate(axis=3)([x, layers[j]])
  j = j -1

  #upsampling
  for i in range(0, 4):
    ff2 = ff2//2
    f = f // 2
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2DTranspose(ff2, 2, strides=(2, 2), padding='same') (x)
    x = Concatenate(axis=3)([x, layers[j]])
    j = j -1


  #classification
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  outputs = Conv2D(1, 1, activation='sigmoid') (x)
```

```
=================================================
Total params: 3823409 (14.59 MB)
Trainable params: 3823409 (14.59 MB)
Non-trainable params: 0 (0.00 Byte)
```

Hình 19. Cấu trúc (A) và tham số đầu vào (B) của model U-net

Sau quá trình fit trọng số, lưu mô hình với tên model1.h5

```
model.save('model1.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`
  saving_api.save_model(
```

Hình 20. Lưu mô hình

Khi load model1.h5 vì metrics mean_iou cần được định nghĩa lại trong custom_objects của load_model

```
def mean_iou(y_true, y_pred):
    yt0 = y_true[:,:,:,0]
    yp0 = K.cast(y_pred[:,:,:,0] > 0.5, 'float32')
    inter = tf.compat.v1.count_nonzero(tf.logical_and(tf.equal(yt0, 1), tf.equal(yp0, 1)))
    union = tf.compat.v1.count_nonzero(tf.add(yt0, yp0))
    iou = tf.where(tf.equal(union, 0), 1., tf.cast(inter/union, 'float32'))
    return iou
```

```
from tensorflow.keras.models import load_model

model1 = load_model('/content/model1.h5', custom_objects={'mean_iou':mean_iou})
```

Hình 21 Định nghĩa lại IoU metrics (A) và Load model1.h5 (B)

Việc phục hồi ảnh theo mô hình deepfillv2 của pytorch cần git clone thư viện về. Model deepfillv2 cần hai đầu vào là ảnh cần phục hồi và phần mask. Trong dự án này, phần mask được lấy từ chính ảnh đã dự đoán được segment.

```
#@title Run this cell for setup { display-mode: "form"}
!git clone https://github.com/vrindaprabhu/deepfillv2_colab.git
!gdown "https://drive.google.com/u/0/uc?id=1uMghKl883-9hDLhSiI8lRbHCzCmmRwV-&export=download"
!mv /content/deepfillv2_WGAN_G_epoch40_batchsize4.pth deepfillv2_colab/model/deepfillv2_WGAN.pth
```

```
Cloning into 'deepfillv2_colab'...
remote: Enumerating objects: 99, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 99 (delta 2), reused 1 (delta 1), pack-reused 96
Receiving objects: 100% (99/99), 571.56 KiB | 5.15 MiB/s, done.
Resolving deltas: 100% (44/44), done.
Downloading...
From: https://drive.google.com/u/0/uc?id=1uMghKl883-9hDLhSiI8lRbHCzCmmRwV-&export=download
To: /content/deepfillv2_WGAN_G_epoch40_batchsize4.pth
100% 64.8M/64.8M [00:01<00:00, 49.4MB/s]
```

```
UPLOAD INPUT FILE
Chọn tệp  Không có tệp nào được chọn  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving input.png to input.png
Uploaded file "input.png" of 626665 bytes


SELECT MASK TYPE TO INPAINT
random free-form  random bbox  upload
Chọn tệp  Không có tệp nào được chọn  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving mask.png to mask.png
Uploaded file "mask.png" of 1021 bytes


PLEASE RUN THE NEXT CELL
```

```
#@title Run to trigger inpainting. { display-mode: "form" }
!python inpaint.py
```

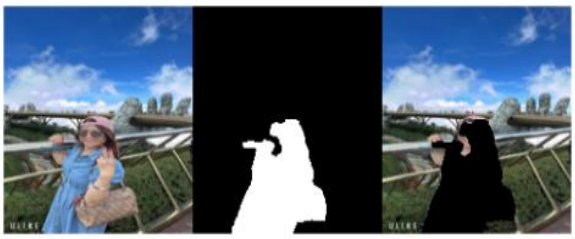
```
-- Generator is created! --
-- Initialized generator with xavier type --
-- INPAINT: Loading Pretrained Model --
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:
  warnings.warn(_create_warning_msg(
-- Inpainting is finished --
```

Hình 22. Thực hiện pretrained model deepfillv2.

Chương 4. Kết quả

Sau quá trình fit model với 100 epochs trọng số mô hình được áp dụng để dự đoán với tập test. Kết quả mô hình là ma trân cùng kích thước phân vùng người. Hàm loss và Mean_Iou cao trên tập test nhưng chỉ đạt 0.6 trên tập validation.

