BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC THĂNG LONG

BÁO CÁO

TRIỂN KHAI MODEL GAN

GIÁO VIÊN HƯỚNG DẪN

TS.NGUYỄN THỊ HUYỀN CHÂU

CN.ĐOÀN TRUNG PHONG

SINH VIÊN THỰC HIỆN

A42304 NGUYỄN VĂN HIẾU

NĂM HỌC: 2022-2023

LỜI GIỚI THIỆU

Tài liệu này cung cấp một cái nhìn tổng quan cho việcphát triển dự án. Bao gồm mục đích, phạm vi, các định nghĩa, thuật ngữ, từ viết tắt, các tham chiếu của dự án này.

Thực tế trong quá trình nghiên cứu, tại mỗi giai đoạn đều xây dựng một tài liệu khác nhau tương ứng với giai đoạn đó. Để giảm thiểu sự phức tạp của các tài liệu trong quá trình giảng dạy, tài liệu này được xây dựng một cách thống nhất trong suốt quá trình thực hiện.

Tài liệu này có phạm vi bao trùm tất cả các giai đoạn của quá trình phát triển dự án cho tới khi sẵn sàng đưa ra để sử dụng.

Mục lục

Chương 1

Giới thiệu chung

1.1 Mô tả bài toán

Ngày nay, trong ngành công nghiệp Anime và phim Pokemon, việc sử dụng trí tuệ nhân tạo đang trở thành một xu hướng quan trọng trong việc phát triển nhân vật. Thông qua việc áp dụng các thuật toán và công nghệ AI, các nhà làm phim có thể tạo ra những nhân vật đa dạng và phức tạp hơn, từ tính cách đến ngoại hình. Điều này không chỉ giúp tiết kiệm thời gian và chi phí mà còn tăng cường sự sáng tạo và mở rộng vũ trụ của câu chuyện. Việc ứng dụng AI trong xây dựng nhân vật không chỉ mang lại lợi ích trong quá trình sản xuất mà còn nâng cao trải nghiệm của khán giả thông qua các nhân vật đầy sức sống và đa chiều.

Mục tiêu của dự án này là phát triển một mô hình trí tuệ nhân tạo có khả năng tự động tạo ra các nhân vật Pokemon. Mô hình này sẽ có khả năng tạo ra các nhân vật với độ đa dạng và tính cách phong phú, giúp giảm bớt gánh nặng cho các nhà sản xuất trong việc phác họa nhân vật. Bằng cách sử dụng các thuật toán và dữ liệu từ series Pokemon trước đây, mô hình sẽ có khả năng tạo ra những nhân vật mới mẻ và độc đáo, đồng thời duy trì tính nhất quán với thế giới Pokemon đã được thiết lập. Điều này sẽ không chỉ tiết kiệm thời gian và chi phí cho quá trình sản xuất mà còn mở ra cánh cửa cho sự sáng tạo không giới hạn và mở rộng vũ trụ Pokemon đến những đỉnh cao mới.

Chương 2

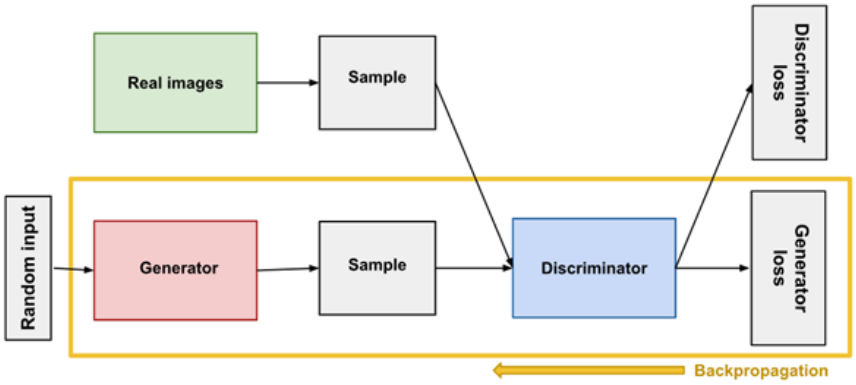
CỞ SỞ LÝ THUYẾT

2.1 Generative Adversarial Network?

Generative Adversarial Network là một mạng đối nghịch tạo sinh là môt hình sinh ra dữ liệu mới giống như dữ liệu trong dataset có sẵn. Mô hình bao gồm 2 mạng là Generator(Bộ tạo sinh) và Discriminator(Bộ phân biệt).

