

# Computer Vision

🚀 Image inpainting

Nhóm 11

Team: Đỗ Minh Tuấn - 20002175

Nguyễn Nhật Tùng - 20002177

Lã Anh Trúc - 20002169



# 1. Tổng quan



Why choose image inpainting?

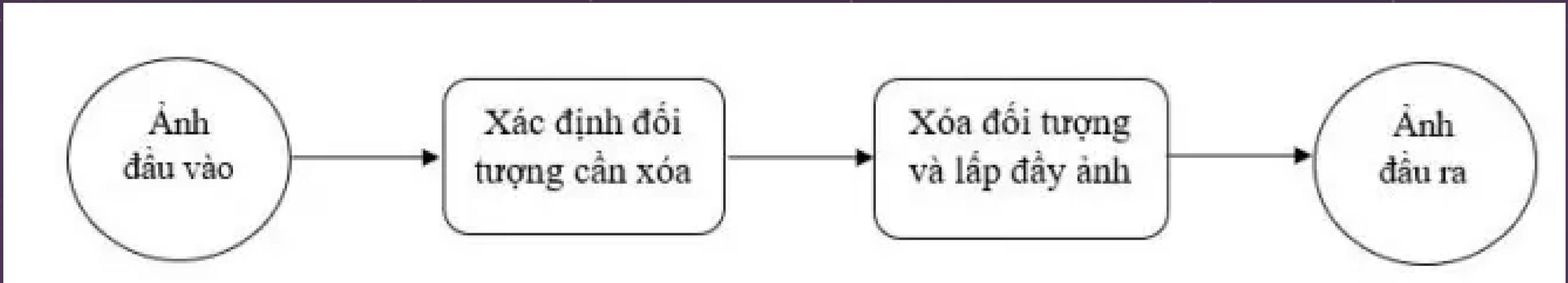
Việc khôi phục ảnh là 1 trong những thứ  
khá mới mẻ và nhiều tiềm năng

Nhu cầu khôi phục ảnh rất nhiều đến từ  
sở thích selfie của giới trẻ, mong muốn phục  
hồi ảnh của người thân, bạn bè, v.v

Mong muốn tìm hiểu về mảng thị giác máy tính



# Hướng tiếp cận của bài toán



Input



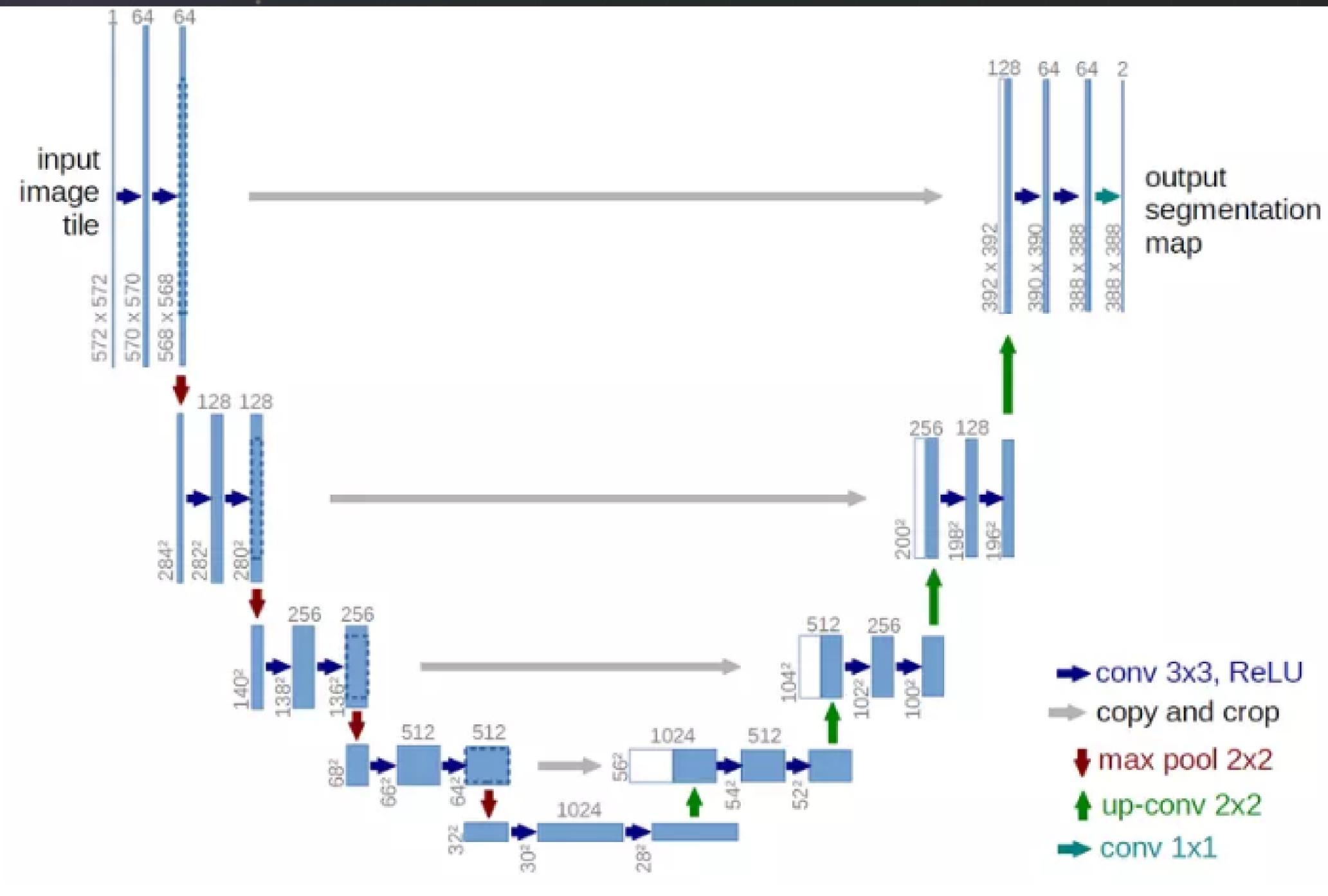
Output

## 🚀 2. Mô hình sử dụng

Mô hình Unet được áp dụng  
để nhầm phân đoạn hình ảnh

Mô hình Deepfillv2 của  
Pytorch để khôi phục ảnh





# Unet

Kiến trúc của nó được tạo thành từ hai phần:

Phần bên trái - trích xuất đặc trưng (encoder) và phần bên phải - giải mã đặc trưng (decoder).

Mục đích của encoder là cô đọng thông tin ngữ cảnh trong khi vai trò của decoder là giải mã thông tin, xác định chính xác các đối tượng.

# Tại sao lại là unet

toàn bộ kiến trúc không hề sử dụng một lớp fully connected nào

=> 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

U-net sử dụng Phương pháp đệm (Padding method), điều này giúp kiến trúc có thể phân đoạn hình ảnh được hoàn toàn

=> độ phân giải không bị hạn chế bởi dung lượng của bộ nhớ GPU



## Model deepfillv2

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.

Model Deepfillv2 được cải tiến từ

- Deepfillv1
- Partial Convolution
- EdgeConnect



# Deepfillv1

gồm 2 thành phần:

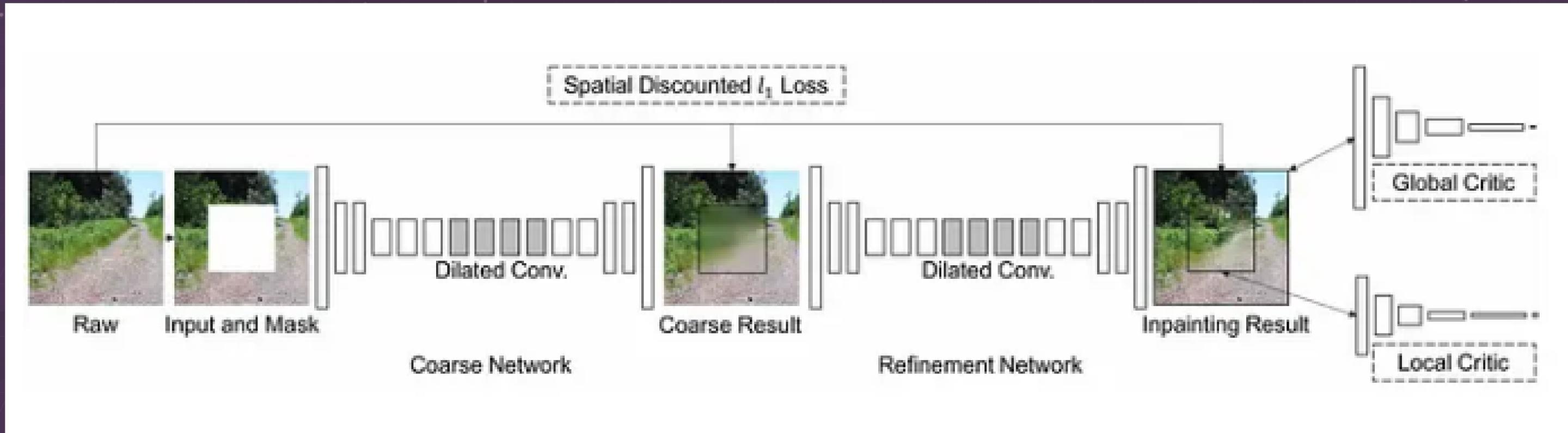
- + Kiến trúc khung ImageInpainting
- + Chú ý ngữ cảnh (Contextual attention).





