**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO MÔN DEEP LEARING**

**STARGAN**

**GVHD : TS. NGUYỄN THIÊN BẢO**

**SVTH : LÊ MINH TÂN**

**TRẦN XUÂN TUẤN**

**NGUYỄN NHẬT HÀO**

**LƯU VĂN TUÂN**

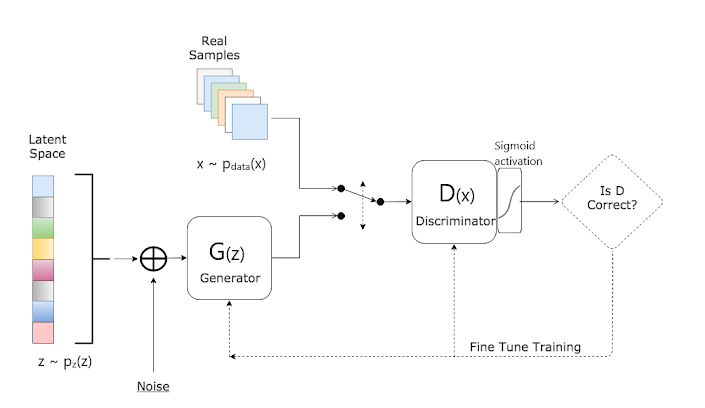
**Tp. Hồ Chí Minh, tháng 06 năm 2019**

# Cơ sở lý thuyết

## Sơ lược về mô hình GAN

Neural network vẫn đang phát triển rất mạnh trong những năm gần đây. Trong năm 2017, 1 loại neural network có tên là GAN (Generative adversarial network) đã phát triển rất mạnh mẽ.

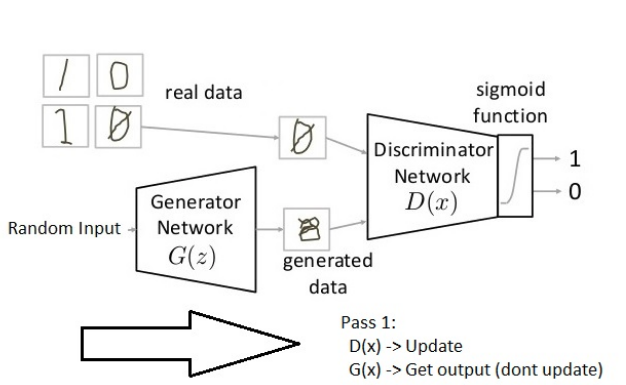
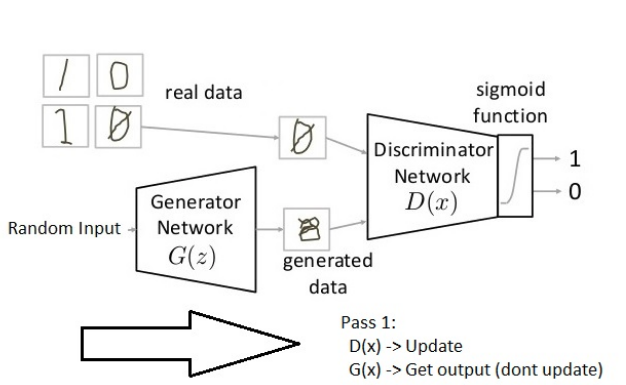
Generative Adversarial Network - tạm dịch Mạng Chống đối Tạo sinh - hình thành trên ý tưởng về sự cạnh tranh của hai mạng neural network:

* Discriminative network (mạng phân biệt): có nhiệm vụ cố gắng phân biệt giữa data thật và data giả mạo.
* Generative network (mạng sinh): sinh ra data giả, và mục tiêu là làm sao cho sinh ra được các data giống với thật nhất, làm Discriminator không thể phân biệt nổi. Các sample được sinh dựa trên các *latent code (mã ngầm)* ***z.***  
  

Generator lấy một random input và cố gắng sinh ra được sample của data. Như mô tả ở hình trên, generator G(z) lấy input z từ pz(z), với z là sample thuộc phân phối xác xuất p(z), được sinh ngẫu nhiên từ latent space, sau đó gán thêm nhiễu (noise). Sample sinh ra từ G(z) được nạp vào Discriminator network D(x). Công việc của Discriminator network là lấy input từ tập train (real sample) và sample được sinh ra từ G (generated sample) và xác định xem sample nào mới là thật. Real sample x được lấy từ phân phối xác suất pdata(x).

