ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH


# KHAI PHÁ DỮ LIỆU - CO3029

---

**Assignment:**

# HIGHLY IMBALANCED CLASSIFICATION USING SAMPLING

---

**Giảng viên hướng dẫn:** PGS.TS Lê Hồng Trang

**Nhóm sinh viên:** Chế Quang Huy - 1812340

Võ Phạm Long Huy - 1812438

TP. Hồ Chí Minh, Tháng 4/2022

# Contents

# 1 Giới thiệu chung về Highly Imbalanced Classification

Mất cân bằng dữ liệu là một trong những hiện tượng phổ biến của bài toán phân loại nhị phân (Binary Classification) như spam email, phát hiện gian lận, dự báo vỡ nợ, chuẩn đoán bệnh lý,.. Trong trường hợp tỷ lệ dữ liệu giữa 2 classes là 50:50 thì được coi là cân bằng. Khi có sự khác biệt trong phân phối giữa 2 classes, chẳng hạn 60:40 thì dữ liệu có hiện tượng mất cân bằng.

Hầu hết các bộ dữ liệu đều khó đạt được trạng thái cân bằng mà luôn có sự khác biệt về tỷ lệ giữa 2 classes. Đối với những trường hợp dữ liệu mất cân bằng nhẹ như tỷ lệ 60:40 thì sẽ không ảnh hưởng đáng kể tới khả năng dự báo của mô hình.

Tuy nhiên nếu hiện tượng **mất cân bằng nghiêm trọng (Highly Imbalanced)** xảy ra, chẳng hạn như tỷ lệ 90:10 sẽ thường dẫn tới ngộ nhận chất lượng mô hình. Khi đó thước đo đánh giá mô hình là độ chính xác (Accuracy) có thể đạt được rất cao mà không cần tới mô hình. Ví dụ, một dự báo ngẫu nhiên đưa ra tất cả đều là nhóm đa số thì độ chính xác đã đạt được là 90%. Do đó không nên lựa chọn độ chính xác làm chỉ số đánh giá mô hình để tránh lạc quan sai lầm về chất lượng.

Trong trường hợp mẫu mất cân bằng nghiêm trọng ta cần phải thay đổi chỉ số đánh giá để đưa ra kết quả hợp lý hơn. Ngoài ra, mất cân bằng dữ liệu nghiêm trọng thường dẫn tới dự báo kém chính xác trên nhóm thiểu số. Bởi đa phần kết quả dự báo ra thường thiên về 1 nhóm là nhóm đa số và rất kém trên nhóm thiểu số. Trong khi tầm quan trọng của việc dự báo được chính xác một mẫu thuộc nhóm thiểu số lớn hơn nhiều so với dự báo mẫu thuộc nhóm đa số. Để cải thiện kết quả dự báo chúng ta cần những điều chỉnh thích hợp để mô hình đạt được một độ chính xác cao trên nhóm thiểu số.

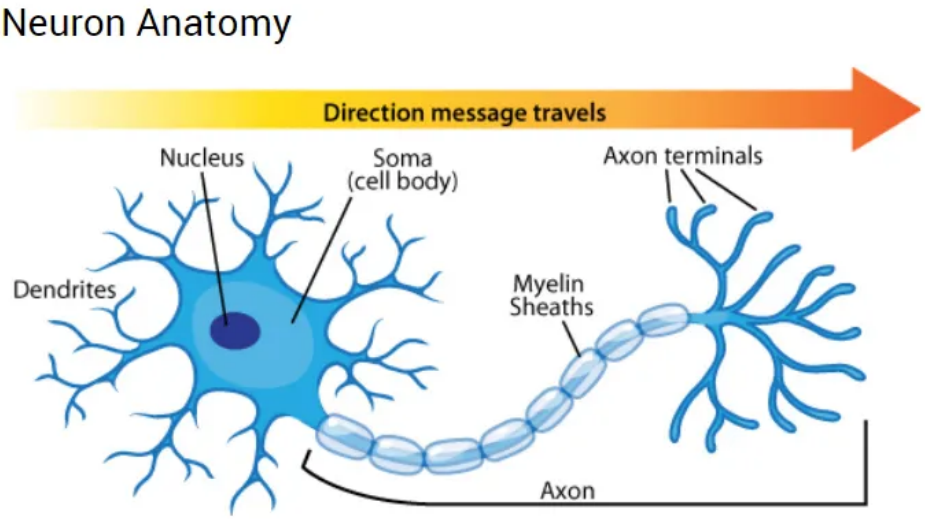
Những điều chỉnh cần thiết giúp cải thiện hiệu năng dự báo của mô hình trong trường hợp xảy ra mất cân bằng nghiêm trọng là gì? Trong bài báo cáo này nhóm em sẽ trình bày về một số phương pháp lấy mẫu nhằm tạo ra một tập dữ liệu cân bằng hơn.

# 2 Cơ sở lý thuyết

## 2.1 Neural Networks

### 2.1.1 Định nghĩa về Neural Netwworks

Neural Networks hay còn gọi là mạng nơ-ron nhân tạo là mạng sử dụng các mô hình toán học phức tạp để xử lý thông tin. Chúng dựa trên mô hình hoạt động của các tế bào thần kinh và khớp thần kinh trong não của con người. Mạng nơ-ron nhân tạo kết nối các nút đơn giản, còn được gọi là tế bào thần kinh. Và một tập hợp các nút như vậy tạo thành một mạng lưới các nút, do đó có tên là mạng nơ-ron nhân tạo.




Hình 1: Hình ảnh về hoạt động của nơ-ron thần kinh con người

Tương tự như bộ não con người, trong mạng nơ-ron nhân tạo, một loạt các thuật toán được sử dụng để xác định và nhận ra các mối quan hệ trong các tập dữ liệu. Mạng nơ-ron nhân tạo được sử dụng trên nhiều công nghệ và ứng dụng khác nhau như trò chơi điện tử, thị giác máy tính, nhận dạng giọng nói, lọc mạng xã hội, dịch tự động và chẩn đoán y tế. Đáng ngạc nhiên là mạng nơ-ron nhân tạo được sử dụng cho các hoạt động truyền thống và sáng tạo, như hội họa và nghệ thuật.

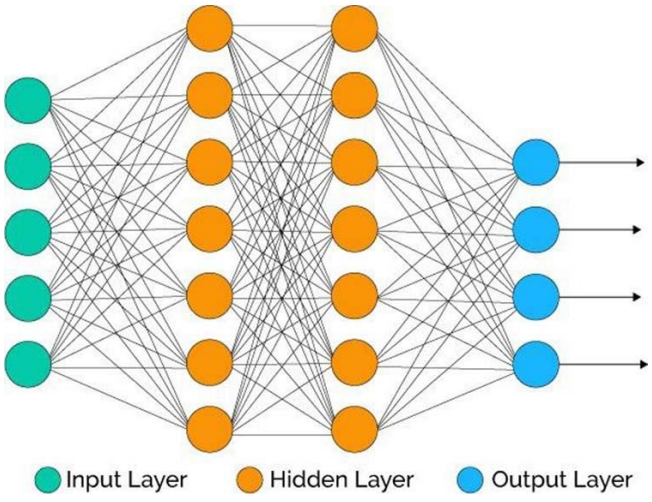
"Neural Networks phản ánh hành vi của não người, cho phép các chương trình máy tính nhận ra các mối quan hệ trong dữ liệu và giải quyết các vấn đề phổ biến trong lĩnh vực AI, học máy (Machine Learning) và học sâu (Deep Learning)."

### 2.1.2 Kiến trúc mạng Neural Networks

Mạng Neural Networks là sự kết hợp của những tầng Perceptron hay còn gọi là Perceptron đa tầng. Và mỗi một mạng Neural Networks thường bao gồm 3 kiểu tầng là:

- *Tầng Input Layer (tầng vào):* Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
- *Tầng Output Layer (tầng ra):* Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.
- *Tầng Hidden Layer (tầng ẩn):* Tầng này nằm giữa tầng vào và tầng ra nó thể hiện cho quá trình suy luận logic của mạng.

