

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

# BÁO CÁO ĐỒ ÁN 1

Đề tài: Khôi phục ảnh nha khoa bằng phương pháp Inpainting

Giảng viên hướng dẫn:

- ThS. Huỳnh Hồ Thị Mộng Trinh

Sinh viên thực hiện:

- Vũ Tuấn Hải - 17520433

SE121.K21

TP. Hồ Chí Minh , tháng 8 năm 2020

## LỜI CẢM ƠN

Trong quá trình thực hiện đề tài, em đã nhận được sự giúp đỡ, hướng dẫn và hỗ trợ từ gia đình, quý thầy cô. Nhờ đó em đã có được những trải nghiệm đáng nhớ và có kết quả như hôm nay. Nay em xin được gửi lời cảm ơn sâu sắc đến:

ThS. Huỳnh Hồ Thị Mộng Trinh, người đã trực tiếp hướng dẫn khóa luận. Trong quá trình thực hiện, cô đã tận tình hướng dẫn, giúp em giải quyết các vấn đề nảy sinh trong quá trình làm đề tài.

PGS.TS. Phạm Thế Bảo, người tạo ra một môi trường học thuật thuận lợi, giúp em có thể trình bày kết quả nghiên cứu cũng như giải đáp các thắc mắc liên quan đến chuyên môn thứ 7 mỗi tuần.

Cuối cùng, em xin được gửi lời cảm ơn tới gia đình là những người động viên và hỗ trợ giúp em những lúc khó khăn.

**Sinh viên**

Vũ Tuấn Hải

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Cộng hòa Xã hội Chủ nghĩa Việt Nam

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Độc lập - Tự do – Hạnh phúc

KHOA CÔNG NGHỆ PHẦN MỀM

---

## ĐỀ CƯƠNG CHI TIẾT

<b>Tên đề tài:</b> Khôi phục ảnh nha khoa bằng phương pháp Inpainting
<b>Giảng viên hướng dẫn:</b> ThS. Huỳnh Hồ Thị Mộng Trinh
<b>Thời gian thực hiện:</b> từ ngày 1/4 đến ngày 28/7
<b>Sinh viên thực hiện:</b> Vũ Tuấn Hải - 17520433
<b>Nội dung đề tài:</b>  Nhận thấy nhu cầu xử lý ảnh nha khoa trong thị trường hiện nay, đặc biệt khi những ứng dụng của GAN như FaceApp (chuyển đổi giới tính) nổi lên như một hiện tượng trong năm 2020. Em quyết định nghiên cứu và xây dựng một model chuyển đổi ảnh từ có mắc cài sang ảnh không còn mắc cài cho những người đang thực hiện niềng răng nhưng muốn biết răng của mình sẽ như thế nào mà không cần tháo mắc cài.  Để thực hiện được đề tài, em đã nghiên cứu và tìm hiểu những bài báo, nghiên cứu mới nhất (2017 – nay) có thể giải quyết bài toán kể trên trong những hội nghị xử lý ảnh và tạp chí nổi tiếng như CVPR, IEER, arxiv, ...  Kết quả nghiên cứu là một model kết hợp model truyền thống xóa mắc cài và một model deep learning thực hiện inpainting.

**Kế hoạch thực hiện:**

Đề tài được thực hiện trong vòng 16 tuần, với nội dung và thời gian cụ thể như sau:

STT	Nội dung thực hiện	Thời gian thực hiện
1	Nghiên cứu và phân rã yêu cầu bài toán	1/4/2020 – 10/4/2020
2	Tìm hiểu về phương pháp phân đoạn ảnh GraphCut	10/4/2020 – 14/4/2020
3	Tìm hiểu về phương pháp inpainting bằng WGAN - GP	15/4/2020 – 20/7/2020
4	Tìm hiểu việc crawl dữ liệu ảnh nha khoa	28/4/2020 – 30/4/2020
5	Hiện thực hóa và chạy thử nghiệm 2 model	1/5/2020 – 28/7/2020
6	Viết báo cáo	15/7/2020 – 28/7/2020

<p>Xác nhận của GVHD</p> <p>(ký và ghi rõ họ tên)</p> <p><b>Huỳnh Hồ Thị Mộng Trinh</b></p>	<p>TP.HCM, ngày 1 tháng 8 năm 2020</p> <p>(ký và ghi rõ họ tên)</p> <p><b>Vũ Tuấn Hải</b></p>
---	---

## MỤC LỤC

LỜI CẢM ƠN .....	1
ĐỀ CƯƠNG CHI TIẾT .....	2
MỤC LỤC .....	4
DANH MỤC HÌNH ẢNH .....	6
DANH MỤC TỪ TIẾNG ANH.....	8
MỞ ĐẦU .....	9
Chương 1. Giới thiệu đề tài.....	10
1.1.    Giới thiệu .....	10
1.2.    Mục tiêu .....	10
Chương 2. Model GraphCut .....	12
2.1. Giới thiệu .....	12
2.2. CIELab.....	14
2.3. SLIC.....	16
2.4. Mapping .....	18
2.5. Boykov - Kolmogorov .....	19
Chương 3. Model Inpainting.....	20
3.1. Giới thiệu .....	20
3.2. GAN.....	22
3.3. DCGAN .....	23
3.4. WGAN .....	24
3.5. WGAN - GP.....	24
3.6. Kiến trúc mạng.....	25
Chương 4. Thực thi model GraphCut .....	29
Chương 5. Thực thi model Inpainting.....	34
Chương 6. Đánh giá.....	35

6.1. Dataset.....	35
6.2. Training.....	36
6.2.1. Data được sử dụng trong quá trình training .....	37
6.2.2. Metric .....	39
Chương 7. Kết luận.....	41
7.1. Kết quả đạt được .....	41
7.2. Thuận lợi & Khó khăn .....	41
7.2.1. Thuận lợi .....	41
7.2.2. Khó khăn.....	41
7.3. Hướng phát triển .....	42
TÀI LIỆU THAM KHẢO .....	43

## DANH MỤC HÌNH ẢNH

Figure 1. Ảnh có mắc cài .....	10
Figure 2. Ảnh không còn mắc cài .....	11
Figure 3. Quy trình xử lý ảnh.....	11
Figure 4. Graph G với 5 đỉnh, nét đứt chính là cut .....	12
Figure 5. 4 loại node thuộc đồ thị .....	13
Figure 6. Đầu vào của model GraphCut, chấm xanh dương đánh dấu vị trí sink node và chấm đỏ đánh dấu vị trí source node.....	13
Figure 7. Đầu ra của model GraphCut .....	14
Figure 8. Quy trình xử lý của model GraphCut .....	14
Figure 9. Mã giả RGB→XYZ .....	15
Figure 10. Mã giả XYZ→CIELab.....	15
Figure 11. Các SP khi khởi tạo có center cách đều nhau nên các boundary sẽ song song và cách đều nhau .....	16
Figure 12. Quy trình khởi tạo với mỗi SP.....	16
Figure 13. Quy trình xử lý với mỗi SP.....	17
Figure 14. Các boundary mới sau khi tính lại center cho tất cả SP .....	17
Figure 15. Đồ thị mẫu sau khi Mapping .....	18
Figure 16. Ảnh trước và sau được Inpainting .....	20
Figure 17. GT.....	21
Figure 18. Binary mask.....	21
Figure 19. GT và binary mask sau khi concat.....	21
Figure 20. Ảnh sau quá trình inpainting. ....	22
Figure 21. Quy trình xử lý của model Inpainting.....	22
Figure 22. Sơ đồ tổng quát của model GAN cổ điển .....	22
Figure 23. Mã giả của model WGAN-GP.....	25
Figure 24. Kiến trúc của toàn bộ model Inpainting .....	25
Figure 25. Mã trận dilated với padding và độ giãn nở khác nhau.....	26
Figure 26. Kiến trúc của mạng Coarse.....	26
Figure 27. Kiến trúc của Contextual Attention Layer.....	27
Figure 28. Attention thể hiện độ tương đồng pixel cần vá khi so với pixel khác trong background. ....	27
Figure 29. Các vị trí tại biên (đánh dấu đỏ) có hiện tượng inconsistency khi không kết hợp với dilated network. ....	28
Figure 30. Mã nguồn mở của tác giả shameempk.....	29

Figure 31. Thực nghiệm với lá cây .....	29
Figure 32. Thực nghiệm với lá cây .....	30
Figure 33. Thực nghiệm với ảnh có mắc cài.....	30
Figure 34. Thực nghiệm với ảnh có mắc cài.....	31
Figure 35. Thực nghiệm với ảnh có mắc cài.....	32
Figure 36. Thực nghiệm với ảnh có mắc cài.....	33
Figure 37. Mã nguồn của tác giả JiahuiYu .....	34
Figure 38. Mã nguồn của tác giả Yabin Zheng.....	35
Figure 39. Quá trình crawl ảnh với keyword là teeth, số lượng chỉ định là 100.....	35
Figure 40. Đánh giá model dựa trên dataset Teeth tự crawl bao gồm 145 ảnh về răng.....	36
Figure 41. Lần lượt từ trái sang phải, GT – GT và binary mask, ảnh sau Inpainting và ma trận Attention Score. ....	39
Figure 42. Đồ thị G_loss với trục hoành là thời gian, trục tung là giá trị G_loss .....	39
Figure 43. Đồ thị AE_loss với trục hoành là thời gian, trục tung là giá trị AE_loss .....	40



## DANH MỤC TỪ TIẾNG ANH

Từ tiếng Anh	Diễn giải
Model	Mô hình
Deep learning	Học sâu, là một tập con trong Machine learning.
Supapixel	Siêu điểm ảnh, là pixel đại diện cho $n$ pixel xung quanh nó
Mapping	Ánh xạ từ $A^* \rightarrow B^*$ với $f: A \rightarrow B$
Histogram	Đồ thị biểu thị phân phối màu của ảnh
GAN	Generative Adversarial Network, mạng đối nghịch tạo sinh
CNN	Convolutional Neural Network. mạng neural tích chập
Model Collapse	Model không phát triển mặc dù tiếp tục training
Loss function	Hàm tính độ lỗi của model
Metric	Số liệu đánh giá model
SUP	Least upper bound, cận trên nhỏ nhất
Fine - tuning	Quy trình đánh giá, thử nghiệm và tìm kiếm và lặp lại cho đến khi tìm được hyperparameter tối ưu

## MỞ ĐẦU

Xử lý ảnh hay rộng hơn là thị giác máy tính là một trong những lĩnh vực trong khoa học máy tính có rất nhiều ứng dụng hữu ích hiện nay. Đặc biệt vào khoảng thời gian từ 2010 đến nay, khi các model deep learning được phát minh và liên tục cải tiến, chất lượng ảnh được xử lý và số lượng ngành nghề trong đời sống cần áp dụng xử lý ảnh được tăng mạnh.

Một trong số đó là lĩnh vực y tế, do số lượng dân số gia tăng và tình hình bệnh tật ngày càng phức tạp, trong khi đó số lượng bác sĩ, y tế không thể đáp ứng đủ. Vì vậy, y tế là ngành rất cần đến những chức năng tự động hóa của máy tính, giải phóng được một số công việc lặp đi lặp lại cho những người trị bệnh như kê khai đơn thuốc, chẩn đoán bệnh qua X – quang hoặc ảnh chụp.

