TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

MÔN XỬ LÝ DỮ LIỆU LỚN

PROJECT 7

GOM CỤM DỮ LIỆU BẰNG PHƯƠNG PHÁP HỌC SÂU KHÔNG GIÁM SÁT – DEEP UNSUPERVISED METHOD

Người hướng dẫn: **THẦY BÙI THANH HÙNG**

Người thực hiện: **CAO NGUYỄN KỲ DUYÊN – 51900491**

NGUYỄN NGỌC THỦY VY – 51900579

Lớp : **19050402**

19050401

Khoá : **23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

MÔN XỬ LÝ DỮ LIỆU LỚN

PROJECT 7

GOM CỤM DỮ LIỆU BẰNG PHƯƠNG PHÁP HỌC SÂU KHÔNG GIÁM SÁT – DEEP UNSUPERVISED METHOD

Người hướng dẫn: **THẦY BÙI THANH HÙNG**

Người thực hiện: **CAO NGUYỄN KỲ DUYÊN – 51900491**

NGUYỄN NGỌC THỦY VY – 51900579

Lớp : **19050402**

19050401

Khoá : **23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn đến thầy Bùi Thanh Hùng đã giúp đỡ, hướng dẫn và dìu dắt em trong quá trình học tập và tìm hiểu về bộ môn "Xử lý dữ liệu lớn". Nhờ như vậy, em có thể thực hiện bài báo cáo này một cách tốt nhất và có thể đạt được một kết quả tốt nhất.

Em cũng xin chân thành cảm ơn đến quý thầy cô trong khoa Công nghệ thông tin đã truyền đạt những kiến thức quý báu giúp chúng em có thể hoàn thành tốt được bài báo cáo này. Khoa đã luôn sẵn sàng chia sẻ các kiến thức bổ ích cũng như chia sẻ các kinh nghiệm tham khảo tài liệu, giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu mà còn giúp ích cho việc học tập và rèn luyện trong quá trình thực hành tại trường Đại học Tôn Đức Thắng nói chung.

Và một lần nữa, em xin bày tỏ lòng biết ơn của mình đến với thầy Bùi Thanh Hùng và chúc thầy sẽ luôn thành công trên con đường dạy học của mình.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của Thầy Bùi Thanh Hùng . Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 26 tháng 4 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Cao Nguyễn Kỳ Duyên

Nguyễn Ngọc Thủy Vy

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Phân cụm dữ liệu là bài toán gom nhóm các đối tượng dữ liệu vào thánh từng cụm (cluster) sao cho các đối tượng trong cùng một cụm có sự tương đồng theo một tiêu chí nào đó.

Tuy nhiên việc phân cụm vẫn còn gặp rất nhiều khó khăn đối với dữ liệu phức tạp do đó đồ án này sẽ tập trung vào cách cải thiện kết quả phân cụm bằng các phương pháp học sâu.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

AE	Autoencoder
VAE	Variational Autoencoder
GAN	Generative Adversarial Networks
BiGAN	Bidirectional Generative Adversarial Network

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH VÀ BẢNG

GOM CỤM DỮ LIỆU BẰNG PHƯƠNG PHÁP HỌC SÂU KHÔNG GIÁM SÁT –DEEP UNSUPERVISED METHOD

1.1 Giới thiệu về bài toán

Dữ liệu là một tập hợp các dữ kiện, chữ số, từ vựng hoặc hình ảnh để mô tả về một sự vật hoặc đối tượng nào đó trong thế giới thực. Đa số dữ liệu rất đa dạng và không có một cấu trúc nhất định, do đó một trong những cách để khai phá dữ liệu là sử dụng phương pháp gom cụm dữ liệu.

Gom cụm dữ liệu hay gom cụm dữ liệu là một phương pháp nhóm các dữ liệu có các đặc điểm tương tự nhau thành các cụm riêng biệt. Gom cụm dữ liệu là môt loại mô hình học máy không giám sát (unsupervised learning), tùy vào mục đích khác nhau mà gom cụm có thể dùng để khai phá thông tin hữu ích, phá hiện giá trị ngoại lai, gán dãn dữ liệu tự động hoặc là giảm kích thước của dữ liệu.

Tuy nhiên việc gom cụm lại rất phụ thuộc vào dữ liệu đầu vào và các cách phân tích dữ liệu khác nhau. Cụ thể là với mỗi loại dữ liệu khác nhau có thể sẽ phải áp dụng mô hình gom cụm khác nhau và cách phân tích dữ liệu đầu vào khác nhau cũng sẽ cho ra kết quả gom cụm khác nhau.

Việc lựa chọn phương pháp và phân tích dữ liệu đầu vào rất quan trọng với bài toán gom cụm. Mặc dù hiện nay mô hình học máy gom cụm dữ liệu đã cho ra kết quả gom cụm khá tốt trên một số bộ dữ liệu đơn giản tuy nhiên đối với một số bộ dữ liệu phức tạp hơn thì vẫn còn là một thách thức lớn. Một trong những cách để cải thiện việc phân tích dữ liệu đầu vào chính là sử dụng các phương pháp học sâu để ánh xạ các đặc trưng có trong dữ liệu từ đó giúp gom cụm hiệu quả hơn thay vì chỉ đơn thuần sử dụng các phương pháp vector hóa dữ liệu thông thường.

1.2 Phân tích yêu cầu của bài toán

1.2.1 Yêu cầu của bài toán

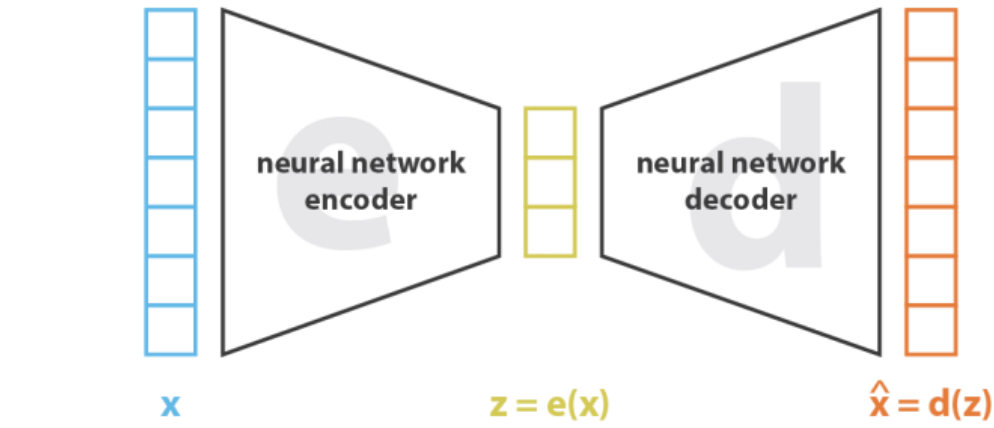
Để xây dựng mô hình gom cụm dữ liệu bằng phương pháp học sâu chúng ta sẽ sử dụng một mô hình học sâu có khả năng học tập cách biểu diễn dữ liệu sao cho có thể tập trung vào các đặc trưng quan trọng nhất của dữ liệu. Sau khi đã trích xuất được các đặc trưng này chúng ta sẽ đưa vào mô hình học máy không giám sát để gom cụm dữ liệu và so sánh kết quả với mô hình gom cụm không sử dụng phương pháp học sâu để đánh giá kết quả.

1.2.2 Các phương pháp giải quyết bài toán

Nhiệm vụ của mô hình học sâu là phải trích xuất được các đặc trưng trong bộ dữ liệu và từ đó đưa các đặc trưng này vào mô hình học không giám sát để gom cụm. Do đó chúng ta sẽ lựa chọn những phương pháp học sâu như sau.