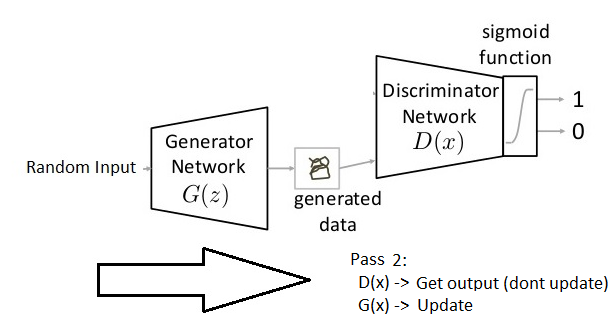
D(x) xử lý vấn đề binary classification bằng cách sử dụng hàm sigmoid, trả về kết quả khoảng từ 0 đến 1, với xác suất đầu ra càng cao thì khả năng sample đó là thật (sample lấy từ tập data) càng lớn, và ngược lại.  
D được huấn luyện để tối đa xác suất gán đúng nhãn cho sample, đồng thời G lại được huấn luyện để tối thiểu khả năng phát hiện của D, tương đương tối thiểu *log(1 - D(G(z))*.

Huấn luyện Discriminator network, cố định Generator network (cố định ở đây nghĩa là chỉ feed-forward, không backpropagation)



D được update Stochastic gradient sau mỗi bước train bằng cách công thêm giá trị:  
 \nabla_{\theta_{d}}\dfrac{1}{m}\sum_{i=1}^{m}\left[\log(D(x^{(i)}) + \log(1-D(G(z)))\right]

Huấn luyện Generator network, cố định Discriminator network



G được update Stochastic gradient bằng cách trừ đi giá trị:

Đang chèn hình ảnh...

## Ý tưởng StarGAN

Mô hình GAN về bản chất rất phù hợp cho yêu cầu dịch ảnh. Tuy nhiên, một số yếu tố khiến GAN trở nên lỗi thời trong việc đáp ứng dịch ảnh đa miền. Chẳng hạn, nếu xem là số miền mô hình cần đáp ứng, số generator trong mô hình là:

Mỗi generator chỉ tạo được dữ liệu theo một cặp miền nhất định, với một miền ứng với dữ liệu giả, miền còn lại là dữ liệu thật. Vậy việc chọn cặp miền ở đây chính là bài toán chỉnh hợp, vì mỗi miền được chọn có một vai trò nhất định không thể hoán đổi. Công thức trên có thể chứng minh lại bằng phép toán chỉnh hợp: Chọn hai miền trong số k miền:

Vậy giả sử nếu ta có bốn miền, số generator cần thiết là . Từ ví dụ này. ta có thể khái quát bài toán chỉnh hợp bằng mô hình.

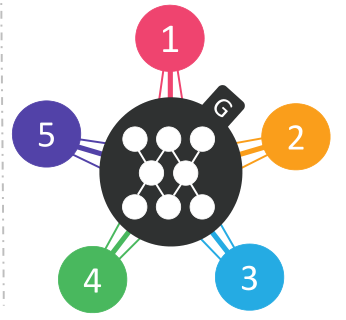
A close up of a sign

Description automatically generated

Chỉ với bốn miền, ta đã phải cần tới mười hai máy generator. Con số này chắc chắn sẽ tăng rất nhanh nếu số lượng miền tăng thêm. Một điểm yếu nữa là khả năng chia sẻ tham chiếu rất kém, vì các generator hoạt động độc lập, với các mục đích rất khác nhau. Ở yêu cầu cao hơn, ta còn mong muốn kết hợp nhiều dataset với nhau để tăng thêm số lượng miền. Đơn giản vì một dataset đôi lúc là không đủ đáp ứng số miền yêu cầu.

Từ những vấn đề trên, ý tưởng StarGAN được ra đời. StarGAN được định nghĩa là một mô hình nâng cấp từ GAN, có khả năng học ánh xạ giữa các tên miền với nhau. Về mặt ý tưởng, thay vì học cách dịch ảnh từ A sang B (ví dụ: nam sang nữ), mô hình nhận đầu vào cùng lúc hai bức ảnh và thông tin miền (quan trọng nhất là nhãn) và học cách dịch ảnh đến tên miền là “nữ”.

Ngoài ra, StarGAN chỉ dùng duy nhất một generator. Việc học dịch ảnh đa miền cũng chính là học cách ánh xạ từ A sang B một cách tự nhiên chứ không cứng nhắc. Ví dụ, dataset MNIST có mỗi ảnh ứng với một nhãn duy nhất (một con số từ 0 đến 9). Mô hình GAN có thể tạo ảnh dựa vào thông tin nhãn đưa vào để tạo ra ảnh fake ứng với nhãn đó. Nhưng với dataset mà mỗi ảnh có nhiều miền (nhãn) thì mô hình cần tự học cách ánh xạ nhờ vào việc học nhãn dưới dạng một vector ngay từ đầu. Mỗi phần tử trong vector ứng với một nhãn, giá trị là -1 nếu không thuộc nhãn đó và ngược lại là 1. Vì chỉ có một generator, ta có thể xem nó là trung tâm như mô hình dưới đây:



Mối quan hệ giữa các miền hình thành ánh xạ này và được hình thành từ đặc điểm chung của các bức hình. Lấy ví dụ đơn giản, nếu ta có 1000 bức ảnh huấn luyện, mỗi ảnh thuộc nhiều miền khác nhau (tóc vàng, tóc nâu, giới tính nam,…), nhưng tất cả 1000 ảnh đó đều thuộc chung một miền là “trẻ tuổi” thì ánh xạ liên quan đến miền trẻ tuổi nhiều khả năng sẽ hiệu quả hơn.