# Deepfillv1

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

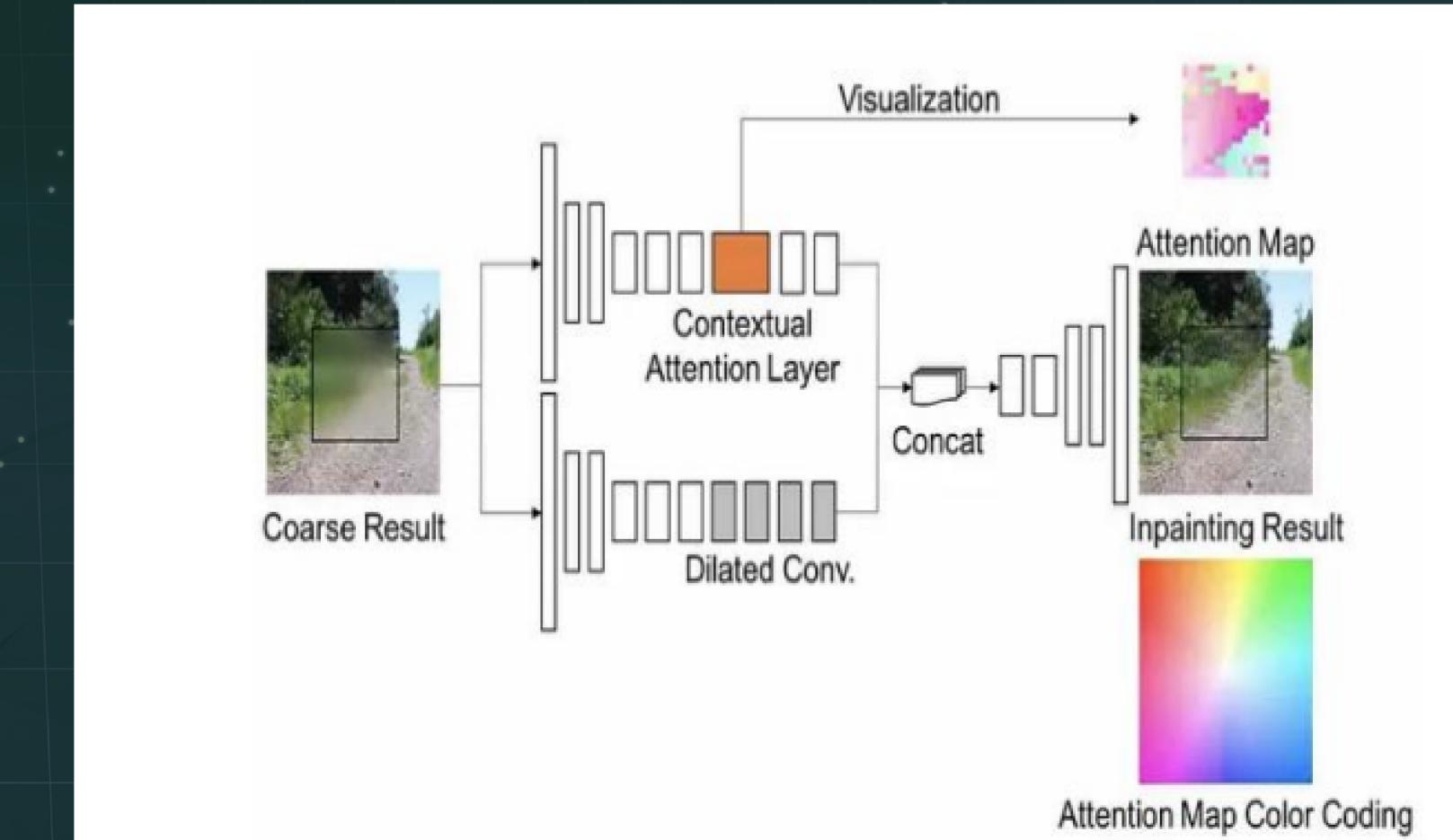
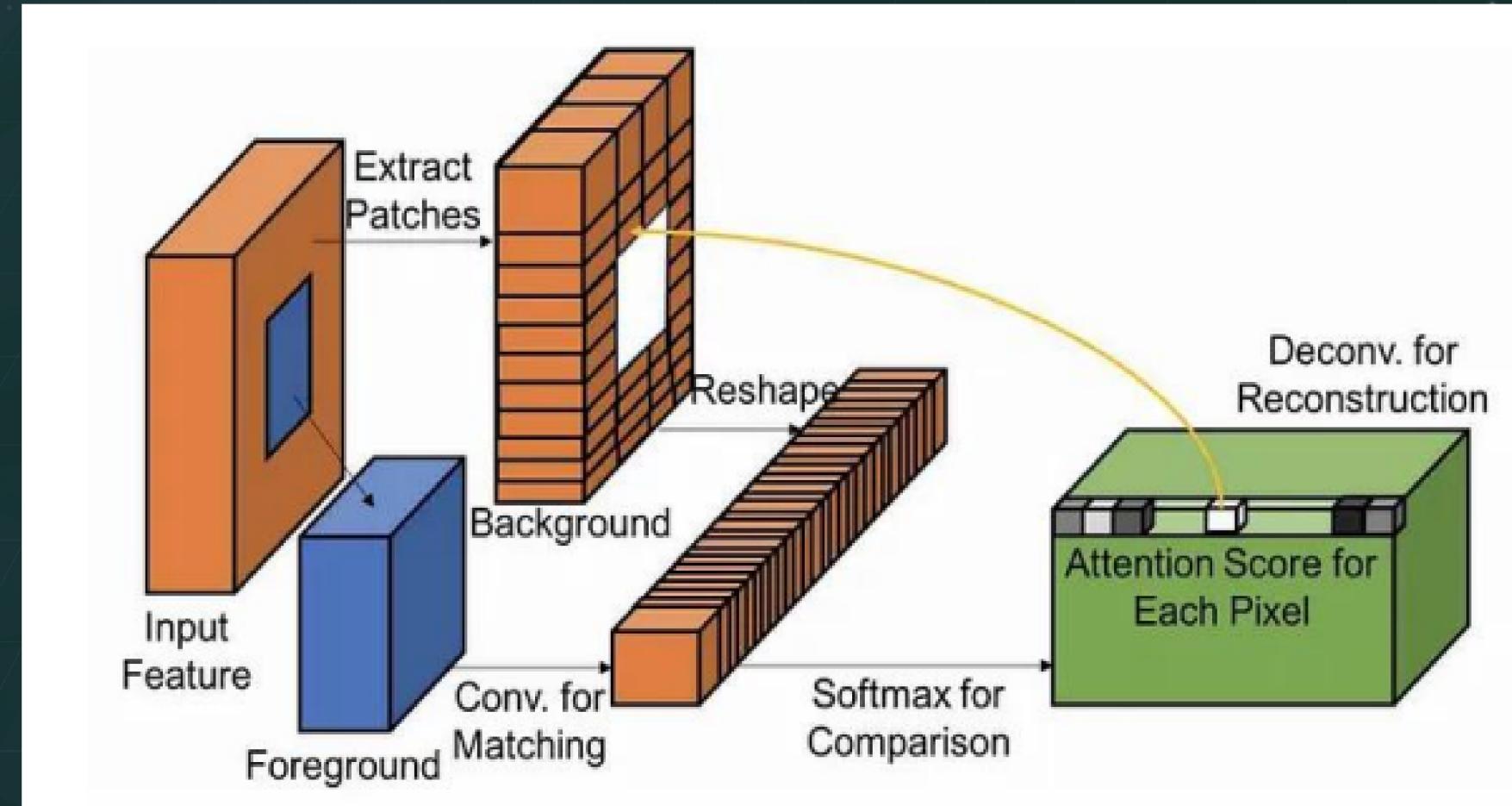


- + Một generator dành cho việc tái tạo và một generator khác dành cho phân loại
- + Bộ phân biệt toàn cục lấy toàn bộ hình ảnh làm đầu vào trong khi bộ phân biệt cục bộ lấy vùng được lấp đầy làm đầu vào



# Deepfillv1

Cơ chế chú ý theo ngữ cảnh: 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



## 🚀 partial convolution

Tách các pixel bị thiếu khỏi các pixel hợp lệ trong quá trình tích chập sao cho kết quả của các tích chập chỉ phụ thuộc vào các pixel hợp lệ