2.2 Cấu trúc mô hình

Model Gan thường có hai phần chính là Generator (Máy tạo) và Discriminator (Máy phân biệt)



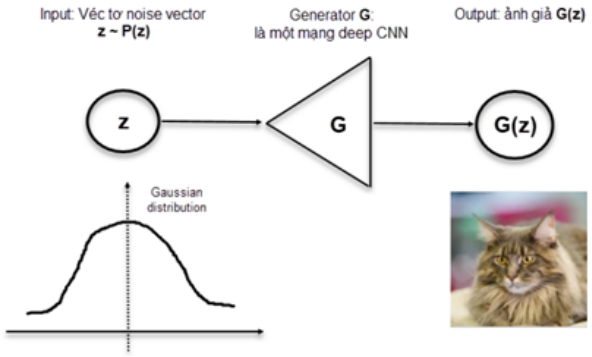
Hình 2.1: Kiến trúc của Gan

– Generator: Có nghiệm vụ học cách sinh ra dữ liệu giả để lừa model Discriminator. Để đánh lừa được Discriminator thì đòi hỏi mô hình phải sinh ra output thực sự tốt. Do đó chất lượng ảnh sinh ra càng giống thật càng tốt.

– **Discriminator: Có nhiệm vụ phân biệt giữa dữ liệu giả được sinh từ mô hình Generator với dữ liệu thật. Nếu Generator không qua mặt được Discriminator thì Generator cần tiếp tục học để có thể đánh lừa được Discriminator, đồng thời Discriminator cũng phải cải thiện khả năng phân biệt của mình vì chất lượng ảnh của Generator ngày càng tốt hơn.**

Có thể coi Generator và Discriminator là hai model xung đột lợi ích khi mà thiệt hại của người này chính là lợi ích của người kia. Mô hình Generator tạo ra dữ liệu giả tốt hơn sẽ làm cho Discriminator phân biệt khó hơn và khi Discriminator phân biệt tốt hơn thì Generator cần phải tạo ra ảnh giống thật hơn để qua mặt Discriminator.

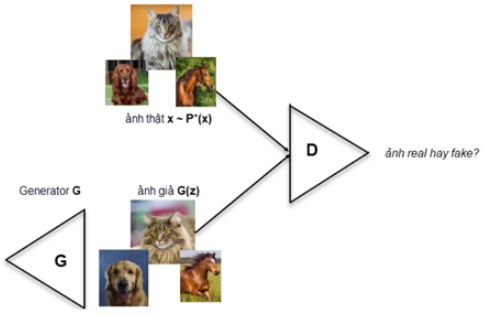
2.2.1 Generator



Hình 2.2: Kiến trúc của Generator

Generator về bản chất là một mô hình sinh nhận đầu vào là một tập hợp các véc tơ Z nhiễu được khởi tạo ngẫu nhiên theo phân phối Gaussian. Ở một số lớp mô hình GAN tiên tiến hơn, input có thể làm một dữ liệu chẳng hạn như bức ảnh, đoạn văn bản hoặc đoạn âm thanh.