Ảnh dưới đây mô tả cách generator và discriminator của starGAN hoạt động:

A screenshot of a cell phone

Description automatically generated

Ở bước huấn luyện, discriminator nhận dữ liệu thật và dữ liệu giả. Theo ví dụ, dữ liệu thật thuộc tên miền A (nam), trong khi dữ liệu giả thuộc tên miền còn lại (nữ).

Generator nhận ảnh đầu thuộc miền “nam”, target domain là nhãn ta cần dịch ảnh đến, ở đây là “nữ”. Một bức ảnh fake được generator tạo ra để thực hiện hai chức năng chính:

* Kết hợp với tên miền gốc (tên miền của ảnh đầu vào là “nam”) đưa vào generator để sinh ra ảnh phù hợp với domain gốc. Ảnh này sẽ đóng vai trò là ảnh đầu vào kết hợp target domain. Đây là vòng tuần hoàn dành riêng cho generator.
* Bức ảnh fake sinh ra ban đầu còn được đưa vào discriminator để phân loại. Giai đoạn này giống với GAN truyền thống.

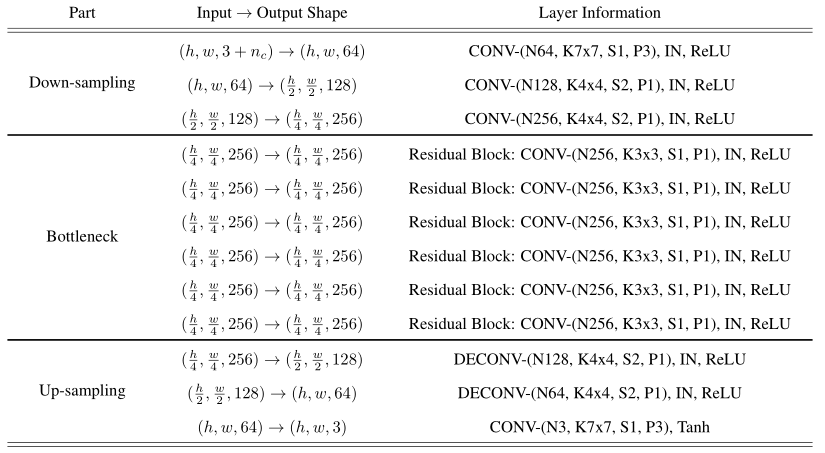
Một khái niệm tiếp theo cần được quan tâm là mask vector. Thực sự để có thể học nhiều tên miền từ hai hay nhiều dataset khác nhau, ta phải học lần lượt các dataset. Giả sử dataset A có miền là “nam”, dataset B (học sau đó) có miền là “nữ” thì sẽ phát sinh khó khăn, khi máy học phải được đưa vào cả hai bức ảnh ứng với hai tên miền cùng lúc. Giải pháp đưa ra là thiết kế mô hình có thể bỏ qua miền nó chưa biết và tập trung cho miền đã hoặc đang được học. Mask vector chính là tên của vector đại diện cho một đối tượng tạm thời chưa được biết đến.

StarGAN có thể được huấn luyện bởi hai tập dữ liệu cùng lúc thông qua mask vector. Gọi là một mask vector có chiều, bằng với số dataset. là vector chứa các vector con , chứa thông tin nhãn trong dataset thứ .

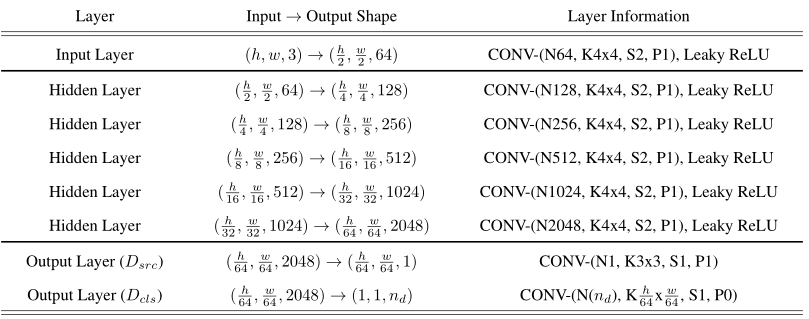
thường là one-hot vector. Lấy ví dụ, với vector , giá trị phần tử tại vị trí bằng 1 khi dữ liệu ứng với nhãn có dataset thứ nhất, bằng 0 trong trường hợp ngược lại. Với những không có dữ liệu (do không ứng với bất kì nhãn nào trong dataset ) thì giá trị toàn bộ là 0.