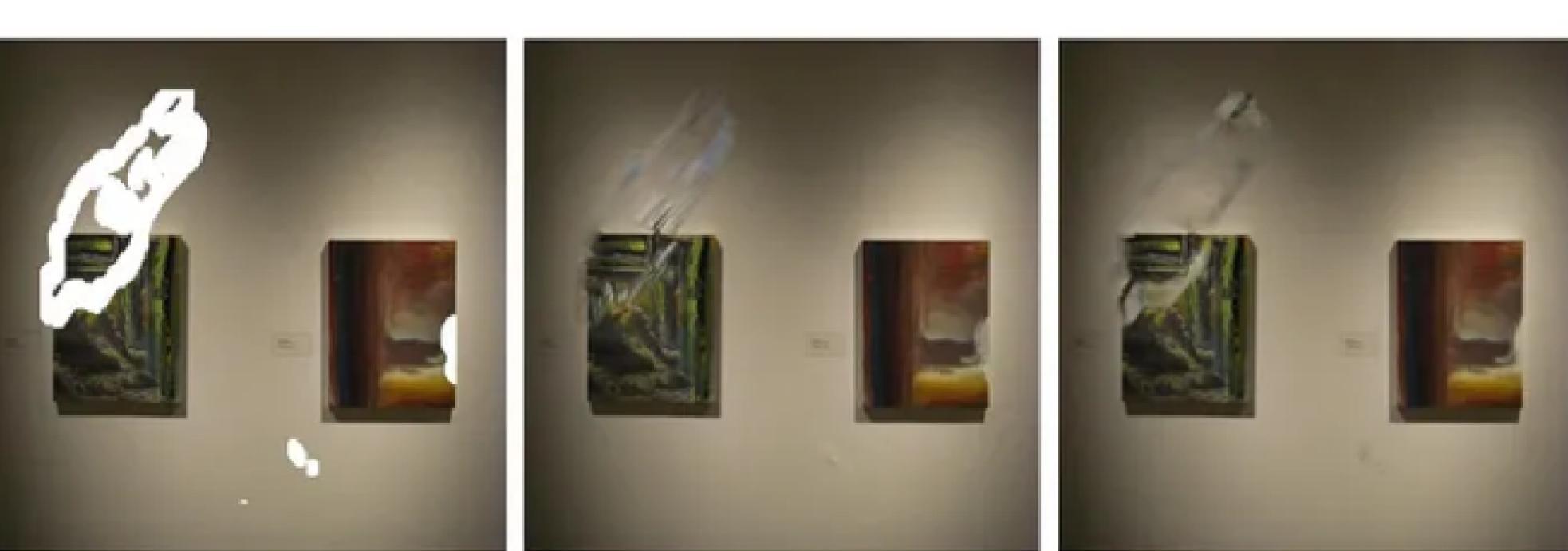
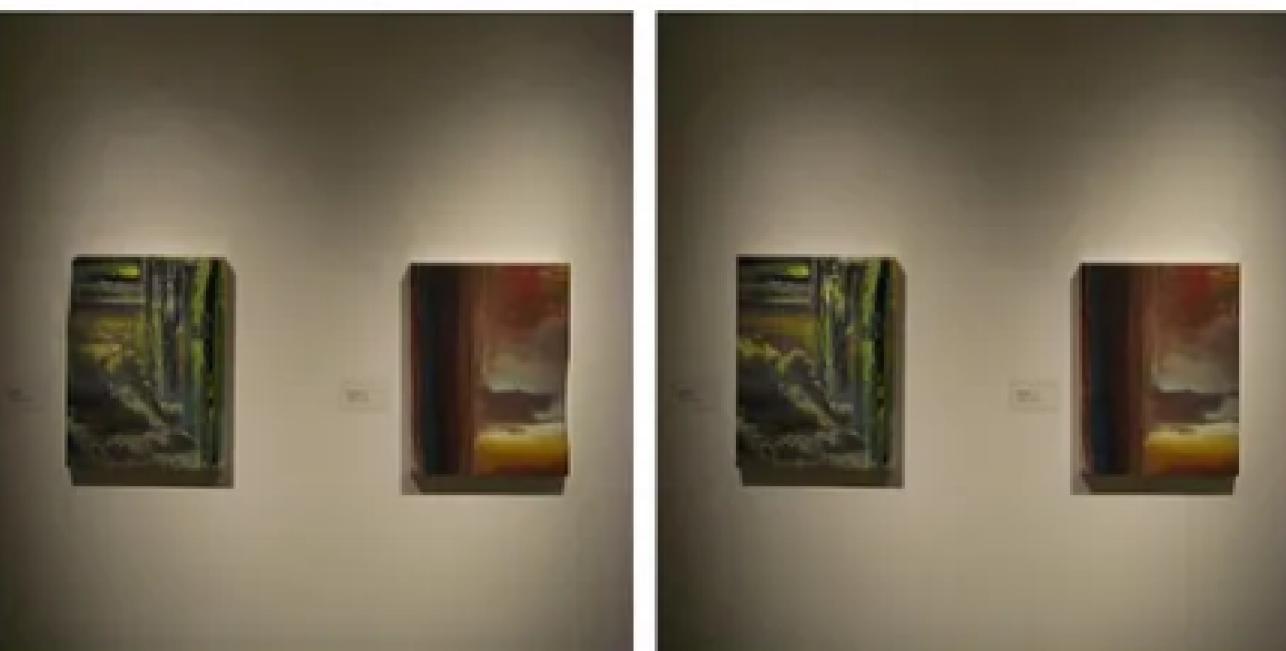


Image with hole

Iizuka et al.[10]

Yu et al.[38]

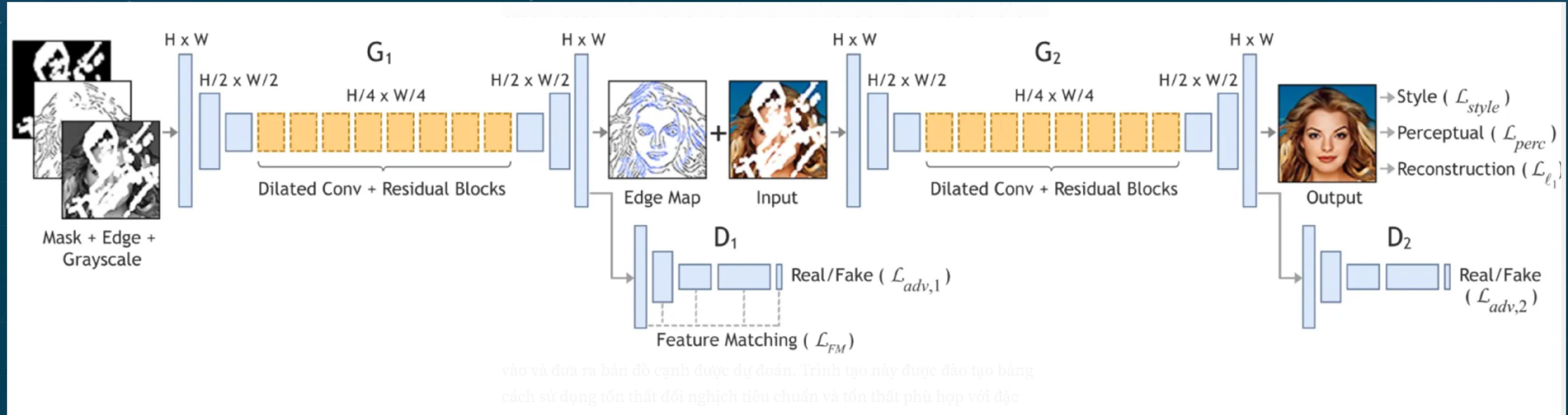


Partial Conv

Ground Truth



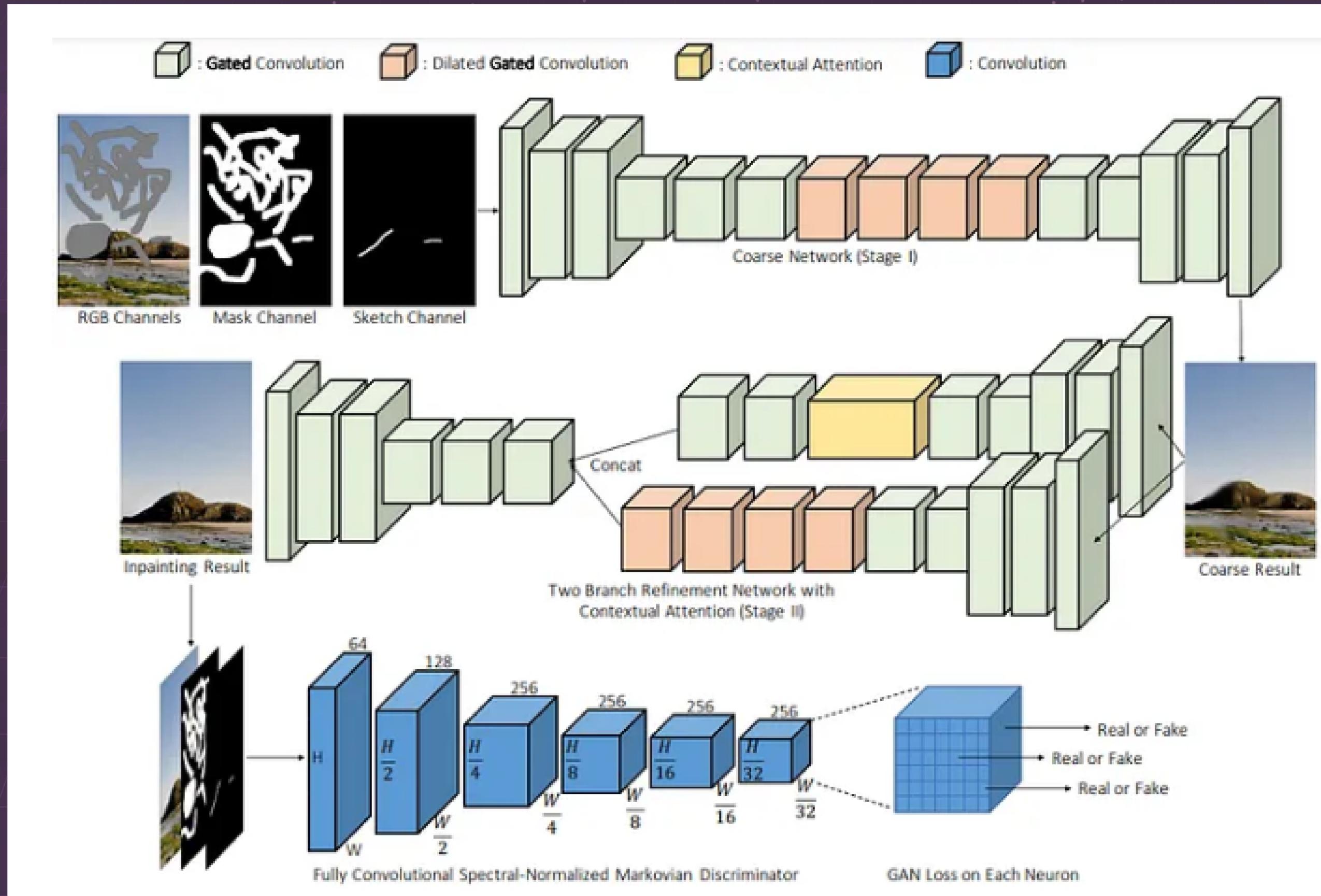
dự đoán cạnh và hoàn thiện hình ảnh.



