PROJECT
Machine Learning

1. Thông tin thành viên

STT	Thành viên 1	Thành viên 2
MSSV	20424008	20424013
Họ và tên	Dương Mạnh Cường	Phạm Nguyễn Mỹ Diễm
Email	20424008@student.hcmus.edu.vn	20424013@student.hcmus.edu.vn

2. Phân công công việc

STT	Phân công công việc	Diễm	Cường	Hoàn thành (%)
1	Tìm hiểu chọn đề tài (https://paperswithcode.com).	X	X	100
2	Đọc báo và xây dựng đồ án.	X	X	100
3	Chọn Dataset phù hợp.	X		100
4	Tìm hiểu về GAN (Cấu trúc mạng)	X	X	100
5	Tìm hiểu về DCGAN (Cấu trúc mạng: Generator và Discriminator).	X	X	100
6	Tìm hiểu về Transposed convolution.		X	100
7	Tìm hiểu về Loss function (Generator và Discriminator).	X		100
8	Một số Tips để build model và train DCGAN.		X	100
9	Thực nghiệm và đánh giá	X	X	100
10	Tìm hiểu về một số vấn đề gặp phải và đưa ra cách giải quyết.	X	X	100
11	Tìm hiểu về LSGAN.	X		100
12	Đưa ra một số kết luận và hướng phát triển cho bài toán.	X	X	100
13	Implement.	X	X	100

3. Giới thiệu bài toán: DCGAN

- Task(T): **Generate** ra những bức ảnh giống với dataset.
- Performance measurement (P): Binary classification: **Accuracy.**
- Experience (E): **Unsupervised Representation Learning.**

4. Xây dựng mô hình học máy

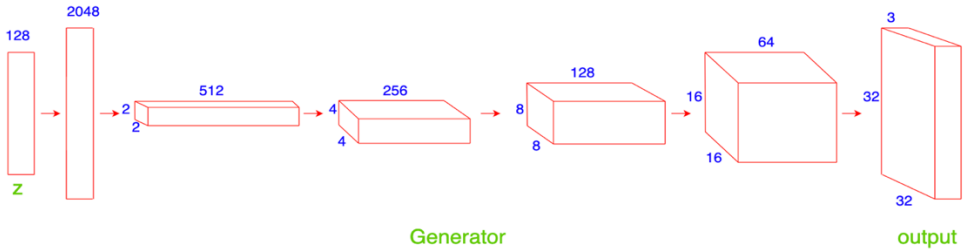
- Data: Bộ dữ liệu CIFAR-10: Bao gồm 60.000 hình ảnh được chia thành 10 classes, với mỗi class chứa 6000 hình ảnh có kích thước 32 x 32. 10 class khác nhau của tập dữ liệu này là: **Airplane, Car, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck**.

- Dataset CIFAR-10 g`ồm 50.00 hình ảnh cho traning set và 10.000 hình ảnh cho test set.
- **Chia lại dữ liệu cho phù hợp với đ`ồán**: Training set: 50.000 hình ảnh được chia thành 10 classes. Test set: 50.000 hình ảnh được chia thành 10 classes.

- Hypothesis set:

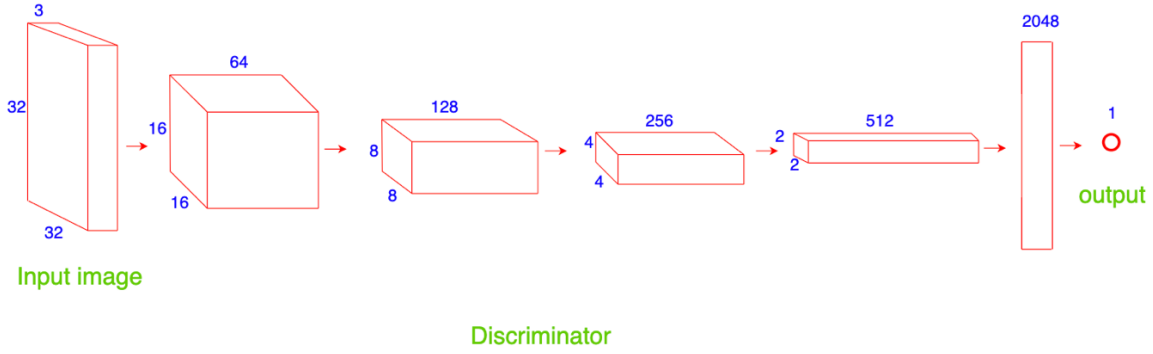
+ Generator: Mạng Generator nhằm mục đích sinh ảnh fake, input là noise vector kích thước 128 và output là ảnh fake cùng kích thước ảnh thật (32 * 32 * 3).