Về mặt lý thuyết, ta có thể chọn giải pháp dễ hiểu hơn là tiền xử lý dữ liệu. Tuy nhiên, hiệu năng và tốc độ sẽ không đạt hiệu quả cao, nhất là với các dataset dung lượng lớn thì việc trải qua hai giai đoạn tiền xử lý và học sẽ tốn rất nhiều thời gian và tài nguyên.

Sang phần thực nghiệm, ta sẽ mô tả lại ý tưởng StarGAN, nhưng dựa trên dataset cụ thể.



*Kiến trúc mạng mô hình generator*



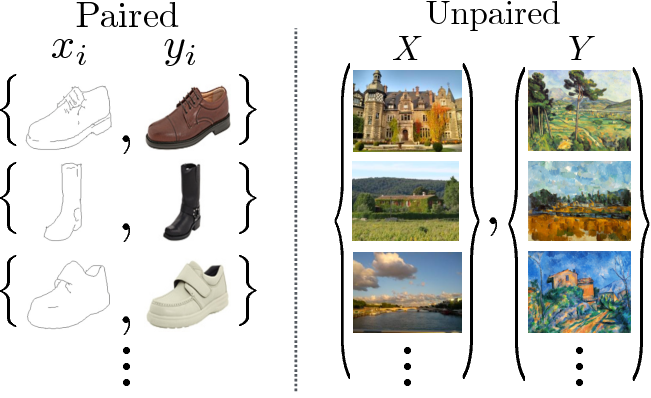
Kiến trúc mạng mô hình discriminator

## Dịch ảnh và dịch ảnh đa miền

### Dịch ảnh

Dịch ảnh (Image-to-Image Translation) là quá trình biến đổi một bức ảnh input X trở thành một bức ảnh Y theo mong muốn của mình

Dịch ảnh có 2 kết quả xảy ra:



* Paired: 2 bức ảnh được xem là paired khi ánh xạ input X và nhận được kết quả tương ứng là Y (G: X  Y)
* Unpaired: ngược lại với paired, Unpaired xảy ra khi input X không thể ánh xạ tương ứng với output Y và ngược lại. Ở kết quả này sẽ có 2 trường hợp xảy ra: input X sẽ ánh xạ với một hoặc nhiều output Y và ngược lại.

### Dịch ảnh đa miền

Dịch ảnh đa miền (Multi-Domain Image-to-Image Translation) mục tiêu là đào tạo một trình tạo G duy nhất để học ánh xạ giữa nhiều miền.

Để đạt được điều này, phải đào tạo G để dịch một hình ảnh đầu vào *x* thành một hình ảnh đầu ra *y* có điều kiện trên nhãn miền đích *c, G (x, c) → y.* Và tạo ngẫu nhiên nhãn tên miền đích *c* để *G* học cách dịch linh hoạt hình ảnh đầu vào. Nó chủ yếu tạo nhãn miền đích *c* để *tt* học cách dịch linh hoạt hình ảnh đầu vào. Chúng tôi cũng giới thiệu một trình phân loại phụ trợ cho phép một trình phân biệt đối xử duy nhất kiểm soát nhiều miền. Đó là, bộ phân biệt đối xử của chúng tôi cung cấp phân phối xác suất trên cả hai nguồn và nhãn chính, D: x → {Dsrc (x), Dcls (x)}.

A screenshot of a cell phone

Description automatically generated

Để làm cho hình ảnh được tạo không thể phân biệt được với hình ảnh thực, chúng tôi chấp nhận mất Adversarial Loss.

L*adv* = E*x* [log *Dsrc*(*x*)] + E*x,c*[log (1 − *Dsrc*(*tt*(*x, c*))]*, (1)*

trong đó *tt* tạo ra một hình ảnh *tt (x, c)* được điều hòa trên cả hình ảnh đầu vào *x* và nhãn miền đích *c*, trong khi *D* cố gắng phân biệt giữa hình ảnh thật và giả. Trình tạo *tt* cố gắng giảm thiểu mục tiêu này, trong khi người phân biệt đối xử *D* cố gắng tối đa hóa nó.