vào và đưa ra bản đồ cạnh được dự đoán. Trình tạo này được đào tạo bằng cách sử dụng tổn thất đối nghịch tiêu chuẩn và tổn thất phù hợp với đặc



# model deepfillv2



### 3. Thực hiện dự án

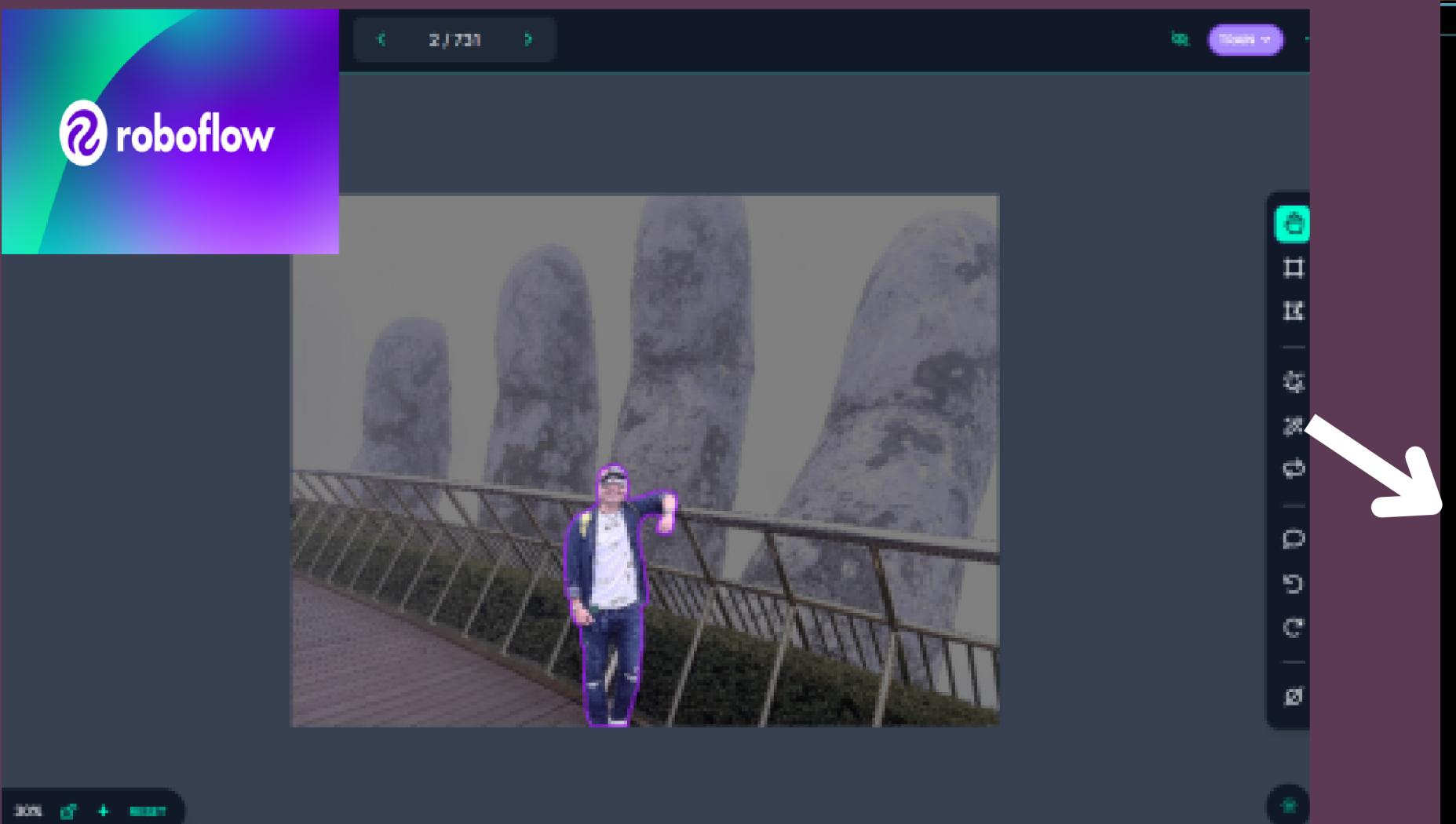
#### 3.1. Chuẩn bị data



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%

### 3. Thực hiện dự án

#### 3.2. Xử lý dữ liệu



D:\Hoc\_tap\TGMT\Cut\_Person2\test\0\_annotations.cocojson\1\_annotations\0\#image\_id  
"height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":56,"license":1,"file\_name":"ctt\_87.jpg.rf.c4e36fc62ab3d72hf013e2ab89621c76.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":57,"license":1,"file\_name":"ctt\_184.jpg.rf.d9dc039ccff58827a8bdd91a9992b425.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":58,"license":1,"file\_name":"ctt\_27.jpg.rf.e482b354c2626889abe9de315b31c497.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":59,"license":1,"file\_name":"cr\_126.jpg.rf.dsdc45f4f660131cbf6a6af83b5c784c.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":60,"license":1,"file\_name":"ctn\_169.jpg.rf.ec191cccd84eab42bf2ea5bc668d914fd.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":61,"license":1,"file\_name":"cr\_128.jpg.rf.e746b9e161f7d1c9b5c3a098d522197d1.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":62,"license":1,"file\_name":"CTT\_51.jpg.rf.f11782e45dc4657a869cd9fb58cd822.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":63,"license":1,"file\_name":"ctt2\_168.jpg.rf.e8cdea2b1fefff843d016f6bcbe0fc6974.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":64,"license":1,"file\_name":"cv\_184.jpg.rf.efc02a77f957a8b3d46498c568c3ca7.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":65,"license":1,"file\_name":"cr\_195.jpg.rf.f92c65538a2bdhc7e4c4c521fec2539.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":66,"license":1,"file\_name":"ctt2\_281.jpg.rf.fsdb542e285a8aa0e8837bbe0c14aaddf.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":67,"license":1,"file\_name":"CTT\_8.jpg.rf.fa154dc299bde12ded2814edf6e5622.jpg","height":648,"width":648,"date\_captured":"2023-12-11T05:58:11+00:00"}, {"id":68,"license":1,"file\_name": "cv\_235.jpg.rf.ff521a8f29662138b633f91ae2de0174.jpg", "height":648,"width":648,"date\_captured": "2023-12-11T05:58:11+00:00"}, {"id":69,"license":1,"file\_name": "ctn\_7.jpg.rf.fffac81a4ad8ch296447e576d8a7821241.jpg", "height":648,"width":648,"date\_captured": "2023-12-11T05:58:11+00:00"}, {"annotations": [{"id": 0, "image\_id": 0, "category\_id": 1, "bbox": [147, 389, 238, 603], "area": 76077.885, "segmentation": [[320, 648, 389, 841, 597, 333, 343, 365, 548, 952, 335, 238, 481, 524, 356, 571, 474, 667, 353, 524, 434, 286, 377, 985, 411, 429, 372, 825, 351, 238, 389, 841, 318, 895, 282, 159, 328, 147, 392, 368, 162, 54, 425, 143, 218, 413, 457, 143, 238, 683, 474, 667, 257, 816, 457, 143, 236, 698, 534, 857, 253, 968, 554, 667, 275, 302, 648, 328, 648]], "iscrowd": 0}, {"id": 1, "image\_id": 1, "category\_id": 1, "bbox": [178, 289, 227, 123, 358, 889], "area": 79694.988, "segmentation": [[365, 819, 497, 575, 314, 322, 441, 274, 314, 322, 484, 755, 328, 517, 349, 976, 289, 987, 289, 111, 231, 179, 292, 154, 298, 872, 343, 89, 221, 639, 486, 277, 182, 51, 447, 361, 178, 454, 541, 782, 138, 621, 636, 844, 405, 577, 648, 373, 131, 684, 889, 365, 819, 497, 575]], "iscrowd": 0}, {"id": 2, "image\_id": 2, "category\_id": 1, "bbox": [318, 233, 384, 232, 407, 222], "area": 115745.398, "segmentation": [[344, 91, 649, 343, 313, 631, 111, 349, 781, 615, 556, 372, 856, 683, 333, 389, 78, 586, 556, 312, 974, 451, 667, 337, 725, 431, 111, 348, 12, 386, 111, 358, 483, 392, 222, 377, 645, 415, 556, 376, 848, 436, 111, 394, 411, 429, 444, 377, 645, 372, 222, 327, 345, 367, 222, 328, 16, 323, 333, 297, 886, 311, 667, 384, 99, 382, 778, 315, 369, 261, 111, 281, 836, 232, 778, 235, 529, 242, 778, 221, 956, 277, 778, 285, 988, 287, 222, 288, 399, 321, 667, 157, 285, 339, 444, 138, 922, 385, 556, 159, 681, 409, 444, 173, 253, 511, 111, 119, 18, 681, 111, 111, 737, 618, 556, 138, 14, 678, 185, 21, 626, 111, 188, 421, 649, 344, 91, 648]], "iscrowd": 0}, {"id": 3, "image\_id": 3, "category\_id": 1, "bbox": [332, 238, 238, 398], "area": 80700, "segmentation": [[492, 222, 618, 451, 111, 566, 667, 431, 111, 587, 5, 462, 222, 496, 667, 485, 556, 559, 167, 483, 333, 591, 667, 518, 588, 333, 511, 111, 551, 333, 588, 889, 472, 5, 528, 518, 333, 521, 111, 567, 5, 517, 778, 684, 167, 552, 222, 687, 5, 551, 111, 579, 167, 562, 222, 564, 167, 551, 111, 562, 5, 552, 222, 428, 333, 561, 111, 398, 333, 557, 778, 342, 5, 548, 288, 333, 513, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 167, 421, 111, 629, 443, 333, 612, 5, 417, 778, 587, 5, 418, 889, 356, 667, 438, 683, 333, 476, 667, 628, 492, 222, 618]], "iscrowd": 0}, {"id": 4, "image\_id": 4, "category\_id": 1, "bbox": [259, 79, 156, 736, 538, 983], "area": 84472.682, "segmentation": [[318, 625, 514, 969, 313, 75, 506, 221, 320, 455, 327, 316, 25, 406, 822, 326, 25, 367, 851, 331, 25, 313, 776, 355, 625, 333, 657, 360, 352, 337, 285, 361, 341, 336, 892, 362, 142, 334, 177, 356, 894, 326, 371, 364, 9, 327, 389, 301, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 167, 421, 111, 629, 443, 333, 612, 5, 417, 778, 587, 5, 418, 889, 356, 667, 438, 683, 333, 476, 667, 628, 492, 222, 618]]}, {"id": 5, "image\_id": 5, "category\_id": 1, "bbox": [259, 79, 156, 736, 538, 983], "area": 84472.682, "segmentation": [[318, 625, 514, 969, 313, 75, 506, 221, 320, 455, 327, 316, 25, 406, 822, 326, 25, 367, 851, 331, 25, 313, 776, 355, 625, 333, 657, 360, 352, 337, 285, 361, 341, 336, 892, 362, 142, 334, 177, 356, 894, 326, 371, 364, 9, 327, 389, 301, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 167, 421, 111, 629, 443, 333, 612, 5, 417, 778, 587, 5, 418, 889, 356, 667, 438, 683, 333, 476, 667, 628, 492, 222, 618]]}, {"id": 6, "image\_id": 6, "category\_id": 1, "bbox": [259, 79, 156, 736, 538, 983], "area": 84472.682, "segmentation": [[318, 625, 514, 969, 313, 75, 506, 221, 320, 455, 327, 316, 25, 406, 822, 326, 25, 367, 851, 331, 25, 313, 776, 355, 625, 333, 657, 360, 352, 337, 285, 361, 341, 336, 892, 362, 142, 334, 177, 356, 894, 326, 371, 364, 9, 327, 389, 301, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 167, 421, 111, 629, 443, 333, 612, 5, 417, 778, 587, 5, 418, 889, 356, 667, 438, 683, 333, 476, 667, 628, 492, 222, 618]]}, {"id": 7, "image\_id": 7, "category\_id": 1, "bbox": [259, 79, 156, 736, 538, 983], "area": 84472.682, "segmentation": [[318, 625, 514, 969, 313, 75, 506, 221, 320, 455, 327, 316, 25, 406, 822, 326, 25, 367, 851, 331, 25, 313, 776, 355, 625, 333, 657, 360, 352, 337, 285, 361, 341, 336, 892, 362, 142, 334, 177, 356, 894, 326, 371, 364, 9, 327, 389, 301, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 167, 421, 111, 629, 443, 333, 612, 5, 417, 778, 587, 5, 418, 889, 356, 667, 438, 683, 333, 476, 667, 628, 492, 222, 618]]}, {"id": 8, "image\_id": 8, "category\_id": 1, "bbox": [259, 79, 156, 736, 538, 983], "area": 84472.682, "segmentation": [[318, 625, 514, 969, 313, 75, 506, 221, 320, 455, 327, 316, 25, 406, 822, 326, 25, 367, 851, 331, 25, 313, 776, 355, 625, 333, 657, 360, 352, 337, 285, 361, 341, 336, 892, 362, 142, 334, 177, 356, 894, 326, 371, 364, 9, 327, 389, 301, 333, 288, 333, 488, 889, 265, 833, 497, 778, 241, 667, 474, 444, 238, 448, 241, 667, 448, 258, 333, 446, 667, 275, 833, 411, 111, 266, 667, 383, 333, 279, 167, 381, 111, 325, 356, 667, 325, 833, 358, 333, 332, 222, 362, 5, 337, 778, 418, 352, 222, 489, 167, 362, 222, 479, 167, 358, 505, 366, 667, 398, 889, 508, 331, 396, 667, 551, 667, 378, 889, 584, 16