1.2.2.1 Phương pháp học sâu Autoencoder:

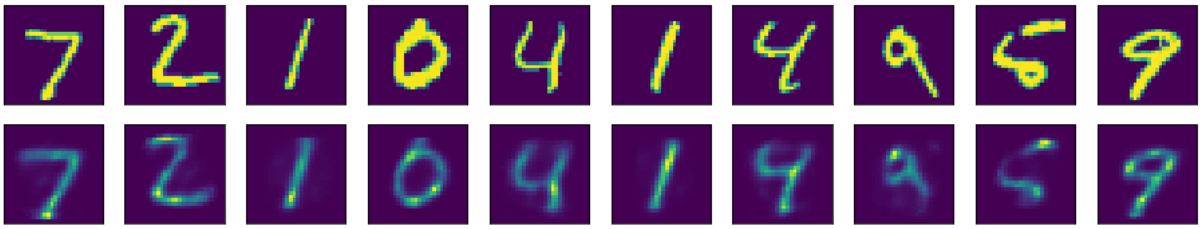
Bộ tự mã hóa – Autoencoder (AE) là một hình mạng nơ ron nhân tạo sử dụng để học dữ liệu không được dán nhãn (unsupervised learning). Autoencoder có 2 phần chính một bộ mã hóa encoder ở đầu vào, và một bộ giải mã ở đầu ra được ngăn cách nhau bằng một nút thắt cổ chai ở giữa – bottleneck hoặc có tên gọi khác là latent vector. Ý tưởng của thuật toán này là sử dụng bộ mã hóa encoder qua mỗi lớp encoder dữ liệu sẽ càng bị ép phải nén lại để trích xuất các đặc trưng của ảnh và tới bottleneck thì sẽ bắt đầu giải mã ngược lại để cố gắng tạo ra những thông tin mới có liên quan chặt chẽ đến thông tin ban đầu.



$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

Hình 1 : Mô hình mạng Autoencode với bên trái là encoder, ở giữa là bottleneck, phần bên phải là decoder và loss function bên dưới

Mặc dù Autoencoder có vẻ giống như là sao chép lại dữ liệu nhưng thật ra Autoencoder chỉ cố gắng tái tạo lại nội dung dữ liệu ban đầu bằng cách tạo ra thông tin bằng những giá trị mới. Bằng cách sử dụng việc ép phải trích xuất các đặc trưng quan trọng trong bộ dữ liệu đầu vào cho nên thuật toán này rất lí tưởng cho việc biểu diễn ánh xạ dữ liệu trong bài toán gom cụm.



Hình 2 : Một ví dụ về kết quả tái tại lại hình ảnh bằng mô hình Autoencoder với hàng bên trên chính là ảnh đầu vào và hàng bên dưới là kết quả đầu ra của mô hình

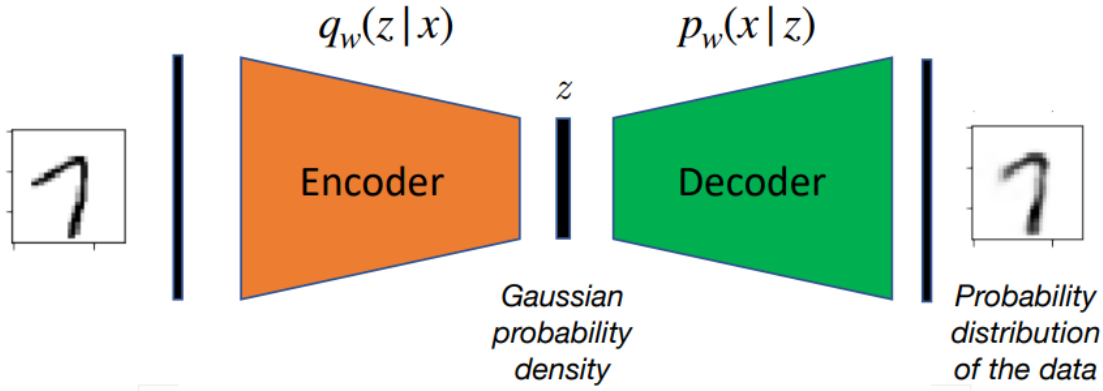
Biến thể của thuật toán Autoencoder:

Variational Autoencoder (VAE) khác với Autoencoder thông thường thay vì mã hóa dữ liệu đầu vào như một vector dữ liệu ở latent space thì mô hình này sẽ mã hóa nó như là một phân bố xác suất.

$$L^{[i]} = -\mathbb{E}_{z\sim q_w(z|x^{[i]})}\left[\log p_w\left(x^{[i]}|z\right)\right] + \mathbf{KL}\left(q_w\left(z|x^{[i]}\right)||p(z)\right)$$

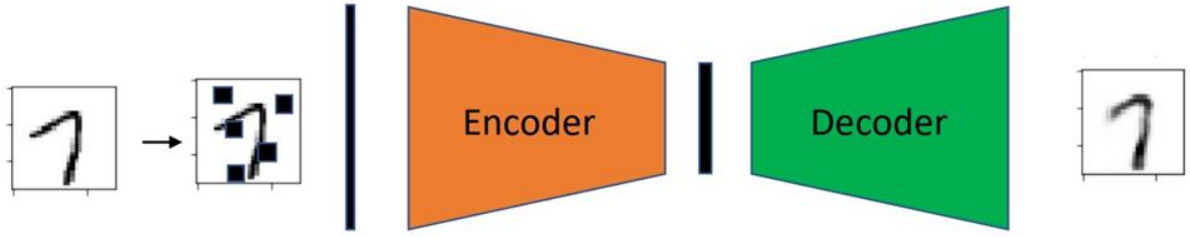
Expected neg. log likelihood term; wrt to encoder distribution

Kullback-Leibler divergence term where $p(z) = \mathcal{N}\left(\mu = 0, \sigma^2 = 1\right)$



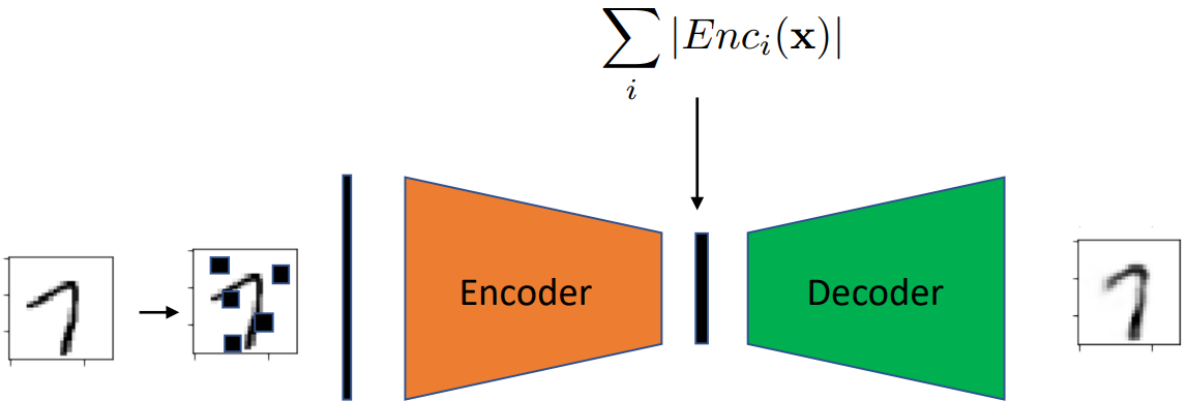
Hình 3 : Mô hình tổng quát của Variational Autoencoder

- Denoising autoencoder (DAE) thêm noise vào dữ liệu đầu vào như ví dụ dưới đây để bắt mô hình phải học thêm cách khử noise dữ liệu.



Hình 4: Mô hình tổng quát của Denoising Autoencoder

• Sparse autoencoder (SAE) thêm hình phạt L1 để chỉ cho phép một số node được phép hoạt động để tạo ra bottleneck mà không cần phải giảm số lượng node để tạo ra bottelneck.