Một số ứng dụng thị giác máy tính trong nha khoa bao gồm chẩn đoán bệnh lý răng [12], tái tạo nguyên hàm với hệ thống CAD/CAM [12], ... Nha khoa là lĩnh vực nhỏ trong y tế có bệnh nhân dồi dào, bởi lẽ mỗi người dân đều phải gặp nha sĩ vài lần để nhổ răng, trám răng hay niềng răng. Đặc biệt giới trẻ ngày nay có nhu cầu làm đẹp là niềng răng rất lớn, nhưng hạn chế của việc niềng răng là thời gian đeo mắc cài khá lâu, khiến nhu cầu chụp ảnh chân dung hay theo dõi sự chuyển biến của răng bị hạn chế. Xác định từ thực tế đó, em đã chọn đề tài **Khôi phục ảnh nha khoa bằng phương pháp Inpainting** làm đề tài cho đề tài 1 của mình. Để minh họa những kiến thức em tìm hiểu được về các model deep learning sinh ảnh, em đã hiện thực hóa bằng thư viện Tensorflow và ngôn ngữ lập trình Python.

Nội dung của đề tài được chia gồm 7 chương:

- Chương 1: Giới thiệu đề tài
- Chương 2: Model GraphCut
- Chương 3: Model Inpainting
- Chương 4: Thực thi model GraphCut
- Chương 5: Thực thi model Inpainting
- Chương 6: Đánh giá
- Chương 7: Kết luận

## Chương 1. Giới thiệu đề tài

### 1.1. Giới thiệu

Vào thế kỉ 20, khi xã hội phát triển, con người bắt đầu quan tâm đến ngoại hình của mình hơn bằng cách chỉnh sửa cơ thể. Một trong số đó là niềng răng, một phương pháp tốn kém, lâu dài nhưng đổi lại người niềng sẽ có một nụ cười đẹp, thuật ngữ "niềng răng" từ đó xuất hiện.

Niềng răng là kỹ thuật sử dụng dụng cụ chính là mắc cài. Các nha sĩ quấn băng kim loại cá nhân xung quanh các răng khác nhau và các khâu đó sẽ được nối với nhau bằng dây cung. Dây cung tạo ra áp lực làm răng di chuyển thẳng hàng từ từ. Vật liệu của mắc cài rất đa dạng, hoàn toàn phụ thuộc vào sở thích cá nhân của nha sĩ hoặc ngân sách của bệnh nhân.

Thời gian niềng răng phụ thuộc vào độ tuổi và tình trạng răng, nhưng đa phần thường rất lâu từ 18 – 24 tháng, thậm chí 3 năm. Điều này gây ảnh hưởng không nhỏ đến một số nhu cầu của người đang niềng răng như chụp ảnh và theo dõi chuyển biến răng của bản thân.

### 1.2. Mục tiêu

Với nhu cầu trên, em xây dựng một model xử lý ảnh có đầu ra vào đầu vào như sau:

Input: ảnh răng có mắc cài, mắc cài có thể có màu sắc, hình dạng và chủng loại khác nhau.

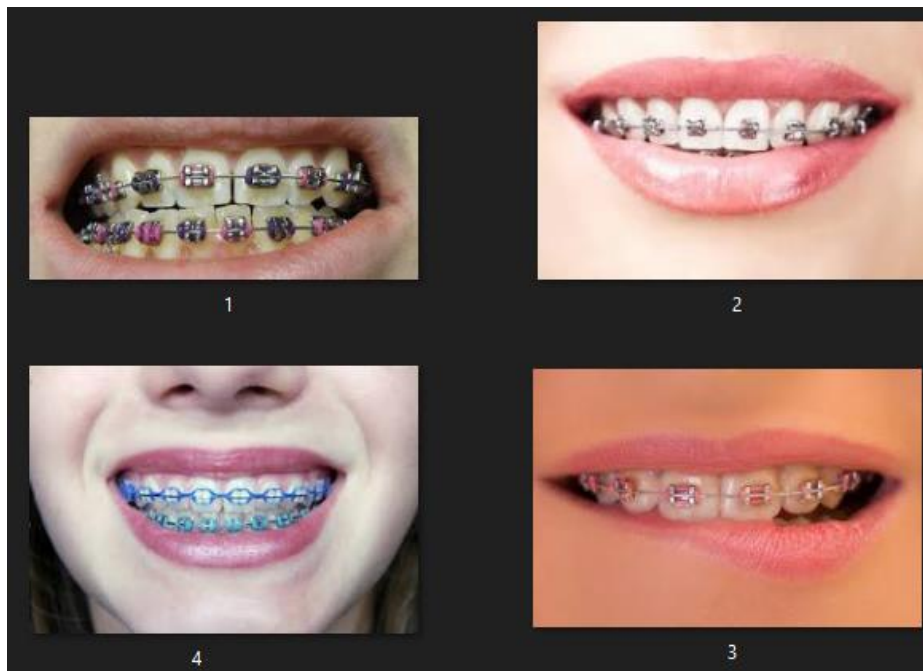


Figure 1. Ảnh có mắc cài

Output: ảnh răng không còn mắc cài, những vùng mắc cài bị xóa được vá sao cho ảnh chân thật nhất.



Figure 2. Ảnh không còn mắc cài

Pipeline: đề tài kết hợp sử dụng 2 model, model GraphCut (mục 2) và model WGAN - GP (mục 3.5).



Figure 3. Quy trình xử lý ảnh

## Chương 2. Model GraphCut

### 2.1. Giới thiệu

Phân đoạn ảnh là thao tác tách vật thể (foreground - object) ra khỏi nền chứa chúng (background). Kết quả của phân đoạn ảnh là đầu vào cho nhiều model khác bao gồm nhận diện (recognize), phân loại (classification), ... Hiện nay có nhiều model deep learning như U – net, Fast – RCNN đã xử lý tốt tuy nhiên yêu cầu thời gian và quá trình training. Để đơn giản hóa thao tác này, em sử dụng GraphCut, một phương pháp thuần xử lý ảnh có độ chính xác không bằng nhưng thời gian phản hồi thấp và không cần giai đoạn training.

GraphCut là phương pháp áp dụng lý thuyết đồ thị. Cho đồ thị  $G(E, V)$ , cut là đường cắt chia  $G$  thành 2 đồ thị con với mincut là đường cut với tổng số lượng edge cắt qua là nhỏ nhất và ngược lại với maxcut. Ví dụ: với đồ thị dưới mincut là 2, và không có cut nào có giá trị 1 vì đồ thị không có cầu.

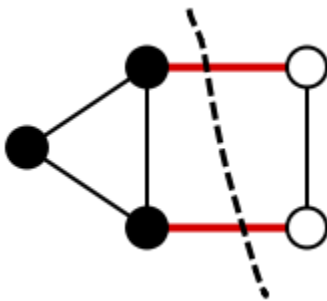


Figure 4. Graph  $G$  với 5 đỉnh, nét đứt chính là cut

Trong mạng dòng chảy (flow network), cho  $G(E, V)$  hữu hạn và mỗi edge  $(u, v)$  có trọng số  $c$  là giá trị thực không âm. Giả sử có hai node, sink node (đỉnh thu) và source node (đỉnh phát) đã được xác định.  $s - t$  cut là đường cut chia  $G$  thành hai tập  $S$  và  $T$  sao cho  $\forall s \in S, s$  là node chứa pixel là foreground và  $\forall t \in T, t$  là node chứa pixel là background. Chúng ta gọi  $s - t$  cut là maxflow khi tổng trọng số của các edge đường cut đi qua là cực đại và ngược lại với minflow.

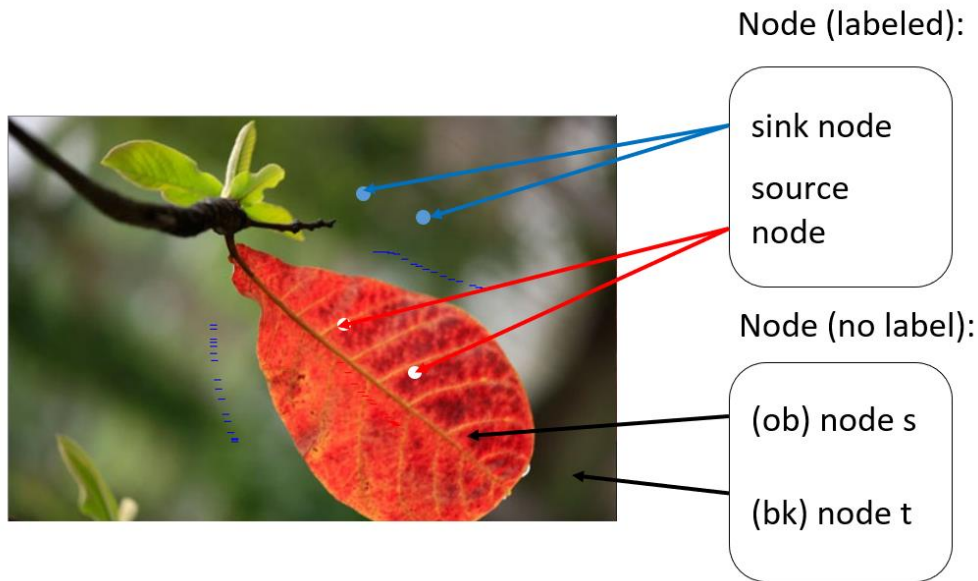


Figure 5. 4 loại node thuộc đồ thị, ta xem sink node là node chứa background và source node chứa foreground

Ứng dụng trong phân đoạn ảnh, đường phân đoạn foreground và background thành hai phần riêng biệt là đường cut thỏa mãn điều kiện là mincut lẫn maxflow. Chúng ta sẽ sử dụng thuật toán Boykov - Kolmogorov trong phần 2.5 để tìm ra đường cut này. Xem mỗi pixel là node và khoảng cách (độ lệch màu) giữa 2 pixel là trọng số  $c(u, v)$  (trong phần 2.4) của edge nối liền chúng, ta có thể ánh xạ từ ảnh bất kì sang đồ thị  $G(E, V)$  tương ứng.

Lấy ví dụ với ảnh  $256 \times 256$ , giả sử chỉ có 1 sink node và 1 source node, chúng ta có số lượng node =  $256 \times 256 = 65536$ , số lượng edge  $(256/4 \times 4 + 256/8 \times 2) \times 255 + 256 \times 2 = 82112$ . Vì kích thước của đồ thị sẽ tăng theo cấp số nhân kích ảnh nên để giảm bớt khối lượng tính toán, chúng ta sử dụng khái niệm Superpixel được trình bày trong phần 2.3.

Input: ảnh màu  $M$  (không gian màu RGB), tập sink node và tập source node.

