**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**---------------**

****

**BÁO CÁO CUỐI KỲ  
Chuyên ngành : Trí tuệ nhân tạo**

Đề tài: Tái tạo hình ảnh nhiệt dựa trên deep learning và phát hiện đối tượng

**GVHD: GV Huỳnh Quốc Bảo**

**SVTH: Phan Thanh Tú 2180608173**

**Trần Mai Trung Nguyên 2180607812**

**Hoàng Lê Hữu Châu 2180608776**

**Lớp : 21DTHD5**

***TP. HCM, tháng 12 năm 2023***

**MỤC LỤC:**

[**I.** **GIỚI THIỆU VÀ CƠ SỞ LÝ THUYẾT** 4](#_Toc156146303)

[ **Các phương pháp deep learning trong xử lý hình ảnh nhiệt** 5](#_Toc156146304)

[ **Một số ưu điểm của Deep Learning:** 6](#_Toc156146305)

[ **Nhược điểm:** 6](#_Toc156146306)

[ **Giới thiệu về Neural Network** 6](#_Toc156146307)

[ **Giới thiệu về GAN** 8](#_Toc156146308)

[ **Cấu trúc mạng GAN** 9](#_Toc156146309)

[ **Giới thiệu về thuật toán liên quan đến Deep Learning là CNN (Convolutional Neural Network – CNN).** 10](#_Toc156146310)

[**II.** **CÔNG CỤ VÀ CÔNG NGHỆ CÓ LIÊN QUAN** 14](#_Toc156146311)

[ **Một số phương pháp và thuật toán quan trọng:** 14](#_Toc156146312)

[ **Các công cụ và thư viện giúp xừ lý và giải quyết** 14](#_Toc156146313)

[**III.** **CÁC BÀI TOÁN LIÊN QUAN** 15](#_Toc156146314)

[ **Convolutional Neural Network** 15](#_Toc156146315)

[ **Convolution Layer** 16](#_Toc156146316)

[ **Pooling layer** 17](#_Toc156146317)

[ **Fully connected layer** 17](#_Toc156146318)

[**IV.** **ÁP DỤNG VÀO BÀI TOÁN** 18](#_Toc156146319)

[ **Mô hinh GAN** 18](#_Toc156146320)

[**Generator** 18](#_Toc156146321)

[**Discriminator** 21](#_Toc156146322)

[ **Mô hình neural network** 23](#_Toc156146323)

[**Logistic regression** 23](#_Toc156146324)

[**Mô hình tổng quát** 25](#_Toc156146325)

[**Kí hiệu** 25](#_Toc156146326)

[**Feedforward** 27](#_Toc156146327)

[**Phép tính convolution** 28](#_Toc156146328)

[**Convolution** 28](#_Toc156146329)

[**Padding** 29](#_Toc156146330)

[**Stride** 30](#_Toc156146331)

[**Ý nghĩa của phép tính convolution** 32](#_Toc156146332)

[Pooling layer 39](#_Toc156146333)

[**Fully connected layer** 41](#_Toc156146334)

[**Visualise convolutional neural network** 41](#_Toc156146335)

[**Mạng VGG 16** 43](#_Toc156146336)

[**V.** **KẾT LUẬN VÀ CÔNG VIỆC TƯƠNG LAI** 44](#_Toc156146337)

[**VI.** **TÀI LIỆU THAM KHẢO** 45](#_Toc156146338)