- Các layer trong mạng:
  - Dense (fully-connected) layer: 128*1 → 2048*1
  - Flatten chuyển từ vector v`êdạng tensor 3d, 2048*1 → 2*2*512
  - Transposed convolution stride=2, kernel=256, 2*2*512 → 4*4*256
  - Transposed convolution stride=2, kernel=128, 4*4*256 → 8*8*128
  - Transposed convolution stride=2, kernel=64, 8*8*128 → 16*16*64
  - Transposed convolution stride=2, kernel=3, 16*16*64 → 32*32*3
- Đ`ầu tiên thì input noise (128) được dùng full-connected layer chuyển thành 2048 ( = 2*2*512). Số 2048 được chọn để reshape v`êdạng tensor 3d (2*2*512). Sau đó transposed convolution với stride = 2 được dùng để tăng kích thước tensor lên d`ần 4*4 → 8*8 → 16*16 → 32*32. Cho tới khi kích thước tensor 32*32 (bằng đúng width, height của ảnh trong CIFAR-10 dataset) thì ta dùng 3 kernel để ra đúng shape của ảnh.



+ **Discriminator** : Mạng Discriminator nhằm mục đích phân biệt ảnh thật từ dataset và ảnh fake do Generator sinh ra, input là ảnh kích thước (32 * 32 *3), output là ảnh thật hay fake (binary classification).

- Mô hình discriminator đối xứng lại với mô hình generator.
- Ảnh input được đi qua convolution với stride = 2 để giảm kích thước ảnh từ 32*32 → 16*16 → 8*8 → 4*4 → 2*2.
- Khi giảm kích thước thì depth tăng d`ần. Cuối cùng thì tensor shape 2*2*512 được reshape v`êvector 2048 và dùng 1 lớp fully connected chuyển từ 2048d v`ê1d.