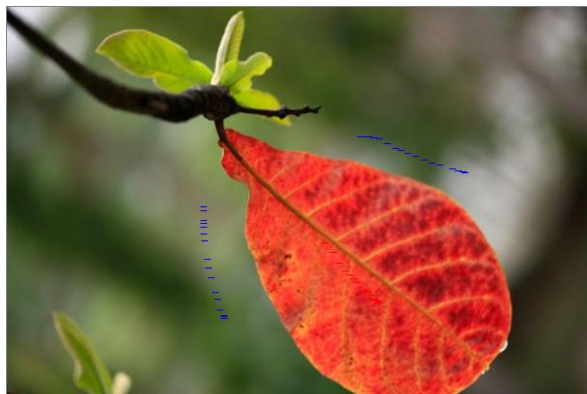


Figure 6. Đầu vào của model GraphCut, chấm xanh dương đánh dấu vị trí sink node và chấm đỏ đánh dấu vị trí source node

Output: những pixel thuộc  $s$  node được giữ nguyên, còn  $t$  node thì chuyển giá trị màu về RGB (255,255,255).



Figure 7. Đầu ra của model GraphCut

Pipeline:

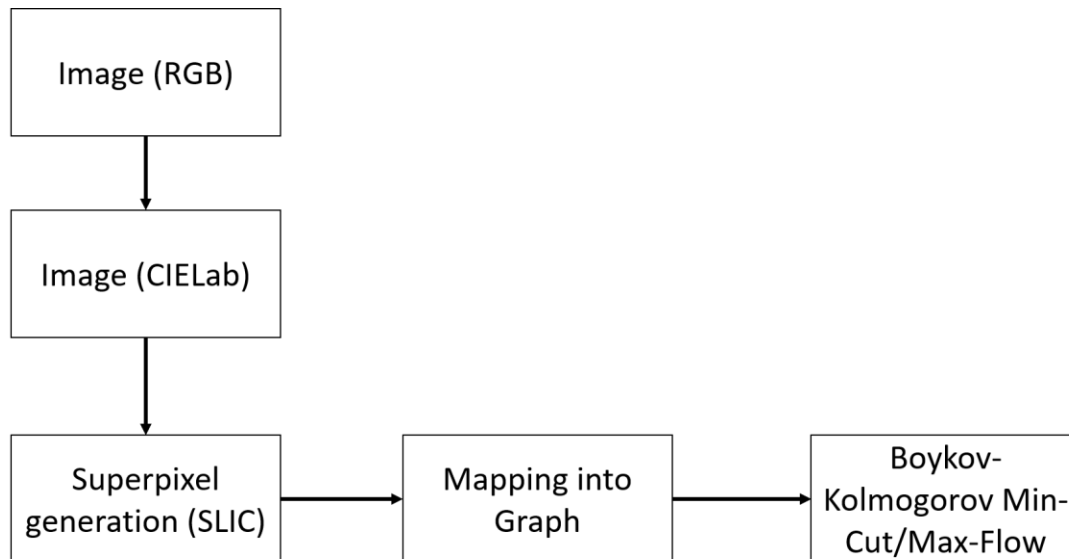


Figure 8. Quy trình xử lý của model GraphCut

## 2.2. CIELab

Để tính toán khoảng cách màu giữa 2 pixel, công thức CIDE2000 phát triển bởi CIE (International Commission on Illumination) được khuyến khích sử dụng. Để áp dụng CIDE2000, ta chuyển ảnh RGB về không gian màu CIELab.

Việc chuyển đổi bao gồm 2 bước:

Bước 1: Chuyển từ RGB sang không gian chuẩn XYZ:



```
//sR, sG and sB (Standard RGB) input range = 0 ÷ 255
//X, Y and Z output refer to a D65/2° standard illuminant.

var_R = ( sR / 255 )
var_G = ( sG / 255 )
var_B = ( sB / 255 )

if ( var_R > 0.04045 ) var_R = ( ( var_R + 0.055 ) / 1.055 ) ^ 2.4
else var_R = var_R / 12.92
if ( var_G > 0.04045 ) var_G = ( ( var_G + 0.055 ) / 1.055 ) ^ 2.4
else var_G = var_G / 12.92
if ( var_B > 0.04045 ) var_B = ( ( var_B + 0.055 ) / 1.055 ) ^ 2.4
else var_B = var_B / 12.92

var_R = var_R * 100
var_G = var_G * 100
var_B = var_B * 100

X = var_R * 0.4124 + var_G * 0.3576 + var_B * 0.1805
Y = var_R * 0.2126 + var_G * 0.7152 + var_B * 0.0722
Z = var_R * 0.0193 + var_G * 0.1192 + var_B * 0.9505
```

Figure 9. Mã giả RGB→XYZ

Bước 2: Chuyển từ không gian chuẩn XYZ sang không gian CIELab:

```
//Reference-X, Y and Z refer to specific illuminants and observers.
//Common reference values are available below in this same page.

var_X = X / Reference-X
var_Y = Y / Reference-Y
var_Z = Z / Reference-Z

if ( var_X > 0.008856 ) var_X = var_X ^ ( 1/3 )
else var_X = ( 7.787 * var_X ) + ( 16 / 116 )
if ( var_Y > 0.008856 ) var_Y = var_Y ^ ( 1/3 )
else var_Y = ( 7.787 * var_Y ) + ( 16 / 116 )
if ( var_Z > 0.008856 ) var_Z = var_Z ^ ( 1/3 )
else var_Z = ( 7.787 * var_Z ) + ( 16 / 116 )

CIE-L* = ( 116 * var_Y ) - 16
CIE-a* = 500 * ( var_X - var_Y )
CIE-b* = 200 * ( var_Y - var_Z )
```

Figure 10. Mã giả XYZ→CIELab



### 2.3. SLIC

Những pixel gần nhau thì có giá trị màu gần như tương tự nhau, do đó chúng ta có thể gom những pixel chung đặc điểm thành một pixel duy nhất có giá trị màu bằng trung bình các pixel cấu thành nó, những pixel mới này được gọi là superpixel (SP). Mỗi SP sẽ có thông tin bao gồm pixel trung tâm (center) và biên với superpixel khác (boundary).

SLIC (viết tắt simple linear Iterative cluster) là một phương pháp ánh xạ ảnh bất kì sang tập  $SP, \forall sp \in SP, sp = \{center, boundary\}$ . Phương pháp này chia thành 2 bước:

Bước 1: Khởi tạo với mỗi center  $C_k = [L_k, a_k, b_k, x_k, y_k]$  trong đó  $L_k, a_k, b_k$  là giá trị màu trong không gian CIELab,  $x_k, y_k$  là tọa độ trong ảnh.



Figure 11. Các SP khi khởi tạo có center cách đều nhau nên các boundary sẽ song song và cách đều nhau

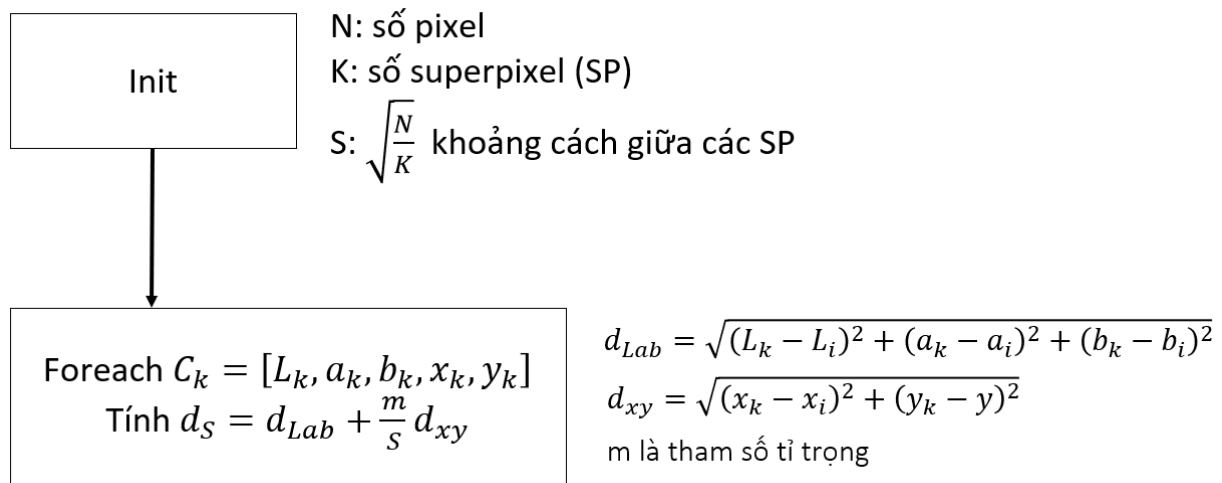


Figure 12. Quy trình khởi tạo với mỗi SP

Bước 2: Tìm center mới cho mỗi SP.

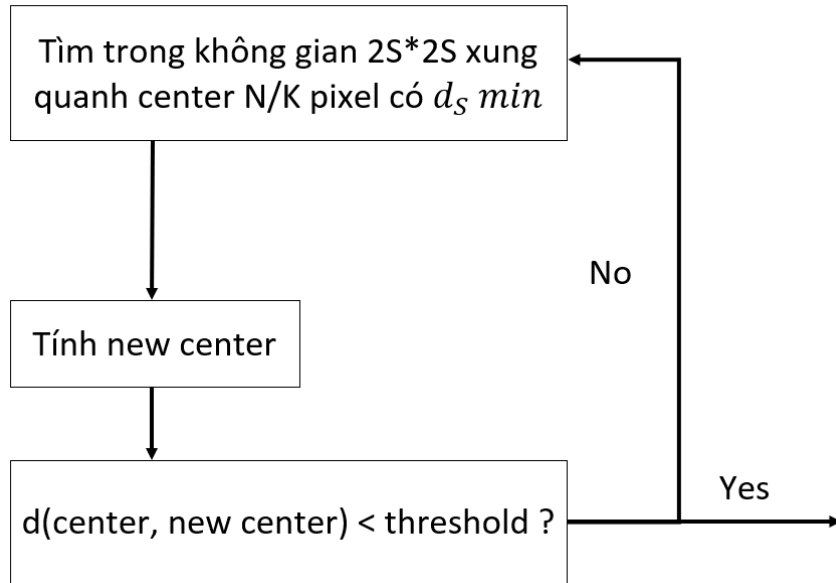


Figure 13. Quy trình xử lý với mỗi SP



Figure 14. Các boundary mới sau khi tính lại center cho tất cả SP

## 2.4. Mapping

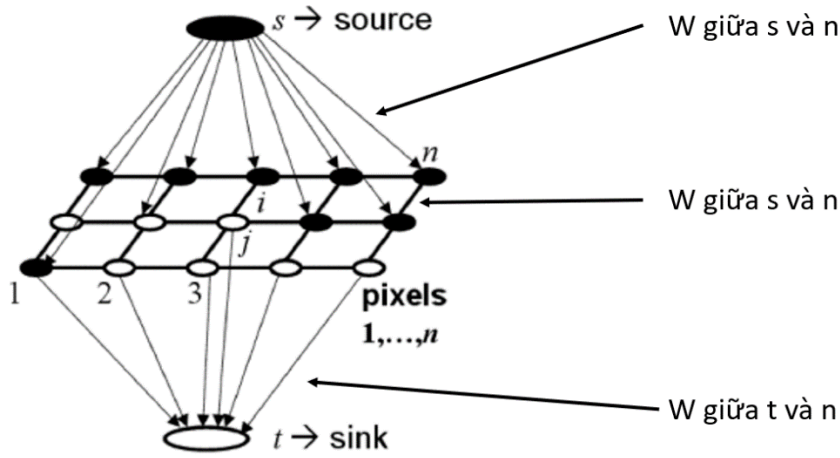


Figure 15. Đồ thị mẫu sau khi Mapping

Sau khi có tập SP, ta ánh xạ chúng sang đồ thị để áp dụng thuật toán Boykov - Kolmogorov. Có hai bước tính toán chính:

Bước 1: Khởi tạo trọng số cho các node bằng với giá trị màu mỗi superpixel với việc xem một SP tương ứng với một node.