2.2.2 Discriminator



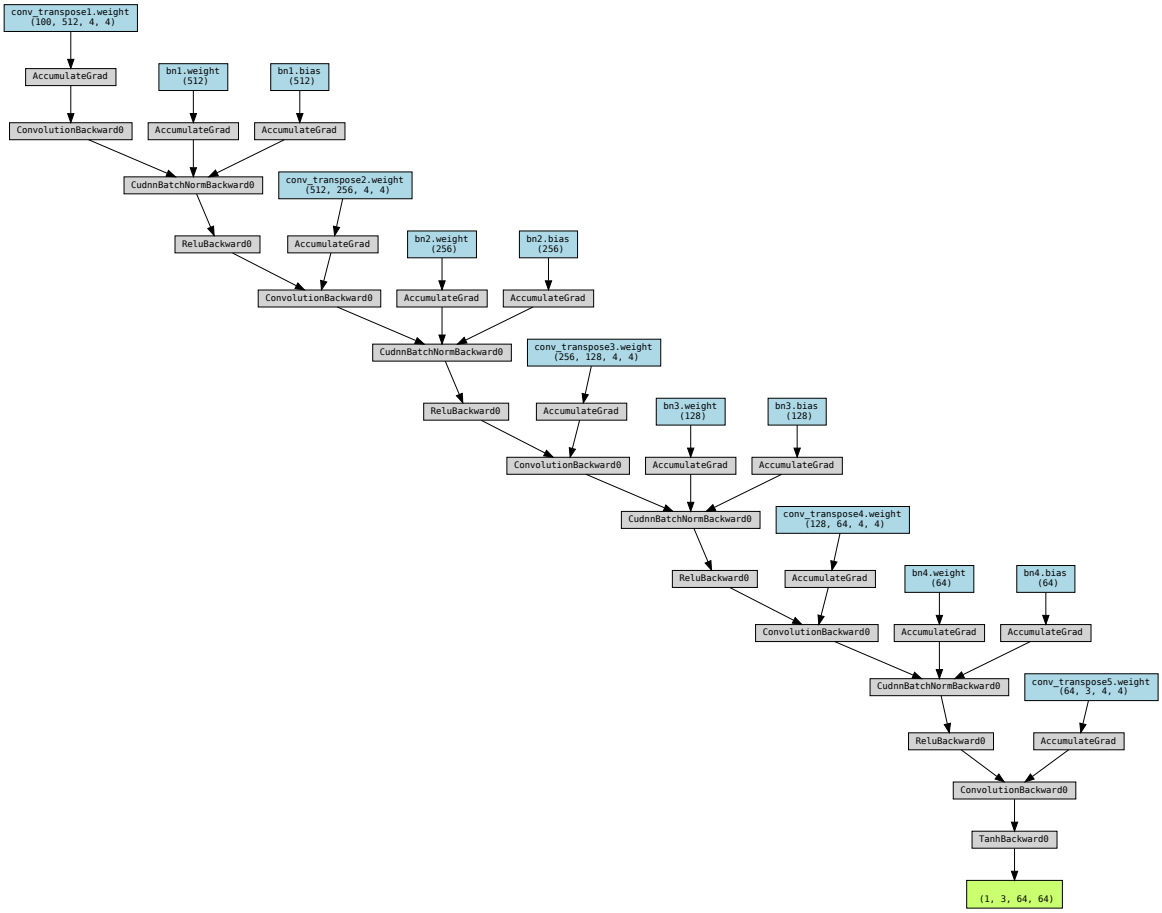
Hình 2.3: Kiến trúc của Discriminator

Mô hình Discriminator sẽ có tác dụng phân biệt ảnh input là thật hay giả. Nhãn của mô hình sẽ là thật nếu ảnh đầu vào của Discriminator được lấy tập mẫu huấn luyện và giả nếu được lấy từ output của mô hình Generator. Về bản chất đây là một bài toán phân loại nhị phân (binary classification) thông thường.

Chương 3

XÂY DỰNG MÔ HÌNH

3.1 Xây dựng Generator



Hình 3.1: Mô tả về cấu trúc của Generator

3.1.1 Kiến trúc mạng

– Hàm Generator này sử dụng mạng neural network tích chập chuyển vị (Convolutional Transpose) để thực hiện quá trình sinh ảnh.

– Bắt đầu từ một đầu vào ngẫu nhiên kích thước 100x1x1 (đại diện cho vectơ ngẫu nhiên) và tạo ra một ảnh có kích thước thích hợp (ví dụ: ảnh màu 3 kênh có kích thước 64x64).

3.1.2 Các lớp của mạng

– convtranspose1: Lớp tích chập chuyển vị với 100 kênh đầu vào và 512 kênh đầu ra, kích thước kernel là 4x4 với bước nhảy 1 và không padding.

– bn1: Lớp Batch Normalization với 512 kênh đầu ra.

– relu1: Hàm kích hoạt ReLU được áp dụng sau Batch Normalization.

– convtranspose2: Lớp tích chập chuyển vị với 512 kênh đầu vào và 256 kênh đầu ra, kích thước kernel là 4x4 với bước nhảy 2 và padding bằng 1.

– bn2: Lớp Batch Normalization với 256 kênh đầu ra.

– relu2: Hàm kích hoạt ReLU được áp dụng sau Batch Normalization.

– convtranspose3, bn3, relu3: Tương tự như lớp thứ hai nhưng với số kênh đầu ra giảm dần xuống 128.