*Lưu ý:* Mỗi một Neural Networks chỉ có duy nhất một tầng vào và 1 tầng ra nhưng lại có thể không hoặc có rất nhiều tầng ẩn.




Hình 2: Kiến trúc mạng Neural Networks

Với mạng Neural Networks thì mỗi nút mạng là một sigmoid nơ-ron nhưng chúng lại có hàm kích hoạt khác nhau. Thực tế, người ta thường sử dụng có cùng loại với nhau để việc tính toán thuận lợi hơn. Tại mỗi tầng, số lượng nút mạng có thể khác nhau còn tùy vào bài toán hoặc cách giải quyết.

Tuy nhiên, khi làm việc người ta sẽ để các tầng ẩn số với số lượng nơ-ron khác nhau. Ngoài ra, những nơ-ron nằm ở tầng thường sẽ liên kết đôi với nhau để tạo thành mạng kết nối đầy

đủ nhất. Khi đó, người dùng có thể tính toán được kích cỡ của mạng dựa vào tầng và số lượng nơ-ron. Ví dụ ở hình trên ta có:

$\rightarrow$ 4 tầng mạng, trong đó có 2 tầng ẩn.

$\rightarrow$ $3 + 4 \times 2 + 1 = 12$ nút mạng.

$\rightarrow$ $(3 \times 4 + 4 \times 4 + 4 \times 1) + (4 + 4 + 1) = 41$ tham số.

### 2.1.3 Ưu điểm và nhược điểm của Neural Networks

**Ưu điểm:**

- Mạng nơ-ron có thể thực hiện các tác vụ mà chương trình tuyến tính không thể thực hiện.
- Khi một phần tử của mạng nơ-ron bị lỗi, nó có thể tiếp tục mà không gặp vấn đề gì bởi tính chất song song của chúng.
- Mạng nơ-ron không cần lập trình lại vì nó tự học.
- Nó có thể được thực hiện một cách dễ dàng mà không gặp bất kỳ vấn đề gì.
- Là hệ thống thông minh, thích ứng, mạng nơ-ron rất mạnh mẽ và vượt trội trong việc giải quyết các vấn đề phức tạp. Mạng nơ-ron rất hiệu quả trong lập trình của chúng và các nhà khoa học đồng ý rằng lợi thế của việc sử dụng ANNs lớn hơn rủi ro.
- Nó có thể được thực hiện trong bất kỳ ứng dụng nào.

**Nhược điểm:**

- Các mạng thần kinh yêu cầu đào tạo để hoạt động.
- Yêu cầu thời gian xử lý cao đối với các mạng nơ-ron lớn.
- Kiến trúc của mạng nơ-ron khác với kiến trúc bộ vi xử lí thông thường vì vậy chúng phải được mô phỏng.

### 2.1.4 Tính phổ dụng Neural Networks

Mạng nơ-ron nhân tạo rất linh hoạt và thích ứng. Mạng nơ-ron nhân tạo được sử dụng trong các hệ thống nhận dạng trình tự và mẫu, xử lý dữ liệu, robot, mô hình hóa,.. ANNs tiếp thu kiến thức từ môi trường xung quanh bằng cách thích ứng với các thông số bên trong và bên ngoài và giúp giải quyết các vấn đề phức tạp khó quản lý. Tổng quát hóa kiến thức để tạo ra phản ứng thích hợp cho các tình huống chưa biết.

**Tính linh hoạt:** Mạng nơ-ron nhân tạo linh hoạt và có khả năng học hỏi, khái quát hóa và thích ứng với các tình huống dựa trên những phát hiện của nó.

**Không tuyến tính:** Chức năng này cho phép mạng thu nhận kiến thức một cách hiệu quả bằng cách học. Đây là một lợi thế khác biệt so với mạng tuyến tính truyền thống, điều này không phù hợp khi mô hình hóa dữ liệu phi tuyến tính.

Mạng nơron nhân tạo có khả năng chịu lỗi lớn hơn mạng truyền thống. Nếu không mất dữ liệu được lưu trữ, mạng có thể khôi phục lỗi trong bất kỳ thành phần nào của nó.

### 2.1.5 Một số ứng dụng của Neural Networks

- Các ứng dụng mạng nơ-ron nhân tạo đã được sử dụng trong lĩnh vực năng lượng mặt trời để mô hình hóa và thiết kế nhà máy tạo hơi nước bằng năng lượng mặt trời.
- Chúng hữu ích trong việc lập mô hình hệ thống, chẳng hạn như trong việc thực hiện ánh xạ phức tạp và xác định hệ thống.
- ANNs được sử dụng để ước tính tải nhiệt của các tòa nhà, hệ số chặn của bộ thu máng hình parabol và tỷ lệ nồng độ cục bộ.
- ANNs được sử dụng trong các ứng dụng đa dạng trong điều khiển, người máy, nhận dạng mẫu, dự báo, y học, hệ thống điện, sản xuất, tối ưu hóa, xử lý tín hiệu và khoa học xã hội/tâm lý.
- Chúng cũng đã được sử dụng để dự đoán luồng không khí trong phòng thử nghiệm thông gió tự nhiên và dự đoán mức tiêu thụ năng lượng của các tòa nhà năng lượng mặt trời.
- Có thể xử lý dữ liệu nhiễu và không đầy đủ và cũng có thể giải quyết các vấn đề phi tuyến tính.
- Việc sử dụng mạng thần kinh nhân tạo trong các hệ thống thông gió và điều hòa không khí, làm lạnh, mô hình hóa, sưởi ấm, dự báo tải, điều khiển hệ thống phát điện và bức xạ mặt trời.

## 2.2 CNNs - Convolutional Neural Networks

### 2.2.1 Định nghĩa về CNNs

Convolutional Neural Network (hay còn gọi là CNNs mạng nơ ron tích chập) là một trong những mô hình Deep Learning vô cùng tiên tiến. CNNs sẽ cho phép bạn xây dựng các hệ thống

thông minh với độ chính xác vô cùng cao. Hiện nay, CNNs được ứng dụng rất nhiều trong những bài toán nhận dạng object trong ảnh. Và kiến thức cụ thể về CNNs đã được lý giải như sau:

**Convolutional:** Đây là một loại cửa sổ dạng trượt nằm trên một ma trận. Những Convolutional Layer sẽ có các Parameter được học để điều chỉnh và lấy ra những thông tin chính xác nhất mà không cần phải chọn Feature. Convolution hay tích chập chính là nhân các phần tử trong ma trận. Sliding Window còn được gọi là Kernel, Filter hoặc Feature Detect và là loại ma trận có kích thước nhỏ.

**Feature:** Feature là đặc điểm, các CNNs sẽ so sánh hình ảnh dựa theo từng mảnh và những mảnh này được gọi là Feature. Thay vì phải khớp các bức ảnh lại với nhau thì CNNs sẽ nhìn ra sự tương động khi tìm kiếm thô các Feature khớp với nhau bằng 2 hình ảnh tốt hơn. Mỗi Feature được xem là một hình ảnh mini có nghĩa chúng là những mảng 2 chiều nhỏ. Các Feature này đều tương ứng với các khía cạnh nào đó của hình ảnh và chúng có thể khớp lại với nhau.