Bước 2: Khởi tạo trọng số cho các edge  $w(u, v)$  bao gồm 3 loại giữa 2 node không label, không label và có label và 2 node có label.

Với 2 node  $u$  ( $center, hist$ ) và  $v$  ( $center, hist$ ) (với hist là histogram):

$$sim(u, v) = e^{(-\frac{cv2.compareHist(u.hist, v.hist)^2}{2 \cdot \sigma^2})} * \frac{1}{distance(u.center, v.center)}$$

với  $cv2.compareHist(H1, H2) \equiv d(H1, H2) = \sum_I \min(H_1(I), H_2(I))$ . Ta có được  $w(u, v) = 1 + sim(u, v)$ .

Với node  $s(center, hist)$ ,  $t(center, hist)$  và  $u(center, hist)$  (không label):

$$pr_{ob} \equiv \frac{hist_{ob}(u)}{hist_{ob}sum}, pr_{bk} \equiv \frac{hist_{bk}(u)}{hist_{bk}sum}$$

Nếu  $pr_{bk} > 0 \rightarrow w(t, u) = 100000, w(s, u) = 0.9 * -\log(pr_{ob})$ .

Nếu  $pr_{ob} > 0 \rightarrow w(s, u) = 100000, w(t, u) = 0.9 * -\log(pr_{ob})$ .

Với node  $s(center, hist)$ ,  $t(center, hist)$  ( $center, hist$ ) và  $u(center, hist)$  (được label):

$u$  (label Source node),  $w(s, u) = 1 + sim, w(t, u) = 0$

$u$  (label Sink node),  $w(t, u) = 1 + \text{sim}$ ,  $w(s, u) = 0$

## 2.5. Boykov - Kolmogorov

Sau khi có đồ thị  $G(E, V)$ , chúng ta áp dụng hàm *Boykov – Kolmogorov*:  $G \rightarrow [S, T]$  để thu được đồ thị S và T.

## Chương 3. Model Inpainting

### 3.1. Giới thiệu

Inpainting là quá trình phục hồi tranh / ảnh, trong đó có chỗ bị hư hỏng hoặc thiếu để có được tranh / ảnh hoàn chỉnh. Inpainting là kỹ thuật có trong mỹ thuật, vật lý và kỹ thuật số như tranh sơn dầu hoặc acrylic, tranh in hóa học, điêu khắc 3 chiều, hoặc hình ảnh kỹ thuật số và video. Trong thị giác máy tính, inpainting xử lý các ảnh lỗi do quá trình truyền và lưu trữ, hoặc muốn tạo ra các ảnh có nội dung mới dựa trên ảnh cũ.



Figure 16. Ảnh trước và sau được Inpainting

WGAN - GP là model sinh ảnh được cải tiến từ GAN (mục 3.2), DCGAN (mục 3.3) và WGAN (mục 3.4). Model WGAN – GP sử dụng trong đề tài thiết kế để phục vụ bài toán inpainting (Img2Img). Một số thuật ngữ được sử dụng trong inpainting bao gồm ground truth (GT) là ảnh gốc ( $H * W$ ), mặt nạ nhị phân (binary mask) là ma trận ( $H * W$ ) (0 là missing pixel và 1 là những pixel còn lại) và hole / missing region là những pixel có vị trí trùng với pixel có giá trị bằng 0 trong binary mask.

Input: ảnh RGB và binary mask.



Figure 17. GT



Figure 18. Binary mask

Output: ảnh sau khi được inpainting.



Figure 19. GT và binary mask sau khi concat



Figure 20. Ảnh sau quá trình inpainting.

Pipeline:

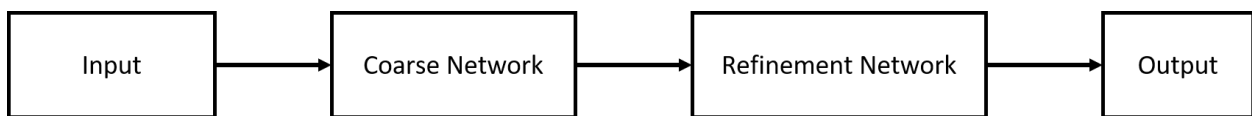


Figure 21. Quy trình xử lý của model Inpainting

### 3.2. GAN

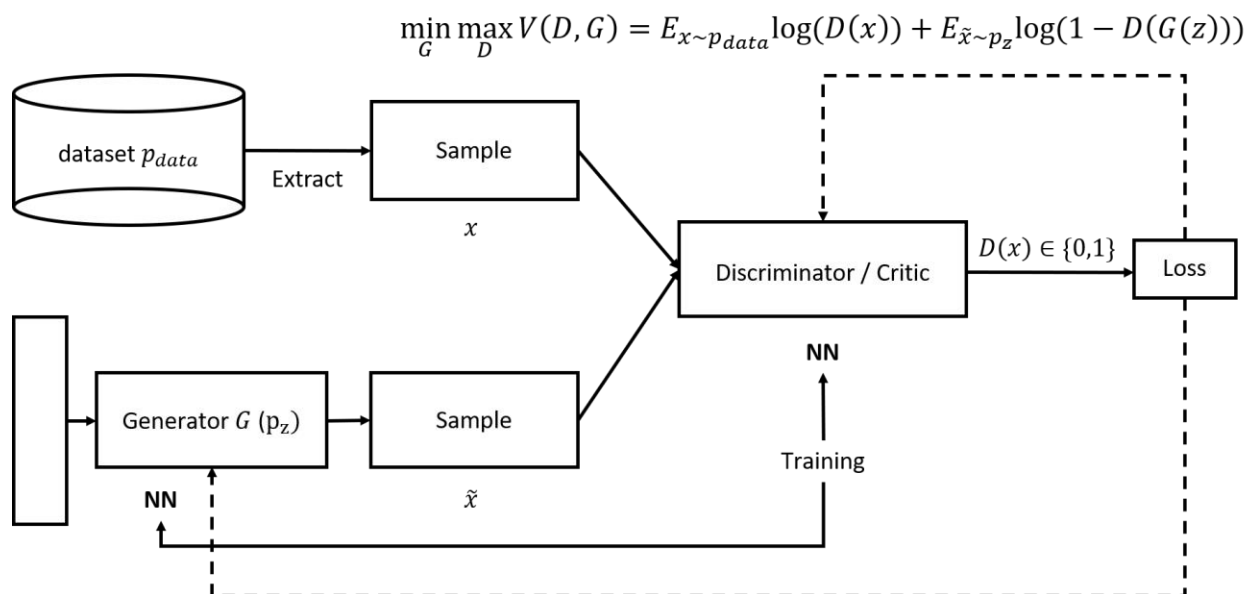


Figure 22. Sơ đồ tổng quát của model GAN cổ điển

GAN là model cổ điển được giới thiệu bởi Ian J. Goodfellow vào năm 2014 đã đạt được rất nhiều thành công lớn trong deep learning nói riêng và AI nói chung. Một số ứng dụng nổi bật của GAN như deepfake, image editing, generate realistic photographs, super resolution, text to image, ...

GAN gồm hai mạng độc lập là generator ( $G$ ) và discriminator ( $D$ ) hay critic ( $C$ ).  $G$  là mạng sinh ảnh với input là noise  $z$  với  $z$  tuân theo phân phối chuẩn  $\mu(0,1)$  hoặc phân phối đều  $U[0,1]$ .  $D$  là mạng classification, với  $D(x) \in \{0,1\}$ , 1 là ảnh thật và 0 là ảnh giả. Cả 2 mạng đều được huấn luyện song song hoặc tuần tự cho đến khi đạt đến cân bằng Nash (Nash equilibrium).

Giả sử  $p$  là phân phối xác suất tập ảnh thật và  $q$  là phân phối xác suất tập ảnh giả. Để đánh giá model tốt hay không (chất lượng ảnh sinh ra cao hay không), chúng ta sử dụng một số metric là phân kỳ Kullback-Leibler (KL divergence):

$$D_{KL}(P||Q) = \sum_{x=1}^N P(x) \log \left( \frac{P(x)}{Q(x)} \right)$$

hoặc phân kỳ Jensen-Shannon (JS divergence):

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P || \frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q || \frac{P+Q}{2})$$

với mục tiêu là  $D_{KL}(P||Q)$  hoặc  $D_{JS}(P||Q)$  xấp xỉ 0. Tuy nhiên, khi kỳ vọng (mean) của  $q$  tăng (một nửa model GAN đang hội tụ) thì giá trị phân kỳ tăng, đạo hàm giảm.  $G$  cập nhật rất ít hoặc hầu như không cập nhật từ gradient descent, hiện tượng này được gọi generator diminishes.

Trong khi đó, loss function nguyên thủy làm mất cân bằng trong quá trình training ( $D$  thường ổn định hơn với  $G$  ở giai đoạn ban đầu). Khi  $q$  quá xa so với  $p$ ,  $-\nabla_{\theta} \log(1 - D(G(z^i))) \rightarrow 0$  ( $\log 1 = 0$ ).  $G$  sẽ khó hội tụ, đây cũng là một vấn đề gây suy giảm chất lượng ảnh của GAN. Do đó, Arjovsky [11] có đề xuất loss function mới và thêm noise vào  $p$  để khiến  $D$  bất ổn định hơn. Tuy nhiên một phương pháp hiệu quả hơn sẽ được giới thiệu ở mục 3.4.

### 3.3. DCGAN

Với nhiệm vụ liên quan tới xử lý ảnh, model CNN được kết hợp với GAN để chất lượng ảnh được sinh ra tốt hơn. Model này được gọi là DCGAN. Một số đặc điểm DCGAN khác với CNN truyền thống như sử dụng stride - 2 conv layer thay cho max pooling layer, sử dụng global average pooling thay cho fully connected layer.



### 3.4. WGAN

Vấn đề trong quá trình training GAN được giới thiệu trong mục 3.2 là generator diminishes và model collapse (ngược lại với generator diminishes) khiến mô hình khó hội tụ. Để khắc phục vấn đề này, [9] đã đề xuất loss function mới trong model mới là Wasserstein GAN (WGAN).

Wasserstein (hay Kantorovich – Rubinstein)  $W$  cũng là một metric đánh giá sự khác biệt giữa 2 phân phối xác suất  $P_r$  (tập ảnh thật) và  $P_g$  (tập ảnh giả) tương tự như phân kỳ  $KL$  và phân kỳ  $JS$  nhưng có đặc điểm quan trọng là đạo hàm mượt tại mọi điểm. Trong loss function mới, mục tiêu cuối cùng của mạng là cố gắng sao cho  $W(P_r, P_g)$  thấp nhất có thể (hay generator sinh ra ảnh có chất lượng gần như ảnh thật nhất).

