Báo cáo đồ án cuối môn

Môn học: Thị giác máy tính

Ứng dụng mạng sinh đối kháng GAN trong vấn đề mất cân bằng dữ liệu

Các thành viên của nhóm
Đinh Tuấn Anh – 19489471
Ngô Đức Bảo – 19487461
Lưu Đức Anh – 19478881
Trần Nam Bá – 19506751

Tóm tắt

Mất cân bằng dữ liệu là một trong những hiện tượng phổ biến của bài toán phân loại, nếu hiện tượng mất cân bằng nghiêm trọng xảy ra thường dẫn tới dự báo kém chính xác trên nhóm thiểu số bởi đa phần kết quả dự báo ra thường thiên về nhóm đa số và rất kém trên nhóm thiểu số. Trong đề tài này, chúng em sẽ tiến hành thử nghiệm xử lý vấn đề mất cân bằng dữ liệu bằng phương pháp sử dụng mạng GAN để tạo mẫu cho nhóm thiểu số trong tập dữ liệu. Huấn luyện mô hình với bộ dữ liệu ban đầu và bộ dữ liệu mới với các mẫu được sinh ra từ mạng GAN. So sánh kết quả đạt được để thấy được hiệu quả của phương pháp này.

1 Giới thiệu

Tầm quan trọng của các vấn đề mất cân bằng trong tập dữ liệu có thể làm sai lệch đáng kể hiệu suất của bộ phân loại, đưa ra sai lệch dự đoán cho nhiều lớp. Không chỉ mất cân bằng lớp mà còn tồn tại nhiều vấn đề mất cân bằng khác nhau trong khoa học dữ liệu. Bên cạnh đó, kích thước của các khiếm khuyết, quy mô tổn thất và sự mất cân bằng phân phối của discriminator cũng được đề cập. Để ngăn chặn các vấn đề mất cân bằng như vậy, nhóm đã tiến hành loại bỏ một phần nhỏ dữ liệu mẫu và chuyển vấn đề thành một framework để phát hiện sự bất thường. Xem xét sự thiếu hụt và đa dạng của dữ liệu bất thường thường được mô hình hóa như một vấn đề phân loại một lớp, với tập dữ liệu huấn luyện chỉ chứa dữ liệu bình thường.

Ý tưởng đằng sau điều này là autoencoder có thể tái tạo lại dữ liệu bình thường với các lỗi nhỏ, trong khi lỗi tái cấu trúc (reconstruction errors) của dữ liệu bất thường thường lớn hơn nhiều. Autoencoder được áp dụng bởi hầu hết các phương pháp dựa trên tái cấu trúc giả định rằng các mẫu bình thường và bất thường có thể dẫn đến các cách embedding khác nhau đáng kể và do đó có thể tận dụng sự khác biệt trong các lỗi tái tạo tương ứng để phân biệt hai loại mẫu.

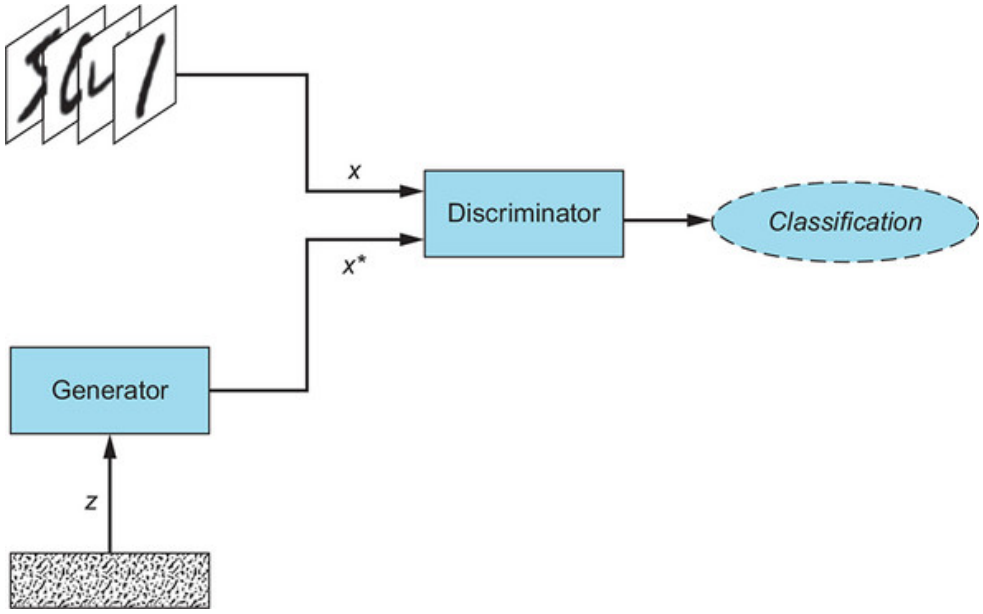
Adversarial training được giới thiệu bằng cách thêm một discriminator sau autoencoder để đánh giá xem hình ảnh gốc hay hình ảnh được tái tạo. the latent vector thể hiện sự phân phối thực của dữ liệu và ánh xạ lại tới latent vector bằng cách tối ưu hóa một pre-trained trên latent vector dựa vào GAN. Tại đây, nhóm sẽ dựa trên mô hình conditional GAN, DCGAN, ACGAN để cải thiện hiệu suất cho các bộ dữ liệu điểm chuẩn như Fashion-MNIST, MNIST.

2 Các kiến thức liên quan

2.1 Mạng sinh đối kháng GAN

GAN được giới thiệu lần đầu tiên bởi Ian Goodfellow và đồng nghiệp của mình qua một báo cáo khoa học vào năm 2014. Vào thời kì đầu, mọi người tập trung vào chức năng phát sinh dữ liệu của GAN. Sau đó, tiềm năng của GAN được khai thác nhiều hơn và nhờ đó mà GAN có thể sử dụng trong nhiều lĩnh vực liên quan đến dữ liệu như image generation, video generation hay voice generation [1].

GAN sử dụng hai model khác nhau, generator và discriminator. Hai network này sẽ cạnh tranh lẫn nhau và sẽ tự phát triển trong quá trình cạnh tranh này. Trong quá trình này, generator sẽ tạo ra dữ liệu mới với mục tiêu là vượt qua được sự kiểm tra về tính xác thực của discriminator. Nhờ vào sự cạnh tranh và phát triển này, chúng ta có thể sử dụng generator để tạo ra dữ liệu với tính xác thực cao [1].



Sơ đồ mạng GAN

Vì ta có 2 mạng Generator và Discriminator với mục tiêu khác nhau, nên cần thiết kế 2 loss function cho mỗi mạng. Discriminator thì cố gắng phân biệt đâu là ảnh thật và đâu là ảnh giả còn Generator sẽ cố gắng sinh ảnh giả. 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 có thể được định nghĩa như sau [2]:

$$\max_D V(D) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\text{recognize real images better}} + \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{recognize generated images better}}$$

Discriminator loss function

Generator sẽ học để đánh lừa Discriminator rằng số nó sinh ra là số thật, hay D(G(z)) -> 1. Hay loss function muốn maximize D(G(z)), tương đương với minimize (1 – D(G(z)))

$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Optimize G that can fool the discriminator the most.