### 2.2.2 Các lớp cơ bản của mạng CNNs

**Convolutional Layer:** Đây là lớp quan trọng nhất của CNN, lớp này có nhiệm vụ thực hiện mọi tính toán. Những yếu tố quan trọng của một convolutional layer là:

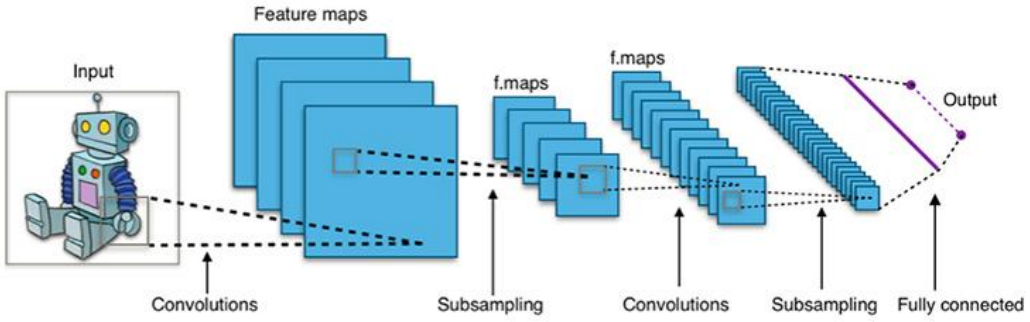
- Filter Map: Để áp dụng vào vùng của hình ảnh. Những Filter Map này được gọi là ma trận 3 chiều, mà bên trong nó là các con số và chúng là Parameter.
- Stride: Có nghĩa là khi bạn dịch chuyển Filter Map theo Pixel dựa vào giá trị trừ trái sang phải. Và sự chuyển dịch này chính là Stride.
- Padding: Là các giá trị 0 được thêm vào với lớp input.
- Feature Map: Nó thể hiện kết quả của mỗi lần Filter Map quét qua input. Sau mỗi lần quét sẽ xảy ra quá trình tính toán.

**Relu Layer:** Đây là hàm kích hoạt trong Neural Networks và hàm này còn được gọi là Activation Function. Hàm kích hoạt có tác dụng mô phỏng các Neuron có tỷ lệ truyền xung qua Axon. Trong Activation Function thì nó còn có hàm nghĩa là: Relu, Leaky, Tanh, Sigmoid, Maxout,.. Hiện nay, hàm Relu được dùng phổ biến và vô cùng thông dụng.

Nó được sử dụng nhiều cho các nhu cầu huấn luyện mạng Neuron thì Relu mang lại rất nhiều ưu điểm nổi bật như việc tính toán sẽ trở nên nhanh hơn. Quá trình sử dụng Relu,

chúng ta cần lưu ý đến vấn đề tùy chỉnh các Learning Rate và theo dõi Dead Unit. Những lớp Relu Layer đã được sử dụng sau khi Filter map được tính ra và áp dụng hàm Relu lên những giá trị của Filter Map.

**Pooling Layer:** Khi đầu vào quá lớn, những lớp Pooling Layer sẽ được xếp vào giữa giữa những lớp Convolutional Layer để làm giảm Parameter. Hiện nay, Pooling Layer có 2 loại chủ yếu là: Max Pooling và Average.




Hình 3: Cấu trúc của CNNs

**Fully Connected Layer:** Lớp này có nhiệm vụ đưa ra kết quả sau khi lớp Convolutional Layer và Pooling Layer đã nhận được ảnh truyền. Lúc này, ta thu được kết quả là Model đã đọc được thông tin của ảnh và để liên kết chúng cũng như cho ra nhiều Output hơn thì ta sử dụng Fully Connected Layer.

Ngoài ra, nếu như Fully Connected Layer có được giữ liệu hình ảnh thì chúng sẽ chuyển nó thành mục chưa được phân chia chất lượng. Cái này khá giống với phiếu bầu rồi chúng sẽ đánh giá để bầu chọn ra hình ảnh có chất lượng cao nhất.

### 2.2.3 Kiến trúc mạng CNNs

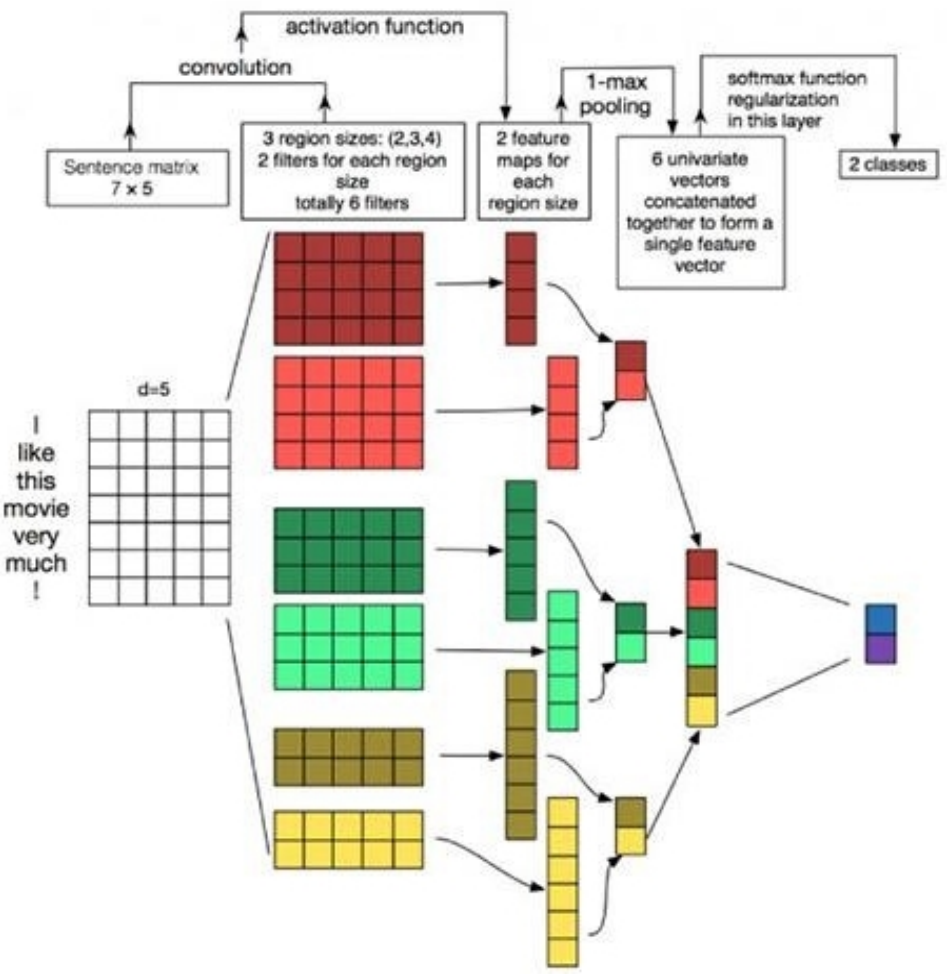
Mạng CNNs là một trong những tập hợp của lớp Convolution bị chồng lên nhau cũng như sử dụng hàm Nonlinear Activation như Relu và tanh để kích hoạt trọng số trong Node. Lớp này sau khi thông qua hàm thì sẽ được trọng số trong các node. Những lớp này sau khi đã thông qua hàm kích hoạt thì có thể tạo ra những thông tin trừu tượng hơn cho những lớp tiếp theo.

Trong mô hình CNNs có tính bất biến và tích kết hợp. Nếu như bạn có cùng một đối tượng mà lại chiếu theo nhiều góc độ khác nhau thì độ chính xác có thể sẽ bị ảnh hưởng. Với chuyển

dịch, quay và co giãn thì Pooling Layer sẽ được sử dụng để giúp làm bất biến những tính chất này. Vì vậy, CNNs sẽ đưa ra kết quả có độ chính xác tương ứng ở từng mô hình.

Trong đó, Pooling Layer sẽ cho bạn tính bất biến đối với phép dịch chuyển, phép co dãn và phép quay. Còn tính kết hợp cục bộ sẽ cho bạn thấy những cấp độ biểu diễn, thông tin từ thấp đến mức độ cao với độ trừu tượng thông qua Convolution từ các Filter. Mô hình CNNs có các layer liên kết được với nhau dựa vào cơ chế Convolution.

Những Layer tiếp theo sẽ là kết quả từ những Convolution từ Layer trước đó, vì thế mà bạn sẽ có các kết nối cục bộ phù hợp nhất. Vậy, mỗi neuron ở lớp sinh ra tiếp theo từ kết quả Filter sẽ áp đặt lên vùng ảnh cục bộ của một Neuron có trước đó. Trong khi huấn luyện mạng, CNNs sẽ tự động học hỏi các giá trị thông qua lớp Filter dựa vào cách thức mà người dùng thực hiện.