Để tính toán  $W$  theo công thức gốc sẽ phát sinh nhiều chi phí nên tác giả đã đề xuất sử dụng đối ngẫu Kantorovich – Rubinstein để đơn giản hóa công thức gốc thành:

$$W(P_r, P_g) = \sup_{D \in \mathcal{D}} E_{x \sim P_r}[D(x)] - E_{\tilde{x} \sim P_g}[D(\tilde{x})]$$

Trong đó sup là cận trên nhỏ nhất và  $\mathcal{D}$  là tập 1 - Lipschitz function (thỏa điều kiện  $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$ ). Như vậy mục tiêu mới của chúng ta là tìm (mô phỏng)  $f$  bằng deep neural network (DNN)  $F$  với layer cuối có output là vô hướng đánh giá chất lượng ảnh  $x$ .

Để  $F$  hội tụ, chúng ta giới hạn weight của mạng  $F$  là  $F_\theta$  trong miền  $(-c, c)$  với  $c$  là hyperparameter.

$$w \leftarrow \text{clip}(w, -c, c)$$

Thực nghiệm cho thấy  $c$  nằm ở khoảng gần với 0.01. Tuy nhiên với model khác hoặc dataset khác  $c$  sẽ khác. Như vậy, để có được  $c$  chúng ta phải fine - tuning. Nhược điểm này sẽ được khắc phục trong mục 3.5.

### 3.5. WGAN - GP

GP viết tắt của gradient penalty. Lấy  $\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x}$  là phân phối mẫu, chúng ta ép đạo hàm của 1-Lipschitz function  $D$  nhỏ hơn 1 bằng cách cộng vào  $W(P_r, P_g)$  lượng  $\lambda \left( \|\nabla_{\hat{x}} D_w(\hat{x})\|_2 \odot (1 - m) - 1 \right)^2, \lambda = 10$  ( $m = 0$  nếu là mask).

Giải thuật cuối cùng của model:

---

**Algorithm 1** WGAN with gradient penalty. We use default values of  $\lambda = 10$ ,  $n_{\text{critic}} = 5$ ,  $\alpha = 0.0001$ ,  $\beta_1 = 0$ ,  $\beta_2 = 0.9$ .

---

**Require:** The gradient penalty coefficient  $\lambda$ , the number of critic iterations per generator iteration  $n_{\text{critic}}$ , the batch size  $m$ , Adam hyperparameters  $\alpha, \beta_1, \beta_2$ .

**Require:** initial critic parameters  $w_0$ , initial generator parameters  $\theta_0$ .

```

1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_{\theta}(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m -D_w(G_{\theta}(\mathbf{z}^{(i)})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while

```

---

Figure 23. Mã giả của model WGAN-GP

### 3.6. Kiến trúc mạng

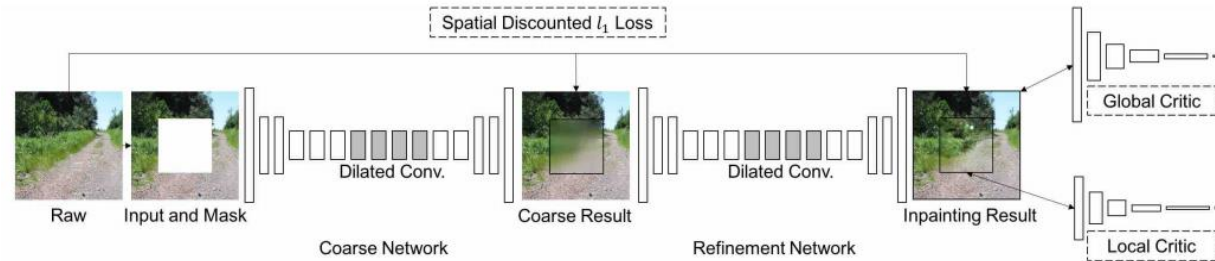


Figure 24. Kiến trúc của toàn bộ model Inpainting

Kiến trúc mạng chia làm 2 giai đoạn:

Giai đoạn 1 (Coarse network): giai đoạn làm thô, từ missing region ảnh được vá sao cho nội dung bên trong chỉ cần tương thích với background. Giai đoạn này model sử dụng thêm Spatial discounted reconstruction loss để đánh giá chất lượng theo khoảng cách từ boundary của missing region đến các missing pixel bên trong, khoảng cách càng nhỏ thì pixel phục hồi càng phải khớp với background ( $\gamma^i$  với  $i$  là khoảng cách đến pixel bên ngoài boundary gần nhất và  $\gamma = 0.99$ ). Các lớp convolutional sử dụng filter là dilated layer.

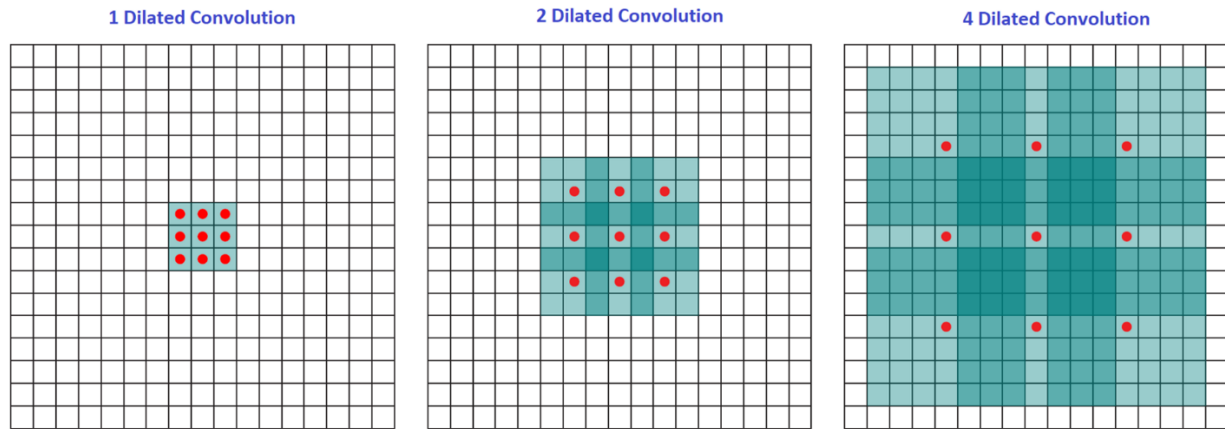


Figure 25. Mã trận dilated với padding và độ giãn nở khác nhau

Dilated layer sử dụng kernel có các phần tử bị cách đều nhau và padding bởi phần tử 0. Khoảng cách cách đều và độ dày của padding tùy thuộc vào kích thước ảnh muốn giãn nở. Khi đi qua nhiều dilated layer, kích thước ảnh được tăng về ban đầu.

Giai đoạn 2 (Refinement network): giai đoạn làm mịn.

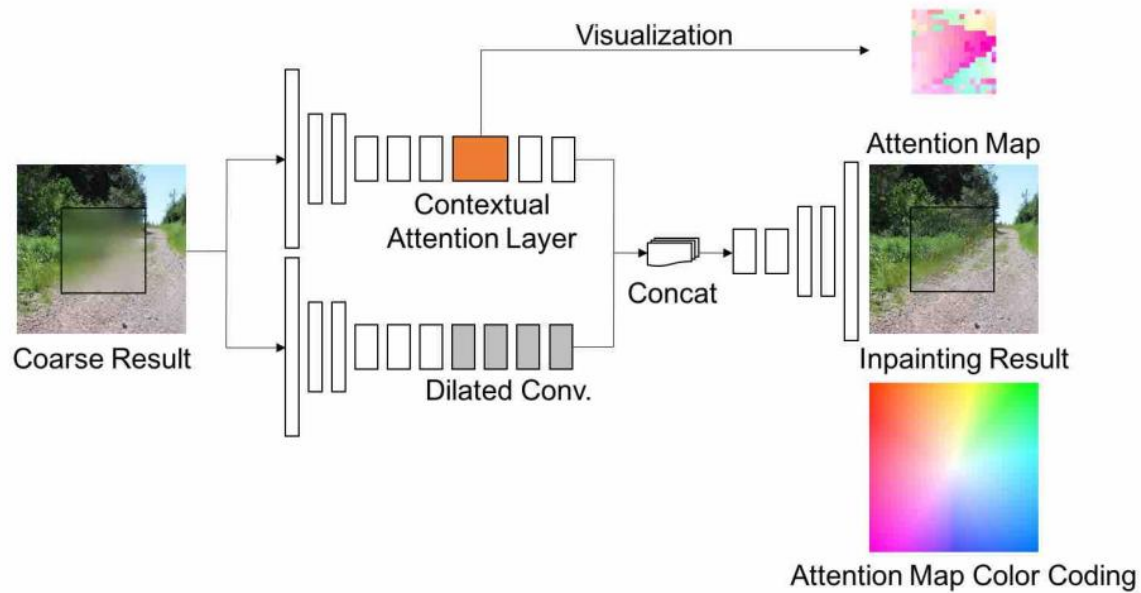


Figure 26. Kiến trúc của mạng Coarse

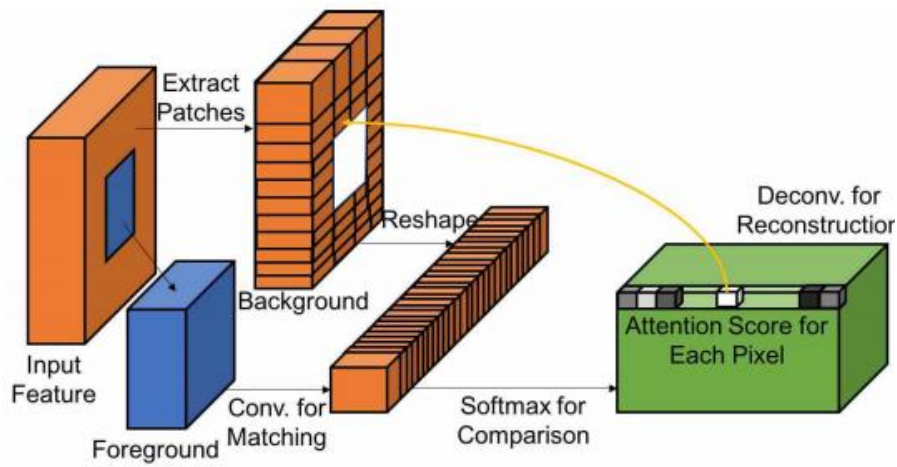


Figure 27. Kiến trúc của Contextual Attention Layer

Refinement network sử dụng kết quả từ hai mạng độc lập Contextual Attention và Dilated Convolutional sau đó concat với nhau. Mạng Contextual Attention tìm những pixel phù hợp trong background và vá vào missing region với 3 bước như sau:

Bước 1: Rút trích các mảnh (patch) 3x3 từ background và reshape để tạo thành filter.

Bước 2: Missing region (foreground) được đưa qua mạng convolutional với filter từ bước 1 và layer cuối cùng sử dụng hàm softmax.

Bước 3: Chúng ta thu được ma trận Attention Score có nghĩa sau. Dựa vào Attention Score, chúng ta xác định được chính xác giá trị màu cần phục hồi dựa vào những pixel khác trong ảnh.



Figure 28. Attention thể hiện độ tương đồng pixel cần vá khi so với pixel khác trong background.