Geneator loss function

ta có thể viết gộp lại loss của mô hình GAN

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].$$

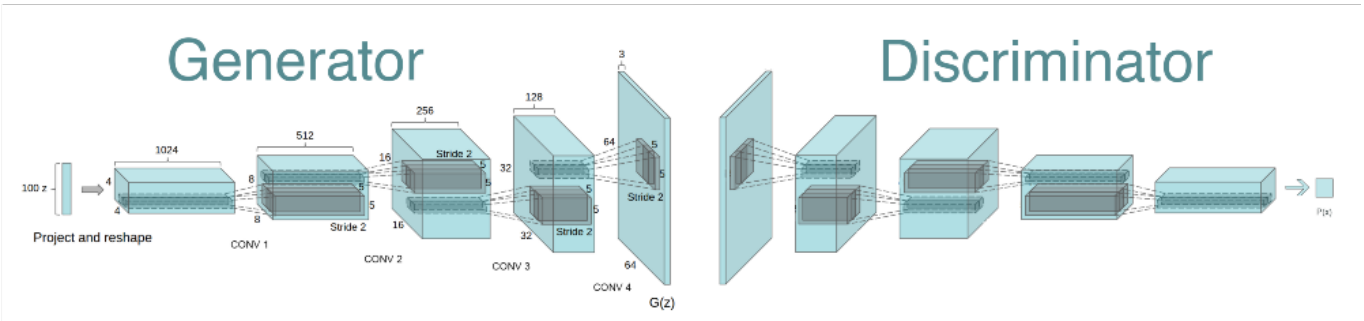
GAN loss function

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.

2.2 Các biến thể mạng GAN sử dụng trong bài

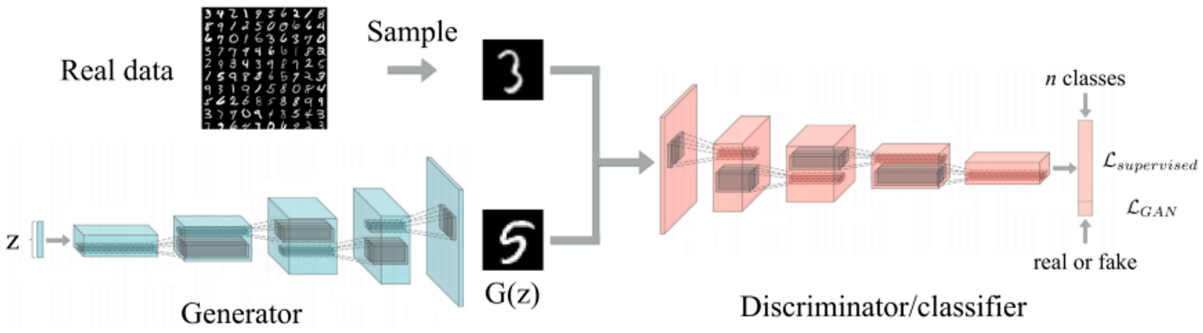
2.2.1 DCGAN – Deep Convolutional GAN

DCGAN, hay Deep Convolutional GAN, là một kiến trúc mạng đối nghịch tổng quát. Cụ thể, nó sử dụng một vài nguyên tắc: Thay thế bất kỳ lớp pooling nào bằng các strided convolutions (Discriminator) và ractional-strided convolutions (Generator). Sử dụng batchnorm trong cả Generator và Discriminator [3].



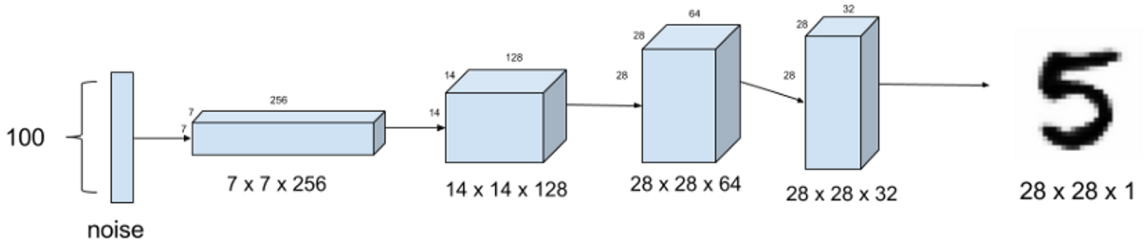
Deep Convolutional GAN

DCGAN là một kiến trúc GAN sử dụng Convolutions. Nó chủ yếu bao gồm các lớp tích chập mà không có các lớp max-pooling hoặc fully-connected.Thay vào đó, nó sử dụng stride convolutional và transposed convolution để lấy mẫu. DCGAN được tạo từ hai mô hình riêng biệt, bộ tạo(generator) và bộ phân biệt (discriminator). Công việc của generator là tạo ra các hình ảnh 'giả' trông giống như hình ảnh huấn luyện. Công việc của discriminator là xem xét một hình ảnh và đưa ra kết quả xem đó là hình ảnh huấn luyện thực hay hình ảnh giả mạo từ generator.



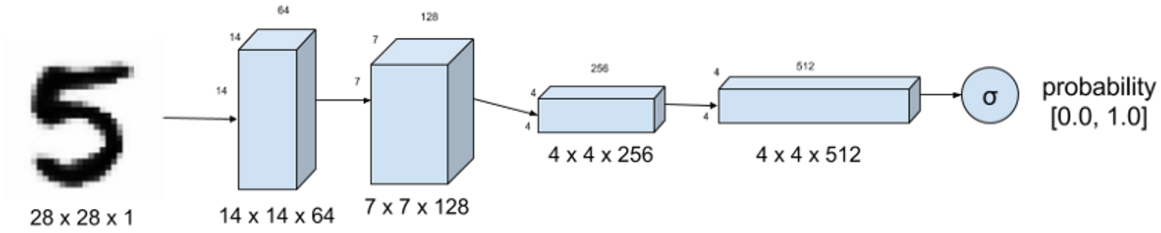
Mô hình kiến trúc của DCGAN trên tập dữ liệu MNIST

Generator, generator tổng hợp các hình ảnh giả, hình ảnh giả được tạo ra từ nhiễu bằng cách sử dụng nghịch đảo của tích chập, được gọi là transposed convolution.Ở giữa các lớp, batch norm giúp ổn định quá trình học. Hàm kích hoạt sau mỗi lớp là một ReLU. Output của sigmoid ở lớp cuối cùng tạo ra hình ảnh giả. Generator sử dụng ConvTranpose2d để mở rộng kích thước hình ảnh từ 7×7 lên 28×28. ConvTranpose2d giúp mở rộng kích thước hình ảnh mà còn tạo ra hình ảnh có chất lượng tốt hơn.