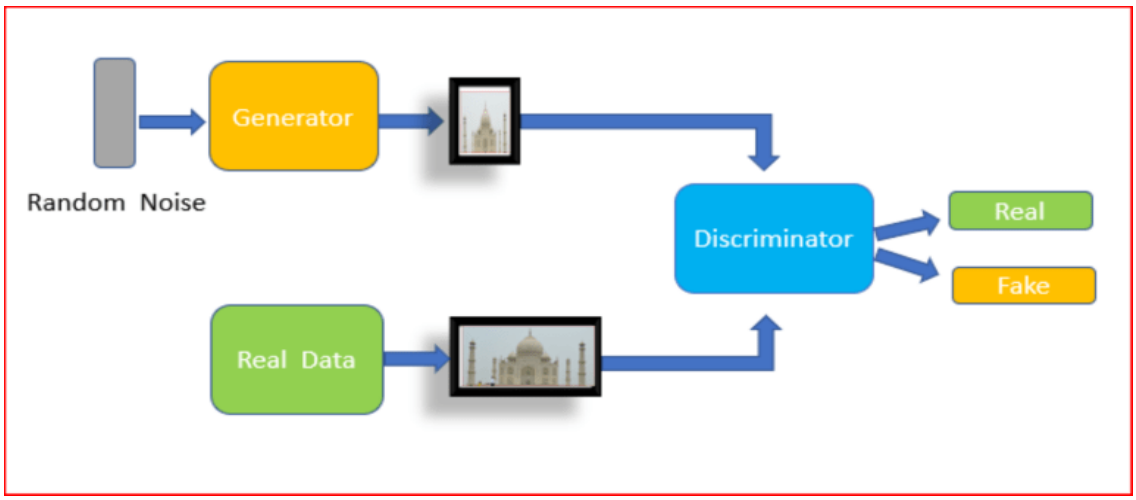


- Algorithm: Thuật toán học dựa trên **Gradient**.
  + Classfication: Vì bài toán phân 2 lớp (real, fake) nên dùng **sigmoid**.
  + Cost function: Vì bài toán binary classification giống với logistic regression nên dùng **Binary Cross Entropy.**
  + Activation function: LeakyReLU, Tanh, Sigmoid function.
    - 1 output layer bao gồm 1 neuron và sử dụng **sigmoid function** để dự đoám sample image 2 lớp (real, fake).
    - ReLU tốc độ hội tụ nhanh hơn hẳn. Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh. Tuy nhiên ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. → Dùng **Leaky ReLU**.
    - Generator model: Sau mỗi lần convolutional như vậy, ta áp dụng LeakyReLU với slope bằng 0.2 ngay sau đó, con số này được cho là mang lại một DCGANs model ổn định.
    - Chúng ta biết rằng, mỗi pixel trong ảnh là các giá trị thuộc ngưỡng [0, 255], và theo điều 1 trong những tips mà Soumith gợi ý để build một DCGANs model ổn định là ta nên chuẩn hóa nó về ngưỡng [-1, 1]. Như vậy Generator model sẽ phát sinh ra các generated sample với các pixel nằm trong phạm vi [-1, 1] và nó sẽ sử **Tanh activation function** kèm theo. Ở output layer, ta dùng thêm một Conv2DTranspose nữa với **Tanh activation fucntion**, kernel size là (3, 3). Tanh activation function giúp đảm bảo từng giá trị pixel luôn nằm trong phạm vi [-1, 1].
  + Optimizer: **Adam**
    - Adam = momentum + adagrad/RMSPop nên nó vừa có khả năng vượt qua các điểm yên ngựa để tìm điểm tối ưu toàn cục.
    - Tốc độ hội tụ nhanh hơn vì những bước đầu nó đi nhanh để đến gần điểm tối ưu. Khi gần đến điểm tối ưu, nó đi chậm lại để tránh vượt qua điểm tối ưu.
    - Optimizer dùng Adam với learning rate là 0.0002 và momentum là 0.5, đây là đề nghị của tác giả cho DCGANs model.

5. Lost function

- Kí hiệu z là noise đầu vào của generator, x là dữ liệu thật từ bộ dataset.
- Kí hiệu mạng Generator là G, mạng Discriminator là D. G(z) là ảnh được sinh ta từ Generator. D(x) là giá trị dự đoán của Discriminator xem ảnh x là thật hay không, D(G(z)) là giá trị dự đoán xem ảnh sinh ra từ Generator là ảnh thật hay không.

- Vì ta có 2 mạng Generator và Discriminator với mục tiêu khác nhau, nên c ần thiết kế 2 loss function cho mỗi mạng.
- **Discriminator** thì cố gắng phân biệt đâu là ảnh thật và đâu là ảnh giả. Vì là bài toán binary classification nên loss function dùng giống với binary cross-entropy loss của bài sigmoid.
- Giá trị output của model qua hàm sigmoid nên sẽ trong (0, 1) nên Discriminator sẽ được train để input ảnh ở dataset thì output g ần 1, còn input là ảnh sinh ra từ Generator thì output g ần 0, hay $D(x) \rightarrow 1$ còn $D(G(z)) \rightarrow 0$.