Mạng dilated mở rộng receptive field với mục đích tổng quát hóa và có kiến trúc tương tự với coarse network. Lý do cần sử dụng hai mạng với cùng mục đích là để đạt được hai yếu tố: chất lượng (resolution) - đặc trưng chi tiết và độ hợp lý (consistency) - đặc trưng tổng thể. Khi chỉ sử dụng chỉ một trong hai, chúng ta gặp vấn đề sau:

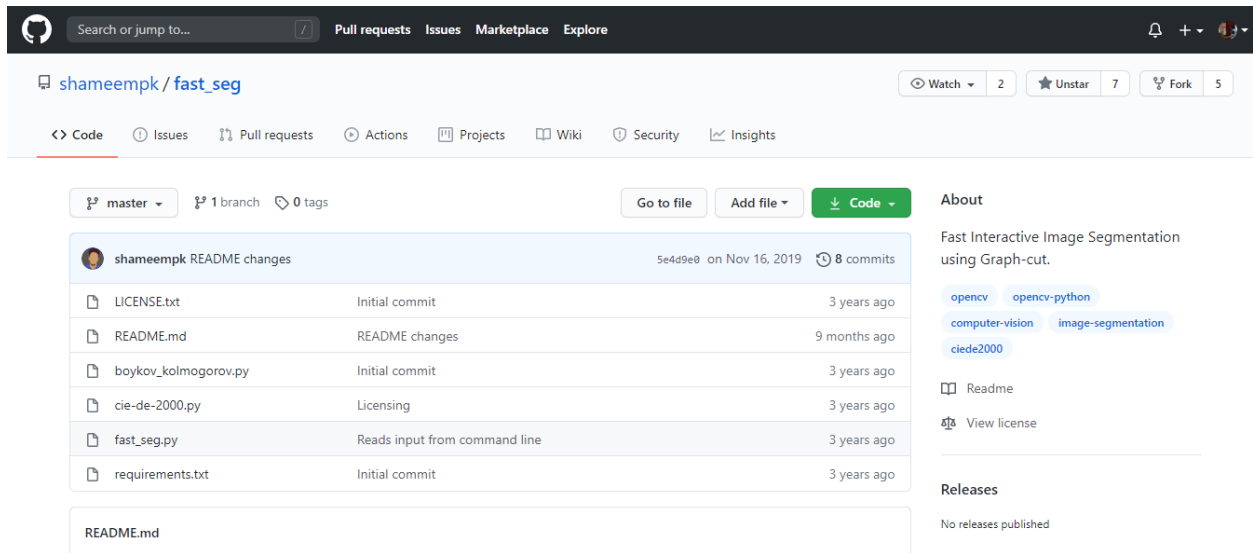


Figure 29. Các vị trí tại biên (đánh dấu đỏ) có hiện tượng inconsistency khi không kết hợp với dilated network.



## Chương 4. Thực thi model GraphCut

Em sử dụng mã nguồn mở của tác giả shameempk tại [đây](#).



shameempk / fast\_seg

Watch 2 Unstar 7 Fork 5

Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags

Go to file Add file Code

shameempk README changes 5e4d9e8 on Nov 16, 2019 8 commits

LICENSE.txt	Initial commit	3 years ago
README.md	README changes	9 months ago
boykov_kolmogorov.py	Initial commit	3 years ago
cie-de-2000.py	Licensing	3 years ago
fast_seg.py	Reads input from command line	3 years ago
requirements.txt	Initial commit	3 years ago

README.md

About

Fast Interactive Image Segmentation using Graph-cut.

opencv opencv-python computer-vision image-segmentation ciede2000

Readme View license

Releases

No releases published

Figure 30. Mã nguồn mở của tác giả shameempk

Tiến hành thử nghiệm và thu được một số kết quả như sau:



Figure 31. Thực nghiệm với lá cây



Figure 32. Thực nghiệm với lá cây



Figure 33. Thực nghiệm với ảnh có mắc cài



Figure 34. Thực nghiệm với ảnh có mắc cài



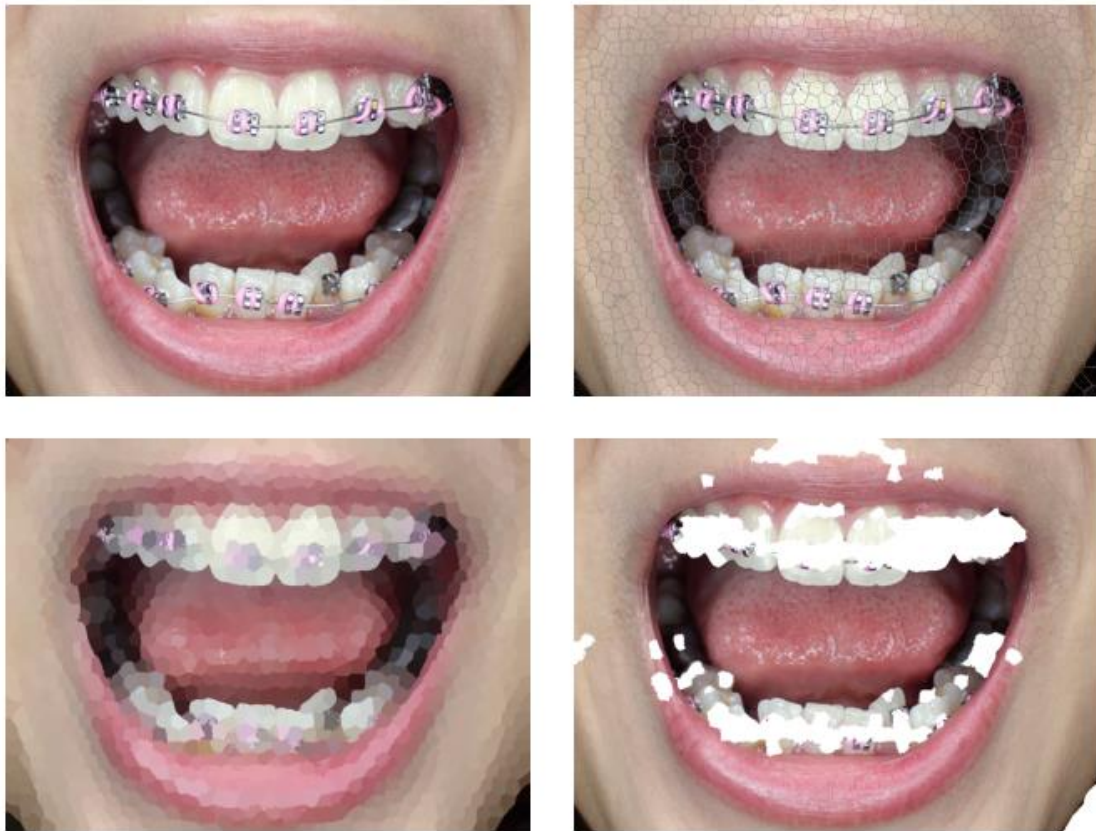


Figure 35. Thực nghiệm với ảnh có mắc cài

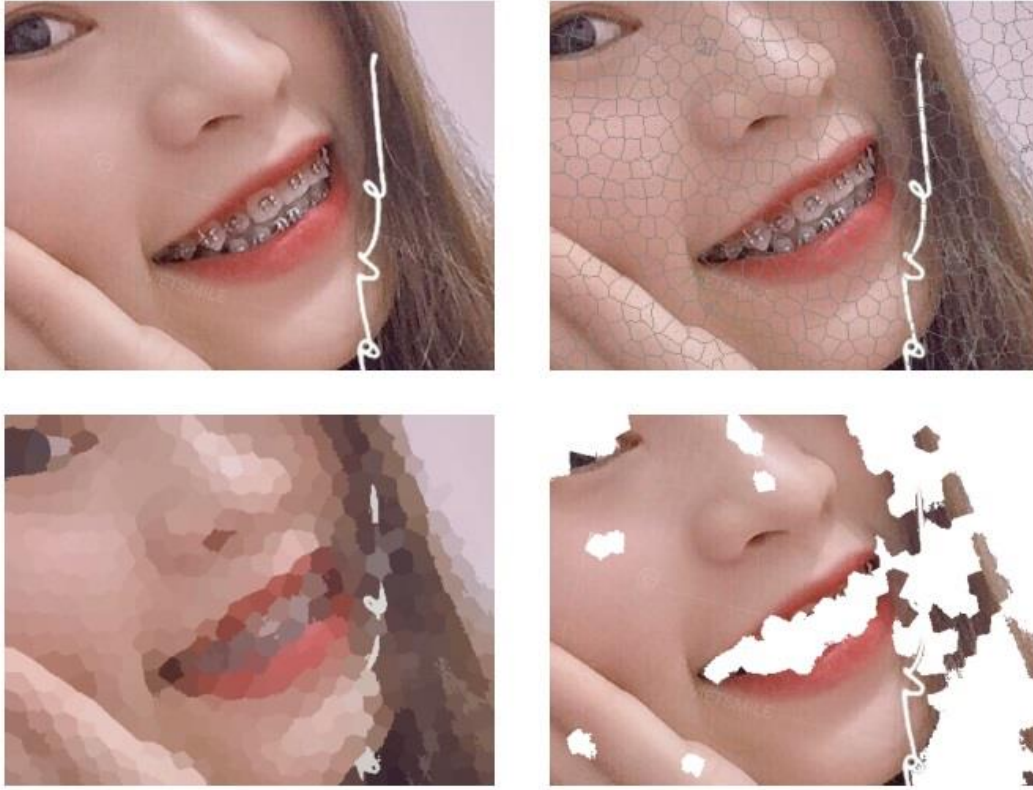


Figure 36. Thực nghiệm với ảnh có mắc cài

Lần lượt từ trái sang phải, từ trên xuống dưới của mỗi mẫu thử nghiệm: ảnh gốc, SP khi khởi tạo, SP hoàn chỉnh và các pixel thuộc đồ thị  $S$ .

## Chương 5. Thực thi model Inpainting

Đề tài có tham khảo mã nguồn mở của tác giả JiahuiYu tại [đây](#) và có chỉnh sửa để phù hợp với mục tiêu và đầu ra của đề tài.

Search or jump to...

Pull requests Issues Marketplace Explore

JiahuiYu / generative\_inpainting

Watch 69 Star 1.9k Fork 510

<> Code Issues 22 Pull requests Actions Security Insights

master 3 branches 0 tags

Go to file Add file Code

JiahuiYu Fix typo. 3a53243 on Jun 10 42 commits

examples/places2	Release v2.0.0	9 months ago
.gitignore	Release v2.0.0	9 months ago
LICENSE	CC 4.0 Attribution-NonCommercial International	3 years ago
README.md	Fix typo.	2 months ago
batch_test.py	Fix missing FLAGS	9 months ago
guided_batch_test.py	Release v2.0.0	9 months ago
inpaint.yml	Release v2.0.0	9 months ago
inpaint_model.py	Add missing FLAGS parameters	9 months ago
inpaint_ops.py	Release v2.0.0	9 months ago
test.ov	Release v2.0.0	9 months ago

About

DeepFill v1/v2 with Contextual Attention and Gated Convolution, CVPR 2018, and ICCV 2019 Oral

[jiahuiyu.com/deepfill/](#)

deepfill image-inpainting generative-adversarial-network attention-model deep-neural-networks tensorflow

Readme

View license

Releases

No releases published

Figure 37. Mã nguồn của tác giả JiahuiYu

## Chương 6. Đánh giá

### 6.1. Dataset

Em sử dụng công cụ tải ảnh tự động từ google image tại [đây](#).