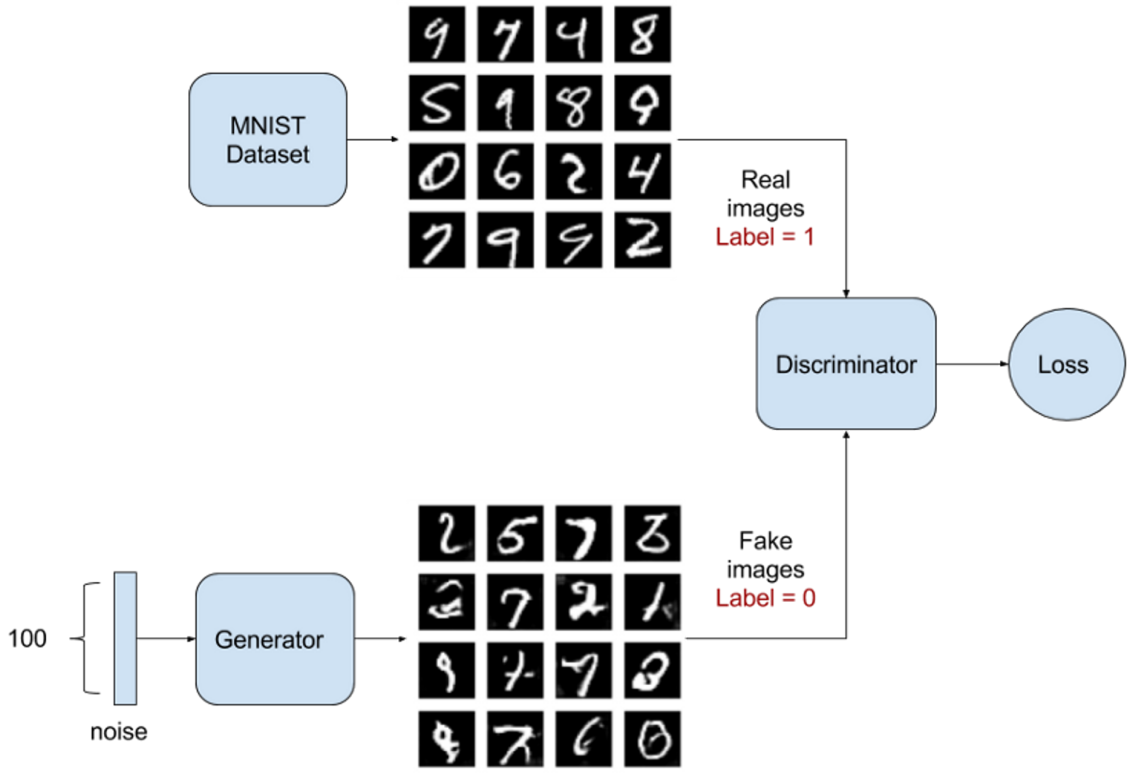


Mô hình Generator tổng hợp ảnh MNIST giả từ nhiễu.

Discriminator, discriminator về cơ bản là một Convolutional Neural Network (CNN). Input là một hình ảnh (28 pixel x 28 pixel x 1 channel). Output là sigmoid (một giá trị vô hướng) để xác suất mức độ thật của hình ảnh. Khác biệt so với CNN điển hình là không có max-pooling giữa các lớp. Thay vào đó, một strided-convolution được sử dụng để downsampling. Chức năng kích hoạt trong mỗi lớp CNN là một a leaky ReLU . Việc dropout từ 0,4 đến 0,7 giữa các lớp sẽ ngăn cản quá trình overfitting.



Mô hình Discriminator cho biết hình ảnh đầu vào của một chữ số thực từ bộ dữ liệu MNIST.

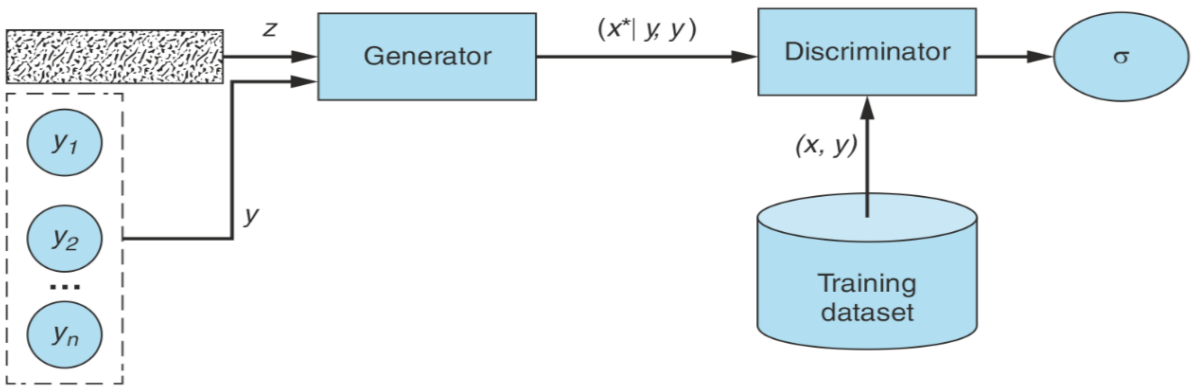


Mô hình discriminator được huấn luyện để phân biệt hình ảnh viết tay thật và giả..

2.2.2 cGAN – Conditional GAN

DCGAN 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. Điều này làm chúng ta không biết trước được ảnh cần tạo thuộc về class nào và đây cũng là hạn chế của DCGAN [4]. 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 . được coi như điều kiện để sinh ảnh nên mô hình mới có tên gọi là conditional GAN [4].



Conditional GAN

Theo như kiến trúc của conditional GAN:

- Đầu vào Generator sẽ nhận vector z ngẫu nhiên và label y. Đầu ra sẽ là các samples giả sao cho giống với label y của nó với G(z, y) = x*|y. Mục tiêu tạo dữ liệu, mẫu giả
- Đầu vào Discriminator sẽ nhận sample thật từ việc training dataset (x, y) và samples giả được tạo từ Generator với (x*|y, y). Đầu ra sẽ là một xác suất đơn (single probability) chỉ ra samples đầu vào có phải là thật hay không hay samples có phù hợp với mỗi label hay không. Mục tiêu phân biệt giữa samples giả cho mỗi label từ Generator và sample thật từ training dataset

2.2.3 ACGAN – Auxiliary Classifier GAN

AC-GAN là một phần mở rộng của cGAN thay đổi Discriminator để dự đoán label của một hình ảnh nhất định thay vì nhận nó làm đầu vào. Nó có tác dụng ổn định quá trình đào tạo và cho phép tạo ra các hình ảnh lớn có chất lượng cao trong khi học một biểu diễn trong latent space độc lập với label [5]. Mô hình Discriminator nhận đầu vào là một ảnh và dự đoán xác xuất độ thật của ảnh và xác suất của ảnh thuộc về mỗi lớp. Mô hình sẽ có 2 lớp đầu ra:

- Lớp đầu tiên gồm một node với hàm kích hoạt sigmoid để dự đoán tính chân thực của ảnh.
- Lớp thứ 2 gồm nhiều node, ứng với mỗi nhãn lớp của dữ liệu. Lớp này sẻ dụng hàm kích hoạt softmax để dự đoán class của ảnh.