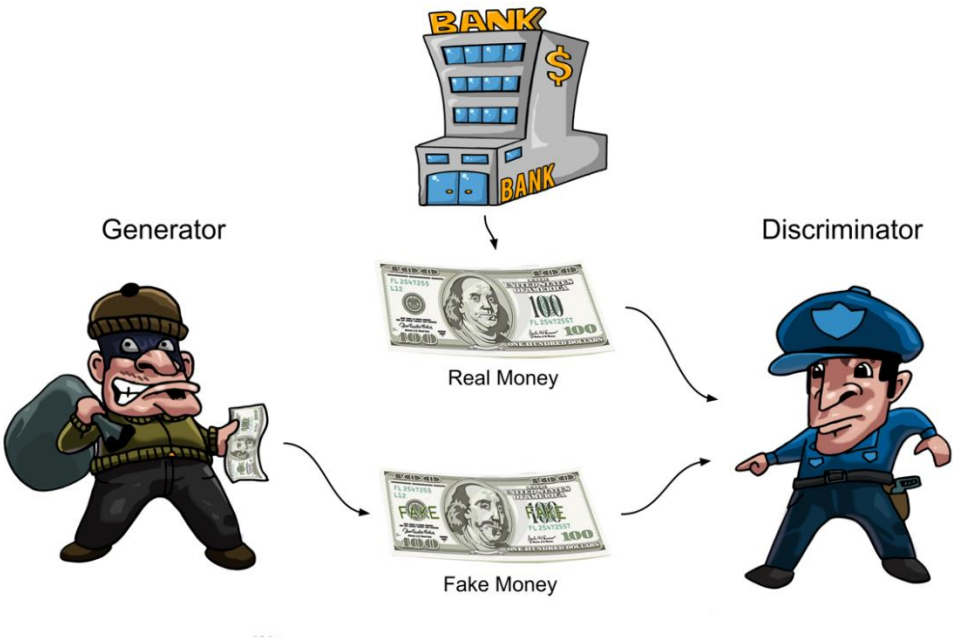


$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2 + \sum_i |Enc_i(\mathbf{x})|$$

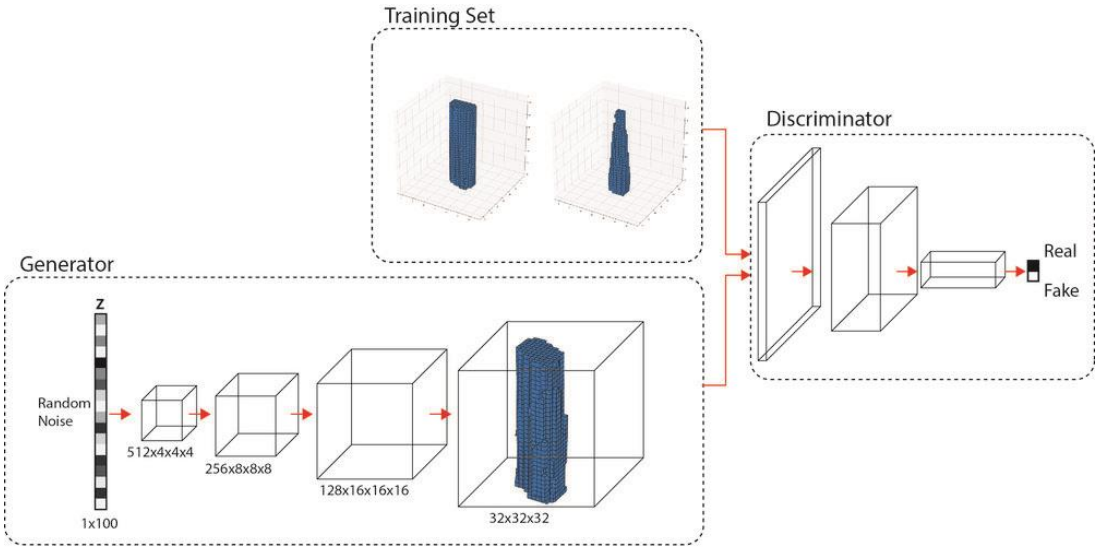
Hình 5: Mô hình tổng quát của Sparse Autoencoder

1.2.2.2 Phương pháp học sâu Generative Adversarial Network:

Mạng sinh đối lập - Generative Adversarial Network (GAN) cũng là một loại mô hình học không có giám sát (unsupervised learning) cũng giống như Autoencoder, GAN cũng tập trung vào việc tạo ra dữ liệu nhưng thay vì nén dữ liệu lại như Autoencoder nó lại tập trung tạo ảnh mới và nhận diện ảnh đó có phải là ảnh thật hay là ảnh được tạo ra từ máy. Cấu trúc của GAN cũng đúng theo như tên gọi, Generative có nghĩa là sinh hoặc tạo ra dữ liệu, Adversarial nghĩa là đối nghịch bởi vì GAN được cấu thành từ 2 mạng gọi là Generator và Discriminator luôn luôn đối nghịch nhau bởi vì trong khi Generator cố gắng tạo ra các dữ liệu giống như thật còn Discriminator lại cố gắng phân biệt đâu là dữ liệu được sinh ra từ Generator và đâu là dữ liệu thật.



Hình 6: Một ví dụ về cách Generator và Discriminator hoạt động, Generator giống như một kẻ xấu chuyên làm tiền giả để đánh lừa cảnh sát, Discriminator thì đóng vai là một cảnh sát chuyên phân biệt đâu là tiền giả được tạo từ kẻ xấu kia và đâu là tiền thật)



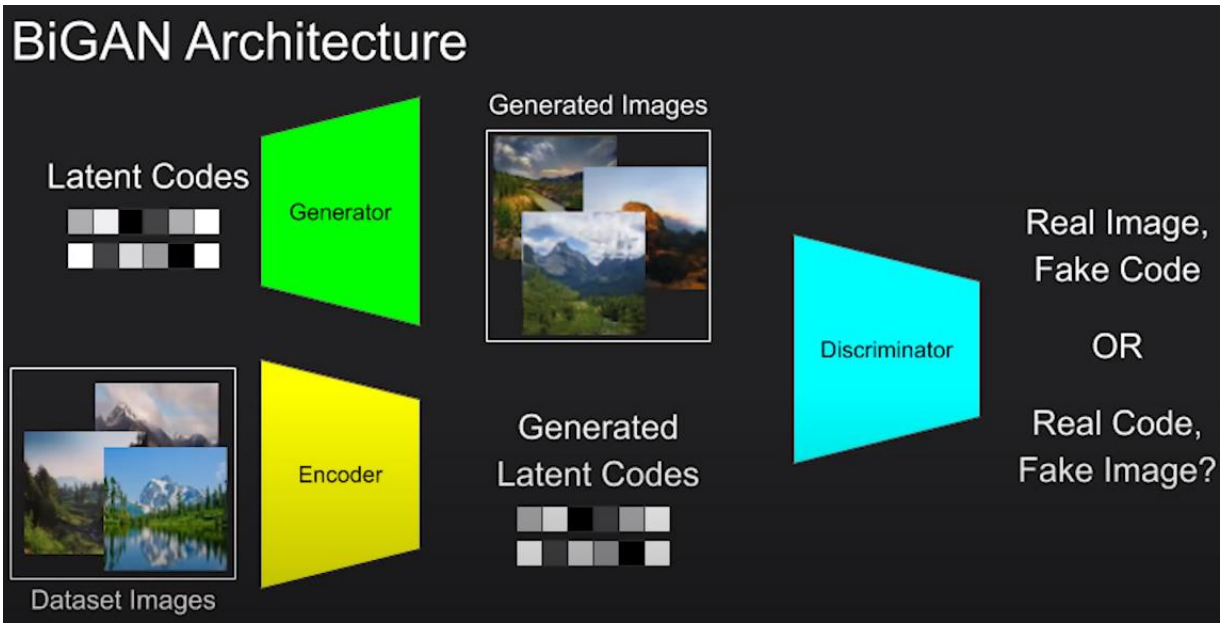
Hình 7: Mô hình GAN với Generator sử dụng một Random Noise để tạo dữ liệu mới và Discriminator nhận đầu vào là gồm cả dữ liệu thật và giả trộn lẫn với nhau để phân biệt