**Domain Classification Loss :** Đối với một hình ảnh đầu vào *x* và nhãn tên miền *c* đã cho, mục tiêu là dịch *x* thành hình ảnh đầu ra *y*, được phân loại chính xác cho miền đích *c*. Để đạt được điều kiện này, chúng tôi thêm một bộ phân loại phụ trợ lên trên *D* và áp đặt mất phân loại miền khi tối ưu hóa cả *D* và *tt*. Đó là, chúng tôi phân tách mục tiêu thành hai thuật ngữ: mất phân loại miền của hình ảnh thực được sử dụng để tối ưu hóa *D* và mất phân loại miền của hình ảnh giả được sử dụng để tối ưu hóa tt. Cụ thể, trước đây được định nghĩa là



Trong đó thuật ngữ *Dcls (c | x)* đại diện cho phân phối xác suất trên các nhãn miền được tính bằng *D*. Bằng cách giảm thiểu mục tiêu này, *D* học cách phân loại hình ảnh thực x cho miền gốc phản ứng của nó *c*. Chúng tôi giả định rằng cặp hình ảnh đầu vào và nhãn tên miền *(x, c ′)* được cung cấp bởi dữ liệu đào tạo. Mặt khác, chức năng mất cho phân loại miền của hình ảnh giả được định nghĩa là



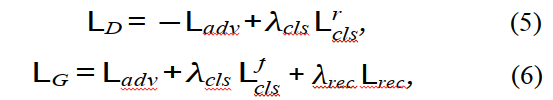
Nói cách khác, *tt* cố gắng giảm thiểu mục tiêu này để tạo ra các hình ảnh có thể được phân loại là miền đích *c.*

**Reconstruction Loss:** Bằng cách giảm thiểu các tổn thất đối nghịch và phân loại*, tt* được đào tạo để tạo ra các hình ảnh thực tế và được phân loại theo miền mục tiêu chính xác của nó. Tuy nhiên, giảm thiểu tổn thất (phương trình (1) và (3)) không bảo đảm rằng hình ảnh được dịch sẽ giữ nội dung của hình ảnh đầu vào trong khi chỉ thay đổi phần liên quan đến tên miền của đầu vào. Để giảm bớt vấn đề này, chúng tôi áp dụng tổn thất liên quan đến chu kỳ cho máy phát điện, được định nghĩa là:



trong đó *tt* lấy hình ảnh đã dịch *tt (x, c)* và nhãn tên miền gốc *c ′* làm đầu vào và cố gắng xây dựng lại hình ảnh gốc *x*. Chúng tôi áp dụng định mức L1 là mất mát tái thiết của chúng tôi. Lưu ý rằng chúng tôi sử dụng một trình tạo hai lần, lần đầu tiên để chuyển một hình ảnh gốc thành một hình ảnh trong miền đích và sau đó để tái tạo lại hình ảnh gốc từ hình ảnh được chiếu.

Cuối cùng, các chức năng mục tiêu để tối ưu hóa *tt* và *D* được viết tương ứng, như sau:



trong đó λcls và λrec là các tham số siêu kiểm soát tầm quan trọng tương đối của phân loại và tái cấu trúc miền, tương ứng, so với tổn thất đối nghịch.

# Thực nghiệm

## Dataset

### CelebA

**CelebFaces Attributes Dataset (**CelebA**)** là một dataset với hơn 200.000 hình ảnh các người nổi tiếng, với hơn 40 thuộc tính (attribute) được sử dụng trong việc nhận diện gương mặt.

* 202,599bức ảnh (mỗi bức ảnh có kích thước 218x178)
* 10,177bức ảnh
* 40 thuộc tính
* 5 toạ độ (landmark locations**)** để tạo đánh dấu vị trí người trong bức ảnh.



Cấu trúc file:

* In-The-Wild Images.zip: dataset ảnh gốc.
* Align&Cropped Images: dataset sau khi được xử lý (resize).
* Landmarks Annotations.txt: tất cả các vị trí đặc điểm của con người trên toạ độ x y.
* Attributes Annotations.txt: tất cả các thuộc tính được đánh nhãn nhị phân (“1” tương ứng với có, “-1” tương ứng với không có).
* Train/Val/Test Partitions.txt: dùng cho việc Train, Valid, Test (label 0 tương ứng với train, label 1 tương ứng với valid, label 2 tương ứng với test).

Việc thực nghiệm với dataset này không khó, khi mô hình đã có sẵn trên mạng và có thể dễ dàng tải về. Tuy nhiên, khi thực hiện trên nhiều dataset (kết hợp với FEI Face), ta sẽ giảm số lượng ảnh học theo quy tắc sau:

* Ảnh được chọn phải có ít nhất một trong số năm nhãn: 'Black\_Hair', 'Blond\_Hair', 'Brown\_Hair', 'Male', 'Young'.
* Số lượng ảnh mỗi nhãn ở mức tương đối (không quá chênh lệch về số lượng ảnh).

Số lượng ảnh được chọn là 360. Toàn bộ ảnh sẽ dùng cho việc train (không trích một phần cho việc test) do chúng chỉ được sử dụng cho demo kết hợp FEI Face, vốn dùng ảnh bên ngoài cụ thể để test.

### FEI Face

**FEI Face** là một dataset với các hình ảnh của người Brazil được thu thập từ tháng 6 năm 2005 đến tháng 3 năm 2006. Dataset bao gồm 2800 hình của 200 người với 14 vị trí khác nhau.