Mô hình được đạo tạo với hai hàm mất mát:

- Binary cross entropy cho lớp đầu ra đầu tiên:

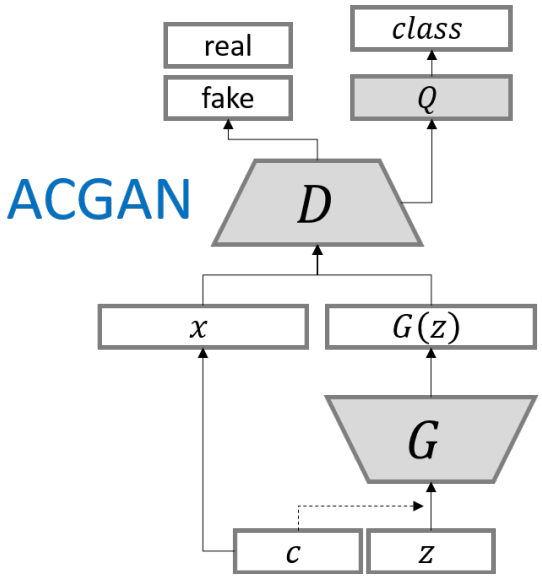
$$L_S = Ex \sim p\text{data}(x)[\log D(x)] + Ez \sim pz(z)[1 - \log D(G(z))]$$

- Categorical cross-entropy loss cho lớp đầu ra thứ hai

$$L_C = \sum_i^C y_i log(f(\hat{y})_i)$$

$$f(\hat{y})_i = \frac{e^{\hat{y}_i}}{\sum_j^C e^{\hat{y}_j}}$$

Nhiệm vụ của mô hình Discriminator là tối đa hóa $L_S + L_C$ và nhiệm vụ của mô hình Generator là tối thiểu hóa $L_S - L_C$



Auxiliary Classifier GAN

3 Phương pháp

Một số mục tiêu xác định ban đầu của nhóm sẽ tiến hành những công việc sau:

- Sử dụng hai bộ dữ liệu MNIST và FashionMNIST, xóa bớt dữ liệu để tạo ra class imbalanced cho bộ dữ liệu và sử dụng làm bộ train cho mạng GAN.
- Train mạng GAN (DCGAN, cGAN, ACGAN) với dữ liệu imbalanced, sau đó dùng mạng Generator sinh dữ liệu bị imbalanced tạo ra bộ dữ liệu thứ 2.
- Huấn luyện mô hình phân loại với 2 bộ dữ liệu imbalanced và balanced (dữ liệu balanced được tạo ra từ mạng GAN) và so sánh hiệu suất mô hình để đánh giá phương pháp này.