Hình 4: CNNs được ứng dụng rộng rãi

Trong đó, cấu trúc cơ bản của CNNs thường bao gồm 3 phần chính là:

- *Local Receptive Field (trường cục bộ):* Lớp này có nhiệm vụ tách lọc dữ liệu, thông tin ảnh và lựa chọn các vùng ảnh có giá trị sử dụng cao nhất.
- *Shared Weights and Bias (trọng số chia sẻ):* Lớp này giúp làm giảm tối đa lượng tham số có tác dụng chính của yếu tố này trong mạng CNNs. Trong mỗi Convolution sẽ có các Feature map khác nhau và mỗi Feature lại có khả năng giúp Detect một vài Feature trong ảnh.
- *Pooling Layer (lớp tổng hợp):* Lớp cuối cùng và có tác dụng làm đơn giản các thông tin đầu ra. Có nghĩa là, sau khi đã hoàn tất tính toán và quét qua các lớp thì đến Pooling Layer để lược bớt các thông tin không cần thiết. Từ đó, cho ra kết quả theo như ý mà người dùng mong muốn.

### 2.2.4 Ưu điểm và nhược điểm của CNNs

**Ưu điểm:**

- Tận dụng được tính năng trích chọn đặc trưng của lớp tích chập và bộ phân lớp được huấn luyện đồng thời.
- Tự động học các đặc trưng của dữ liệu để thiết lập các đặc trưng mới và phân lớp dữ liệu.
- Lượng tham số nhỏ hơn nhiều so với mạng kết nối đầy đủ, vì các lớp Pooling giúp trích xuất đặc trưng của ảnh khiến ma trận ảnh đi qua mỗi lớp nhỏ dần đi, nên lượng tham số cần học cũng giảm theo.
- Có tính kháng dịch chuyển. Vì loại đi những dữ liệu thừa có ảnh hưởng đến vị trí của đối tượng nên mạng nhận diện chính xác một đối tượng nằm ở các vị trí khác nhau trong một ảnh.

**Nhược điểm:**

- Dễ bị Overfit do lượng tham số cần tìm quá lớn và Data quá nhỏ.
- Mạng không phân loại được cùng một loại đối tượng nhưng ở vị trí khác nhau trong ảnh.

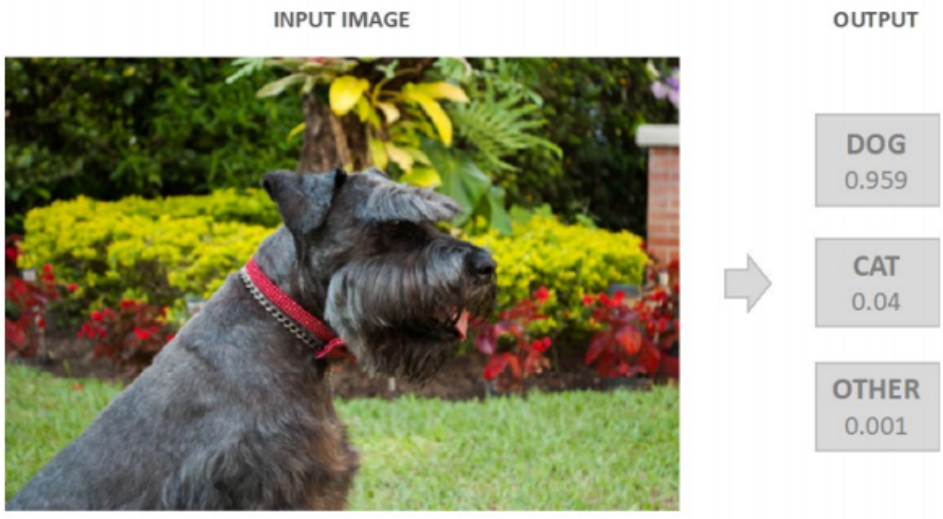
### 2.2.5 Ứng dụng của CNNs

Mặc dù CNN chủ yếu được sử dụng cho các vấn đề về Computer Vision, nhưng điều quan trọng là đề cập đến khả năng giải quyết các vấn đề học tập khác của họ, chủ yếu liên quan đến tích chuỗi dữ liệu.

*Ví dụ:* CNNs đã được biết là hoạt động tốt trên chuỗi văn bản, âm thanh và video, đôi khi kết hợp với các mạng khác quả cầu kiến trúc hoặc bằng cách chuyển đổi các chuỗi thành hình ảnh có thể được xử lý của CNNs. Một số vấn đề dữ liệu cụ thể có thể được giải quyết bằng cách sử dụng CNNs với chuỗi dữ liệu là các bản dịch văn bản bằng máy, xử lý ngôn ngữ tự nhiên và gắn thẻ khung video, trong số nhiều người khác.

**Classification:** Đây là nhiệm vụ được biết đến nhiều nhất trong Computer Vision. Ý tưởng chính là phân loại nội dung chung của hình ảnh thành một tập hợp các danh mục, được gọi là nhãn.

*Ví dụ:* Phân loại có thể xác định xem một hình ảnh có phải là của một con chó, một con mèo hay bất kỳ động vật khác. Việc phân loại này được thực hiện bằng cách xuất ra xác suất của hình ảnh thuộc từng lớp, như được thấy trong hình ảnh sau.



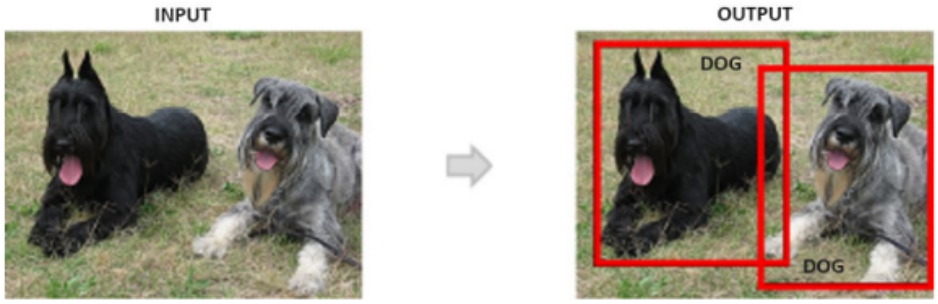

Hình 5: Hình ảnh về Classification

**Localization:** Mục đích chính của Localization là tạo ra một hộp giới hạn mô tả vị trí của đối tượng trong hình ảnh. Đầu ra bao gồm một nhãn lớp và một hộp giới hạn. Tác vụ này

có thể được sử dụng trong cảm biến để xác định xem một đối tượng ở bên trái hay bên phải của màn hình.


Hình 6: Hình ảnh về Localization

**Detection:** Nhiệm vụ này bao gồm thực hiện Localization trên tất cả các đối tượng trong ảnh. Các đầu ra bao gồm nhiều hộp giới hạn, cũng như nhiều nhãn lớp (một cho mỗi hộp). Nhiệm vụ này được sử dụng trong việc chế tạo ô tô tự lái, với mục tiêu là có thể xác định vị trí các biển báo giao thông, đường, ô tô khác, người đi bộ và bất kỳ đối tượng nào khác có thể phù hợp để đảm bảo trải nghiệm lái xe an toàn.




Hình 7: Hình ảnh về Ditection

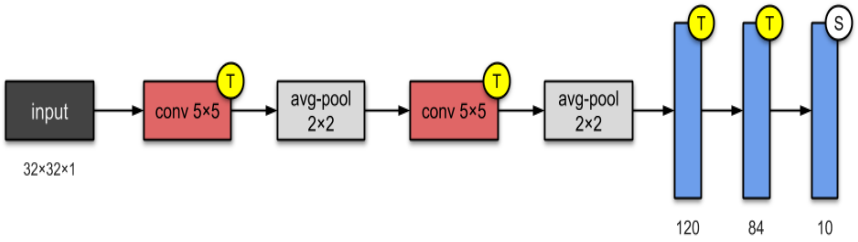
**Segmentation:** Nhiệm vụ ở đây là xuất ra cả nhãn lớp và đường viền của mỗi đối tượng hiện diện trong hình ảnh. Điều này chủ yếu được sử dụng để đánh dấu các đối tượng quan trọng của hình ảnh cho phân tích sâu hơn.