**Báo Cáo**

**MEMU:** Tái tạo hình ảnh nhiệt dựa trên deep learning và phát hiện đối tượng

Tóm Tắt : Gần đây, việc áp dụng deep learning trong xử lý hình ảnh nhiệt đang thu hút sự chú ý. Máy ảnh nhiệt độ đang được sử dụng rộng rãi trong nhiều lĩnh vực như giám sát thông minh, nhận diện sinh trắc học và theo dõi sức khỏe. Tuy nhiên, chi phí cao và vấn đề mờ trong hình ảnh do chuyển động vật thể, chuyển động của máy ảnh và cài đặt tiêu điểm đã tạo ra thách thức. Để giải quyết vấn đề này, nghiên cứu mới đề xuất sử dụng deep learning, đặc biệt là mạng GAN (Generative Adversarial Networks), để tái cấu trúc và làm mịn hình ảnh nhiệt. Mạng GAN có khả năng tạo ra hình ảnh sắc nét hơn và bảo toàn chi tiết cấu trúc.Nghiên cứu này đề xuất một phương pháp gọi là deblur-SRRGAN để tái cấu trúc hình ảnh nhiệt độ. Đồng thời, họ cũng đề xuất sử dụng mạng Mask R-CNN nhẹ để phát hiện đối tượng trong hình ảnh nhiệt đã được tái cấu trúc. Các thí nghiệm được tiến hành trên cơ sở dữ liệu tự thu thập và cơ sở dữ liệu mở, cho thấy phương pháp mới này vượt trội hơn so với các phương pháp hiện có, đánh dấu bước tiến đáng kể trong việc xử lý hình ảnh nhiệt bằng deep learning

# **GIỚI THIỆU VÀ CƠ SỞ LÝ THUYẾT**

* Xử lý hình ảnh nhiệt bằng deep learning là một lĩnh vực nghiên cứu và ứng dụng quan trọng trong thị giác máy tính, trong đó deep learning (học sâu) được sử dụng để xử lý dữ liệu hình ảnh được thu thập từ máy ảnh nhiệt hoặc cảm biến nhiệt. Ứng dụng của hình ảnh nhiệt mở ra nhiều tiềm năng trong nhiều lĩnh vực, bao gồm:

1. Y tế và Chăm sóc sức khỏe: Hình ảnh nhiệt được sử dụng để theo dõi sức khỏe, chẩn đoán bệnh và giám sát trạng thái sức khỏe của người dùng thông qua việc đo nhiệt độ cơ thể.
2. An ninh và Giám sát: Trong giám sát thông minh, hình ảnh nhiệt đóng vai trò quan trọng trong việc nhận diện đối tượng, phát hiện xâm nhập, và giám sát môi trường.
3. Công nghiệp và Hàng không vũ trụ: Sử dụng hình ảnh nhiệt để kiểm tra các thiết bị, phát hiện sự cố và giúp trong việc duy trì an toàn trong các môi trường nguy hiểm.
4. Nông nghiệp và Môi trường: Hình ảnh nhiệt cũng có thể được sử dụng trong nông nghiệp để giám sát cây trồng, phát hiện độ ẩm và nhu cầu tưới tiêu, cũng như giám sát môi trường tự nhiên.

## **Các phương pháp deep learning trong xử lý hình ảnh nhiệt**

* Tái cấu trúc hình ảnh: Sử dụng mạng neural để tái tạo hình ảnh nhiệt độ với độ phân giải cao hơn hoặc loại bỏ nhiễu, mờ.
* Phân loại và Nhận diện đối tượng: Sử dụng deep learning để phân loại và nhận diện đối tượng trong hình ảnh nhiệt độ.
* Phân tích chuỗi dữ liệu: Trong trường hợp video nhiệt, sử dụng mạng nơ-ron như Recurrent Neural Networks (RNNs) hoặc Long Short-Term Memory (LSTM) để phân tích thông tin từ các khung hình liên tiếp.
* Sử dụng mạng GAN: Generative Adversarial Networks (GANs) có thể được áp dụng để tạo ra hình ảnh nhiệt giả mạo có chất lượng cao.
* Lĩnh vực này đang tiếp tục phát triển để tận dụng những tiềm năng và ứng dụng rộng rãi của hình ảnh nhiệt, và deep learning chơi vai trò quan trọng trong việc tạo ra các mô hình thông minh và hiệu quả để xử lý dữ liệu này.
* Deep Learning là một nhánh của Machine Learning và Khoa học dữ liệu bắt chước cách con người thu được những kiến thức cụ thể, giúp máy tính tự huấn luyện chính nó thực hiện những tác vụ tương tự con người, hỗ trợ cho việc dịch ngôn ngữ, phân loại các hình ảnh, nhận dạng giọng nói.
* Deep Learning về cơ bản là một mạng nơ-ron gồm ba lớp trở lên, những mạng lưới nơ-ron này thu thập từ dữ liệu lớn (Big Data), học từ những dữ liệu đó và cố gắng “bắt chước” hành vi của não bộ con người. Công nghệ này hoạt động bằng cách khám phá các cấu trúc phức tạp trong dữ liệu mà chúng trải nghiệm, cụ thể là xây dựng các mô hình tính toán bao gồm nhiều lớp xử lý, mạng có thể tạo ra nhiều mức trừu tượng để biểu diễn dữ liệu.