```
i= 99 loss= 0.016232402995228767 val_loss= 0.5371387600898743 mean_iou= 0.9148619771003723 val_mean_iou= 0.5857987403869629
1/1 [==============================] - 0s 18ms/step
```

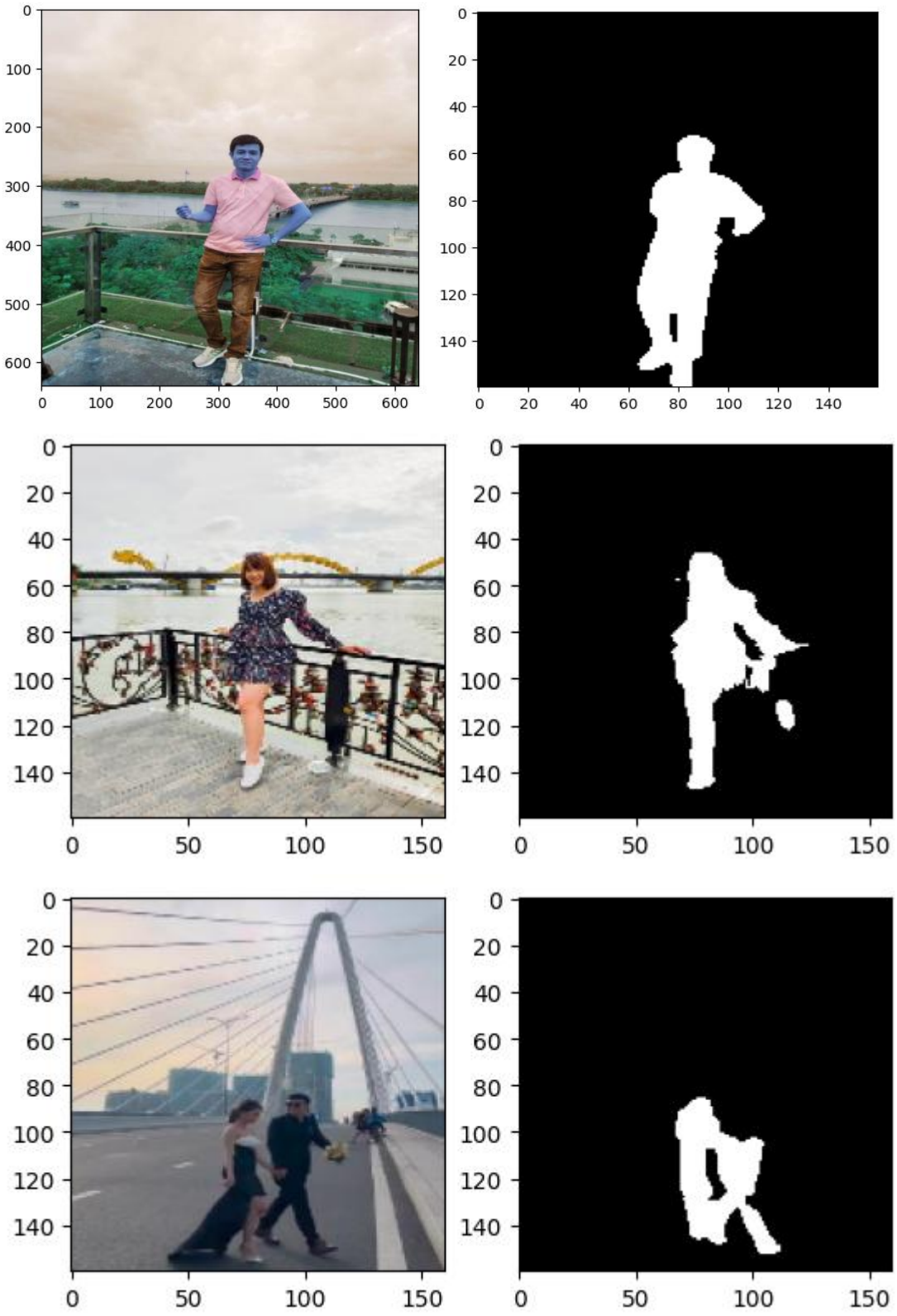