– convtranspose4, bn4, relu4: Tương tự như lớp thứ ba nhưng với số kênh đầu ra giảm dần xuống 64.

– convtranspose5, tanh: Lớp tích chập chuyển vị cuối cùng với 64 kênh đầu vào và 3 kênh đầu ra (đại diện cho ảnh màu), kích thước kernel là 4x4 với bước đi 2 và padding bằng 1. Hàm kích hoạt Tanh được áp dụng để đảm bảo các giá trị pixel nằm trong khoảng [-1, 1].

3.1.3 Hàm forward

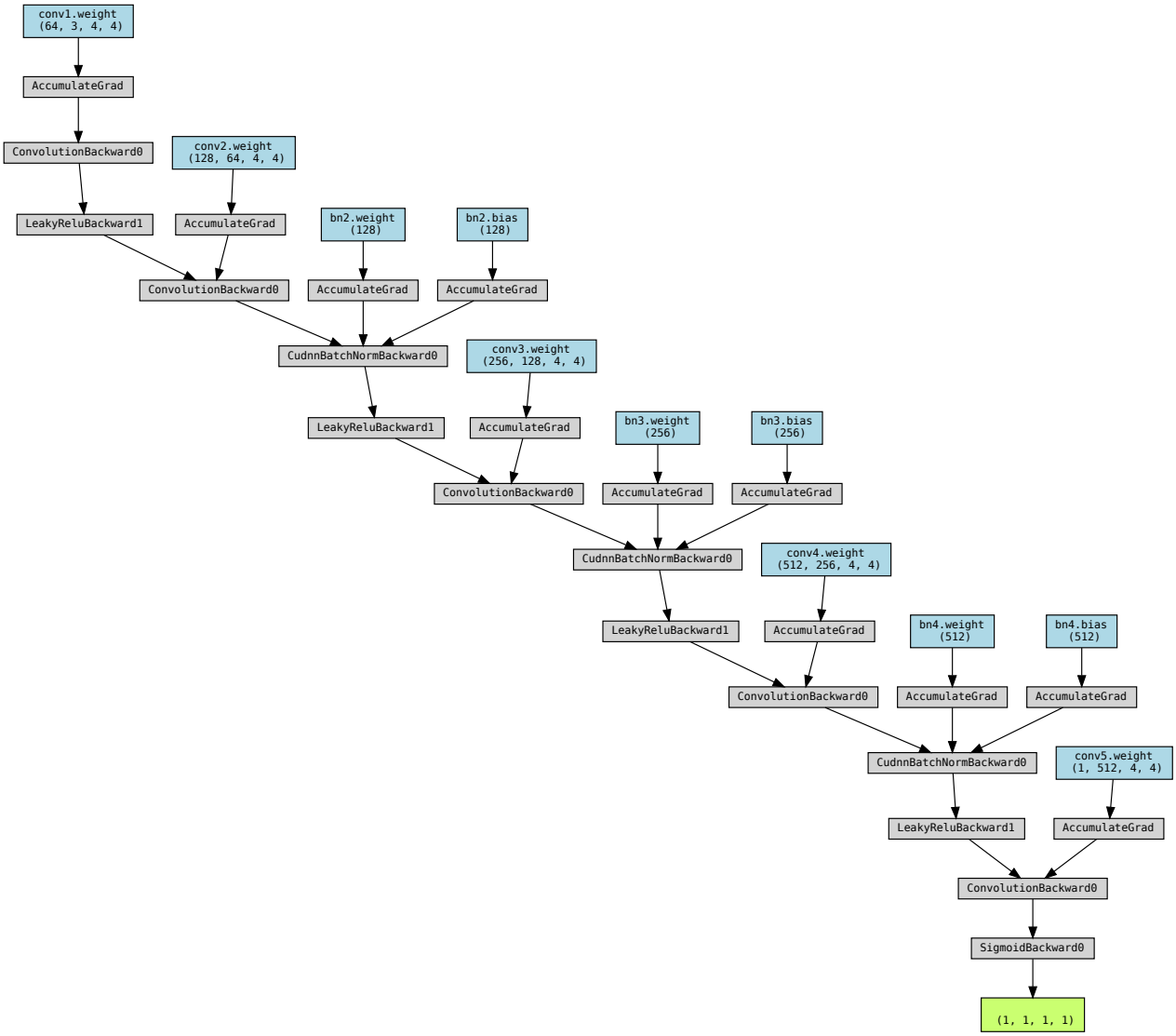
Hàm forward thực hiện các phép tích chập chuyển vị xen kẽ với các lớp Batch Normalization và hàm kích hoạt ReLU cho đến khi đạt được ảnh đầu ra. Cuối cùng, hàm kích hoạt Tanh được áp dụng để đảm bảo rằng các giá trị pixel của ảnh nằm trong khoảng [-1, 1], phù hợp với các giá trị của dữ liệu ảnh thông thường.