Nguồn: https://medium.com/datadriveninvestor/generative-adversarial-network-gan-using-keras-ce1c05cfdfd3

- Hay nói cách khác là loss function muốn maximize $D(x)$ và minimize $D(G(z))$. Ta có minimize $D(G(z))$ tương đương với maximize $(1 - D(G(z)))$. Do đó loss function của **Discriminator** được viết lại thành.
- E là kì vọng, hiểu đơn giản là lấy trung bình của tất cả dữ liệu, hay maximize $D(x)$ với x là dữ liệu trong traning set.
- Generator sẽ học để đánh lừa Discriminator rằng số nó sinh ra là số thật, hay $D(G(z)) \rightarrow 1$. Hay loss function muốn maximize $D(G(z))$, tương đương với minimize $(1 - D(G(z)))$
- Do đó ta có thể viết gộp lại loss của mô hình GAN:

$$\max_D V(D) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_z(z)}[log\,(1 - D(G(z)))]$$

recognize real images better *recognize generated images better*

Nguồn: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

- E là kì vọng, hiểu đơn giản là lấy trung bình của tất cả dữ liệu, hay maximize $D(x)$ với x là dữ liệu trong traning set.
- Generator sẽ học để đánh lừa Discriminator rằng số nó sinh ra là số thật, hay $D(G(z)) \rightarrow 1$. Hay loss function muốn maximize $D(G(z))$, tương đương với minimize $(1 - D(G(z)))$

$$\min_D V(G) = E_{z \sim p_z(z)}[log\,(1 - D(G(z)))]$$

Optimize G that can fool the discriminator the most.

Nguồn: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

- Do đó ta có thể viết gộp lại loss của mô hình GAN:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_z(z)}[log\,(1 - D(G(z)))]$$

Nguồn: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

- Từ hàm loss của GAN có thể thấy là việc train Generator và Discriminator đối nghịch nhau, trong khi D cố gắng maximize loss thì G cố gắng minimize loss.
- Quá trình train GAN kết thúc khi model GAN đạt đến trạng thái cân bằng của 2 models, gọi là Nash equilibrium.

6. Kết quả thực nghiệm

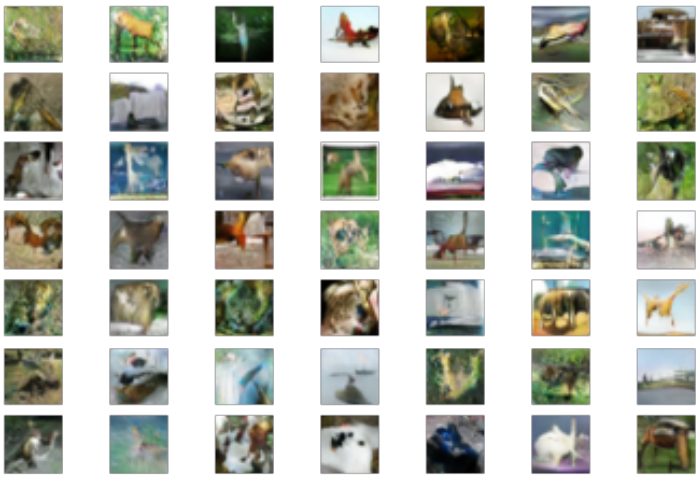
- Ảnh CIFAR-10 được scale về (-1, 1) để cùng scale với ảnh sinh ra bởi generator khi dùng tanh activation.

Ảnh sinh ra bởi generator sau 10 epochs

- Tuy nhiên sau 100 epoch thì mạng đã học được thuộc tính của ảnh trong dữ liệu CIFAR-10 và có thể sinh ra được hình con ếch, nai, …

Ảnh sinh ra bởi generator sau 100 epochs

- Và sau 200 epoch thì mạng đã học được thuộc tính của ảnh trong dữ liệu CIFAR-10 và có thể sinh ra được hình xe tải, con nai, con chim..