Các bức ảnh là ảnh màu, nền trắng, được chụp thẳng đứng với các vị trí được xoay 180 độ. Các bức ảnh có cùng kích thước 640x480.Người được chụp là học sinh hoặc nhân với của FEI với tỷ lệ nam/nữ là 50/50.

Lưu ý rằng khi thực nghiệm, nhóm chỉ lấy ảnh chụp chính diện. Như vậy, có tổng cộng 400 ảnh sẽ được dùng đến và 200 ảnh cho mỗi nhãn. Tỉ lệ chỉ train/test là 90%/10%.

## Huấn luyện

### CelebA

Input: Hình ảnh, file gán label cho từng hình tương ứng

Output: Model đã được train

Argument:

* Mode: Chế độ train hay test
* Dataset: lựa chọn dataset
* Image size: kích thước ảnh
* Dimension of domain labels: Số thuộc tính
* Sample directory: Thư mực chứa sample
* Log directory: Thư mục chứa log
* Lesult directory: thư mục chứa kết quả
* Model directory: thư mục chứa model
* Attribute: Những thuộc tính được chọn (tóc vàng, tóc nâu, giới tính nam, …)

Demo sử dụng model đã được train sẵn vì model tự train với 200 000 ảnh mất ít nhất 4 ngày cần nhiều tài nguyên và thời gian.

### FEI Face

Sau mỗi một đợt n iteration (trong demo này, n = 10), chương trình sẽ lưu lại mô hình, log và sample. Input để ra sample không từ thư mục test, mà là thư mục train và được lấy ngẫu nhiên 16 ảnh lúc mới bắt đầu train.

Mô hình sử dụng đã được huấn luyện qua 10530 iterations. Do cảm thấy chất lượng đầu ra ổn, nhóm dừng lại tại mốc này.

Cấu hình máy:

|  |  |  |
| --- | --- | --- |
| **STT** | **Phần cứng** | **Cấu hình** |
| 1 | CPU | Intel Xeon E3 1231v3 3.4 GHz |
| 2 | RAM | 12GB DDR3 |

Argument:

--mode train --dataset FEI --image\_size 128 --c\_dim 2 --sample\_dir stargan\_fei/samples --log\_dir stargan\_fei/logs --model\_save\_dir stargan\_fei/models --result\_dir stargan\_fei/results

Nhóm train chỉ thông qua CPU, thời gian train và các thông tin khác để ở mục kết quả.

### CelebA kết hợp FEI Face

Số lượng ảnh huấn luyện bằng với tổng số ảnh của demo CelebA và demo FEI Face. Thời gian thực hiện chắc chắn là lâu hơn. Cụ thể, nhóm tốn hơn mười hai phút để hoàn tất mỗi mười iteration. Cấu hình máy, tham chiếu không đổi so với demo FEI Face.

Mục tiêu quan trọng nhất của demo nhằm nêu lên lợi ích của việc sử dụng nhiều dataset khi mỗi dataset lại có những điểm mạnh, điểm yếu khá rõ ràng. Chẳng hạn, ảnh của CelebA được chụp từ nhiều hướng khuôn mặc khác nhau, chứa ảnh nền (yếu tố gây nhiễu) nhưng lượng ảnh rất đồ sộ, nhiều miền. Ngược lại, FEI Face chứa các ảnh có phần nền trắng, chụp chính diện.

Cấu hình máy:

|  |  |  |
| --- | --- | --- |
| **STT** | **Phần cứng** | **Cấu hình** |
| 1 | CPU | Intel Xeon E3 1231v3 3.4 GHz |
| 2 | RAM | 12GB DDR3 |

Argument:

--mode=train --dataset Both --image\_size 128 --c\_dim 5 --c2\_dim 2 --sample\_dir stargan\_both/samples --log\_dir stargan\_both/logs --model\_save\_dir stargan\_both/models --result\_dir stargan\_both/results

Nhóm train chỉ thông qua CPU, thời gian train và các thông tin khác để ở mục kết quả.

## Kết luận

### CelebA

Chuẩn bị data:

* Tại thư mục data/celeba/image: Chứa các hình đươc đánh số theo thứ tự.
* Tại thư mục data/celeba: Chuẩn bị file txt ghi tổng số hình muốn test và gán nhãn theo mẫu có sẵn, đặt tên là list\_attr\_celeba.txt.