GAN dựa trên cách hoạt động của mình cho nên GAN là một mô hình tạo dữ liệu rất mạnh mẽ có thể ứng dụng rất tốt cho nhu cầu tạo mới và tái tạo dữ liệu ví dụ như là GAN có thể được sử dụng để phát hiện hình ảnh tăng nhãn áp giúp chẩn đoán sớm, điều cần thiết để tránh mất thị lực một phần hoặc toàn bộ, GAN có thể được sử dụng để làm lão hóa các bức ảnh chụp khuôn mặt để cho biết diện mạo của một cá nhân có thể thay đổi như thế nào theo tuổi tác, GAN cũng có thể được sử dụng để tô vẽ các đối tượng địa lý còn thiếu trong bản đồ và ứng dụng tạo ra khuôn mặt nam nữ…

Hình 8: Ứng dụng nổi tiếng của GAN đó là tạo ra khuôn mặt người, phía trên là những hình ảnh người không có thật mà chỉ do máy tính sử dụng GAN tạo ra

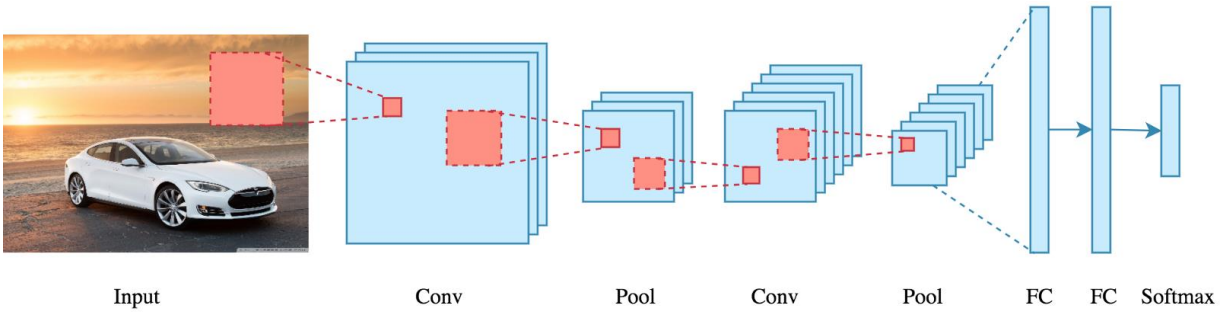
Biến thể của thuật toán GAN:

Bidirectional Generative Adversarial Network (BiGAN) – là một loại mô hình GAN nhưng có sử dụng thêm một bộ mã hóa Encoder chuyển dữ liệu thành một latent codes mục đích là yêu cầu Discriminator phải học thêm cách phân biệt được đâu là latent codes thật của ảnh và đâu là latent codes của Generator sử dụng để tạo ra ảnh.



Hình 9 : Cấu trúc một mạng BiGAN với thêm một bộ Encoder để Discriminator học được đâu là bộ latent codes đúng của anh

DCGAN (Deep Convolution GAN) – mạng này vẫn hoạt động như GAN thông thường nhưng áp dụng Convolutional Neural Network CNN để tăng khả năng xử lí dữ liệu ảnh so với mạng Neural Network truyền thống.

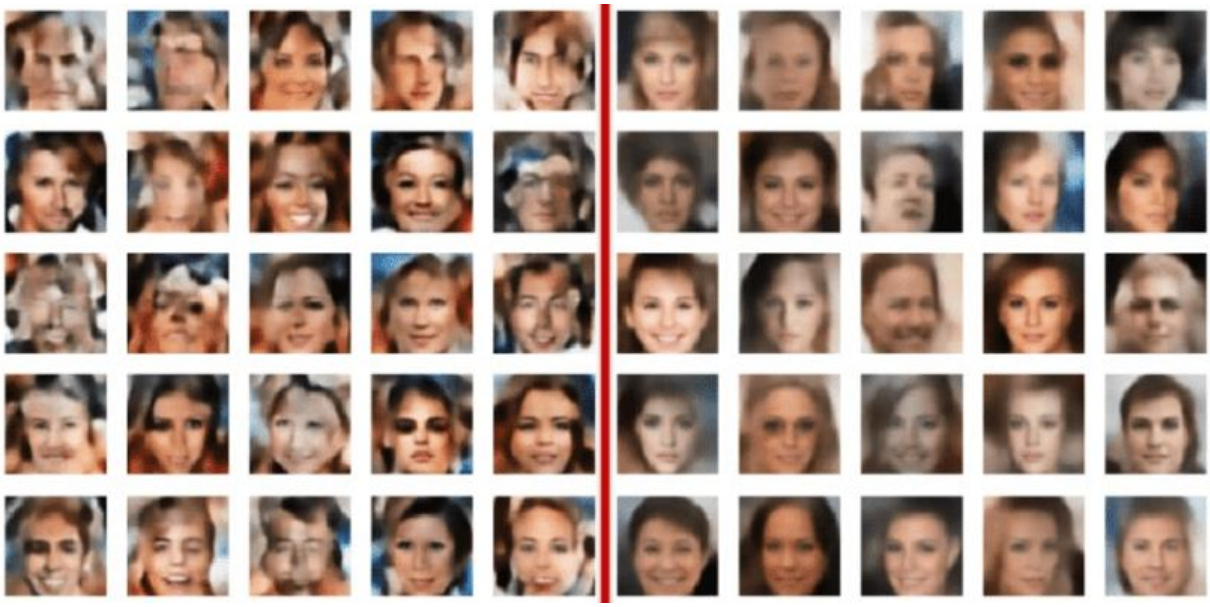


Hình 10: Một ví dụ về mạng CNN sử dụng các lớp tích chập (convolution) để trích xuất các đặc trưng trong ảnh hiệu quả hơn

1.2.3 Phương pháp đề xuất giải quyết bài toán

Theo như các phương pháp đã trình bên trên chúng ta sẽ thấy Variational Autoencoder và BiGAN đáp ứng rất tốt yêu cầu của bài toán:

Đối với Autoencoder thông thường, mô hình Autoencoder luôn tập trung vào mã hóa và giải mã tái tạo lại dữ liệu đầu ra càng giống đầu vào càng nhiều càng tốt tuy nhiên nó lại không quan tâm đến đặc trưng này có hợp lí hay không và điều này lại dễ dẫn đến overfitting. Do đó Variational Autoencoder sẽ biến đổi dữ liệu ở latent space thành một phân bố xác suất để giúp cho mô hình Autoencoder giảm được hiện tượng overfitting hay nói cách khác Variational Autoencoder chính là mô hình nâng cao của Autoencoder. Sau khi huyến luyện xong mô hình Variational Autoencoder chúng ta có thể đem phần Encoder này đi trích xuất các thông tin quan trọng có trong dữ liệu để gom cụm dữ liệu.



Hình 11: Hình ảnh được tái tạo lại bởi lần lượt Autoencoder(bên trái) và Variational Autoencoder (bên phải)

BiGAN cũng là một mô hình nâng cao của mô hình GAN, BiGAN thì sử dụng thêm một bộ mã hóa Encoder để tạo ra các latent code và yêu cầu Discriminator phải

phân biệt luôn cả latent code. BiGAN sau khi huyến luyện xong mô hình thì cũng vừa học được cách tạo dữ liệu và cách trích xuất thông tin từ dữ liệu thành các latent code. Chúng ta sẽ lấy phần Encoder này đem đi trích xuất dữ liệu và sau đó gom cụm dữ liệu.

Theo như các phân tích bên trên cả 2 đều có thể đáp ứng tốt yêu cầu bài toán tuy nhiên Variational Autoencoder vẫn sẽ có một số điểm nhỉnh hơn do BiGAN hay GAN vẫn tồn tại một số nhược điểm đáng lưu tâm như sau:

Generator có thể bị sập (GAN collapse) do không thể tạo ra được dữ liệu đa dạng mới mà chỉ tập trung chỉ tạo một đầu vào phù hợp.

BiGAN và GAN rất nhạy cảm với việc điều chỉnh tham số của mô hình cho nên đa số các mô hình dạng phải điều chỉnh tham số thủ công khá nhiều.