## **Một số ưu điểm của Deep Learning**

1. Kiến trúc của mạng nơ-ron có khả năng thích ứng và có thể dễ dàng sửa đổi để giải quyết một loạt các vấn đề.
2. Có khả năng giải quyết nhiều vấn đề phức tạp với độ chính xác cao.
3. Mức độ tự động hóa, tự điều chỉnh và tự tối ưu hóa cao.
4. Có khả năng tính toán song song, hiệu suất cao và xử lý lượng lớn dữ liệu.

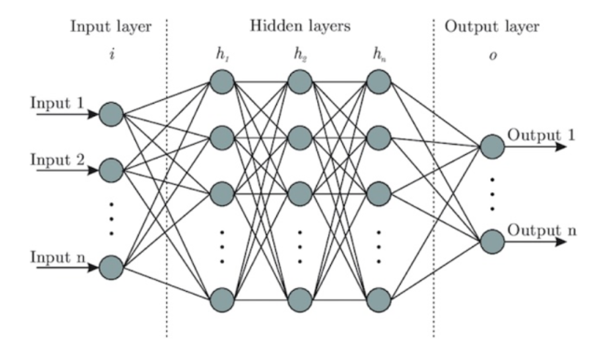
## **Nhược điểm**

1. Để khai thác hết khả năng của Deep Learning, cần phải có một lượng lớn dữ liệu.
2. Do phải xử lý nhiều mô hình phức tạp nên chi phí tính toán cao.
3. Không có nền tảng lý thuyết vững chắc để lựa chọn các công cụ tối ưu Deep Learning tốt nhất.

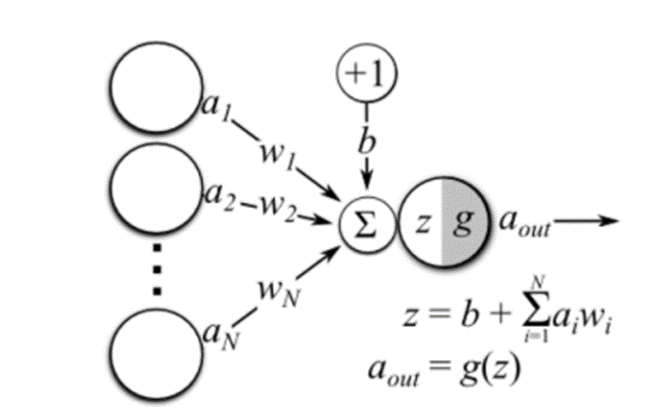
## **Giới thiệu về Neural Network**

Mạng thần kinh là mạng nhiều lớp bao gồm các nơ-ron hoặc các nút. Các nơ-ron này là đơn vị xử lý cốt lõi của mạng nơ-ron. Chúng được thiết kế để hoạt động giống như bộ não của con người. Chúng tiếp nhận dữ liệu, tự đào tạo để nhận ra các mẫu trong dữ liệu và sau đó dự đoán kết quả đầu ra.