*Ví dụ:* tác vụ này có thể được sử dụng để phân định rõ ràng khu vực tương ứng với khối u trong hình ảnh phổi của bệnh nhân. Hình sau mô tả cách vật thể quan tâm được phác thảo và gán nhãn.


Hình 8: Hình ảnh về Segmentation

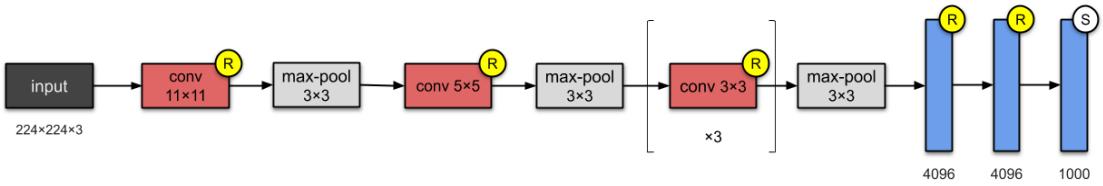
### 2.2.6 Các mạng CNNs thông dụng

**LeNet (1998):** Là mạng đầu tiên áp dụng tích chập 2 chiều.



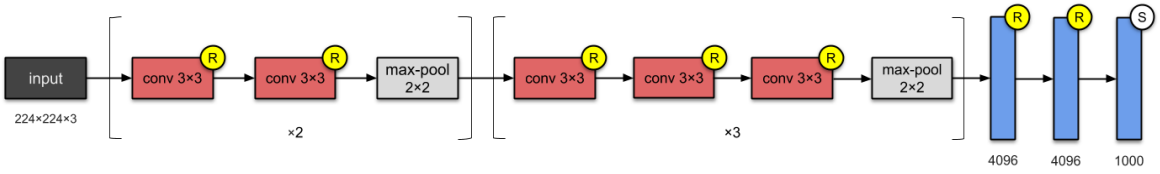

Hình 9: Kiến trúc mạng LeNet

**AlexNet (2012):** Làm mạng áp dụng CNNs đầu tiên chiến thắng trong cuộc thi ImageNet. Phá vỡ lối mòn sử dụng các đặc trưng thủ công từ các thuật toán truyền thống như HOG, SHIFT, SURF thay cho các đặc trưng được huấn luyện trong các tác vụ học có giám sát của thị giác máy tính.



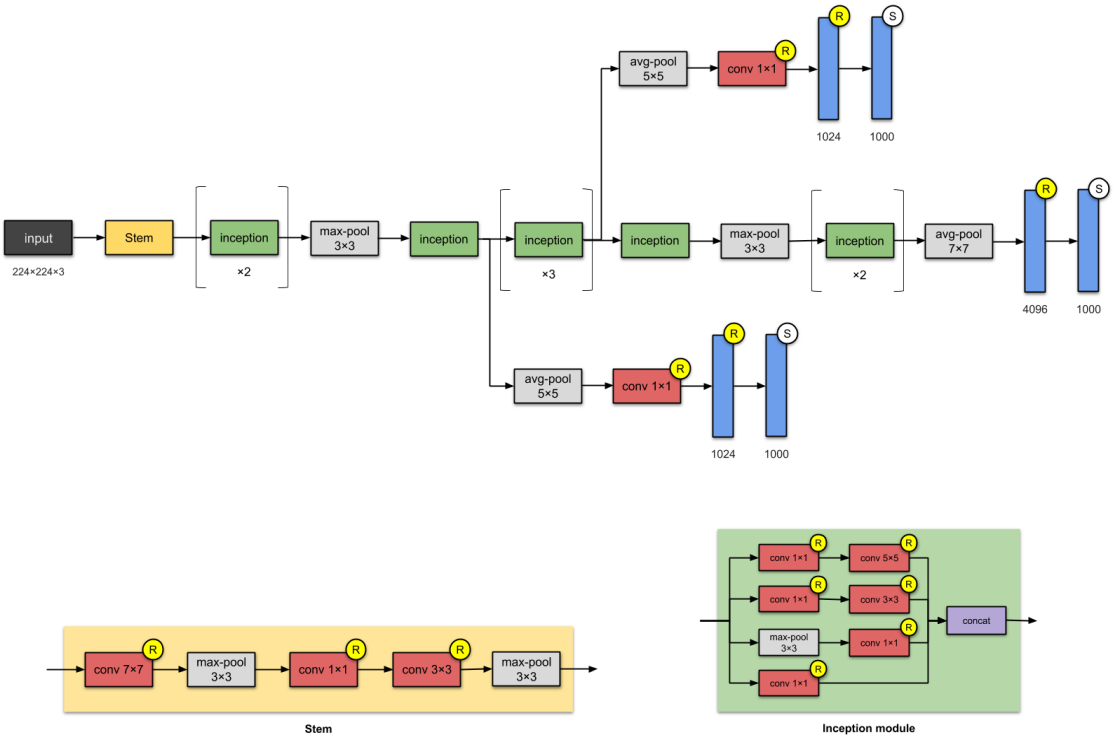

Hình 10: Kiến trúc mạng AlexNet

**VGG-16 (2014):** Hình thành một xu hướng cải thiện độ chính xác của các mạng học sâu thông qua gia tăng độ sâu của chúng.



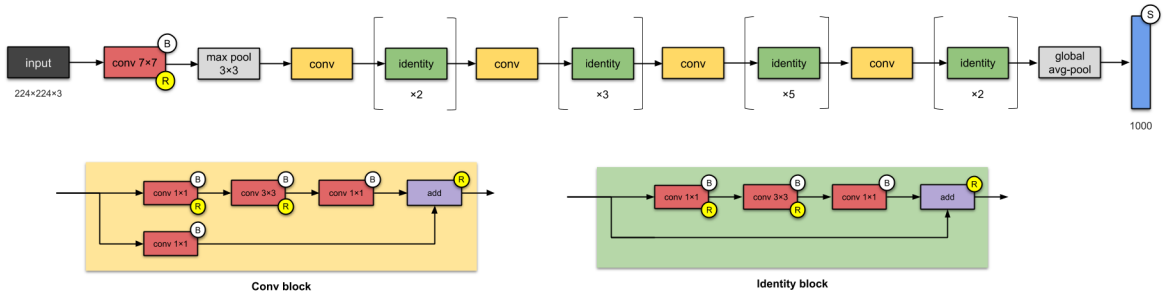

Hình 11: Kiến trúc mạng VGG-16

**GoogleNet - InceptionV1 (2014):** Kết hợp nhiều bộ lọc có kích thước khác biệt vào cùng một khối. Định hình kiến trúc khối cho các kiến trúc mạng CNNs chuẩn sau này.