4 Thực nghiệm

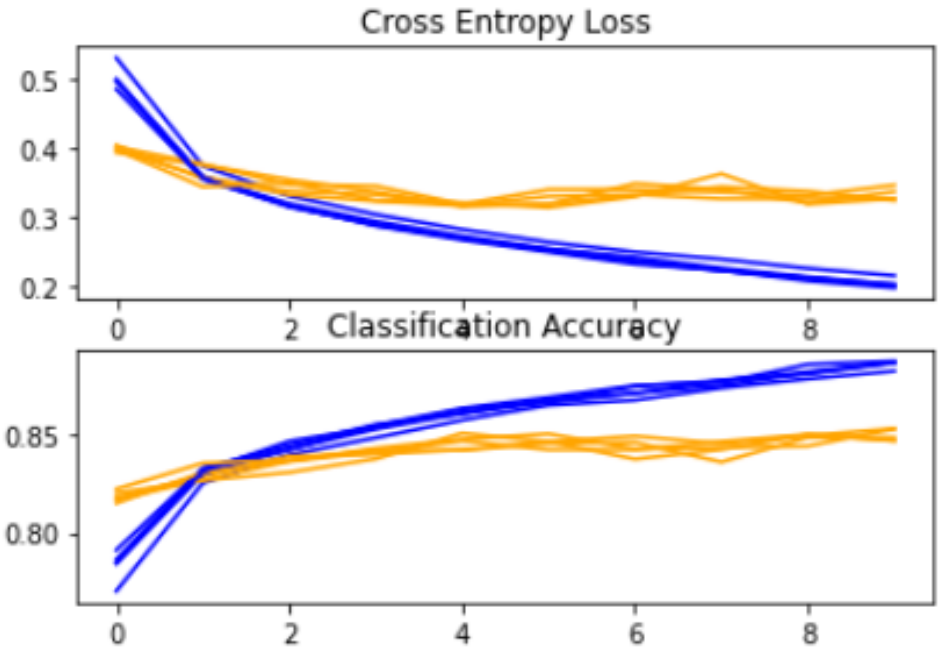
4.1 Dữ liệu

- MNIST với 10 class

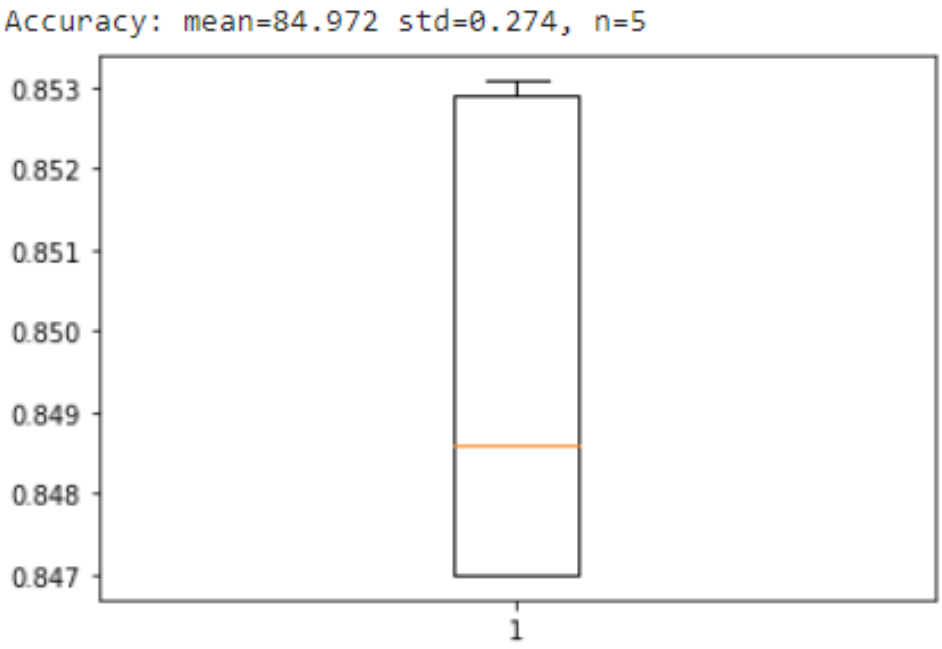
4.2 Kết quả

- DCGAN

Kết quả của mô hình phân loại với dữ liệu được tạo bởi DC-GAN



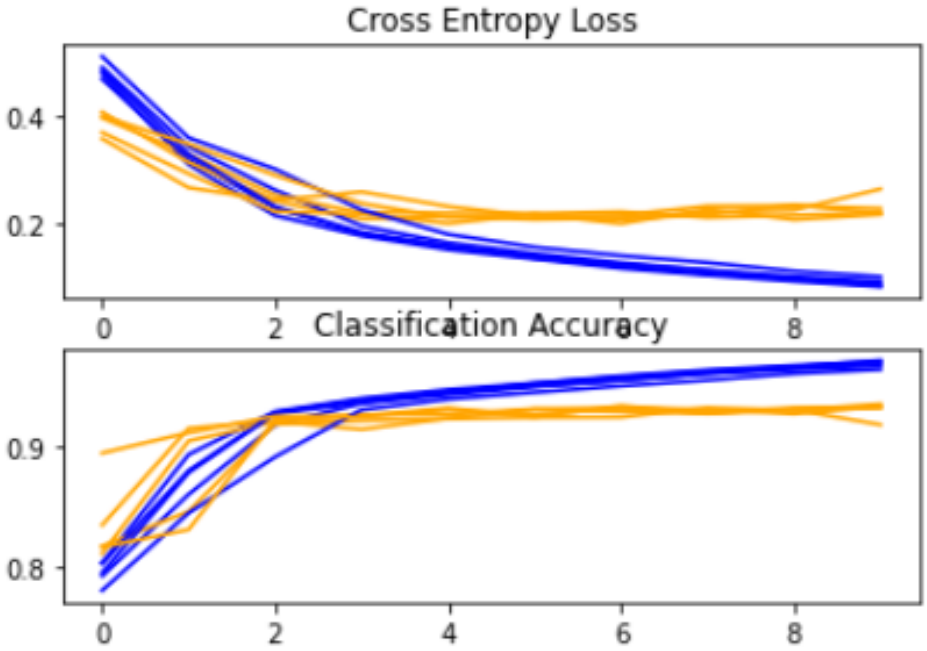
Ảnh được tạo bởi mô hình DC-GAN



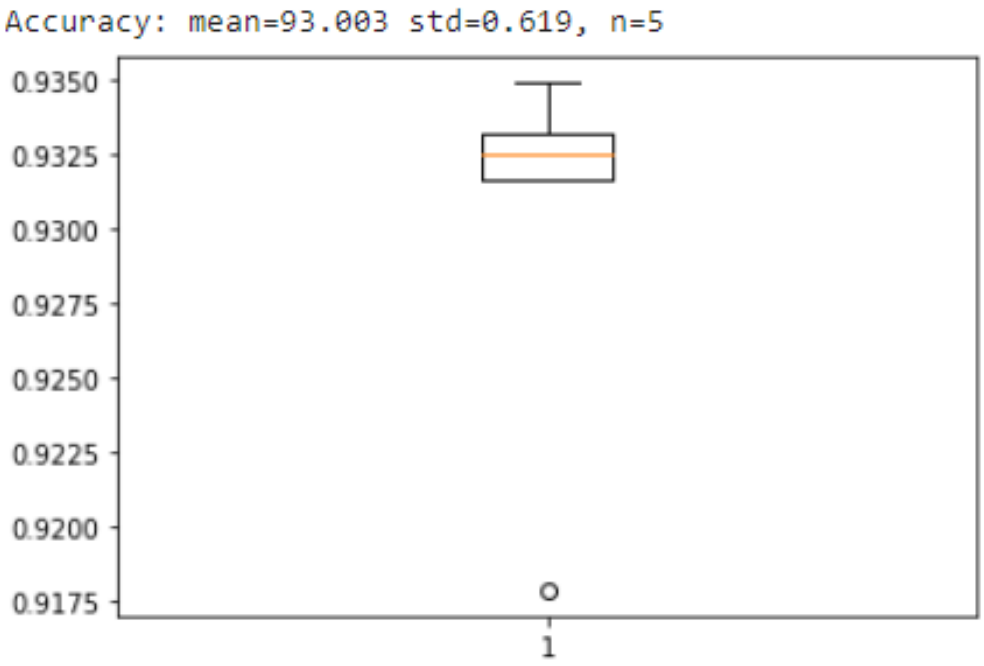
Kết quả của mô hình phân loại với dữ liệu được tạo bởi DC-GAN