3.2 Xây dựng Discriminator

3.2.1 Kiến trúc mạng

– Hàm Discriminator này sử dụng một mạng neural network tích chập (Convolutional Neural Network) để phân loại ảnh đầu vào là thật hay giả.

– Đầu vào của mạng là một ảnh có kích thước phù hợp (ví dụ: ảnh màu 3 kênh có kích thước 64x64).



Hình 3.2: Mô tả về cấu trúc của discriminator

3.2.2 Các lớp của mạng

– conv1: Lớp tích chập với 3 kênh đầu vào (ảnh màu), 64 kênh đầu ra, kernel size là 4x4, bước đi là 2 và padding bằng 1.

– conv2: Lớp tích chập tiếp theo với 64 kênh đầu vào và 128 kênh đầu ra, kernel size là 4x4, bước đi là 2 và padding bằng 1.

– bn2: Lớp Batch Normalization với 128 kênh đầu ra.

– conv3: Lớp tích chập tiếp theo với 128 kênh đầu vào và 256 kênh đầu ra, kernel size là 4x4, bước đi là 2 và padding bằng 1.

– bn3: Lớp Batch Normalization với 256 kênh đầu ra

– conv4: Lớp tích chập tiếp theo với 256 kênh đầu vào và 512 kênh đầu ra, kernel size là 4x4, bước đi là 2 và padding bằng 1.

– bn4: Lớp Batch Normalization với 512 kênh đầu ra.

– conv5: Lớp tích chập cuối cùng với 512 kênh đầu vào và 1 kênh đầu ra (đại diện cho xác suất ảnh là thật hay giả), kernel size là 4x4, stride là 1 và không padding.

– sigmoid: Hàm kích hoạt Sigmoid được áp dụng để chuyển đổi giá trị đầu ra thành một giá trị xác suất nằm trong khoảng [0, 1].

– leakyrelu: Hàm kích hoạt Leaky ReLU được sử dụng trước các lớp tích chập để giảm vấn đề về vanishing gradient.

3.2.3 Hàm forward

Hàm forward thực hiện các phép tích chập xen kẽ với Batch Normalization và hàm kích hoạt Leaky ReLU cho đến khi đạt được đầu ra dự đoán xác suất của ảnh là thật hay giả.

3.3 Loss function

Dưới đây là cách tính lại sai số (loss) sử dụng hàm nn.BCELoss() trong mô hình Generative Adversarial Network (GAN):

3.3.1 Mất mát của Discriminator

- Đối với ảnh thật: errDreal = criterion(outputreal,reallabels)
- Đối với ảnh fake: errDfake =criterion(outputreal,reallabels)
- Tổng mất mát của Discriminator:epochDloss = errDreal + errDfake

3.3.2 Mất mát của Generator

Mất mát của Generator được tính từ kết quả đánh giá của Discriminator với các ảnh fake

- errG = criterion(output,reallabels)
- Tổng mất mát của Generator: epochGloss=errG

Lưu ý:

- outputreal là kết quả đầu ra của Discriminator cho các ảnh thật.
- outputfake là kết quả đầu ra của Discriminator cho các ảnh fake được tạo ra bởi Generator.
- reallabels là nhãn tương ứng cho ảnh thật (trong trường hợp này, là 1)
- fakelabels là nhãn tương ứng cho ảnh fake (trong trường hợp này, là 0).
- 'criterion' đại diện cho hàm mất mát, trong trường hợp này là hàm nn.BCELoss() (Binary Cross Entropy Loss).