* **Một mạng nơ-ron cơ bản có ba lớp:**
* Lớp đầu vào
* Lớp ẩn
* Lớp đầu ra



* **Hoạt động cơ bản của mạng nơ-ron như sau:**
* Hãy xem xét một hình ảnh, mỗi pixel được cung cấp làm đầu vào cho mỗi nơ-ron của lớp đầu tiên, các nơ-ron của một lớp được kết nối với các nơ-ron của lớp tiếp theo thông qua các kênh.
* Mỗi kênh này được gán một giá trị số được gọi là trọng số.
* Các đầu vào được nhân với trọng số tương ứng và tổng trọng số này sau đó được đưa vào làm đầu vào cho các lớp ẩn.
* Đầu ra từ các lớp ẩn được chuyển qua một hàm kích hoạt sẽ xác định xem liệu nơ-ron cụ thể có được kích hoạt hay không.
* Các nơ-ron được kích hoạt sẽ truyền dữ liệu đến các lớp ẩn tiếp theo. Theo cách này, dữ liệu được truyền qua mạng, đây được gọi là Truyền dẫn chuyển tiếp.
* Trong lớp đầu ra, nơ-ron có giá trị cao nhất sẽ dự đoán kết quả đầu ra. Các kết quả đầu ra này là các giá trị xác suất.
* Đầu ra dự đoán được so sánh với đầu ra thực tế để thu được lỗi. Thông tin này sau đó được chuyển trở lại thông qua mạng, quá trình này được gọi là Backpropagation.
* Dựa trên thông tin này, trọng số được điều chỉnh. Chu kỳ lan truyền tiến và lùi này được thực hiện nhiều lần trên nhiều đầu vào cho đến khi mạng dự đoán đầu ra chính xác trong hầu hết các trường hợp.
* Bước này kết thúc quá trình đào tạo của mạng nơ-ron. Trong một số trường hợp, thời gian dành cho việc đào tạo mạng nơ-ron có thể tăng cao.
* Trong hình ảnh dưới đây, ai là tập hợp các đầu vào, wi là trọng số, z là đầu ra và g là bất kỳ hàm kích hoạt nào.



## **Giới thiệu về GAN**

GAN thuộc nhóm generative model. Generative là tính từ nghĩa là khả năng sinh ra, model nghĩa là mô hình. Vậy hiểu đơn giản generative model nghĩa là mô hình có khả năng sinh ra dữ liệu. Hay nói cách khác, GAN là mô hình có khả năng sinh ra dữ liệu mới. Ví dụ như những ảnh mặt người ở dưới bạn thấy là do GAN sinh ra, không phải mặt người thật. Dữ liệu sinh ra nhìn như thật nhưng không phải thật.

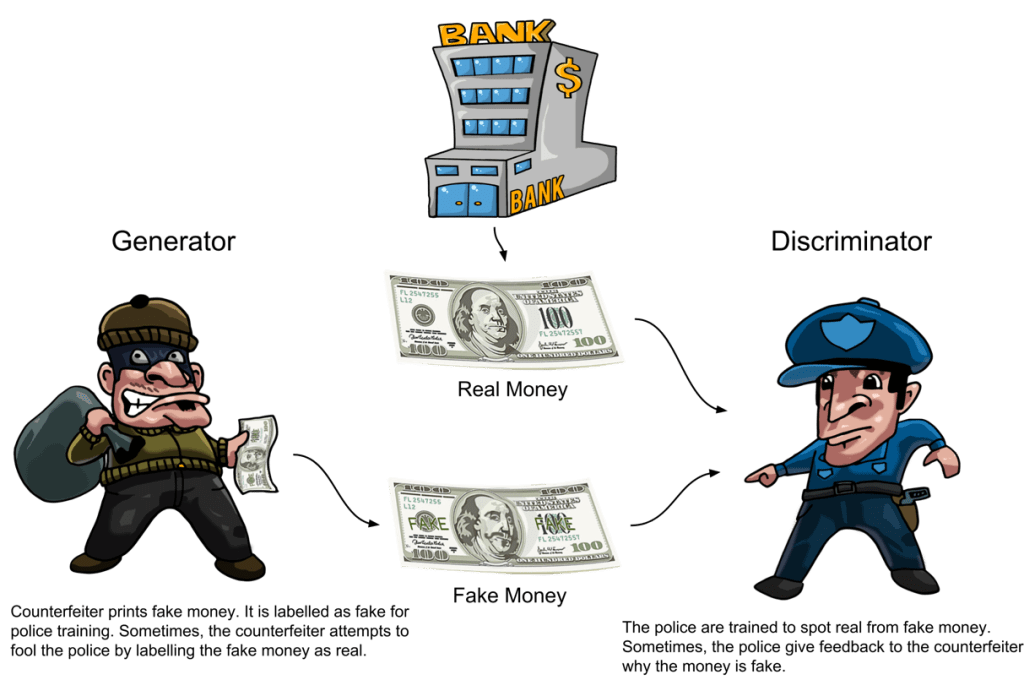


GAN viết tắt cho Generative Adversarial Networks. Generative giống như ở trên, Network có nghĩa là mạng (mô hình), còn Adversarial là đối nghịch. Tên gọi như vậy là do GAN được cấu thành từ 2 mạng gọi là Generator và Discriminator, luôn đối nghịch đầu với nhau trong quá trình train mạng GAN.