- cGAN

Ảnh được tạo bởi mô hình C-GAN

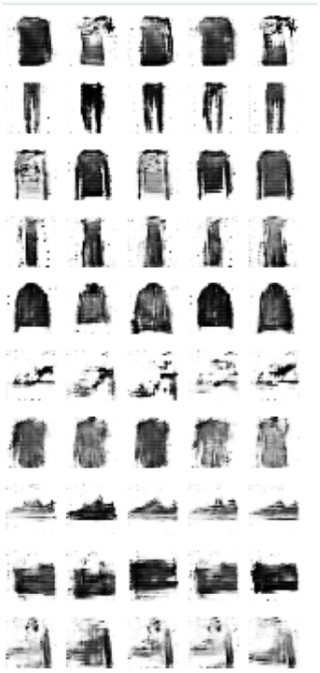


Kết quả của mô hình phân loại với dữ liệu được tạo bởi C-GAN

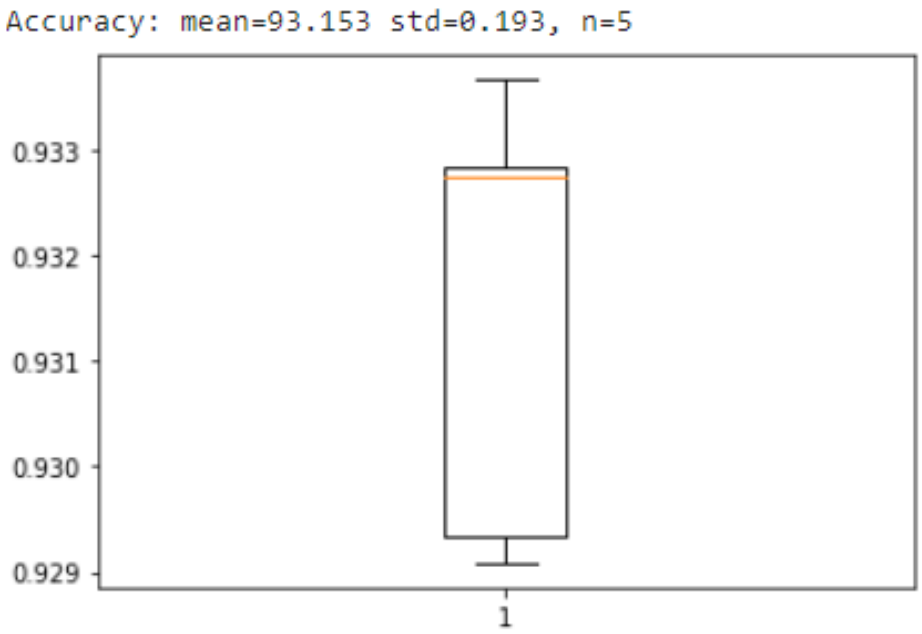


Kết quả của mô hình phân loại với dữ liệu được tạo bởi C-GAN

- ACGAN

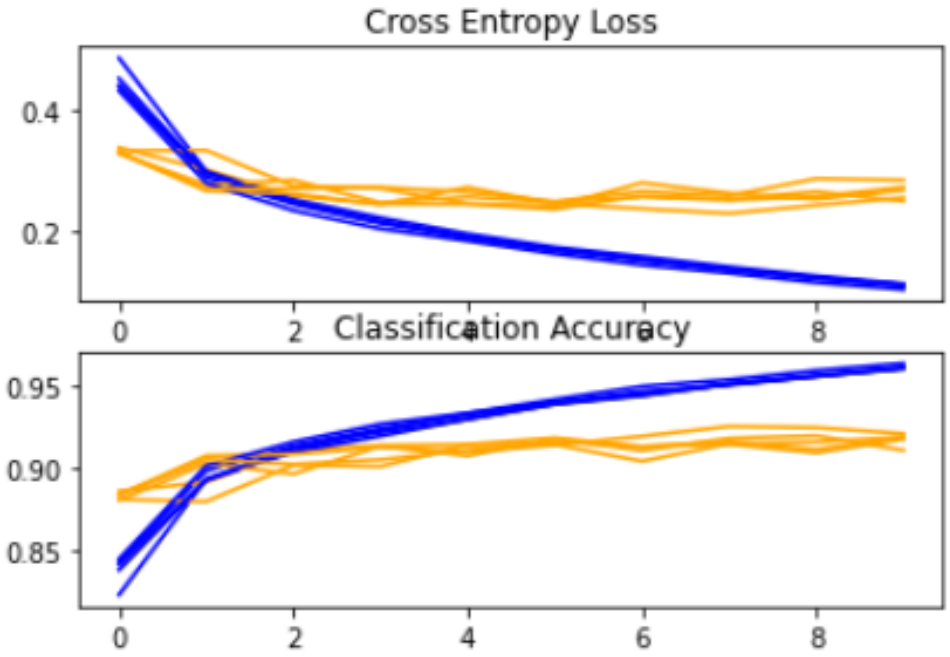


Ảnh được tạo bởi mô hình AC-GAN

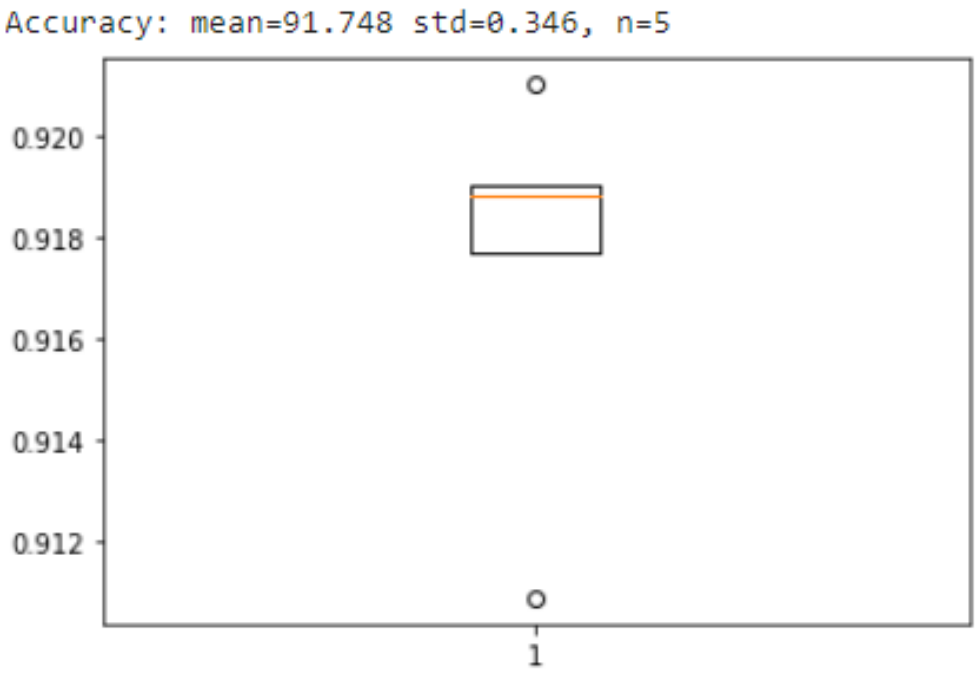


Kết quả của mô hình phân loại với dữ liệu được tạo bởi AC-GAN

- Dữ liệu bị thiếu



Kết quả của mô hình phân loại với dữ liệu bị thiếu



Kết quả của mô hình phân loại với dữ liệu bị thiếu

5 Kết luận

5.1 Nhận xét

Kết quả của mô hình phân loại được huấn luyện với dữ liệu cân bằng được tạo với các mô hình C-GAN và AC-GAN có kết quả tốt hơn khi huẩn luyện với dữ liệu bị thiếu ban đầu.

5.2 Hướng phát triển

- Có thể sử dụng với các mô hình GAN phức tạp hơn cùng với dữ liệu đa dạng hơn để đưa bài toán đến gần với thực tế.
- Nghiên cứu và áp dụng nhiều các mô hình GAN vào bài toán Semantic Segmentation sẽ chuyển từ ảnh thật sang các ảnh phân khúc.