Chương 4

TRAINING

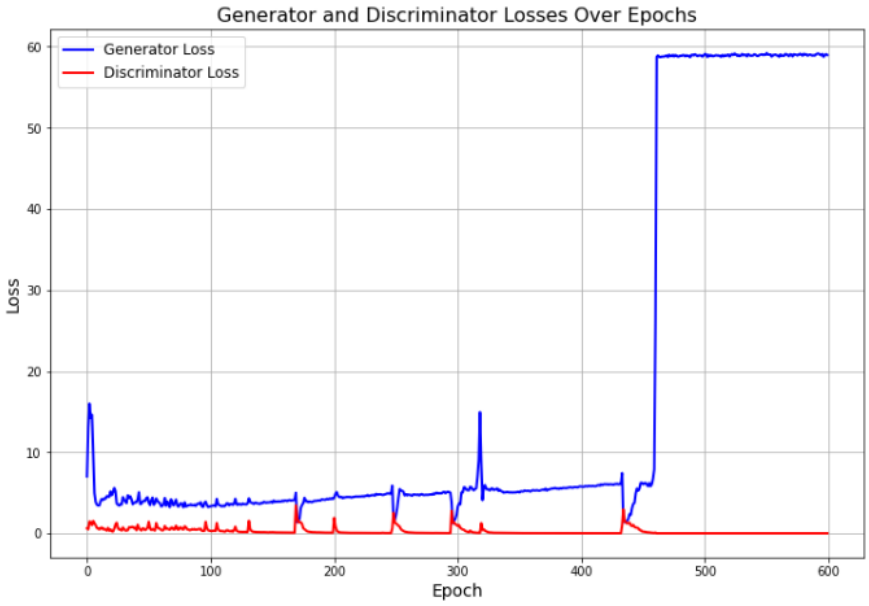
4.1 Data

Tập dữ liệu được download từ trên kaggle tại:https://www.kaggle.com/datasets/kvpratama/pokemo
images-dataset

Tập dữ liệu gồm có 2 file ảnh giống nhau nhưng khác nhau về định ảnh lag jpg và png. Mỗi tập dữ liệu gồm có 819 bức ảnh Pokemon.

4.2 Result

Sau khi train xong model ta thu được kết quả như sau:



Hình 4.1: Kết quả sau khi train mô hình

Như đã thấy từ mức 0 đến 400 epoch ta thấy Mô hình vẫn học tốt nhưng đến đoạn sau 400 epoch loss Generator gần như duy trì một giá trị ổn định và không

thay đổi theo thời gian. Điều này xảy ra khi Generator học cách tạo ra một loại ảnh hoặc một số lượng rất nhỏ các ảnh mà Discriminator dễ dàng phân biệt, thay vì học cách tạo ra nhiều dạng ảnh khác nhau.

Điều này có thể do model chưa được xây dựng thực sự tốt, hàm loss chưa được xây dựng phù hợp với model ngoài ra còn có thể do tập dữ liệu chưa đủ tốt cần bổ sung thêm dữ liệu.

4.3 Đánh giá mô hình

4.3.1 Ưu điểm

– Có thể thấy mô hình đã có thể sinh ra hình ảnh các Pokemon với tốc độ khá nhanh

4.3.2 Nhược điểm

– Mặc dù mô hình đã có thể sinh ra được các hình ảnh Pokemon nhưng mà chất lượng hình ảnh còn chưa cao. – Mô hình chỉ sinh ra một số kiểu dáng nhất định của Pokemon.

4.4 Hướng phát triển tiếp

– Thu thập thêm data để mô hình có thể học tốt hơn.

– Xây dựng lại các hàm loss để có thể tối ưu tốt hơn

– Sử dụng một số phương pháp trong xử lý ảnh để có thể cho kết quả được nhìn rõ hơn.

– Xây dựng website cho phép người dùng gửi ảnh Pokemon phác thảo và mô hình sẽ sinh màu cho bức ảnh đó.

Tài liệu tham khảo

[1] Khoa học dữ liệu - Khanh's blog
https://phamdinhkhanh.github.io/2020/07/13/GAN.html

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros
Image-to-Image Translation with Conditional Adversarial Networks
https://arxiv.org/abs/1611.07004, 2016

[3] Pokemon GAN
https://www.kaggle.com/code/sanjanabhute03/pokemon-gan-project

[4] Pokemon DCGAN
https://github.com/kvpratama/gan/tree/master/pokemon$_dcgan$

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
Generative Adversarial Networks
https://arxiv.org/abs/1406.2661, 2014