Tóm lại GAN là mạng để sinh dữ liệu mới giống với dữ liệu trong dataset có sẵn và có 2 mạng trong GAN là Generator và Discriminator.

## **Cấu trúc mạng GAN**

GAN cấu tạo gồm 2 mạng là Generator và Discriminator. Trong khi Generator sinh ra các dữ liệu giống như thật thì Discriminator cố gắng phân biệt đâu là dữ liệu được sinh ra từ Generator và đâu là dữ liệu thật có.

Minh họa các mạng trong GAN. Nguồn: <https://dzone.com/articles/working-principles-of-generative-adversarial-netwo>

Ví dụ bài toán giờ là dùng GAN để generate ra tiền giả mà có thể dùng để chi tiêu được. Dữ liệu có là tiền thật.

Generator giống như người làm tiền giả còn Discriminator giống như cảnh sát. Người làm tiền giả sẽ cố gắng làm ra tiền giả mà cảnh sát cũng không phân biệt được. Còn cảnh sát sẽ phân biệt đâu là tiền thật và đâu là tiền giả. Mục tiêu cuối cùng là người làm tiền giả sẽ làm ra tiền mà cảnh sát cũng không phân biệt được đâu là thật và đâu là giả và thế là mang tiền đi tiêu được.

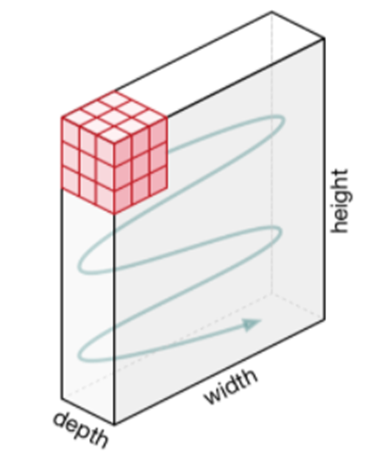
Trong quá trình train GAN thì cảnh sát có 2 việc: 1 là học cách phân biệt tiền nào là thật, tiền nào là giả, 2 là nói cho thằng làm tiền giả biết là tiền nó làm ra vẫn chưa qua mắt được và cần cải thiện hơn. Dần dần thì thằng làm tiền giả sẽ làm tiền giống tiền thật hơn và cảnh sát cũng thành thạo việc phân biệt tiền giả và tiền thật. Và mong đợi là tiền giả từ GAN sẽ đánh lừa được cảnh sát

## **Giới thiệu về thuật toán liên quan đến Deep Learning là CNN (Convolutional Neural Network – CNN).**

* Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một dạng tiến bộ của mạng nơ-ron nhân tạo, được thiết kế đặc biệt để xử lý và phân tích dữ liệu hình ảnh và video.
* CNN đã trở thành công cụ quan trọng trong lĩnh vực thị giác máy tính và xử lý hình ảnh, nơi nó có thể tự động học và trích xuất các đặc trưng từ dữ liệu hình ảnh, giúp giải quyết nhiều vấn đề, từ nhận diện vật thể đến phân loại hình ảnh.
* **Cấu trúc chính của CNN bao gồm các thành phần sau:**

1. Lớp Convolutional (Convolutional Layer)

CNN sử dụng lớp tích chập để áp dụng các bộ lọc hoặc kernel lên hình ảnh đầu vào để nhận diện các đặc trưng như cạnh, góc, màu sắc và các đặc điểm cục bộ khác.