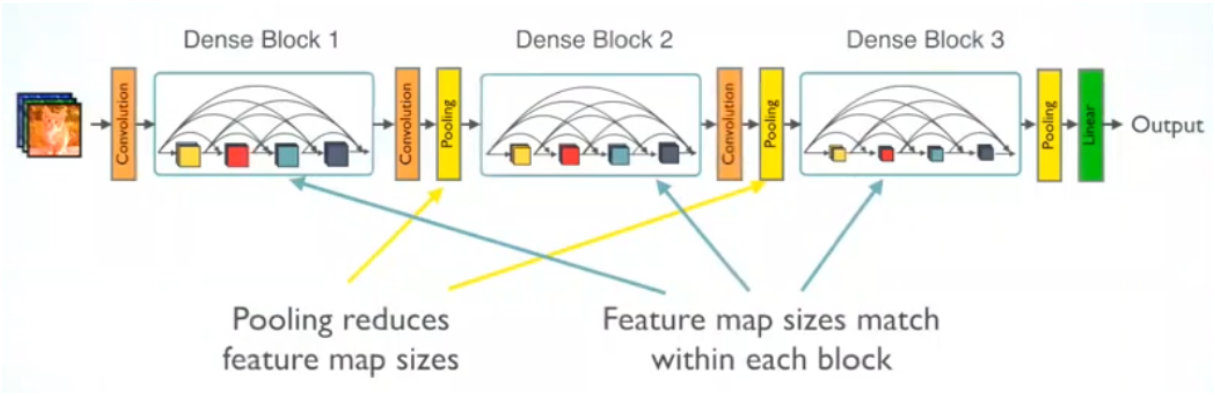

Hình 12: Kiến trúc mạng GoogleNet

**ResNet-50 (2015):** Sử dụng kết nối tắt để ánh xạ các đầu vào từ những Layer trước đó tới những Layer sau. Là kiến trúc mạng rất sâu nhưng có số tham số nhỏ hơn nhờ kế thừa những kỹ thuật từ GoogleNet.




Hình 13: Kiến trúc mạng ResNet-50

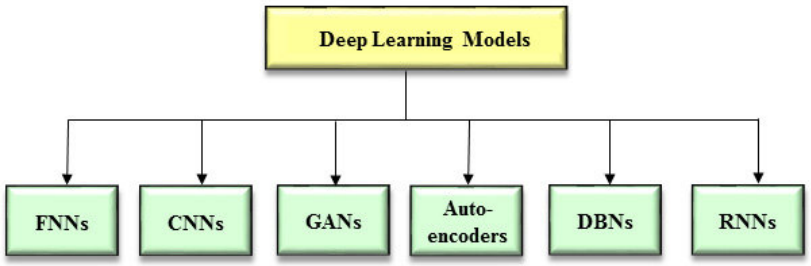
**DenseNet (2016):** Là bước phát triển tiếp theo cua ResNet khi kế thừa kiến trúc khối và phát triển kết nối tắt theo một mạng dày đặc.




Hình 14: Kiến trúc mạng DenseNet

## 2.3 Các biến thể khác liên quan khác của mạng Neural Networks

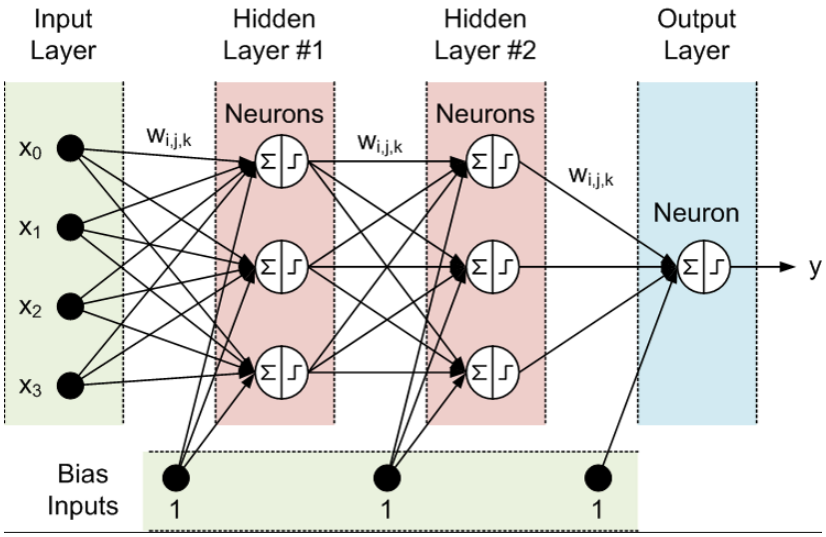
Chúng ta có 6 biến thể liên quan đến Deep Learning, trong đó có CNNs và RNNs đã tìm hiểu ở những mục bên trên còn có:




Hình 15: Những biến thể của Deep Learning

### 2.3.1 Recurrent Neural Networks (RNNs)

Mô hình RNNs là một lớp mạng nơ-ron nhân tạo trong đó các kết nối giữa các nút tạo thành một đồ thị có hướng hoặc vô hướng dọc theo một trình tự thời gian, có khả năng xử lý các phép đo chuỗi thời gian hoặc dữ liệu tuần tự như chuỗi DNA. Thuật ngữ "mạng nơ-ron hồi quy" (RRNs) được sử dụng để chỉ lớp mạng có đáp ứng xung vô hạn, trong khi "mạng nơ-ron tích chập" (CNNs) dùng để chỉ lớp đáp ứng xung hữu hạn. RNNs có thể cần nhiều thời gian tính toán hơn CNNs vì chúng khó có thể thực hiện song song. RNNs được sử dụng để xử lý ngôn ngữ tự nhiên ứng dụng nhiều trong Deep Learning.

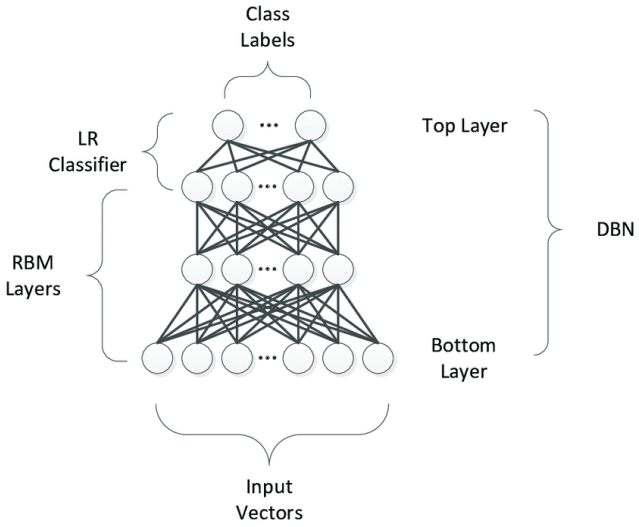



Hình 16: Mạng RNNs

### 2.3.2 Deep Belief Networks (DBNs)

Trong học máy, mạng niềm tin sâu (DBNs) là một mô hình đồ họa tổng quát, hoặc cách khác là một lớp của mạng nơ-ron sâu, bao gồm nhiều lớp biến tiềm ẩn (Hidden Unit), với các kết nối giữa các lớp nhưng không phải giữa các đơn vị bên trong mỗi lớp. Ngoại trừ lớp đầu tiên và lớp cuối cùng, mọi lớp trong DBNs thực hiện vai trò kép bằng cách đóng vai trò là lớp ẩn đối với các nút đến trước nó và là lớp đầu vào cho các nút đến sau.

DBNs có thể được xem như một thành phần của các mạng đơn giản, không được giám sát, chẳng hạn như máy Boltzmann (RBM) hoặc máy mã tự động, trong đó mỗi lớp ẩn của mạng con đóng vai trò là lớp hiển thị cho lớp tiếp theo. Một số ứng dụng của mạng niềm tin sâu sắc là nhận dạng, phân cụm và tạo ra hình ảnh, chuỗi video và dữ liệu ghi lại chuyển động.




Hình 17: Mạng DBNs

### 2.3.3 Generative Adversarial Networks (GANs)

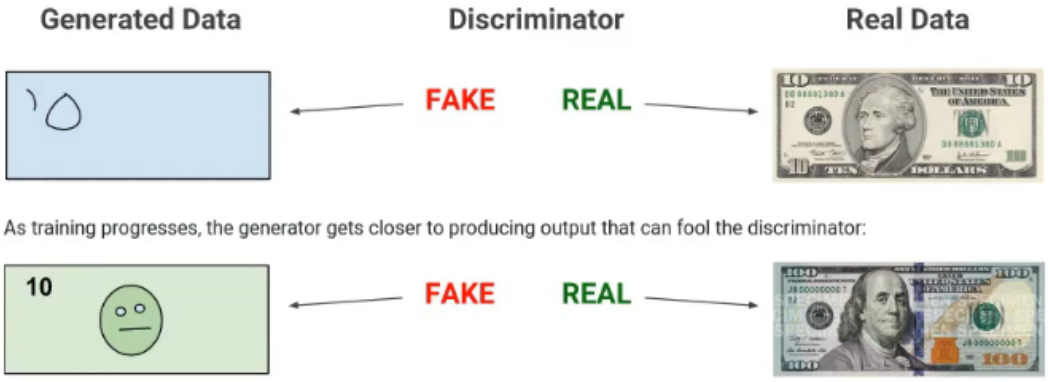
Mạng GANs gồm 2 thành phần:

**Generator:** Đây là mô hình sinh, dùng để sinh ra những dữ liệu giả.

**Discriminator:** Mô hình phân biệt, dùng để phân biệt dữ liệu mà Generator tạo ra với những dữ liệu có trong tập dữ liệu.