Và với những lí do đã phân tích bên trên chúng ta sẽ sử dụng mô hình Variational Autoencoder để học cách trích xuất dữ liệu và đem mô hình đi trích xuất đặc trưng cho mô hình gom cụm.

1.3 Phương pháp giải quyết bài toán

1.3.1 Mô hình tổng quát

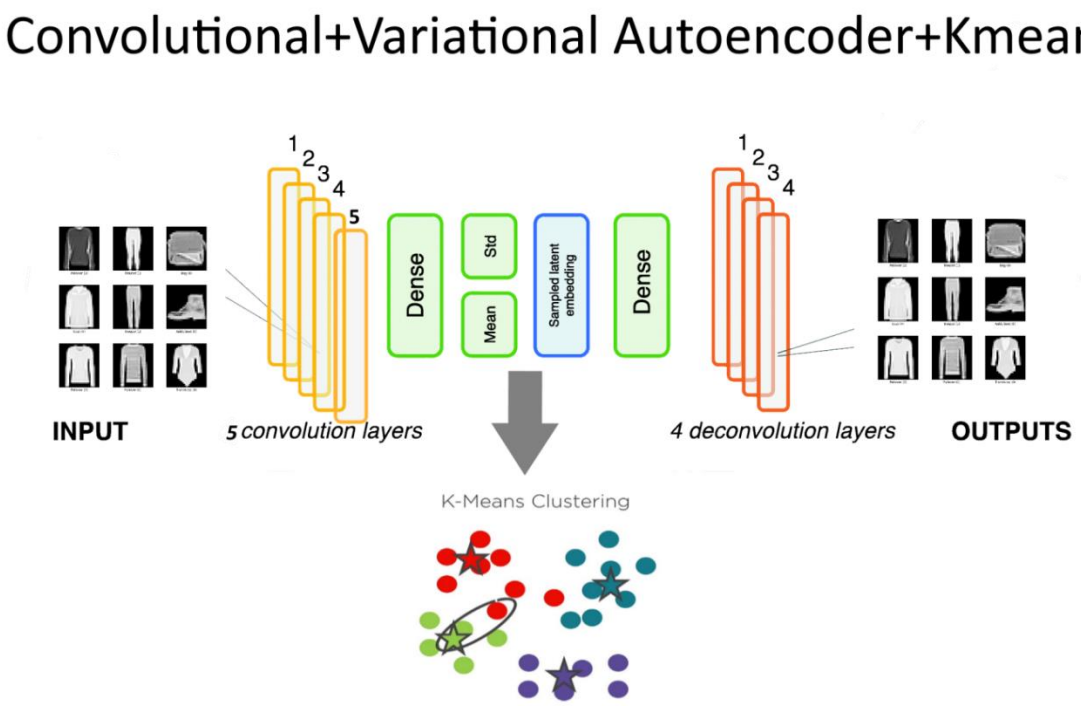
Mô hình tổng quát của Variational Autoencoder để giải quyết bài toán sẽ có dạng như hình bên dưới, mô hình sẽ có 4 phần chính:

Phần 1: Gồm 5 lớp Convolutional kết hợp lại với nhau để tạo ra phần Encoder với input là dữ liệu dạng ảnh. Kết hợp với 4 lớp Convolutional ở phần sau để tạo ra phần Decoder với output là ảnh đã tái tạo lại.

Phần 2: Lớp dense đóng vai trò làm phẳng dữ liệu để chuẩn mã hóa dữ liệu thành phân bố xác suất.

Phần 3: Phần latent space gồm 3 lớp mean và std chính là đại diện cho trung bình μ và độ lệch chuẩn $sigma$. Lớp sampled latent embedding chính là lớp ra tạo vector z phân phối chuẩn từ lớp mean và lớp std.

Phần 4: Gom cụm dữ liệu bằng K-Means. Đối phương pháp Variational Autoencoder thì ta sẽ sử dụng các vector z được tạo ra ở lớp sampled latent embedding để ánh xạ dữ liệu.



Hình 12: Mô hình tổng quát của Variational Autoencoder

1.3.2 Đặc trưng của mô hình đề xuất

1.3.2.1 Tạo bộ mã hóa encoder:

Để xây dựng mô hình Autoencoder cũng như mô hình Variational Autoencoder trước tiên cần phải xây dựng một mã hóa Encoder. Sử dụng 1 lớp input dữ liệu có dạng 28*28*1 và 5 lớp Convolutional lần lượt có các tham số cấu hình như sau:

filters=1,kernel_size=(3,3),activation='relu',name='encoder_convolutional_layer_1'

filters=32,kernel_size=(3,3),activation='relu',name='encoder_convolutional_layer_2'

filters=64,kernel_size=(3,3),activation='relu',name='encoder_convolutional_layer_3'

filters=64,kernel_size=(3,3),activation='relu',name='encoder_convolutional_layer_4'

filters=64,kernel_size=(3,3),activation='relu',name='encoder_convolutional_layer_5'

1.3.2.2 Tạo bộ giải mã decoder:

Bộ giải mã decoder cũng được xây dựng khá giống như bộ mã hóa encoder với một lớp đầu vào có giá trị latent space dim (ở đây là 2) tuy nhiên chúng ta sẽ sử dụng 4 lớp Conv2DTranspose để tăng kích thước decoder để giải mã dữ liệu có cấu hình như sau:

filters=64,kernel_size=(3,3),padding='same',strides=1,activation='relu',name="decoder_conv_tran_layer_1"

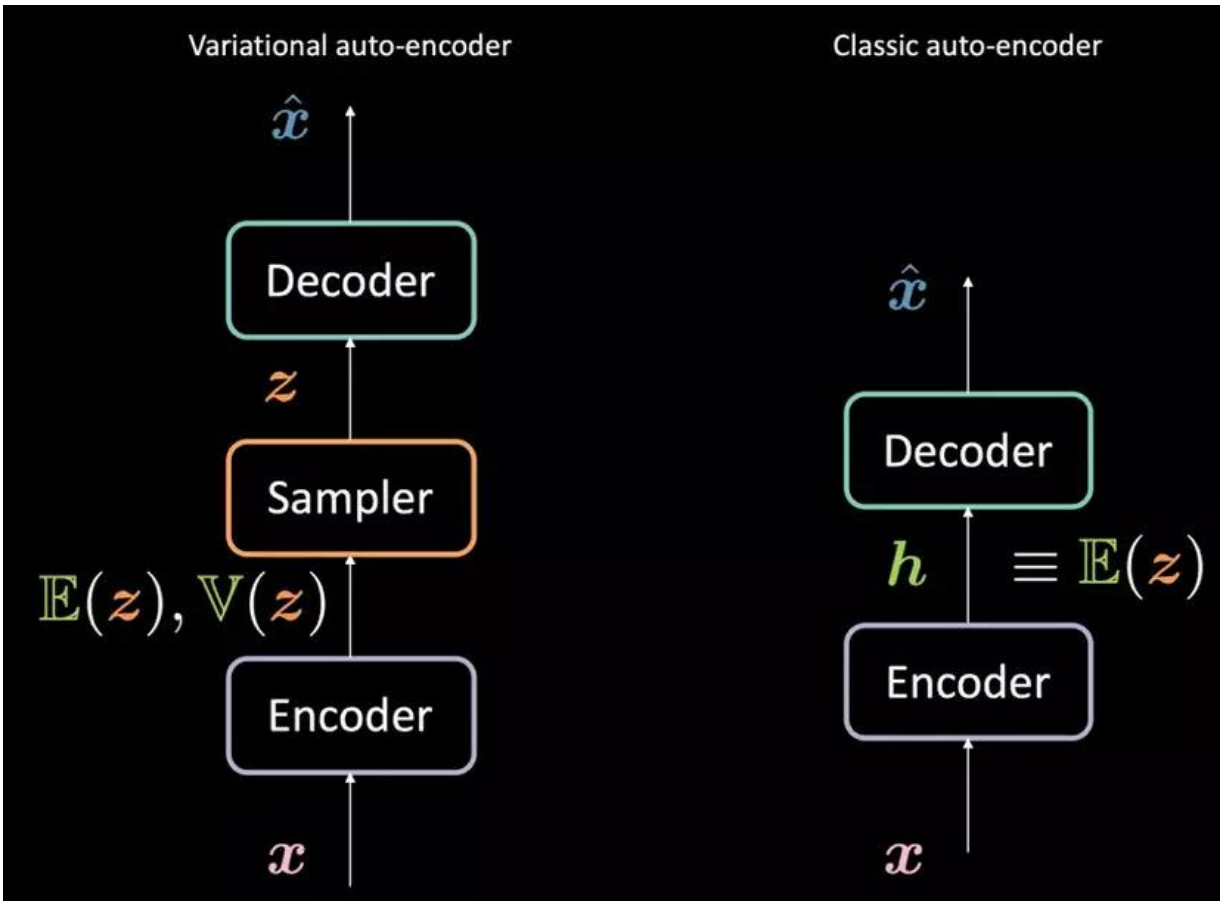
filters=64, kernel_size=(3, 3), padding='same', strides=2,activation='relu', name="decoder_conv_tran_layer_2"