### 3. Thực hiện dự án

#### 3.3. Xây dựng mô hình

##### Xử lý dữ liệu đầu vào

```
▶ 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]])
```

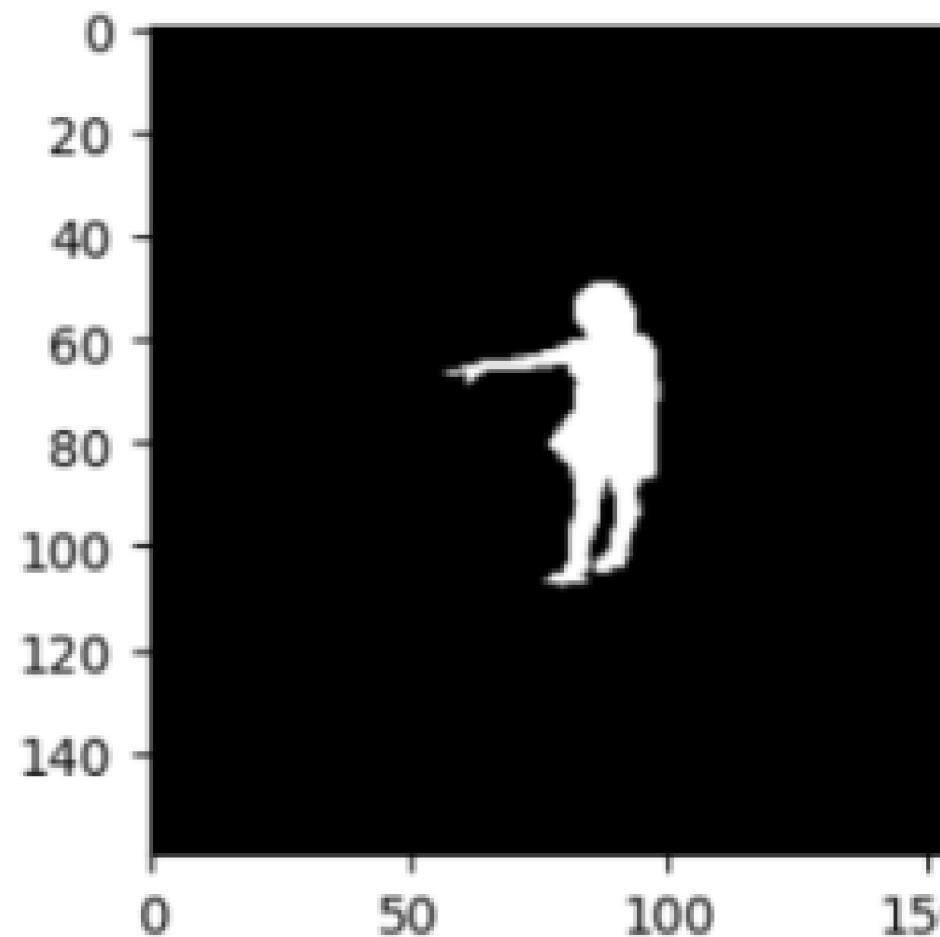
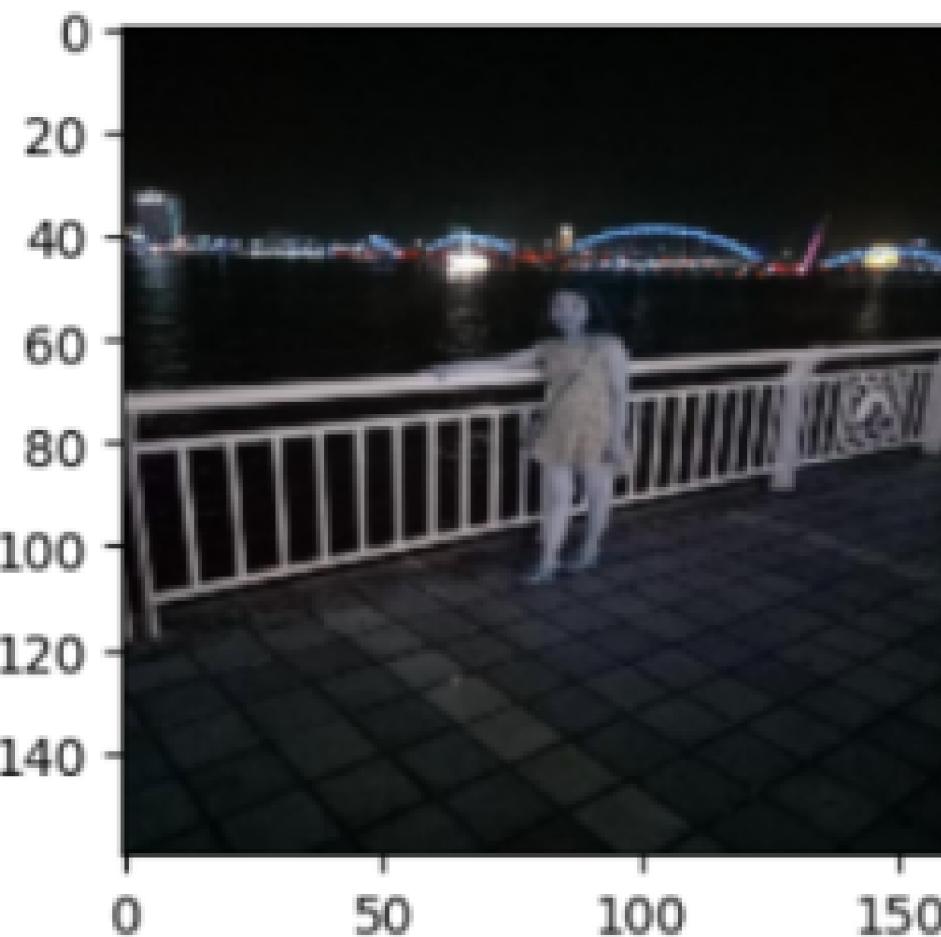
```
[ ] x_train = []
y_train = []
x_valid = []
y_valid = []
x_test = []
y_test = []
for i in range(int(3*len(train_data)/4)):
    img = cv2.imread(train_path+train_data[i][1])
    img = cv2.resize(img, (160, 160))/255.
    img = np.array(img, dtype=np.float32)
    x_train.append(img)

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

# Xử lý dữ liệu đầu vào