GAN đã trở thành một trong những lĩnh vực nghiên cứu nổi bật nhất trong học sâu và chủ

yếu được áp dụng trong lĩnh vực thị giác máy tính và xử lý hình ảnh (chẳng hạn như tạo hình ảnh).




Hình 18: Mạng GANs

# 3 Một số kỹ thuật về Sampling

Có hai cách tiếp cận để tạo ra một tập dữ liệu cân bằng từ tập dữ liệu mất cân bằng đó là Under Sampling và Over Sampling. Đây là hai kỹ thuật được sử dụng trong Khai phá dữ liệu và phân tích để sửa đổi các lớp dữ liệu trở nên cân bằng hơn. Under Sampling và Over Sampling còn được gọi là Resampling (lấy mẫu lại).

## 3.1 Upper Sampling

Việc lấy mẫu làm cân bằng tập dữ liệu bằng cách giảm kích thước của lớp trội. Phương pháp này được sử dụng khi số lượng dữ liệu là đủ. Bằng cách giữ tất cả các mẫu trong lớp hiếm và chọn ngẫu nhiên một số mẫu trong lớp trội, một tập dữ liệu cân bằng mới có thể được lấy ra để lập mô hình tiếp theo. Thông thường Under Sampling được sử dụng khi lượng dữ liệu được thu thập lớn hơn mức lý tưởng và có thể giúp các công cụ khai thác dữ liệu có thể xử lý hiệu quả.

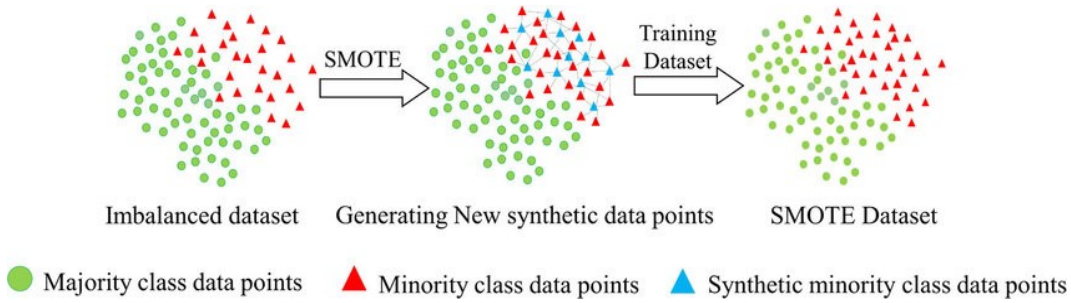
## 3.2 Over Sampling

Ngược lại, Over Sampling được sử dụng khi số lượng dữ liệu không đủ. Phương pháp này nhằm cố gắng cân bằng số liệu bằng cách tăng kích thước của các classes chiếm thiểu số. Thay vì loại bỏ các mẫu lớn, các classes hiếm mới được tạo ra bằng cách sử dụng một số phương

pháp như: lặp lại, Bootstrapping,.. Lưu ý rằng không có lợi thế tuyệt đối của một phương pháp Resampling này so với một phương pháp khác. Việc áp dụng hai phương thức này phụ thuộc vào trường hợp sử dụng mà nó áp dụng cho và tập dữ liệu chính nó. Một kỹ thuật Over Sampling phổ biến đó là SMOTE (Synthetic Minority Over Sampling Technique).

**SMOTE (Synthetic Minority Over Sampling Technique):** Thuật toán SMOTE được đề xuất năm 2002, nhằm giải quyết vấn đề mất cân bằng dữ liệu. Đây là một trong những cách tiếp cận nổi tiếng nhất do sự đơn giản và hiệu quả của nó. Cụ thể SMOTE sinh thêm phần tử nhân tạo bằng cách như sau: đầu tiên tìm các phần tử gần nhất với mỗi phần tử của lớp thiểu số; sau đó chọn ngẫu nhiên một trong số đó; cuối cùng sinh thêm phần tử nhân tạo trên đoạn thẳng nối phần tử đang xét và được lựa chọn bằng cách tính độ lệch giữa véc tơ thuộc tính của phần tử lớp thiểu số đang xét và láng giềng của nó.

SMOTE là một kỹ thuật Over Sampling trong đó các mẫu tổng hợp được tạo ra cho lớp thiểu số. Thuật toán này giúp khắc phục vấn đề Over Fitting do lấy mẫu quá mức ngẫu nhiên. Nó tập trung vào không gian đặc trưng để tạo ra các trường hợp mới với sự trợ giúp của phép nội suy giữa các trường hợp tích cực nằm với nhau.




Hình 19: Kỹ thuật SMOTE

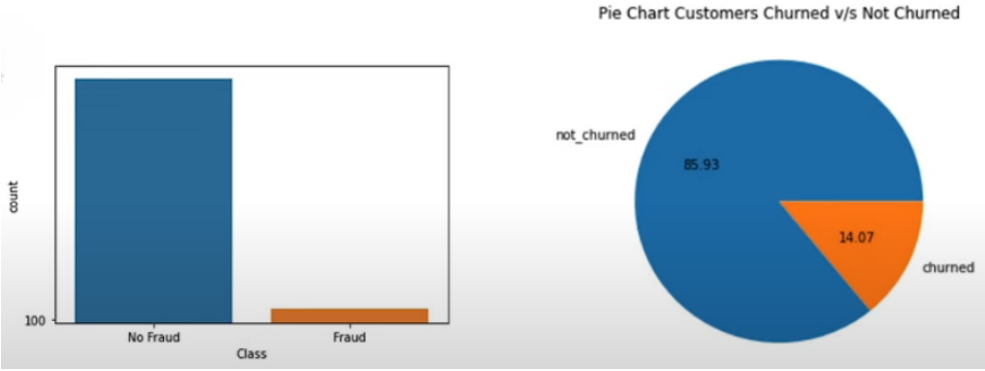
Mặc dù thuật toán này khá hữu ích, nhưng nó có một số nhược điểm đi kèm với nó.

- Các thể hiện tổng hợp được tạo ra theo cùng một hướng, tức là được nối với nhau bằng một đường nhân tạo các thể hiện đường chéo của nó. Điều này làm phức tạp về mặt quyết định đối với một số thuật toán phân loại.
- SMOTE có xu hướng tạo ra số "không" lớn của các điểm dữ liệu nhiễu trong không gian đối tượng.

# 4 Một số ứng dụng trong việc phát hiện bất thường

Một số ứng dụng phổ biến của việc phát hiện bất thường trong cuộc sống mà nhóm đã tìm hiểu được, ví dụ như:

**Phát hiện các giao dịch giả mạo của ngân hàng:** Trong ngân hàng sẽ có những giao dịch đúng với khách hàng đã thực hiện và một số giao dịch do hacker thực hiện khi đã hack được thẻ của khách hàng.




Hình 20: Phát hiện bất thường trong giao dịch ngân hàng

**Dự đoán trong Y tế:** phát hiện bệnh nhân có tỷ lệ bị bệnh ung thư dựa trên phân tích dữ liệu có sẵn từ một vùng nào đó.