filters=64, kernel_size=(3, 3), padding='same', strides=2, activation='relu', name="decoder_conv_tran_layer_3"

filters=1, kernel_size=(3, 3), padding='same', strides=1, activation='sigmoid', name="decoder_conv_tran_layer_4"

1.3.2.3 Tạo bottleneck hay latent space:

Khác với Autoencoder thông thường, Variational Autoencoder sử dụng 2 lớp mean và std để tạo ra 2 vector $\mathbb{E}(x)$ và $\mathbb{V}(z)$ và sử dụng lớp sampled latent embedding để tạo ra vector z chứa phân phối chuẩn dựa trên phương sai và trung bình tương ứng.



Hình 13 : So sánh giữa mô hình VAE và AE thông thường

1.3.2.4 Xây dựng loss function cho Variational Autoencoder:

Hàm loss function gồm 2 thành phần reconstruction loss và regularization loss theo công thức bên dưới:

$$L(x, \hat{x}) = l_{reconstruct} + \boldsymbol{KL}(z, N(0, I_d))$$

Reconstruction loss: dùng để reconstruct lại input ban đầu của chính mô hình Autoencoder thông thường. Các hàm loss thông dụng là Mean Square Error hay Mean Absolute Error. Trong trường hợp dữ liệu các categorie hoặc ảnh nhị phân, ta có thể sử dụng binary cross entropy.

Regularization loss: sử dụng KL divergence (khoảng cách giữa 2 phân phối xác suất) giữa phân phối chuẩn với trung bình $\mathbb{E}(x)$ và phương sai $\mathbb{V}(z)$ với phân phối chuẩn d chiều N(0, I_d) như công thức bên dưới:

$$\boldsymbol{KL}\big(z, N(0, I_d)\big) = \frac{1}{2}\sum_{i=1}^{d}(\mathbb{V}(z_i) - log\mathbb{V}(z_i) - 1 + \mathbb{E}(z_i)^2)$$

1.3.2.5 Xây dựng lại tham số ở lớp sample - Reparameterization trick:

Như ta đã thấy bên trên lớp sampled latent embedding tạo ra vector z phân phối chuẩn từ phương sai và trung bình tương ứng. Tuy nhiên điều này lại ngăn mô hình Variational Autoencoder thực hiện việc lan truyền ngược backpropagation. Do đó chúng ta cần phải sửa lại cách tính vector z theo như công thức dưới đây:

$$z = \mathbb{E}(z) + \epsilon \odot \sqrt{\mathbb{V}(z)}$$

Trong đó $\epsilon \sim N(0, I_d)$. Cũng bởi vì $\mathbb{E}(z)$ và $\mathbb{V}(z)$ chính là output của encoder nên chúng ta đã có thể thực hiện việc lan truyền ngược.

1.3.2.6 Gom cụm K-means:

Theo như yêu cầu bài toán là gom cụm dữ liệu không có nhãn cho nên thuật toán chúng ta sử dụng sẽ là k-means. Đối với k-means chúng ta sẽ phải lưu ý 2 vấn đề đó là chọn lựa giá trị k cụm và cách ánh xạ dữ liệu đầu vào.

Đối với chọn giá trị k cụm thì chúng ta sẽ chọn số cụm bằng với số cụm gốc của bộ dữ liệu cụ thể là k=10 (10 cụm).

Labels

Each training and test example is assigned to one of the following labels:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

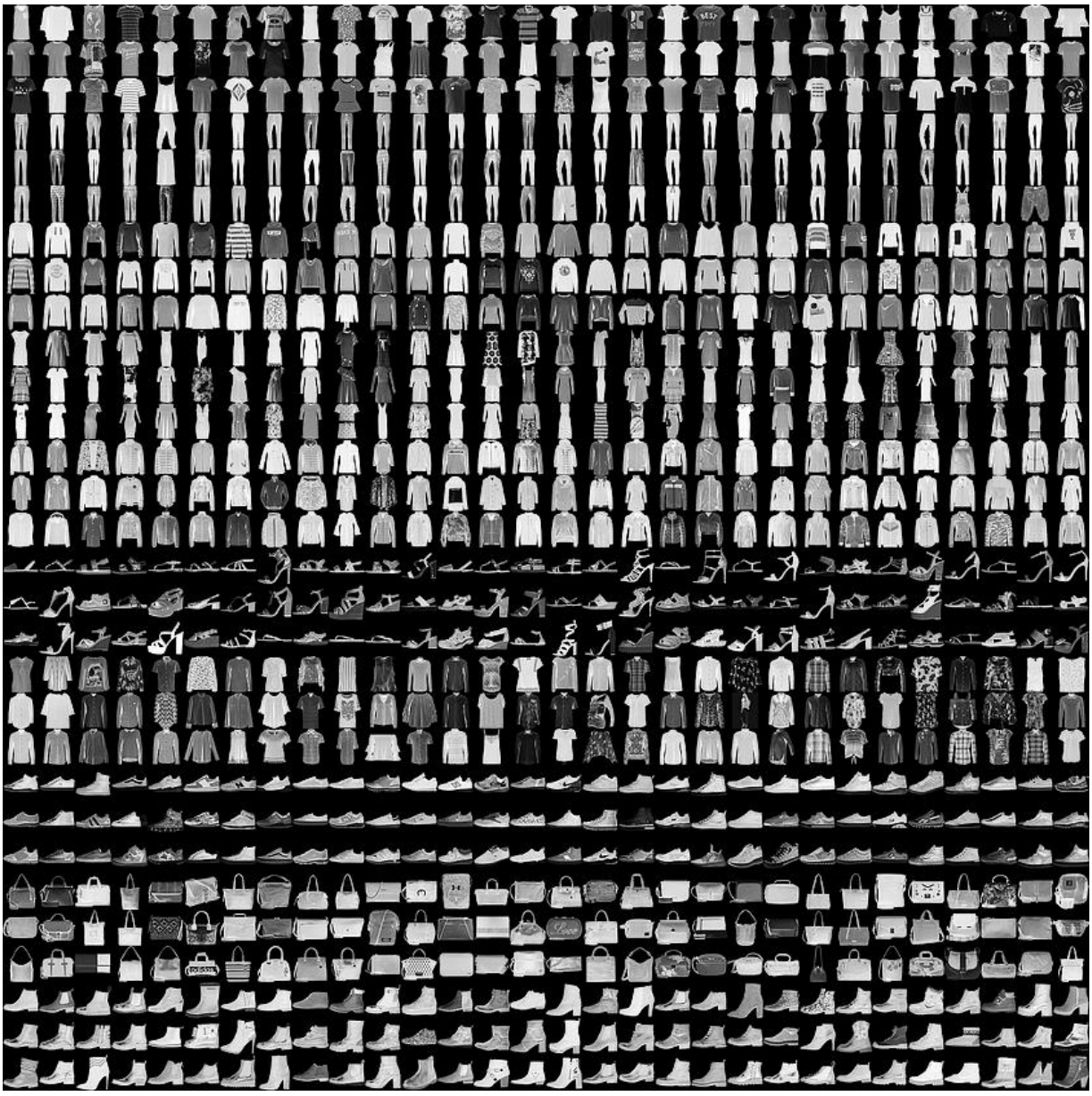
Hình 14 : Hình số nhãn thật của dữ liệu

Đối với việc ánh xạ dữ liệu chúng ta chỉ cần tách nguyên phần encoder của mô hình Variational Autoencoder để ánh xạ dữ liệu và đưa vào mô hình k-means để phân cụm.