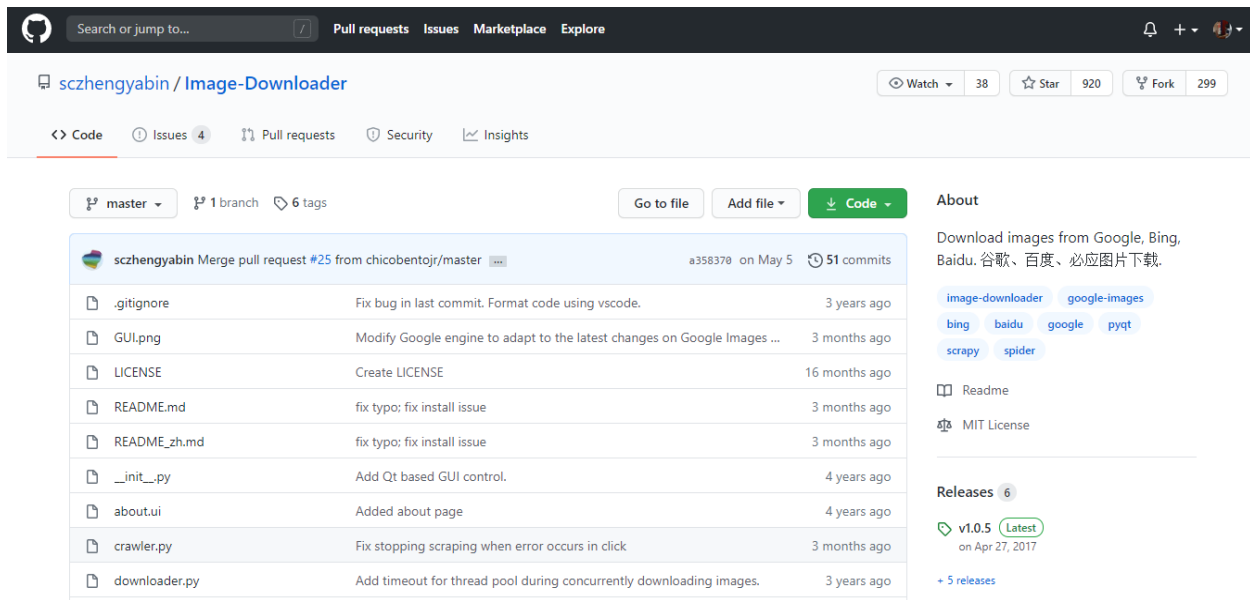


Figure 38. Mã nguồn của tác giả Yabin Zheng

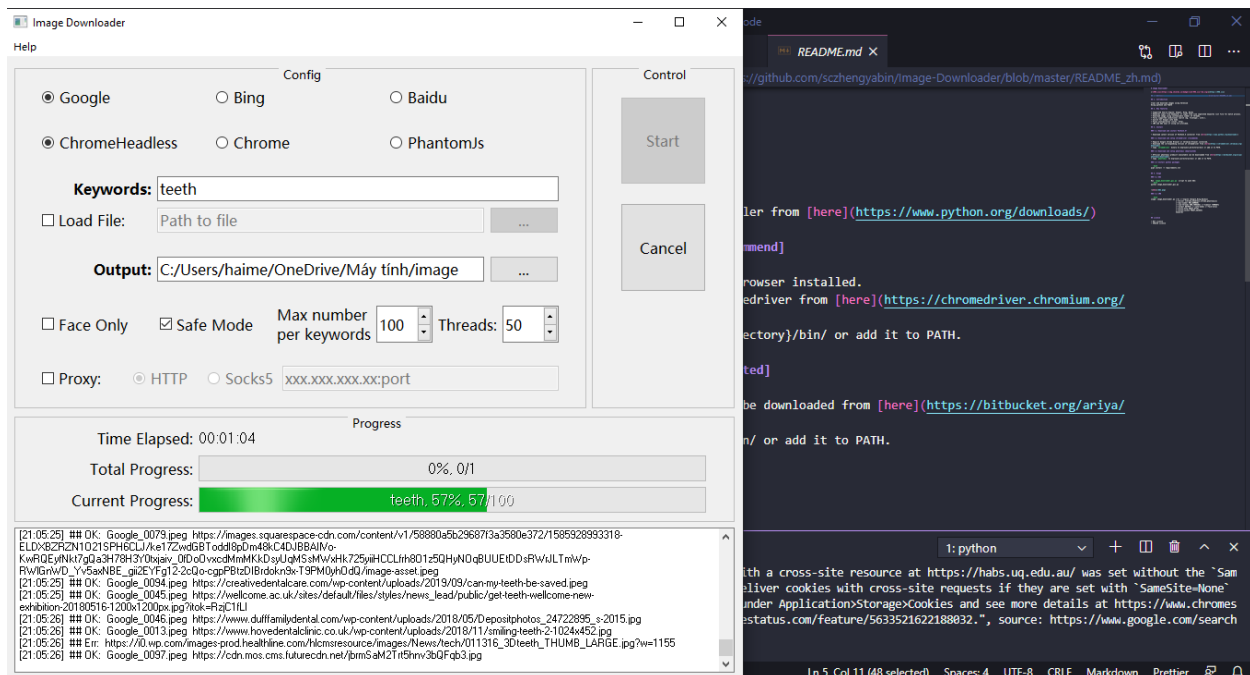


Figure 39. Quá trình crawl ảnh với keyword là teeth, số lượng chỉ định là 100.



Figure 40. Đánh giá model dựa trên dataset Teeth tự crawl bao gồm 145 ảnh về răng

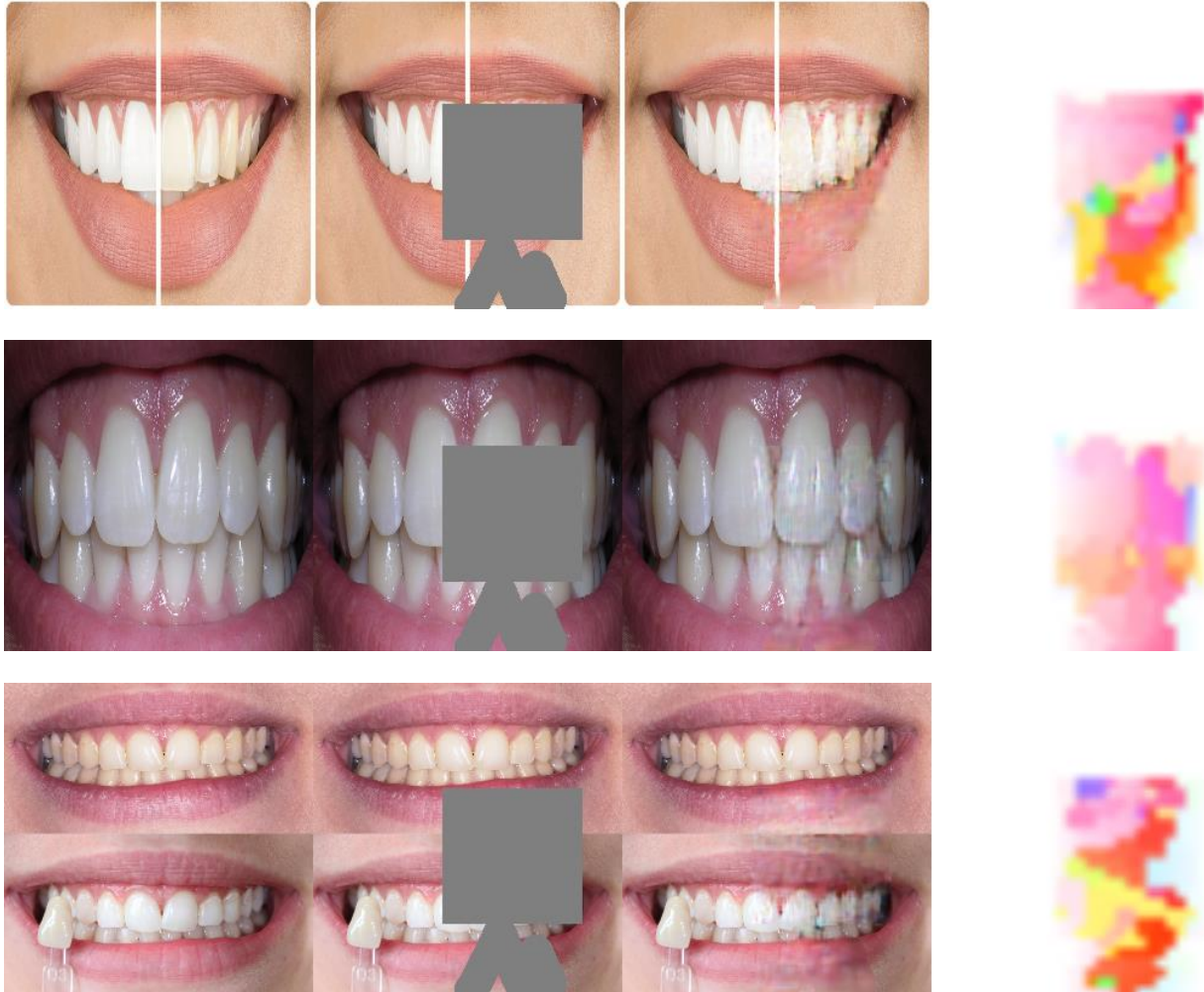
Vì chính sách của Google nên không thể crawl với số lượng lớn. Do vậy, dataset tương đối nhỏ, độ nhiễu lớn và kích thước đa dạng.

## 6.2. Training

Model có tổng cộng 9999294 weight và được implement trên Tensorflow 1.15, phần cứng CPU Intel Xeon E3 12xx v2 (Ivy Bridge) 3 GHz (8 processor), không có GPU. Model thử nghiệm xử lý khoảng 4s trên CPU với ảnh 256x256. Thời gian training đến ngày 23/7 là 39 ngày, tổng số epoch hoàn thành là 4.