# 5 Hiện thực và Thực nghiệm

## 5.1 Datasets sử dụng để thực nghiệm SMOTE

Datasets xử dụng từ nguồn kaggle.com mà thầy đã cho ở đề bài. Nhóm đã hiện thực kỹ thuật SMOTE để xử lý mất cân bằng dữ liệu để xử lý gian lận trong giao dịch thẻ tín dụng.

Sau đây là link datasets và demo kết quả:

- https://www.kaggle.com/mlg-ulb/creditcardfraud
- Link dữ liệu mất cân bằng cần xử lý: https://bitly.com.vn/wewfyg
- Link demo xử lý dữ liệu thông qua SMOTE chạy trên Jupyter: https://bitly.com.vn/jt5vws

## 5.2 Các bước thực hiện giải thuật SMOTE

**Bước 1:** Import các thư viện Python sẽ sử dụng.

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline
```

```
import warnings
warnings.filterwarnings('ignore')
```

**Bước 2:** Đọc dữ liệu datasets Credit Card được lấy từ trang Kaggle theo đề tài.

```
df = pd.read_csv("C:/Users/Admin/Downloads/archive/creditcard.csv")
df.sample(5)
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 | V24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **225062** | 144086.0 | 2.014303 | 0.222328 | -1.730133 | 0.517234 | 0.304026 | -1.365999 | 0.364623 | -0.437297 | 0.416314 | ... | 0.222693 | 0.836251 | -0.054554 | -0.114675 |
| **76060** | 56379.0 | -0.499146 | 1.225395 | 2.125937 | 2.726520 | 0.248282 | 0.593994 | 0.465769 | 0.206636 | -1.652407 | ... | -0.125575 | -0.360794 | -0.217753 | 0.000089 |
| **89720** | 62698.0 | -0.534420 | 0.918204 | 2.042266 | 0.455937 | 0.032230 | 0.639126 | 0.083430 | -0.890892 | 0.473683 | ... | 0.692664 | -0.084574 | -0.182048 | -0.430580 |
| **134611** | 80885.0 | 1.211280 | 0.639500 | -0.452050 | 0.897389 | 0.166469 | -1.167578 | 0.360975 | -0.218111 | -0.240264 | ... | -0.101293 | -0.185911 | -0.017291 | 0.296851 |
| **191073** | 129094.0 | -0.651906 | -5.522984 | -2.372308 | 0.216573 | -2.216428 | 0.065825 | 1.114007 | -0.374656 | -0.414680 | ... | 0.620974 | -1.537952 | -0.842458 | -0.388440 |

Hình 21: Datasets của Credit Card

**Bước 3:** Thống kê sự chênh lệch của các trường hợp sử dụng thẻ tín dụng đã giao dịch hợp lệ và giao dịch gian lận.

Trong đó:

- 0 chỉ ra số giao dịch hợp lệ.
- 1 chỉ ra số giao dịch gian lận.

```
df.Class.value_counts()
```

```
Out[4]: 0    284315
        1       492
        Name: Class, dtype: int64
```

Hình 22: Kết quả thống kê sự chênh lệch của các giao dịch

**Bước 4:** Khai báo các biến và sử dụng thư viện imblearn.over_sampling để dùng SMOTE.

```
X = df.drop('Class',axis='columns')
y = df['Class']
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='minority')
X_sm, y_sm = smote.fit_resample(X, y)
y_sm.value_counts()
```

```
Out[8]: 0    227452
        1    227452
        Name: Class, dtype: int64
```

Hình 23: Dữ liệu được cân bằng sau khi sử dụng SMOTE

# 6 Phân tích và đánh giá (Accuracy, Confusion Matrix)

## 6.1 Accuracy

Cách đơn giản và hay được sử dụng nhất là Accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số giao dịch hợp lệ và tổng số giao dịch được thực hiện trong datasets.

**Bước 5:** Import thư viện sklearn để tách dữ liệu trong datasets thành 2 thành phần. Gọi là *training* và *test*. Theo đó SMOTE sẽ dựa vào trông tin ở *training* để dự đoán xem mỗi dữ liệu trong *test* tương ứng với loại giao dịch nào (gian lận hay hợp lệ). Dữ liệu được dự đoán này sẽ được đối chiếu với các giao dịch hợp lệ của mỗi dữ liệu trong *test* để đánh giá hiệu quả của SMOTE.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import accuracy_score
def log_reg(X_train, y_train, X_test, y_test, weights):
    if weights==-1:
        model = LogisticRegression()
    else:
        model = LogisticRegression(class_weight={0:weights[0], 1:weights[1]})

    model.fit(X_train, y_train)
    acc = model.score(X_test, y_test)
    print("Accuracy", acc, "\n")

    y_pred = model.predict(X_test)
    print("preds", y_pred[:5], "\n")

    cl_rep = classification_report(y_test,y_pred)
    print(cl_rep)

    return y_pred
```

```
weights = -1 # pass -1 to use Logistics Regression without weights
log_reg(X_train, y_train, X_test, y_test, weights)
```

```
Accuracy 0.9739725304679668

preds [0 0 1 0 1]

              precision    recall  f1-score   support

           0       0.97      0.98      0.97     56863
           1       0.98      0.96      0.97     56863

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```

Hình 24: Kết quả Accuracy sau khi được hiện thực

## 6.2 Confusion Matrix

Cách tính sử dụng Accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là Confusion Matrix.

**Bước 6:** Import thư viện confusion_matrix.

```
from sklearn.metrics import confusion_matrix
y_predict = log_reg(X_train, y_train, X_test, y_test, weights)
cnf_matrix = confusion_matrix(y_test, y_predict)
print('Confusion matrix:')
print(cnf_matrix)
```

```
preds [0 0 1 0 1]

              precision    recall  f1-score   support

           0       0.97      0.98      0.97     56863
           1       0.98      0.96      0.97     56863

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726

Confusion matrix:
[[55904   959]
 [ 2001 54862]]
```

Hình 25: Kết quả Confusion Matrix sau khi được hiện thực

# 7 Kết luận

Như vậy tôi đã giới thiệu xong một số các phương pháp chính đối phó với hiện tượng mất cân bằng dữ liệu. Trong quá trình xây dựng mô hình, đặc biệt là các mô hình phân loại nhị phân (2 classes) các bạn sẽ thường xuyên gặp lại hiện tượng này. Mất cân bằng dữ liệu sẽ dẫn tới mô hình dự báo kém chính xác và đa phần kết quả dự báo bị thiên về nhãn đa số. Trong trường hợp đó, các thước đo như Accuracy cũng không phải là một Metric tốt để đánh giá mô hình. Qua

bài viết này các bạn sẽ có thêm những phương pháp hữu hiệu để đối phó với các tình huống mất cân bằng dữ liệu. Tùy vào từng bài toán và từng bộ dữ liệu mà Data Scientist có thể lựa chọn một hoặc kết hợp một vài phương pháp để cải thiện hiệu năng mô hình.

# 8 Tài liệu tham khảo

[1 ] Recurrent Neural Network (Phần 1): *Tổng quan và ứng dụng*. URL: https://www.viblo.asia

[2 ] SMOTE: *Synthetic Minority Over-sampling Technique*. URL: https://arxiv.org/pdf/1106.1813.pdf

[3 ] Credit Card Fraud Detection. URL: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[4 ] SMOTE for Imbalanced Classification with Python. URL: https://www.machinelearningmastery.com

[5 ] Khoa học dữ liệu. URL: https://www.phamdinhkhanh.github.io

[6 ] Thuật toán CNN - Convolutional Neural Network: *TopDev*.

[7 ] Bài 3: Neural network: *Deep Learning cơ bản*. URL: https://www.nttuan8.com

[8 ] Bài 33: Các phương pháp đánh giá một hệ thống phân lớp. URL: https://machinelearningcoban.com/2017/08/31/evaluation/

[9 ] Bài 24 - Mất cân bằng dữ liệu (imbalanced dataset). URL: https://phamdinhkhanh.github.io/2020/02/17/ImbalancedData.html