1.4 Thực nghiệm

1.4.1 Dữ liệu

Đồ án gom cụm dữ liệu sử dụng dữ liệu dạng ảnh đến từ cơ sở dữ liệu MNIST (Modified National Institute of Standards and Technology database). MNIST là nguồn dữ liệu chứa các ảnh theo các chủ đề như là fashion, hand writing, … và trong đồ án này chúng ta sẽ sử dụng gom cụm dữ liệu ảnh có chủ đề fashion.



Hình 15: Mẫu dữ liệu Fashion của MNIST

Bộ dữ liệu fashion gồm tất cả 70000 ảnh và 70000 nhãn của ảnh và mỗi tấm ảnh là ảnh xám có kích thước 28x28. Trong tổng số 70000 tấm ảnh chúng ta chia ra 60000 ảnh làm train và 10000 làm phần test.

1.4.2 Xử lý dữ liệu

Do dữ liệu của chúng ta đã sẵn ở dạng ảnh xám cho nên không cần phải biến đổi ảnh sang ảnh xám nữa mà chỉ chuyển dữ liệu ảnh từ np.uint8 sang np.float32 để không làm ảnh hướng đến kết quả tính toán trong mô hình Variational Autoencoder.

1.4.3 Công nghệ sử dụng

Ngôn ngữ sử dụng	Python 3.7
Thư viện sử dụng	Keras, matplotlib, tensorflow và sklearn
Môi trường thực thi	Google Colab

Bảng 1: Mô tả công nghệ sử dụng cho thực nghiệm bài toán dự đoán liên kết

1.4.4 Cách đánh giá

Đồ án sẽ sử dụng lần lượt 2 thang đo đó là :

- Adjusted Rand Index, để tính được Adjusted Rand Index ta cần phải tính được chỉ số Rand Index. Rand Index tính độ chính xác bằng cách xem xét tất cả cặp mẫu và đếm các cặp được chỉ định trong các cụm giống nhau hoặc khác nhau, và sau khi đã có được chỉ số Rand Index ta sẽ biến đổi thành Adjusted Rand Index theo 2 công thức sau:

o Công thức tính chỉ số Rand Index:

$$RI = \frac{\text{Number of Agreeing Pairs}}{\text{Number of Pairs}}$$

(Giá trị RI sẽ nằm trong khoảng từ 0 đến 1)

- Công thức tính chỉ số Adjusted Rand Index:

$$ARI = \frac{\text{RI - Expected RI}}{\text{Max(RI) - Expected RI}}$$

(ARI có giá trị sẽ nằm trong khoảng từ 0 đến 1. Với 0 là không tương đương và 1 là khi các cụm giống hệt nhau)

- Fowlkes-Mallows index dung để đo mức độ giống nhau của 2 nhóm của một tập hợp các điểm. Nó được tính theo công thức như sau :

- TP = True Positive số cặp điểm thuộc cùng một cụm trong nhãn thật và nhãn dự đoán.
- FP = False Positive số cặp điểm thuộc cùng một cụm trong các nhãn thật nhưng không nằm trong nhãn dự đoán.
- FN = False Negative số cặp điểm thuộc cùng một cụm trong các nhãn dự đoán nhưng không nằm trong nhãn thật.

Adjusted Rand Index và Fowlkes-Mallows index lần lượt dung để đo điểm số chính xác cho cả 2 phương pháp:

- Có sử dụng mô hình encoder của Variational Autoencoder để ánh xạ dữ liệu sau đó gom cụm.
- Không sử dụng bất kì mô hình học sâu nào để ánh xạ dữ liệu thay vào đó chỉ đơn thuần sử dụng vector để ánh xạ dữ liệu và gom cụm.

1.5 Kết quả đạt được

Tham số cụ thể hóa của thực nghiệm:

- Đồ án sử dụng chia dữ liệu huấn luyện và dữ liệu test là 80%-20%.
- Tổng số lớp tất cả là 15 lớp của encoder và 10 của decoder.

Model: "encoder_model"

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, 28, 28, 1)]	0	[]
encoder_convolutional_layer_1 (Conv2D)	(None, 26, 26, 1)	10	['encoder_input[0][0]']
encoder_normalization_layer_1 (BatchNormalization)	(None, 26, 26, 1)	4	['encoder_convolutional_layer_1[0][0]']
encoder_convolutional_layer_2 (Conv2D)	(None, 24, 24, 32)	320	['encoder_normalization_layer_1[0][0]']
encoder_normalization_layer_2 (BatchNormalization)	(None, 24, 24, 32)	128	['encoder_convolutional_layer_2[0][0]']
encoder_convolutional_layer_3 (Conv2D)	(None, 22, 22, 64)	18496	['encoder_normalization_layer_2[0][0]']
encoder_normalization_layer_3 (BatchNormalization)	(None, 22, 22, 64)	256	['encoder_convolutional_layer_3[0][0]']
encoder_convolutional_layer_4 (Conv2D)	(None, 20, 20, 64)	36928	['encoder_normalization_layer_3[0][0]']
encoder_normalization_layer_4 (BatchNormalization)	(None, 20, 20, 64)	256	['encoder_convolutional_layer_4[0][0]']
encoder_convolutional_layer_5 (Conv2D)	(None, 18, 18, 64)	36928	['encoder_normalization_layer_4[0][0]']
encoder_normalization_layer_5 (BatchNormalization)	(None, 18, 18, 64)	256	['encoder_convolutional_layer_5[0][0]']
flatten (Flatten)	(None, 20736)	0	['encoder_normalization_layer_5[0][0]']
encoder_mu (Dense)	(None, 2)	41474	['flatten[0][0]']
encoder_log_variance (Dense)	(None, 2)	41474	['flatten[0][0]']
lambda (Lambda)	(None, 2)	0	['encoder_mu[0][0]', 'encoder_log_variance[0][0]']

Total params: 176,530
Trainable params: 176,080
Non-trainable params: 450

Hình 16 : Thống kê các thông số trong encoder

```
Model: "decoder_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 2)]               0

 decoder_dense_layer_1 (Dens  (None, 3136)             9408
 e)

 reshape (Reshape)           (None, 7, 7, 64)          0

 decoder_convolutional_tranp  (None, 7, 7, 64)         36928
 ose_layer_1 (Conv2DTranspos
 e)

 decoder_normalization_layer  (None, 7, 7, 64)         256
 _1 (BatchNormalization)

 decoder_convolutional_tranp  (None, 14, 14, 64)       36928
 ose_layer_2 (Conv2DTranspos
 e)

 decoder_normalization_layer  (None, 14, 14, 64)       256
 _2 (BatchNormalization)

 decoder_convolutional_tranp  (None, 28, 28, 64)       36928
 ose_layer_3 (Conv2DTranspos
 e)

 decoder_normalization_layer  (None, 28, 28, 64)       256
 _3 (BatchNormalization)

 decoder_convolutional_tranp  (None, 28, 28, 1)        577
 ose_layer_4 (Conv2DTranspos
 e)

=================================================================
Total params: 121,537
Trainable params: 121,153
Non-trainable params: 384
```