```
[ ] 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>



# Xây dựng mô hình có cấu trúc U-net

## IoU metrics (Intersection-Over-Union)

```
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})
```

```
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)
```

# pretrained model deepfillv2

```
#@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=1uMghK1883-9hDLhsII8lRbHCzCmmRwV-&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=1uMghK1883-9hDLhsII8lRbHCzCmmRwV-&export=download
To: /content/deepfillv2_WGAN_G_epoch40_batchsize4.pth
100% 64.8M/64.8M [00:01<00:00, 49.4MB/s]
```

# chèn input và mask từ model1.h5

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 1821 bytes

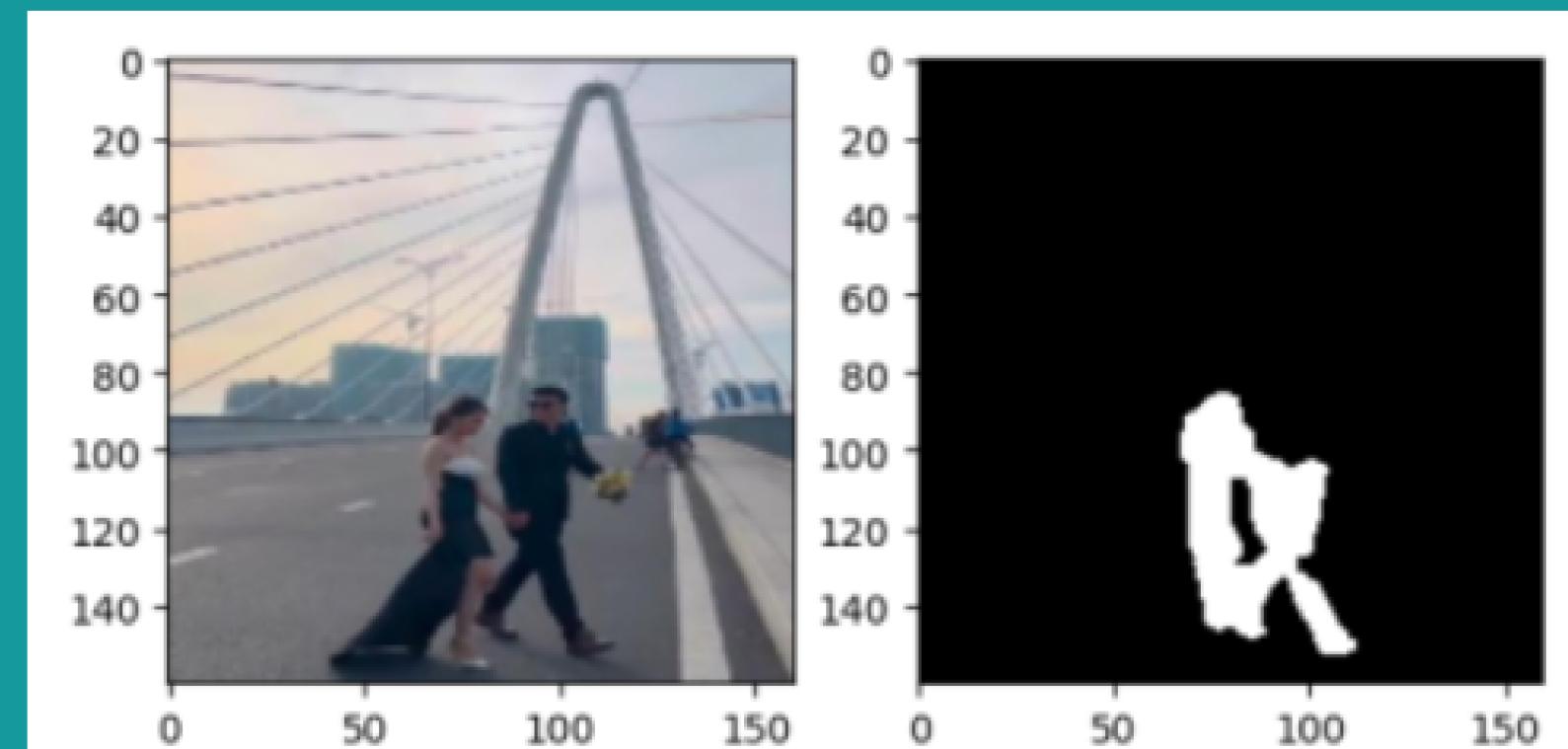
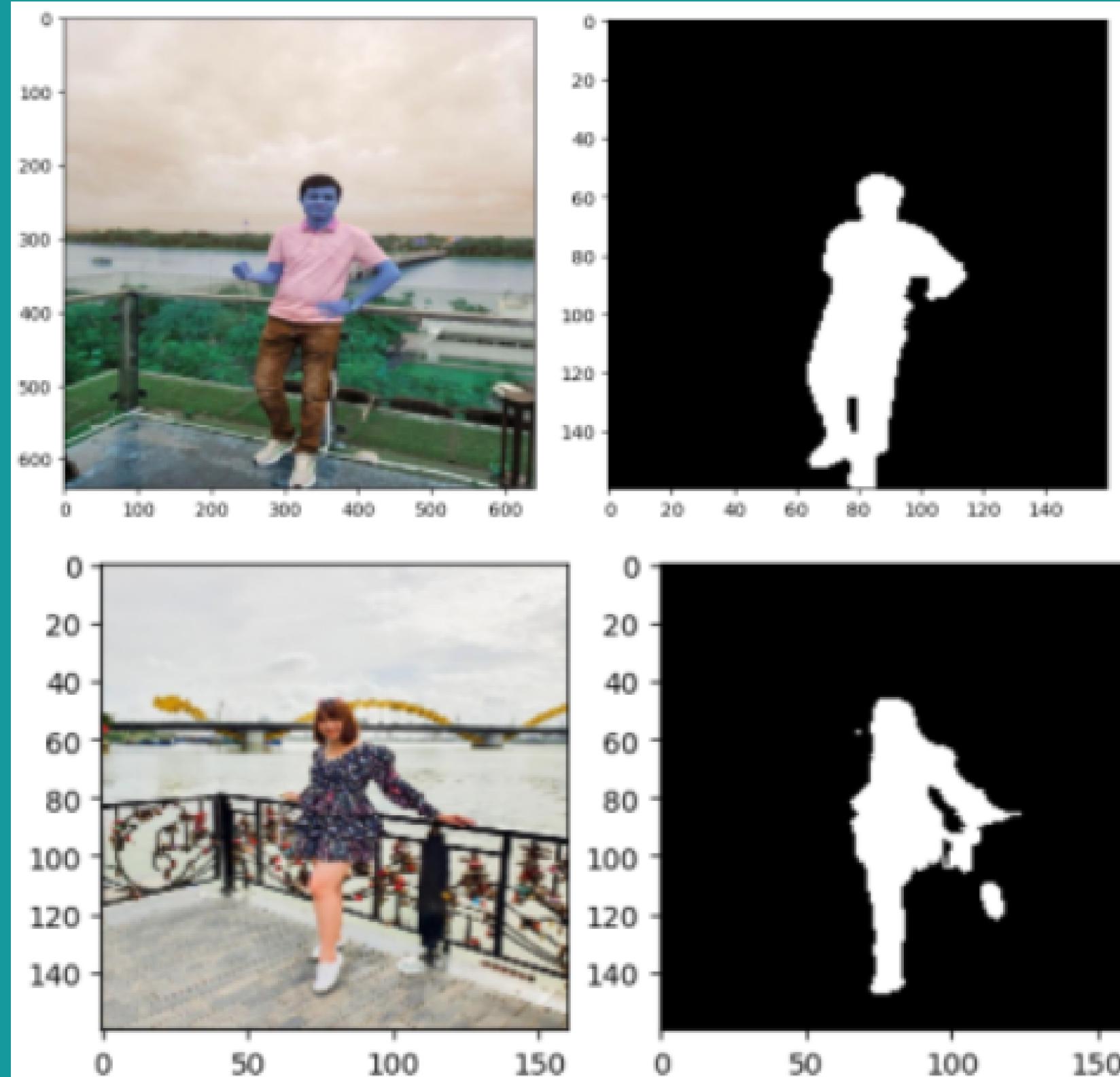
PLEASE RUN THE NEXT CELL