Câu lệnh:   
python main.py --mode test --dataset CelebA --image\_size 128 --c\_dim 5    
                           --selected\_attrs Black\_Hair Blond\_Hair Brown\_Hair Male Young    
                           --model\_save\_dir='stargan\_celeba\_128/models'    
                            --result\_dir='stargan\_celeba\_128/results'

Kết quả:    
(sử dụng model đã được train vói 200000 iters)

1. Với  hình trong data celeba:



1. Thử nghiệm với 2 tấm hình tự cho:



1. Các thông số tại iter 200000

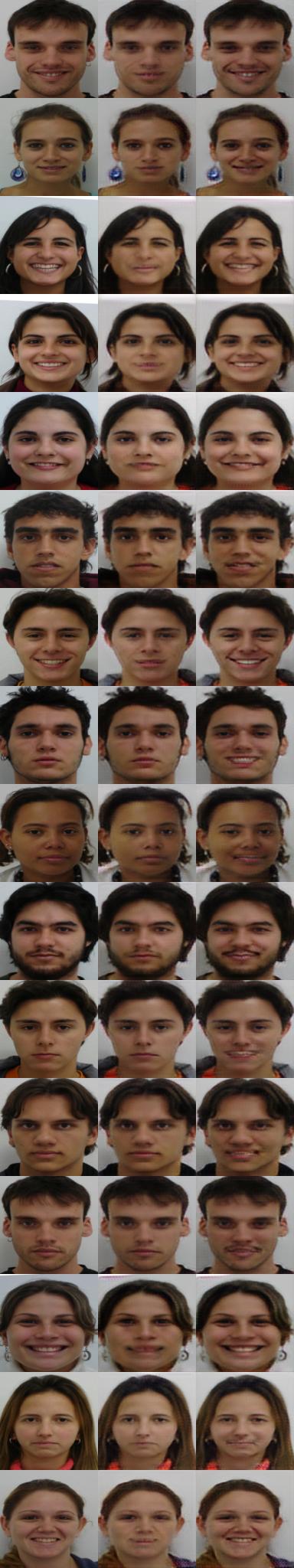
|  |  |  |
| --- | --- | --- |
| **STT** | **Thông số** | **Giá trị** |
| 1 | D/loss\_cls | 1.43 |
| 2 | D/loss\_fake | 9.56 |
| 3 | D/loss\_gp | 0.0038 |
| 4 | D/loss\_real | -14.82 |
| 5 | G/loss\_cls | 3.304 |
| 6 | G/loss\_fake | -8.875 |
| 7 | G/loss\_rec | 0.2577 |

Nhận xét:  Với những hình tự cho thì kết quả thu được không được mượt như những hình đưa vào train (Do kính thước, khung hình, …) Nhưng kết qả thu được rất khả quan.

### FEI Face

Arguments:

--mode test --dataset FEI --image\_size 128 --c\_dim 2 --fei\_image\_dir data/FEIFace/test --sample\_dir stargan\_fei/samples --log\_dir stargan\_fei/logs --model\_save\_dir stargan\_fei/models --result\_dir stargan\_fei/results --test\_iters 10530



*Ảnh sample tại iteration 10530.*



*Test thử từ ảnh bên ngoài*

Thông số tại iter 10350 như sau:

|  |  |  |
| --- | --- | --- |
| **STT** | **Thông số** | **Giá trị** |
| 1 | D/loss\_cls | 6.5565e-7 |
| 2 | D/loss\_fake | 0.7345 |
| 3 | D/loss\_gp | 8.2564e-3 |
| 4 | D/loss\_real | -2.125 |
| 5 | G/loss\_cls | 2.47e-3 |
| 6 | G/loss\_fake | -1.024 |
| 7 | G/loss\_rec | 0.06722 |
| 8 | Thời gian | 7 ngày, từ 14/5 đến 20/5. Có vài ngày mở máy 24/24.  Mỗi 10 iterations tốn hơn 7 phút. |
| 9 | Hình thức train | CPU-only. |

Thực tế là kết quả thu về phụ thuộc rất nhiều vào ảnh đầu vào. Nhiều lúc chất lượng đầu ra rất tốt, nhưng đôi lúc khuôn mặt tương đối biến dạng trong một số trường hợp.