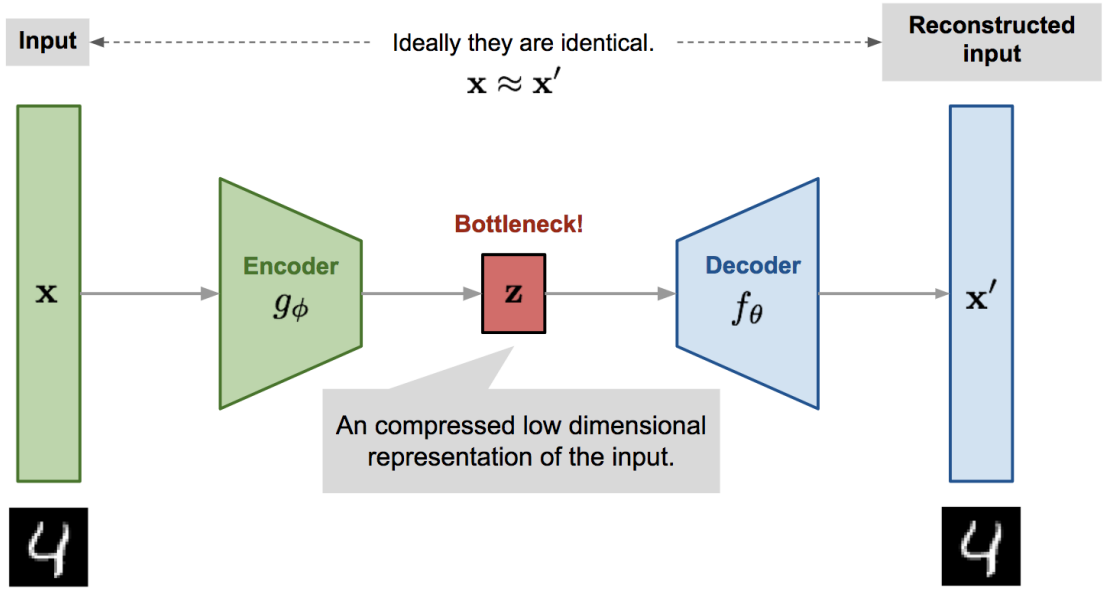
Các mô hình sinh ảnh trending hiện tại

Autoencoder

Autoencoder là một dạng đặc biệt của mạng nơ ron được huấn luyện để bắt chước dữ liệu đầu vào. Đầu vào loại mạng neural này không có nhãn, tức là mạng có khả năng học không giám sát (Unsupervised Learning). [6]

Autoencoder bao gồm 3 phần chính [6]:

- Decoder: Module giúp mạng giải nén các biểu diễn tri thức và tái cấu trúc lại dữ liệu từ dạng mã hóa của nó, mô hình học dựa trên việc so sánh đầu ra của Decoder với đầu vào ban đầu (Input của Encoder).
- Bottleneck: Module chứa các biểu diễn tri thức được nén (chính là output của Encoder), đây là phần quan trọng nhất của mạng bởi nó mang đặc trưng của đầu vào, có thể dùng để tái tạo ảnh, lấy đặc trưng của ảnh, ...
- Encoder: Module có nhiệm vụ nén dữ liệu đầu vòa thành một biểu diễn được mã hóa (coding), thường nhỏ hơn một vài bậc so với dữ liệu đầu vào.

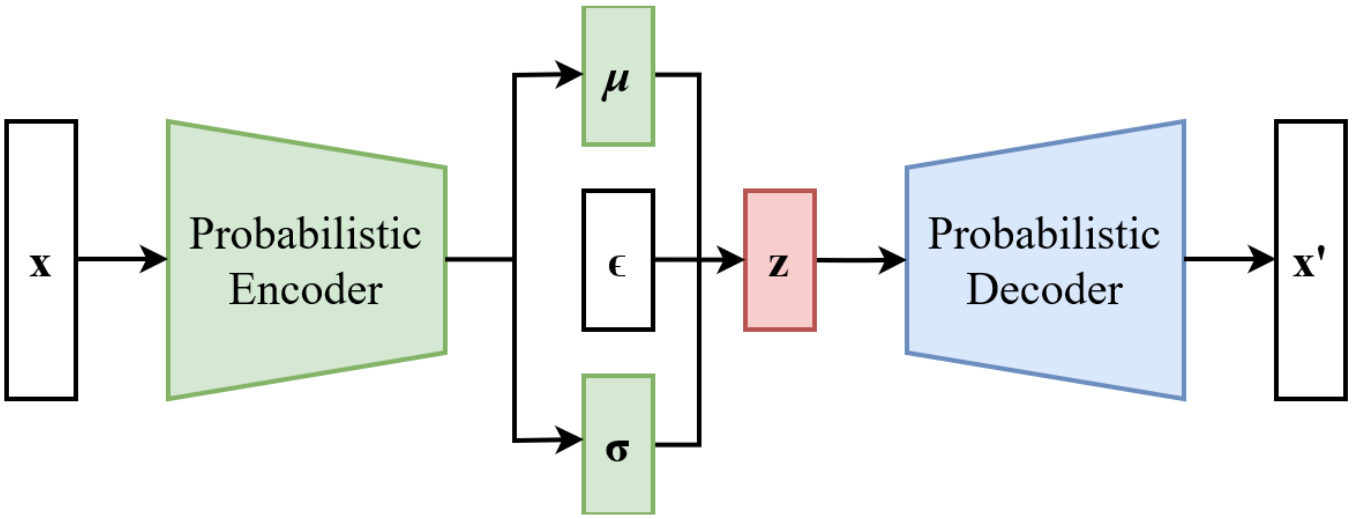


Sơ đồ cơ bản của Autoencoder

Hàm loss gồm reconstruction loss và regularization loss

VAE

VAE là một autoencoder với xác suất lấy dữ liệu đầu vào có chiều dữ liệu lớn và nén nó thành một biểu diễn nhỏ hơn. Khác với autoencoder đơn thuần sẽ map input vào latent space vector mà VAE ánh xạ dữ liệu đầu vào thành các tham số của phân bố xác suất, chẳng hạn như giá trị trung bình và phương sai của Gaussian. Cách tiếp cận này tạo ra một latent space vector có cấu trúc, liên tục, rất hữu ích cho việc tạo hình ảnh. [7]



Sơ đồ của Variational Autoencoder

Stable Diffusion

Stable Diffusion là một mô hình latent diffusion và cũng là mô hình học sâu giúp chuyển văn bản thành hình ảnh được phát hành vào năm 2022. Nó chủ yếu được sử dụng để tạo hình ảnh chi tiết dựa trên mô tả văn bản, mặc dù nó cũng có thể được áp dụng cho các tác vụ khác như inpainting, outpainting và generate img2img nhờ vào một guidance text prompt. [8]

Mô hình Stable Diffusion gồm 3 phần: [9]:

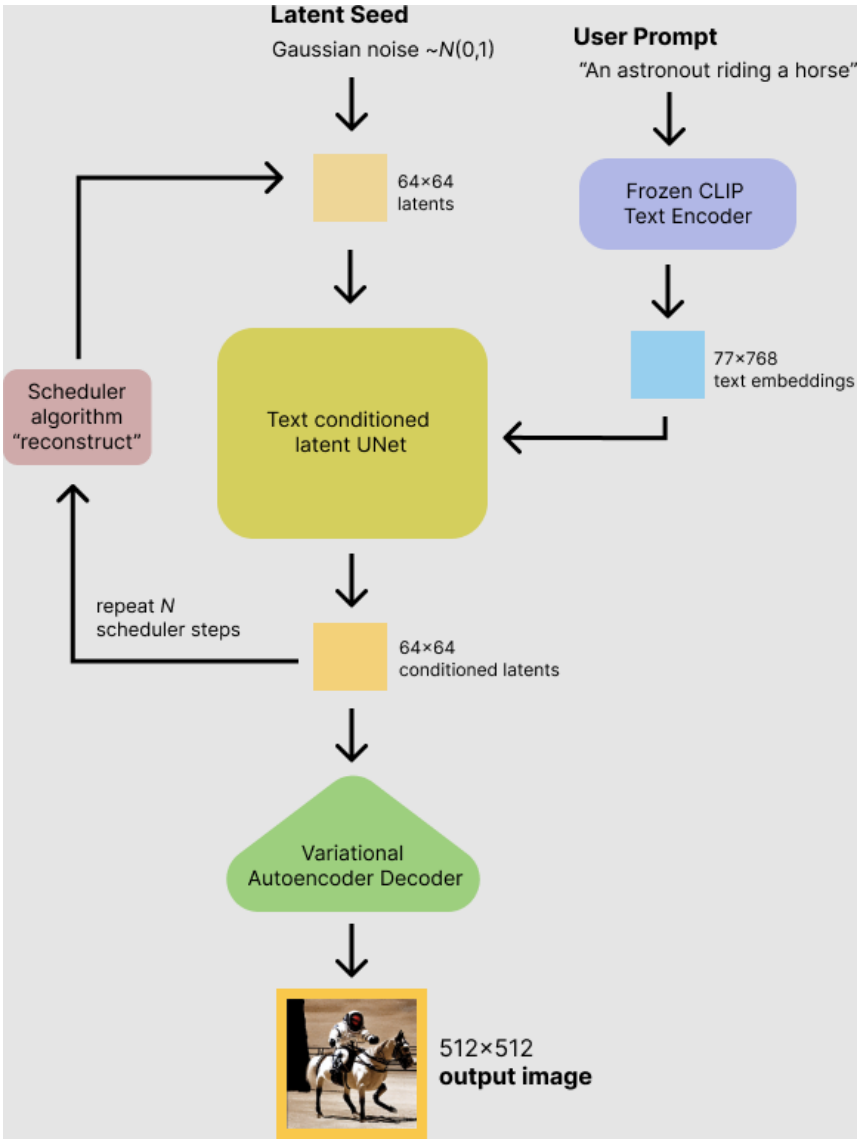
- VAE
  Mô hình VAE có hai phần, Encoder và Decoder. Encoder được sử dụng để chuyển đổi hình ảnh thành latent space, sẽ đóng vai trò là đầu vào cho mô hình U-Net. Ngược lại, Decoder biến latent space trở lại thành hình ảnh [9].
  Trong quá trình đào tạo latent diffsuion, Encoder được sử dụng để nhận các latent space (latents) của hình ảnh cho quá trình forward diffusion, áp dụng ngày càng nhiều noise ở mỗi bước. Trong quá trình suy luận, các denoised latents được tạo ra bởi quá

trình reverse diffusion được chuyển đổi trở lại thành hình ảnh bằng bộ giải mã VAE. Như chúng ta sẽ thấy trong quá trình suy luận, chúng ta chỉ cần Decoder của VAE [9].

- U-net
  U-Net có phần Encoder và phần Decoder, cả hai đều bao gồm các khối ResNet. Encoder nén hình ảnh thành ảnh có độ phân giải thấp hơn và Decoder biểu diễn hình ảnh có độ phân giải thấp hơn trở lại biểu diễn hình ảnh ban đầu có độ phân giải cao hơn được cho là ít noise hơn. Cụ thể hơn, đầu ra U-Net dự đoán dư lượng nhiễu có thể được sử dụng để tính toán biểu diễn denoised image được dự đoán [9].
  Để ngăn U-Net mất thông tin quan trọng trong khi downsampling, các kết nối cắt ngắn thường được thêm vào giữa các ResNet downsampling của Encoder với các ResNet upsampling của Decoder. Ngoài ra, U-Net khuếch tán ổn định có thể điều chỉnh đầu ra của nó khi nhúng văn bản thông qua các lớp cross-attention. Các lớp cross-attention được thêm vào cả phần Encoder và Decoder của U-Net, thường là giữa các khối ResNet [9].

- Text Encoder
  text-encoder chịu trách nhiệm chuyển đổi input prompt, ví dụ: "Một phi hành gia cưỡi ngựa" vào một embedding space mà U-Net có thể hiểu được. Nó thường là một Encoder dựa trên transformer đơn giản, ánh xạ một chuỗi mã thông báo đầu vào thành một chuỗi latent text-embeddings [9].
  Lấy cảm hứng từ Imagen, Stable Diffusion không đào tạo text-encoder trong quá trình đào tạo và chỉ sử dụng text-encoder đã được đào tạo của CLIP, CLIPTextModel [9].

Tại sao khuếch tán tiềm ẩn nhanh và hiệu quả? Vì latent diffusion hoạt động trên một không gian có chiều thấp, nó làm giảm đáng kể các yêu cầu về bộ nhớ và tính toán so với các mô hình pixel-space diffusion. Ví dụ: autoencoder được sử dụng trong Stable Diffusion có hệ số giảm là 8. Điều này có nghĩa là hình ảnh có hình dạng (3, 512, 512) trở thành (3, 64, 64) trong latent space, yêu cầu $8 \times 8 = 64$ lần bộ nhớ ít hơn.



Sơ đồ của Stable Difusion

References

[1] Q. Tran, "Generative adversarial network," Apr 2020. [Online]. Available: https://tnquangblog.wordpress.com/2020/03/19/generative-adversarial-network/

[2] nttuan8, "Bài 1: Giới thiệu về gan," Feb 2021. [Online]. Available: https://nttuan8.com/bai-1-gioi-thieu-ve-gan/

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[4] Khanh, "Khoa học dữ liệu." [Online]. Available: https://phamdinhkhanh.github.io/2020/08/09/ConditionalGAN.html

[5] J. Brownlee, "How to develop an auxiliary classifier gan (ac-gan) from scratch with keras," Jan 2021. [Online]. Available: https://machinelearningmastery.com/how-to-develop-an-auxiliary-classifier-gan-ac-gan-from-scratch-with-keras/

[6] "Autoencoder," Nov 2022. [Online]. Available: https://en.wikipedia.org/wiki/Autoencoder

[7] "Convolutional variational autoencoder nbsp;: nbsp; tensorflow core." [Online]. Available: https://www.tensorflow.org/tutorials/generative/cvae

[8] "Stable diffusion," Dec 2022. [Online]. Available: https://en.wikipedia.org/wiki/Stable_Diffusion

[9] "Stable diffusion with diffusers." [Online]. Available: https://huggingface.co/blog/stable_diffusion