# chạy model

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

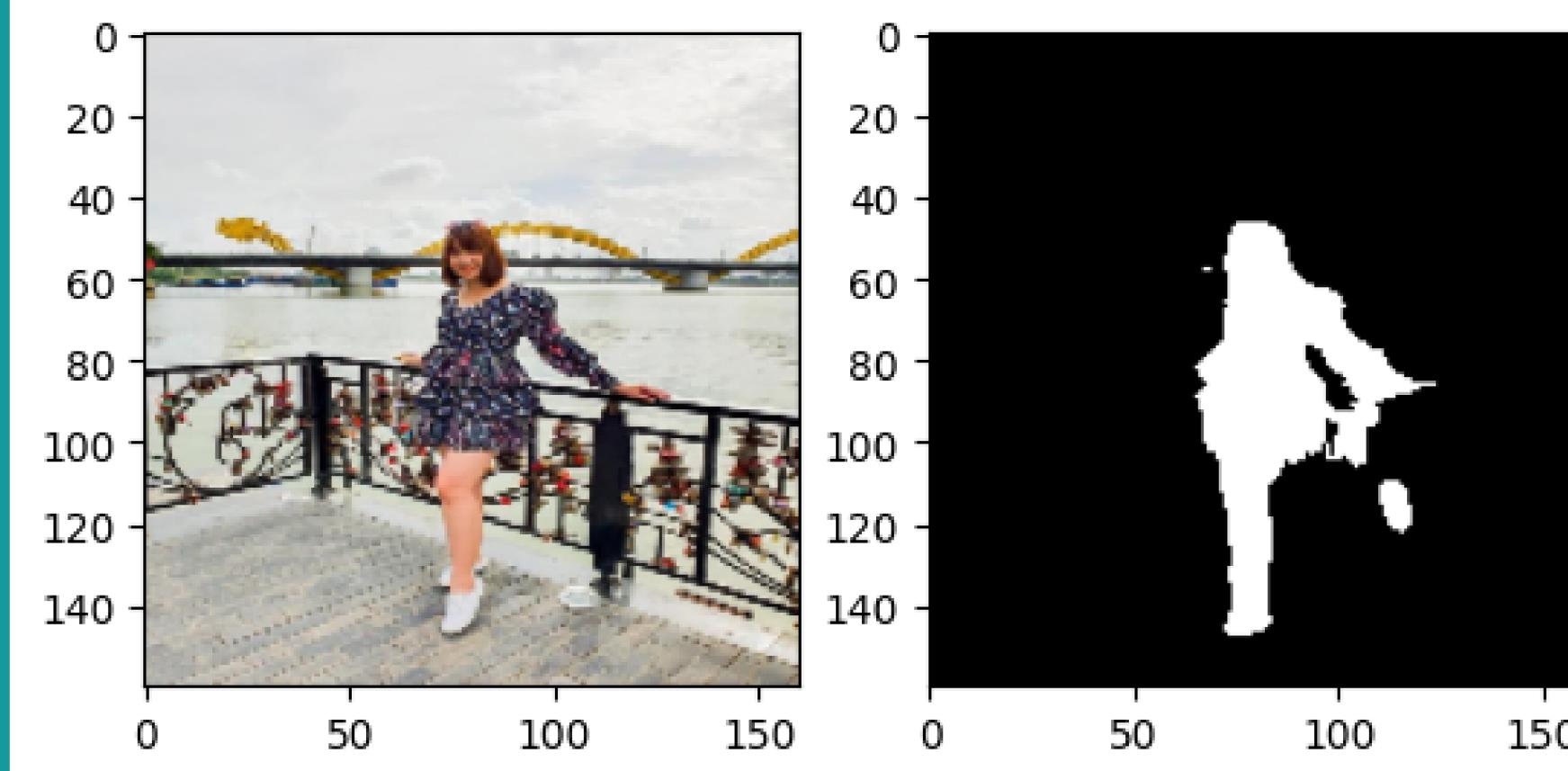
-- Generator is created!
-- Initialized generator with xavier type --
-- INPAINT: Loading Pretrained Model --
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:
    warnings.warn(_create_warning_msg(
-- Inpainting is finished --
```

# 4. Kết quả

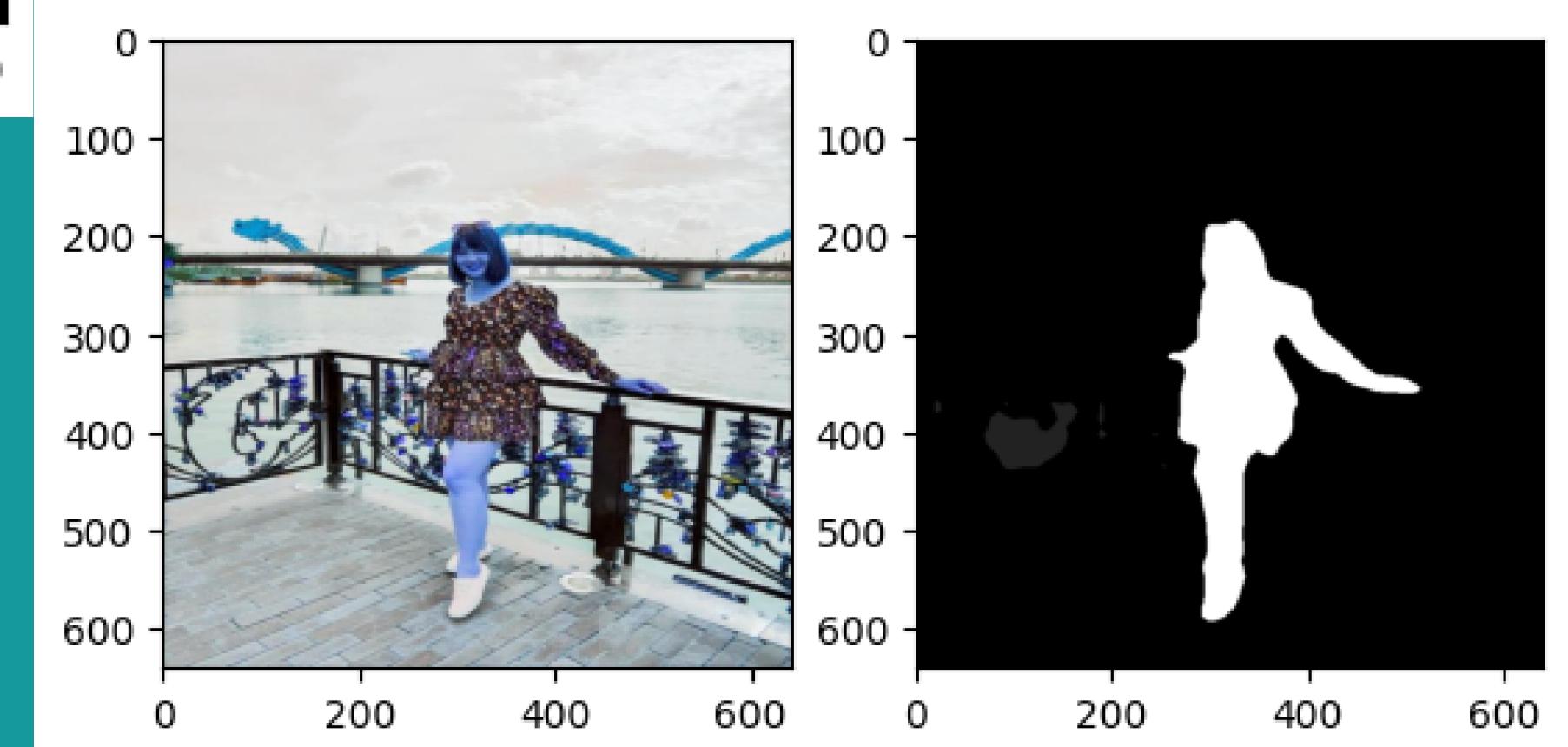


# 4. Kết quả

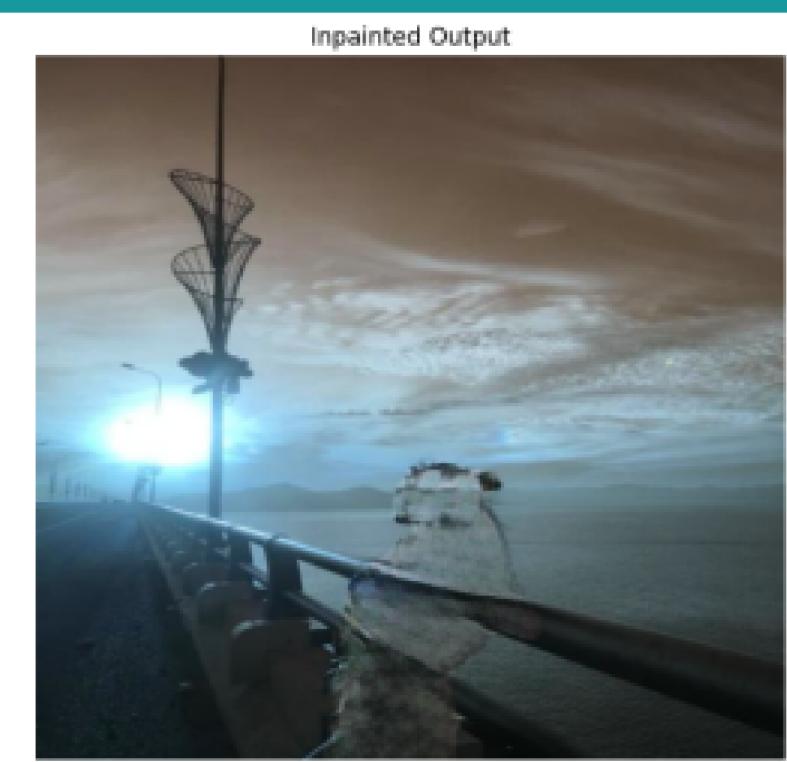
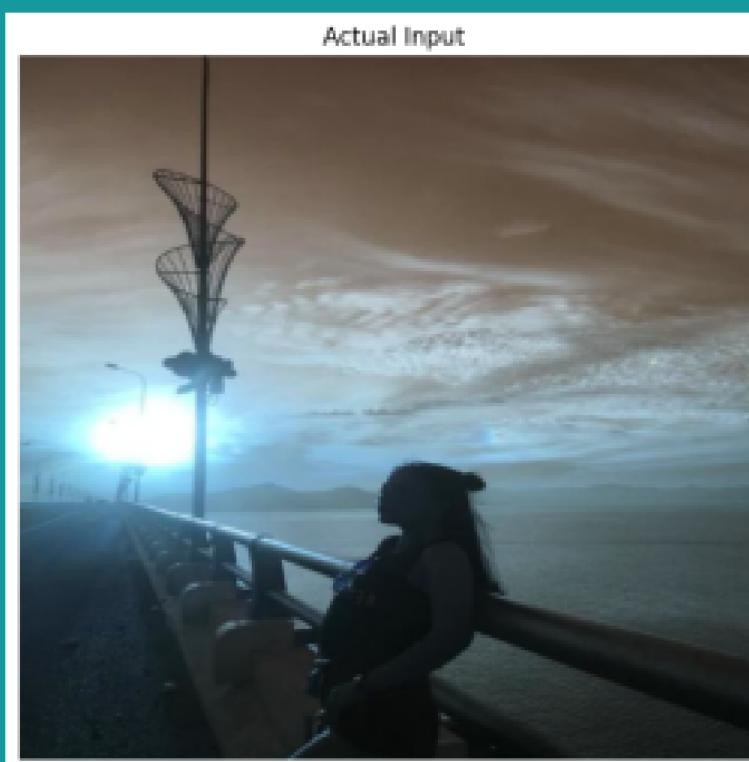
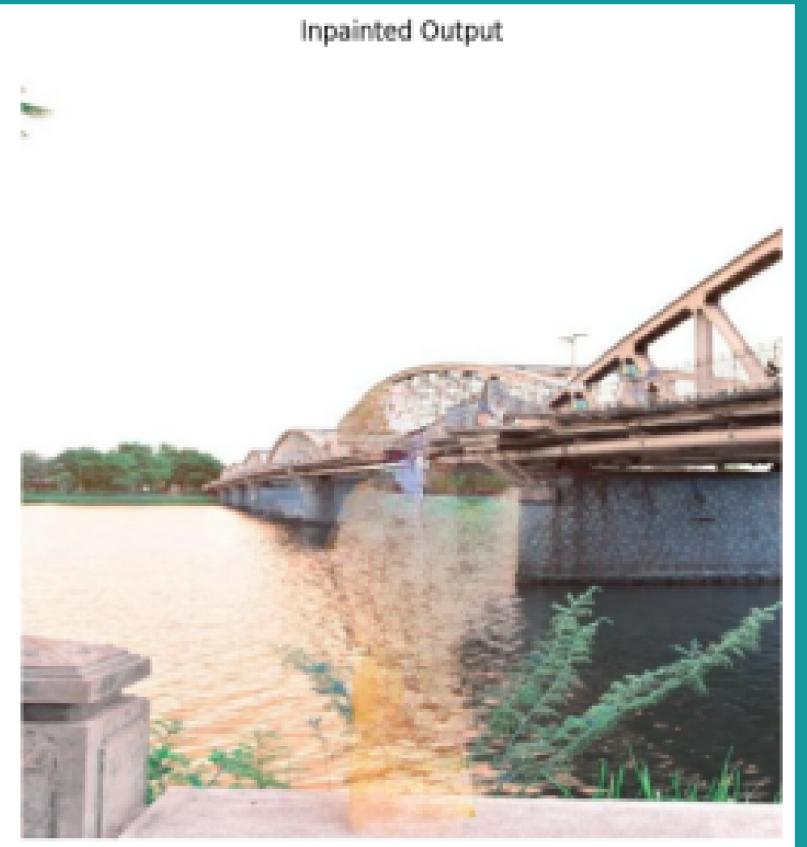
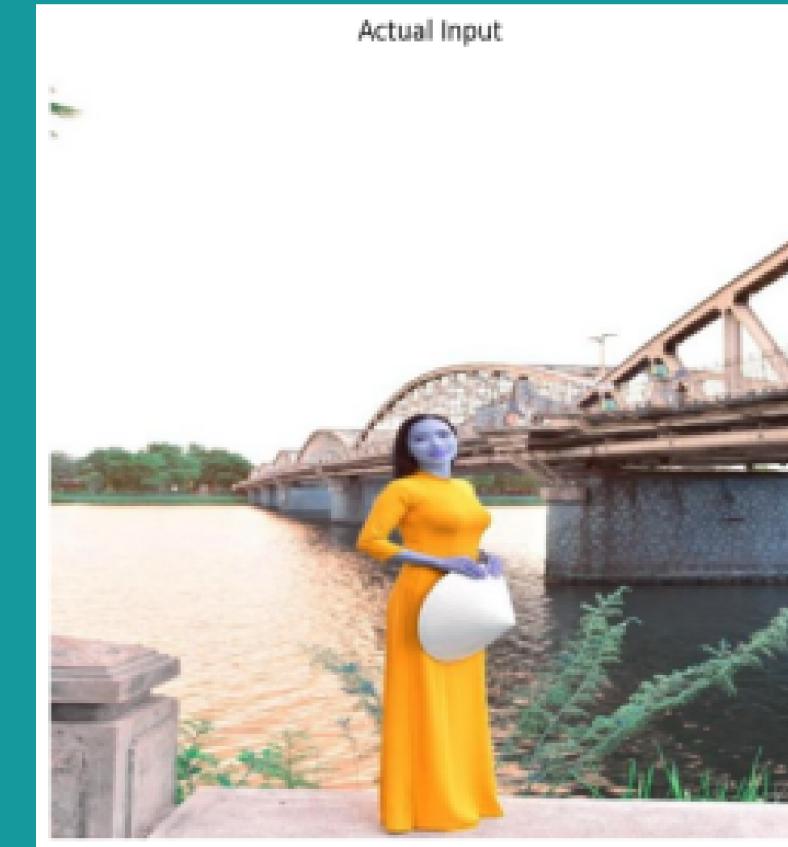
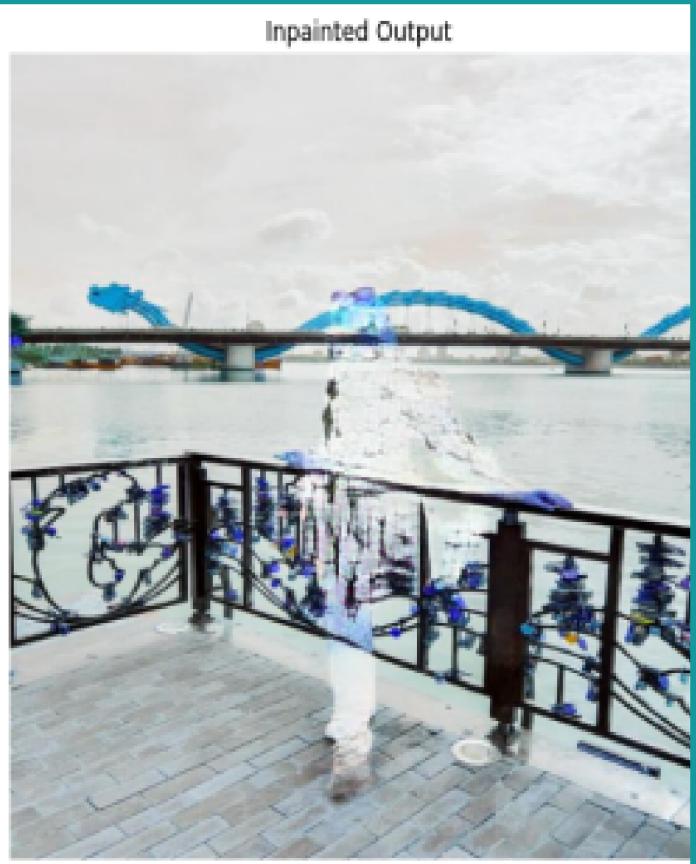