### 6.2.1. Data được sử dụng trong quá trình training



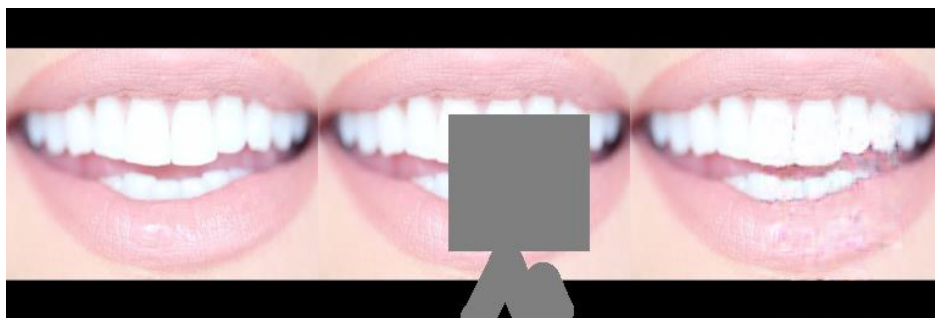
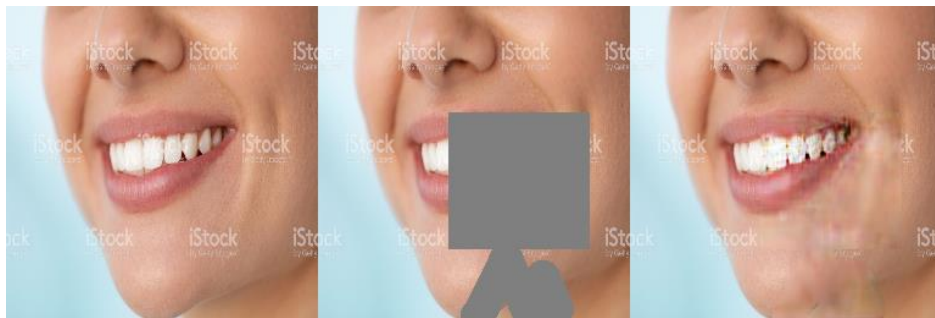
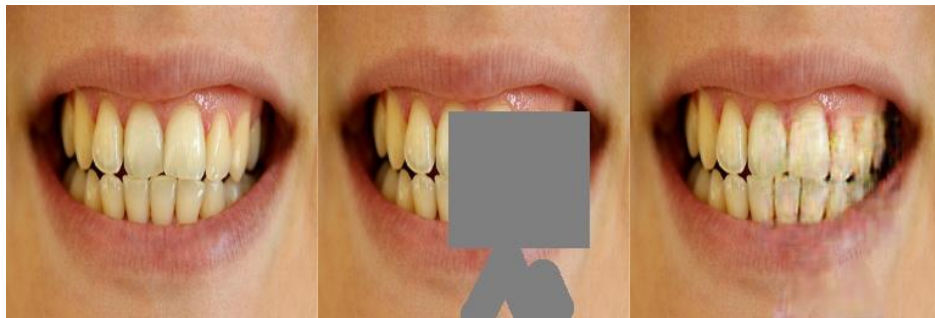




Figure 41. Lần lượt từ trái sang phải, GT – GT và binary mask, ảnh sau Inpainting và ma trận Attention Score.

## 6.2.2. Metric

### 6.2.2.1. G loss – 11.38%

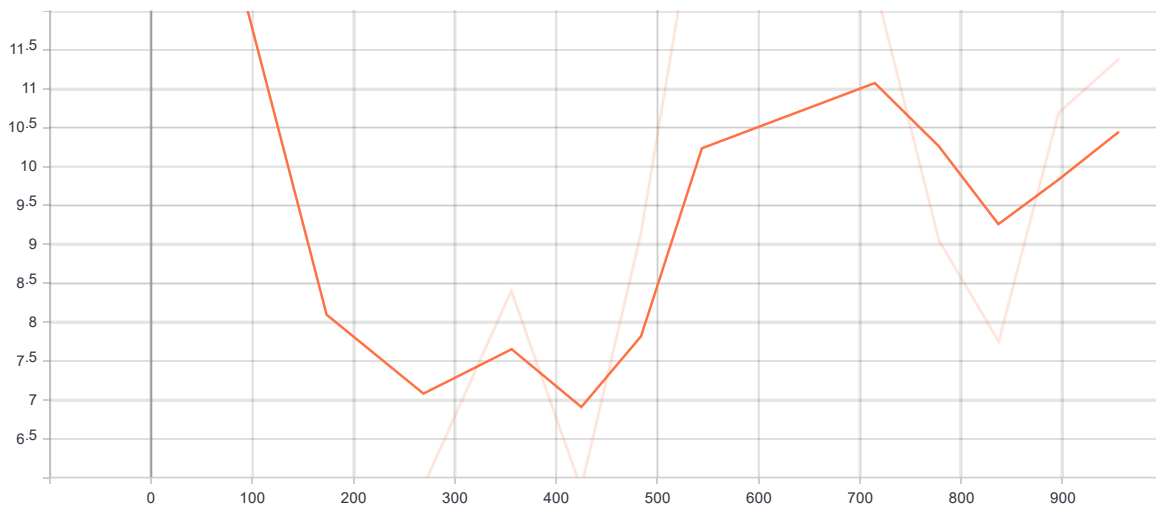


Figure 42. Đồ thị G loss với trục hoành là thời gian, trục tung là giá trị G loss



### 6.2.2.2. AE loss – 14.51%

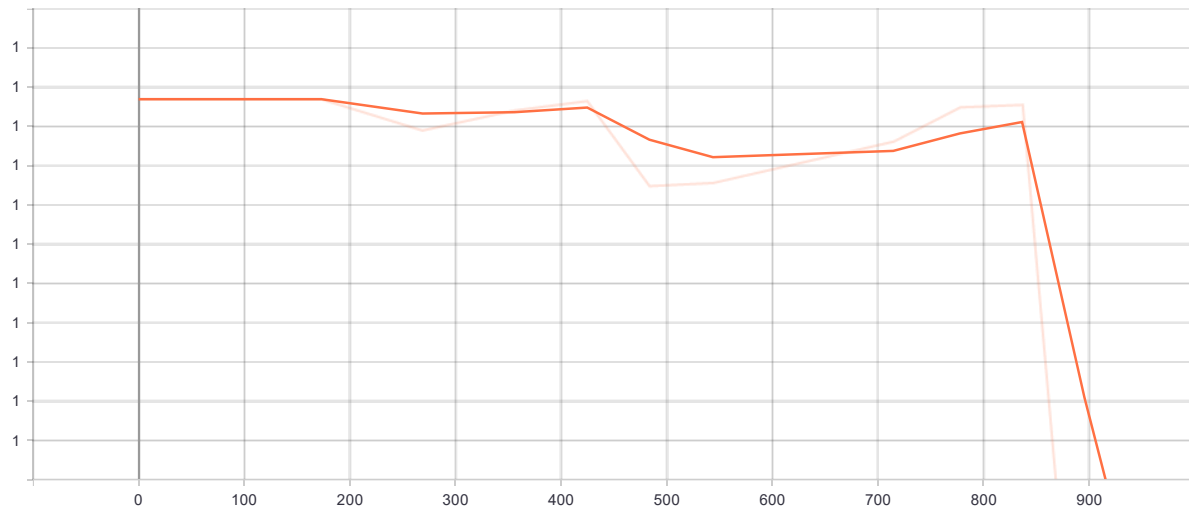


Figure 43. Đồ thị AE loss với trục hoành là thời gian, trục tung là giá trị AE loss

## Chương 7. Kết luận

### 7.1. Kết quả đạt được

Đề tài đã sử dụng những bài báo, tài liệu mới nhất về xử lý ảnh tại các tạp chí uy tín như CVPR2018, CVPR2019 để áp dụng vào một bài toán cụ thể. Đề tài đạt được những nội dung sau:

- Tìm hiểu, cài đặt và xây dựng model tách vật thể ra khỏi nền (hay cụ thể là tách mắc cài khỏi ảnh) bằng Python.
- Tìm hiểu, cài đặt và triển khai công cụ crawl dữ liệu tự động trên google image
- Tìm hiểu các model sinh ảnh GAN, DCGAN, WGAN, WGAN - GP.
- Xây dựng model sinh ảnh liên quan đến răng hàm lợi.

Hạn chế:

- Khi xóa mắc cài, model còn gặp nhiều lỗi khi có dây cung và không thể đạt tỉ lệ 100%, model thường xóa lan ra những vùng lân cận.
- Chất lượng ảnh sinh ra từ model WGAN phụ thuộc vào số lượng ảnh trong dataset và thời gian training (số lượng epoch).

Nhận định chung: sau khi thực hiện đề tài, em thấy mình cần thêm phần cứng mạnh và tìm hiểu những phương pháp khác để tăng chất lượng xử lý ảnh.

### 7.2. Thuận lợi & Khó khăn

#### 7.2.1. Thuận lợi

- Giảng viên hướng dẫn tận tâm, hỗ trợ em tối đa từ việc định hướng đề tài, phần cứng và hướng dẫn viết báo cáo.
- Thường xuyên gặp gỡ với các nhóm xử lý ảnh khác nên được trao đổi và học hỏi nhiều kinh nghiệm.
- Mã nguồn và các bài báo liên quan đều được công khai với giấy phép mã nguồn mở.

#### 7.2.2. Khó khăn

- Đề tài yêu cầu kiến thức sâu về deep learning và những ý tưởng mới nên ít tài liệu tiếng anh lẫn tiếng việt để tìm hiểu và ứng dụng.
- Đề tài thuộc một mảng chuyên ngành hẹp và ít được quan tâm.
- Model cần sử dụng nhiều tài nguyên tính toán và dataset lớn để đạt được độ chính xác cao hơn.
- Chưa có điều kiện thực nghiệm với số lượng lớn.

### 7.3. Hướng phát triển

- Chuyển đổi sang model CycleGAN.
- Tăng cường dataset bằng các kỹ thuật data argument và sử dụng transfer learning để giảm thời gian training.
- Xây dựng GUI trên Web để tương tác dễ hơn.

## TÀI LIỆU THAM KHẢO

- [1] M. S. P. Dinesh Naik, "Fast Interactive Superpixel Based Image Region Generation," *IJITEE*, pp. 1-7, 6 2019.
- [2] A. S. K. S. A. L. Radhakrishna Achanta, "SLIC Superpixels," *EPFL Technical*, pp. 1-15, 2010.
- [3] O. B. K. A. Anders P. Eriksson, "Image Segmentation Using Minimal Graph Cuts," *SSBA Symposium on Image Analysis*, pp. 1-3, 2016.
- [4] Z. L. J. Y. X. S. X. L. T. S. H. Jiahui Yu, "Generative Image Inpainting with Contextual Attention," *CVPR*, pp. 1-15, 2018.
- [5] Z. L. J. Y. X. S. X. L. T. H. Jiahui Yu, "Free-Form Image Inpainting with Gated Convolution," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4471-4480, 2019.
- [6] T. Khanh, "vincentherrmann.github.io," 2017 7 7. [Online]. Available: <https://vincentherrmann.github.io/blog/wasserstein/>. [Accessed 5 6 2020].
- [7] V. Herrmann, "vincentherrmann," 24 2 2017. [Online]. Available: <https://vincentherrmann.github.io/blog/wasserstein/>.
- [8] J. Hui, "medium.com," 14 6 2018. [Online]. Available: [https://medium.com/@jonathan\\_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490](https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490).
- [9] e. a. Ishaan, "Improved training of wasserstein gans," *Advances in neural information processing systems*, pp. 5767-5777, 2017.
- [10] L. B. Martin ArjovskySoumith Chintala.  
]
- [11] S. C. L. B. Martin Arjovsky, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.  
]
- [12] O. Bandura, "dentistryiq.com," 24 1 2018. [Online]. Available:  
] <https://www.dentistryiq.com/dentistry/pathology/article/16367502/computer-vision-in-dentistry-using-cad-and-cbct-systems-to-detect-tooth-pathology#:~:text=Pathology->

,Computer%20vision%20in%20dentistry%3A%20Using%20CAD%20and%20CBCT%20systems%20to,computed%. [Accessed 3 4 2020].

[13 O. Chudakov, "scnsoft.com," 4 4 2014. [Online]. Available: <https://www.scnsoft.com/blog/the-state-of-the-art-in-dental-image-analysis>. [Accessed 4 4 2020].