Hình 17 : Thống kê các thông số trong decoder

- Số epoch là 20, batch_size là 32.

```
Epoch 1/20
1500/1500 [==============================] - 634s 422ms/step - loss: 51.9531 - val_loss: 303.9249
Epoch 2/20
1500/1500 [==============================] - 603s 402ms/step - loss: 38.8350 - val_loss: 220.7361
Epoch 3/20
1500/1500 [==============================] - 599s 399ms/step - loss: 36.0171 - val_loss: 35.0173
Epoch 4/20
1500/1500 [==============================] - 607s 405ms/step - loss: 35.1120 - val_loss: 33.9641
Epoch 5/20
1500/1500 [==============================] - 604s 402ms/step - loss: 34.4881 - val_loss: 34.2192
Epoch 6/20
1500/1500 [==============================] - 603s 402ms/step - loss: 33.9968 - val_loss: 33.3089
Epoch 7/20
1500/1500 [==============================] - 603s 402ms/step - loss: 33.6321 - val_loss: 32.9278
Epoch 8/20
1500/1500 [==============================] - 604s 403ms/step - loss: 35.2635 - val_loss: 34.1144
Epoch 9/20
1500/1500 [==============================] - 599s 399ms/step - loss: 33.7532 - val_loss: 32.9037
Epoch 10/20
1500/1500 [==============================] - 604s 403ms/step - loss: 33.1996 - val_loss: 32.4742
Epoch 11/20
1500/1500 [==============================] - 617s 411ms/step - loss: 33.0015 - val_loss: 32.3946
Epoch 12/20
1500/1500 [==============================] - 605s 403ms/step - loss: 32.8321 - val_loss: 32.6240
Epoch 13/20
1500/1500 [==============================] - 609s 406ms/step - loss: 32.6853 - val_loss: 36.7137
Epoch 14/20
1500/1500 [==============================] - 601s 401ms/step - loss: 33.5121 - val_loss: 119.9234
Epoch 15/20
1500/1500 [==============================] - 602s 402ms/step - loss: 33.0821 - val_loss: 175.6801
Epoch 16/20
1500/1500 [==============================] - 607s 405ms/step - loss: 32.4219 - val_loss: 154.3559
Epoch 17/20
1500/1500 [==============================] - 609s 406ms/step - loss: 32.2125 - val_loss: 147.0215
Epoch 18/20
1500/1500 [==============================] - 603s 402ms/step - loss: 32.1152 - val_loss: 32.0414
Epoch 19/20
1500/1500 [==============================] - 608s 405ms/step - loss: 32.0417 - val_loss: 31.6370
Epoch 20/20
1500/1500 [==============================] - 608s 405ms/step - loss: 31.9714 - val_loss: 31.7717
<keras.callbacks.History at 0x7f133891f990>
```

Hình 18: Chỉ số loss sau mỗi epoch

Phương pháp	Adjusted Rand Index	Fowlkes-Mallows Score
K-means có áp dụng Encoder để ánh xạ dữ liệu.	0.3845373306139258	0.45335759046570256

K-means không áp dụng bất kì phương pháp học sâu nào để ánh xạ dữ liệu.	0.3437875481067457	0.416379084460129

Bảng 2 : So sánh kết quả thực nghiệm của mô hình.

Do đã sử dụng phương pháp học sâu để trích xuất các đặc trưng trong bộ dữ liệu mà kết quả phân cụm của k-means đã cải thiện được đôi chút cụ thể là khoảng 10%.

1.6 Kết luận

1.6.1 Kết quả đạt được của bài toán trên:

Đồ án đã tìm hiểu về các phương pháp giải quyết bài toán gom cụm dữ liệu sử dụng học sâu autoencoder, đồng thời đồ án cũng đề xuất phương pháp học sâu BiGAN.

1.6.2 Hạn chế:

Mặc dù đã áp dụng thành công phương pháp học sâu vào phân cụm K-means tuy nhiên kết quả lại không cải thiện được nhiều như dự tính ban đầu là 20%-30%.

1.6.3 Hướng phát triển:

Nhóm sẽ tập trung tìm cách hiệu chỉnh tham số mô hình để cho ra kết quả tốt hơn và cũng đồn thời tìm hiểu thêm nhiều phương pháp học sâu mới đặc biệt là phương pháp Autoencoder kết hợp với BiGAN.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Autoencoder Trong Deep Learning. (n.d.). Https://Tek4.Vn/. Retrieved May 23, 2022, from https://tek4.vn/autoencoder-trong-deep-learning-tu-hoc-tensorflow

[2] Linear Regression và Gradient descent. (n.d.). Https://Nttuan8.Com/. Retrieved May 23, 2022, from https://nttuan8.com/bai-1-linear-regression-va-gradient-descent/#Gradient_descent-2

[3] VAE – Phiên bản nâng cấp của Auto Encoder. (n.d.). Https://Trituenhantao.Io/. Retrieved May 23, 2022, from https://trituenhantao.io/kien-thuc/vae-phien-ban-nang-cap-cua-auto-encoder/#:~:text=VAE%20l%C3%A0%20vi%E1%BA%BFt%20t%E1%BA%AFt%20c%E1%BB%A7a,qua%20n%C3%BAt%20th%E1%BA%AFt%20c%E1%BB%95%20chai

[3] Model GAN. (n.d.). Https://Phamdinhkhanh.Github.Io/. Retrieved May 23, 2022, from https://phamdinhkhanh.github.io/2020/07/13/GAN.html

Tiếng Anh

[1] Variational AutoEncoder. (n.d.). Https://Keras.Io/. Retrieved May 23, 2022, from https://keras.io/examples/generative/vae/

[2] Keras-BiGAN. (n.d.). Https://Github.Com/Manicman1999. Retrieved May 23, 2022, from https://github.com/manicman1999/Keras-BiGAN/blob/master/bigan.py

[3]] GAN_Tutorial. (n.d.). Https://Github.Com/Nttuan8. Retrieved May 23, 2022, from https://github.com/nttuan8/GAN_Tutorial/blob/master/G1/GAN-MNIST.ipynb

[4] Find the mean vector of a Matrix. (n.d.). Https://Www.Geeksforgeeks.Org/. Retrieved May 23, 2022, from https://www.geeksforgeeks.org/find-mean-vector-matrix/

[5] Scikit Learn - Clustering Performance Evaluation. (n.d.). Https://Www.Tutorialspoint.Com/. Retrieved May 23, 2022, from https://www.tutorialspoint.com/scikit_learn/scikit_learn_clustering_performance_evaluation.htm

[6] Stanislas Morbieu. (n.d.). Accuracy: from classification to clustering evaluation. Https://Smorbieu.Gitlab.Io/. Retrieved May 23, 2022, from https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/#:~:text=Accuracy%20for%20clustering%20For%20clustering%2C%20we%20have%20to,K%5D%20where%20K%20is%20the%20number%20of%20clusters.

[7] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui & Daniel Cremers. (2021, November 19). Clustering with Deep Learning: Taxonomy and New Methods. Https://Arxiv.Org/. Retrieved May 23, 2022, from https://arxiv.org/pdf/1801.07648v1.pdf

[8] Unsupervised Clustering with Autoencoder. (n.d.). Https://Ai-Mrkogao.Github.Io/. Retrieved May 23, 2022, from https://ai-mrkogao.github.io/reinforcement%20learning/clusteringkeras/

TỰ ĐÁNH GIÁ

(Với nhóm có 2 thành viên)

Câu	Nội dung	Điểm chuẩn	Tự chấm	Ghi chú
1 (8.5)	**1.1 Giới thiệu về bài toán**	0.5	0.5	
	1.2 Phân tích yêu cầu của bài toán	1.0	1.0	
	1.3 Phương pháp giải quyết bài toán	1.5	1.0	
	1.4 Thực nghiệm	4	3	
	1.5 Kết quả đạt được	1	0.5	
	1.6 Kết luận	0.5	0.5	
2	**Điểm nhóm**	0.5đ	0.5	
3	**Báo cáo (chú ý các chú ý 2,3,4,6 ở trang trước, nếu sai sẽ bị trừ điểm nặng)**	1đ	1	
Tổng điểm			8	