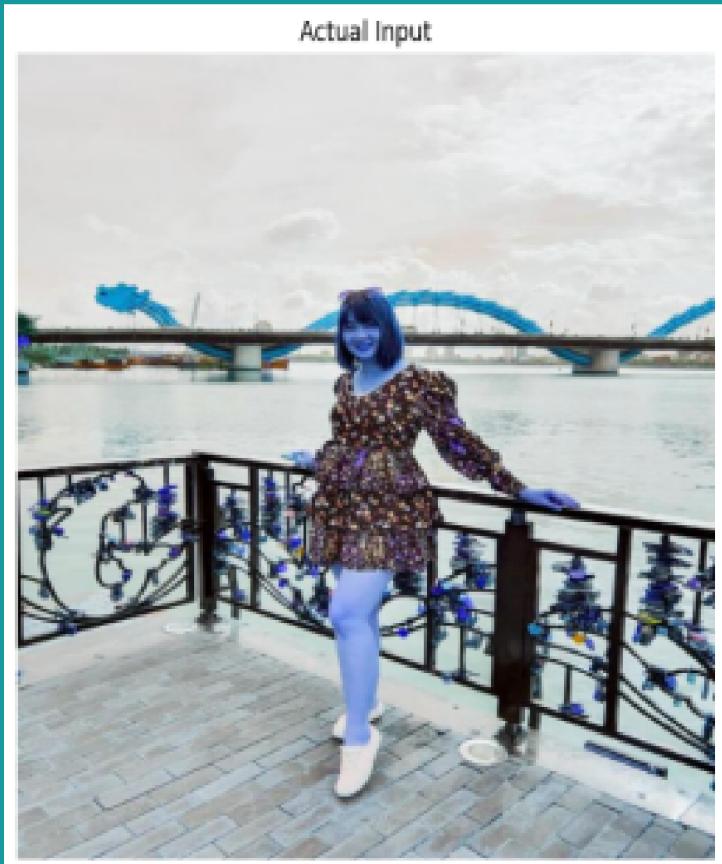
## Unet model



Deeplabv3\_resnet101



# 4. Kết quả



# Tài liệu tham khảo

- [1] <https://en.wikipedia.org/wiki/U-Net>
- [2] 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>
- [3] 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>
- [4] 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>
- [5] 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>
- [6] <https://github.com/nipponjo/deepfillv2-pytorch>
- [7] <https://roboflow.com/>
- [8] Tan, M., & Le, Q. v. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. <http://arxiv.org/abs/1905.11946>

THANK

YOU