```
i= 100 loss= 0.015953538939356804 val_loss= 0.5946705937385559 mean_iou= 0.9157262444496155 val_mean_iou= 0.5985620021820068
1/1 [==============================] - 0s 31ms/step
```

Hình 23. Fit weight của model U-net

Lí do vì thiếu tài nguyện trên google colab và máy tính cái nhân, nên cần trong hai file colab khác nhau, trong đó 1 file chỉ fit dữ liệu của mô hình có cấu trúc Unet và một file dùng để dự đoán mask đồng thời chạy pretrained model deepfillv2.

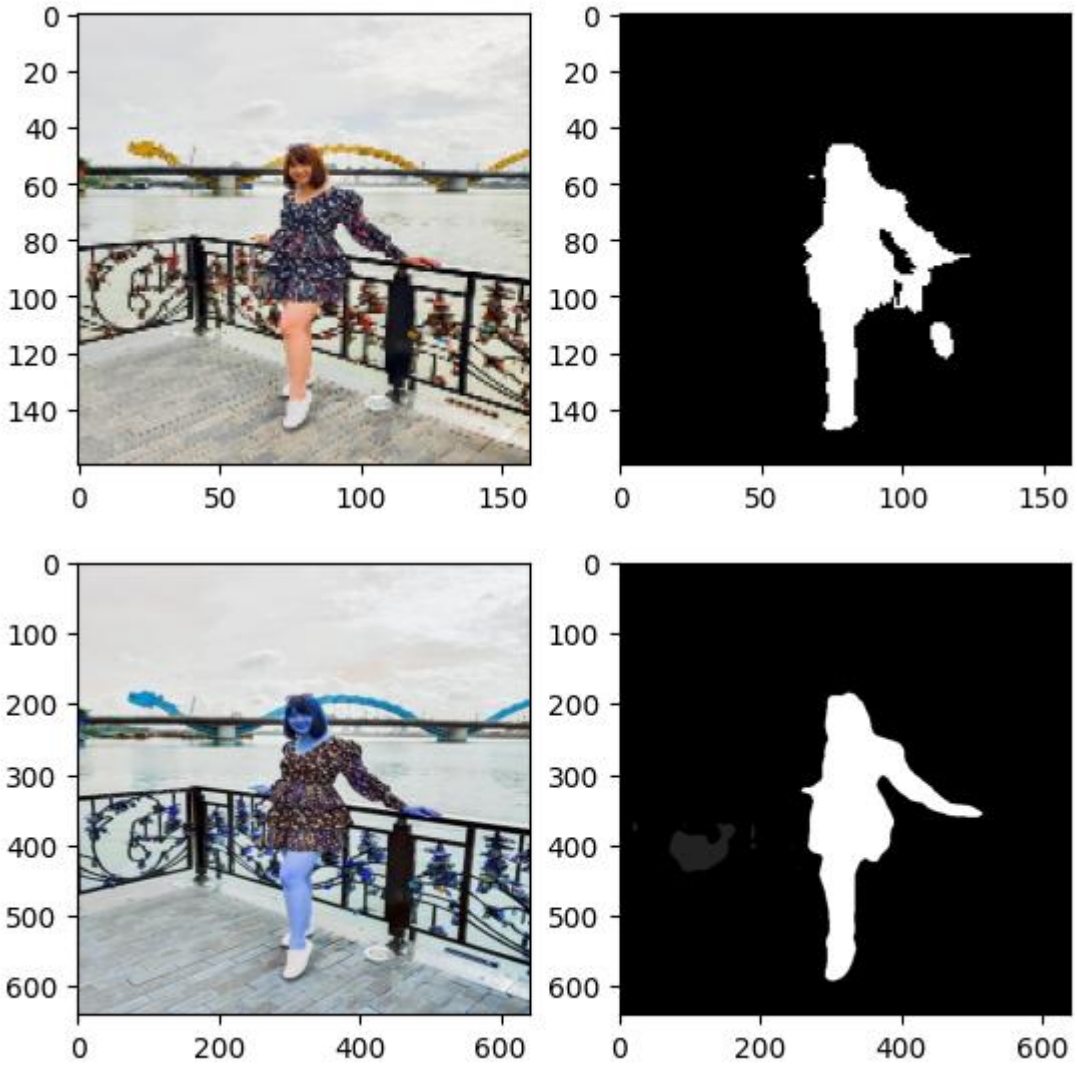
Vì trọng số weight của model huấn luyện vẫn chưa tốt nên đầu ra của vùng phân đoạn người vẫn chưa thể dự đoán đầy đủ/thiếu. Ma trận mask đã phần nào dự đoán được vị trí và phân vùng được người trong bức ảnh như trong Hình 24. Trong đó mô hình có độ chính xác không cao so khi ảnh đầu với có nhiều người trở lên. Theo đó mô hình Unet chủ yếu sử dụng xác định vật thể khối u trong y tế, thường không dùng để phân đoạn được đa đối tượng đều là người trong ảnh. Vì vậy mô hình sẽ phù hợp hơn đối với các tấm ảnh selfie cá nhân.



Hình 24. Dự đoán với trọng số của mô hình model1.h5

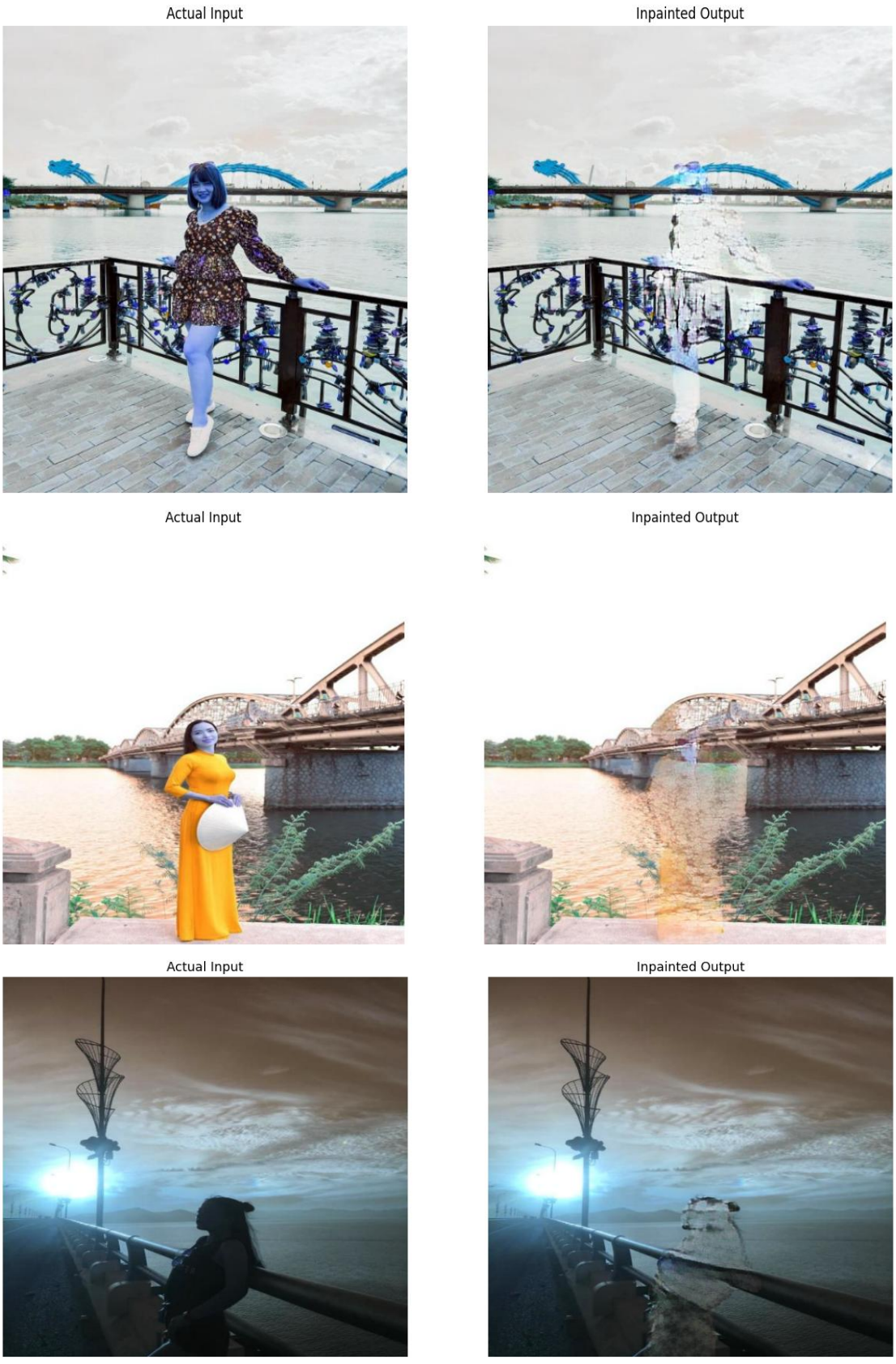
Ngoài ra vẫn tồn tại các model pretrained dùng cho segment vật thể khác như deeplabv3. Theo đó deeplab là mô hình phân đoạn thực hiện segment từng lớp khác nhau, hiện tại phiên bản v3 gồm 21 class như nhận diện người, chó, mèo,… So sánh với mô hình sử dụng cấu trúc Unet của nhóm chúng tôi