Ảnh sinh ra bởi generator sau 300 epochs

7. Vấn đề của bài toán và giải quyết như thế nào?

- Vấn đề 1: Conditional GAN
  - Khi ta train DCGAN xong rồi dùng generator để sinh ảnh mới giống trong dataset mình không kiểm soát được là ảnh sinh ra giống category nào trong dataset.
  - Ví dụ như dùng DCGAN để sinh các ảnh trong bộ CIFAR-10, thì khi train xong và dùng generator sinh ảnh thì mình không biết được ảnh sinh ra đó thuộc class nào (Airplane, Car, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck)
  - Bài toán muốn kiểm soát được generator sinh ra ảnh theo 1 category nhất định. Ví dụ có thể

chỉ định generator sinh ra ảnh con nai chẳng hạn. Mô hình đấy gọi là Conditional GAN (cGAN).
    - Đây có thể được xem như một bước đột phát của GAN vì trên thực tế có rất nhiều những bức ảnh mà ta sẽ phải định hướng kết quả về hình dạng, format. cGAN cũng tạo ra những đột phá mới về chất lượng hình ảnh và sự ổn định trong quá trình huấn luyện.
- Vấn đề 2: LSGAN
    - Cấu trúc mạng DCGAN với thành phần là Generator và Discriminator, GAN loss function.
    - Tuy nhiên GAN loss function không tốt, nó bị vanishing gradient khi train generator → Hàm LSGAN (Least Squares Generative Adversarial Networks) giải quyết vấn đề trên.
- Vấn đề 3: GAN evaluation
    - Mình dùng DCGAN train model xong rồi dùng Generator để sinh ảnh. Tuy nhiên mình chưa có cách nào để đánh giá xem chất lượng ảnh sinh ra đã tốt chưa, chủ yếu chỉ nhìn vào mắt thường để đánh giá.
    - Cách đấy không khách quan cũng như không áp dụng một cách hệ thống trên dữ liệu lớn, nên cần phương pháp để đánh giá chất lượng ảnh GAN - DCGAN sinh ra chất lượng tốt hay không.
    - Một số cách để đánh giá chất lượng ảnh GAN (DCGAN) sinh ra: Inception Score (IS) và Fréchet Inception Distance (FID).

8. Kết luận và hướng phát triển

- DCGAN (deep convolutional GAN) là mô hình GAN áp dụng trong các tác vụ của xử lý ảnh.
- Nhược điểm của DCGAN là chúng ta không thể kiểm soát được bức ảnh được sinh ra thuộc class nào mà nó được tạo ra hoàn toàn ngẫu nhiên.
- cGAN sẽ giúp chúng ta sinh ra được ảnh thuộc một class cụ thể theo ý muốn dựa trên một thông tin được bổ sung vào mô hình là nhãn y. y được coi như điều kiện để sinh ảnh nên mô hình mới có tên gọi là conditional GAN.
- GAN loss function không tốt, bị vanishing gradient khi train generator thì LSGAN đã giải quyết được vấn đề.
- Một số cách để đánh giá chất lượng ảnh GAN (DCGAN) sinh ra: Inception Score (IS) và Fréchet Inception Distance (FID).
- LSGAN hoạt động ổn định hơn GAN thông thường trong quá trình học. Qua đó, chúng em sẽ vận dụng model LSGAN trên những bộ dữ liệu khác như ImageNet,…

9. Tài liệu tham khảo

- [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks] (*https://paperswithcode.com/paper/unsupervised-representation-learning-with-1*)

- [Least Squares Generative Adversarial Networks] (*https://paperswithcode.com/paper/least-squares-generative-adversarial-networks*)

- [GANs in Action: Deep learning with Generative Adversarial Networks 1st Edition] (*https://www.amazon.com/GANs-Action-learning-Generative-Adversarial/dp/1617295566*)

- [How to Train a GAN? Tips and tricks to make GANs work] (https://github.com/soumith/ganhacks)

- [Book: Deep Learning cơ bản] (*https://drive.google.com/file/d/1lNjzISABdoc7SRq8tg-xkCRRZRABPCKi/view*)

- [Introduction - Google Developers] (*https://developers.google.com/machine-learning/gan*)

- [Generative Adversarial Networks Wiki] (*https://en.wikipedia.org/wiki/Generative_adversarial_network*)

- [Bài 45 - Conditional GAN (cGAN)] (*https://phamdinhkhanh.github.io/2020/08/09/ConditionalGAN.html*)

- [f-divergence wiki] (*https://en.wikipedia.org/wiki/F-divergence*)