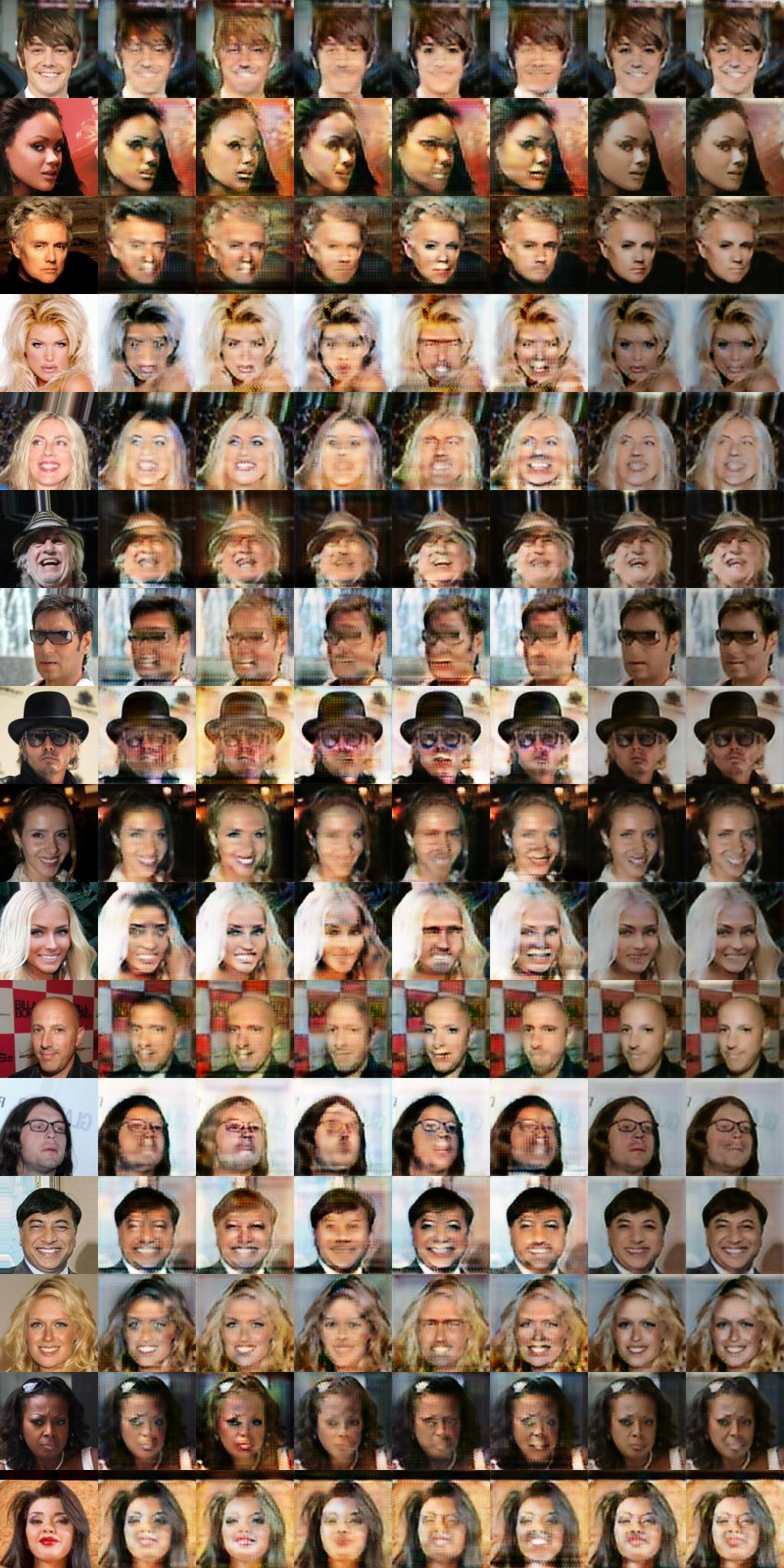
### CelebA kết hợp FEI Face

Arguments:

--mode test --dataset Both --image\_size 128 --c\_dim 5 --c2\_dim 2 --sample\_dir stargan\_both/samples --log\_dir stargan\_both/logs --model\_save\_dir stargan\_both/models --result\_dir stargan\_both/results --test\_iters 13580

Thông số tại iter 13580:

|  |  |  |
| --- | --- | --- |
| **STT** | **Thông số** | **Giá trị** |
| 1 | D/loss\_cls | 3.8743e-7 |
| 2 | D/loss\_fake | 5.579 |
| 3 | D/loss\_gp | 0.01257 |
| 4 | D/loss\_real | -7.262 |
| 5 | G/loss\_cls | 4.9174e-6 |
| 6 | G/loss\_fake | -5.863 |
| 7 | G/loss\_rec | 0.06146 |
| 8 | Thời gian | 17 ngày, từ 22/5 đến 7/6. Có vài ngày mở máy 24/24, có ngày chỉ train trong thời gian ngắn để máy nghỉ.  Mỗi 10 iterations tốn hơn 12 phút. |
| 9 | Hình thức train | CPU-only. |



*Sample tại iteration 13580*



*Test thử ảnh bên ngoài*

Để ý rằng kết quả sẽ có chút khác biệt nếu so với hai demo trước. Đơn giản vì cả hai dataset đều ảnh hưởng đến đầu ra chứ không riêng bất cứ một mô hình nào cả. ngoài ra, 13580 là chưa đủ nếu so với model được train sẵn 200000.

Tham khảo:

Github: <https://github.com/yunjey/stargan>

Paper: <https://arxiv.org/abs/1711.09020>

Deep Learning Notes: StarGAN: <https://mc.ai/deep-learning-notes-stargan/>

StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation: <https://lyusungwon.github.io/generative-models/2019/01/15/stargan.html>