thì mô hình pretrained deeplabv3_resnet101 có thể nhẫn diện với độ chính xác cao hơn nhưng vì mô hình này nhận diện nhiều class nên vẫn tồn tại các vật thể khác nhận diện vào khung hình. Đồng thời ohương pháp đệm (Padding method) trong cấu trúc Unet, điều này giúp kiến trúc có thể phân đoạn hình ảnh được mượt mà hơn dù kích thước huấn luyện model đầu ra chỉ là 160x160.



Hình 25. So sánh mô hình model1.h5 (A) và deeplabv3_resnet101 (B)

Hình 26 là đầu ra của mô hình đầu ra của pretrained model deepfillv2. Qua cấu trúc đã nói bên trên, ta dùng 2 đầu vào là ảnh ban đầu và mask đã dự đoán từ model1.h5. Đánh giá kết quả có thể thực hiện khá tốt xóa người trong ảnh và khôi phục lại phần khung cảnh đằng sau. Từ đó, có thể nhận diện rõ ràng hơn địa điểm trong ảnh. Dù vậy vẫn tồn tại một vài yếu tố ảnh hưởng như phần bóng của đối tượng vẫn ảnh hưởng tới mô hình khôi phục ảnh.



Hình 26. Dự đoán đầu ra của mô hình Deepfillv2

Kết luận

Trong dự án này, kết quả đầu ra thu được ma trận phân đoạn (segment) dù chưa tối ưu trên tập validtion cũng như kết quả test vẫn chưa có độ chĩnh xác quá cao cụ thể là 90% trên tập train nhưng chỉ có 60% trên validation. Kết quả được sự so sánh khả quan với pretrained model deeplabv3_resnet101, quá đó có thể cải thiện dần khả năng huấn luyện của mô hình. Chúng tôi đã phần nào đó giải quyết được hướng đi của bài toán là phân đoạn ảnh và có thể khôi phục lại phần khung cảnh đằng sau. Quá trình thực hiện vẫn chưa đạt được mong muốn như tài nguyên, bộ nhớ không đủ hay chưa đủ thời gian để phân đoạn số lượng ảnh lớn hơn và vẫn dùng luôn pretrained model deepfillv2. Hướng phát triển tương lai của dự án này là tạo ra sản phẩm được sử dụng để xóa vật thể mong muốn và khôi phục bức ảnh.

Google Drive:

https://colab.research.google.com/drive/1zALB9PZGHrKh2WJyJwj2eWeuAyxoU1qo?usp=sharing

https://colab.research.google.com/drive/1OG-vhWTm8S0RB4314zdv_4vh-qxYYpL9?usp=sharing

Tài liệu tham khảo

[1] https://en.wikipedia.org/wiki/Computer_vision
[2] https://en.wikipedia.org/wiki/Image_segmentation
[3] https://en.wikipedia.org/wiki/U-Net
[4] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. (2018). *Free-Form Image Inpainting with Gated Convolution*. http://arxiv.org/abs/1806.03589
[5] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. S. (2018). *Generative Image Inpainting with Contextual Attention*. http://arxiv.org/abs/1801.07892
[6] Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., & Catanzaro, B. (2018). *Image Inpainting for Irregular Holes Using Partial Convolutions*. http://arxiv.org/abs/1804.07723
[7] Nazeri, K., Ng, E., Joseph, T., Qureshi, F. Z., & Ebrahimi, M. (2019). *EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning*. http://arxiv.org/abs/1901.00212
[8] https://github.com/nipponjo/deepfillv2-pytorch
[9] https://roboflow.com/
[10] Tan, M., & Le, Q. v. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. http://arxiv.org/abs/1905.11946
[11] https://keras.io/api/applications/mobilenet/
[12] https://pytorch.org/hub/ultralytics_yolov5/
[13] https://www.tensorflow.org/
[14] https://pytorch.org/
[15] https://wiki.cloudfactory.com/docs/mp-wiki/metrics/iou-intersection-over-union