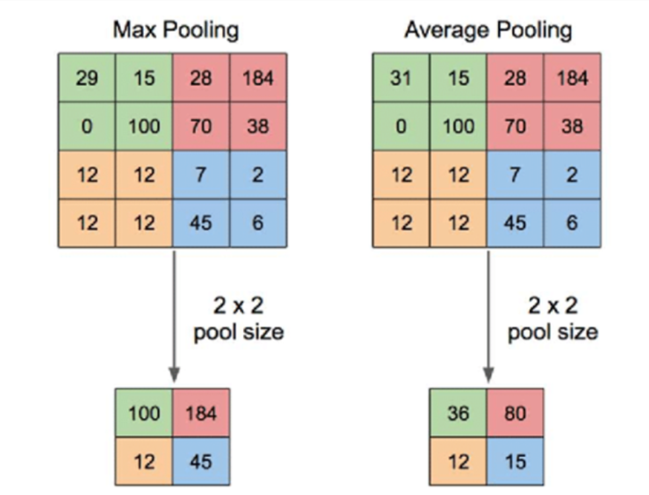


1. Lớp Activation (Activation Layer)

Sau mỗi lớp tích chập, một hàm kích hoạt như ReLU (Rectified Linear Activation) thường được áp dụng để tăng tính phi tuyến tính và kích hoạt các đặc trưng.

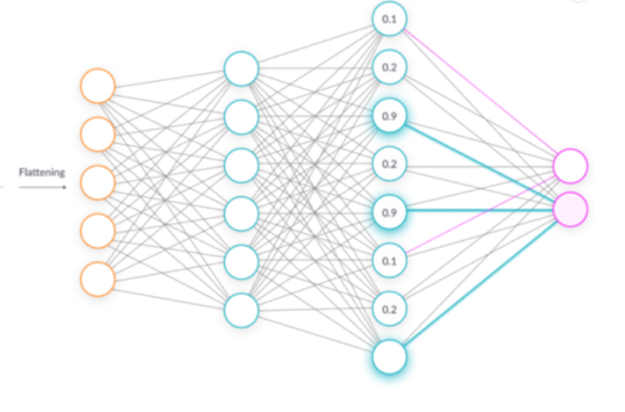
1. Lớp Pooling (Pooling Layer)

Lớp này thường được sử dụng sau các lớp tích chập để giảm kích thước của feature map và giảm chi phí tính toán. Max Pooling và Average Pooling là các phương pháp phổ biến được sử dụng trong lớp này.



1. Lớp Fully Connected (Fully Connected Layer)

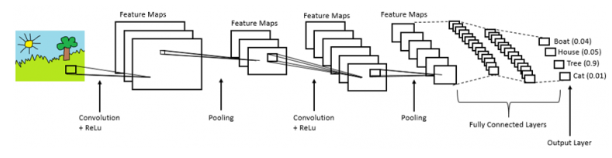
Lớp cuối cùng của CNN, thường là các lớp nơ-ron kết nối đầy đủ. Nó kết hợp các thông tin từ các feature map để thực hiện nhiệm vụ phân loại hoặc dự đoán

.

1. Lớp Dropout (Dropout Layer)

Đôi khi, lớp dropout được thêm vào để ngăn chặn việc overfitting bằng cách loại bỏ một số nơ-ron ngẫu nhiên trong quá trình huấn luyện.

* CNN chủ yếu được sử dụng để trích xuất đặc điểm từ hình ảnh với sự trợ giúp của các lớp. Phổ biến nhất trong các ứng dụng của CNN là phân loại hình ảnh, trong đó mỗi hình ảnh đầu vào được chuyển qua một loạt các lớp để nhận được giá trị xác suất từ ​​0 đến 1.

****

* CNN học cách phân biệt và tự động trích xuất các đặc trưng từ dữ liệu hình ảnh thông qua việc điều chỉnh trọng số của các bộ lọc trong quá trình huấn luyện. Quá trình này giúp mô hình hiểu được cấu trúc cấp cao của hình ảnh và dễ dàng nhận diện các đặc trưng phức tạp, như khuôn mặt, đối tượng, hoặc vật thể trong hình ảnh.
* CNN được áp dụng rộng rãi trong nhiều lĩnh vực như nhận diện vật thể, nhận diện khuôn mặt, xử lý ảnh y tế và nhiều ứng dụng khác trong thị giác máy tính.

# **CÔNG CỤ VÀ CÔNG NGHỆ CÓ LIÊN QUAN**

* Trong lĩnh vực xử lý hình ảnh nhiệt bằng deep learning, có một số phương pháp và thuật toán được áp dụng để cải thiện xử lý và phân tích dữ liệu hình ảnh nhiệt.

## **Một số phương pháp và thuật toán quan trọng:**

1. Super-Resolution (Tái cấu trúc siêu độ phân giải)

SRGAN (Super-Resolution Generative Adversarial Network): Một mô hình GAN được sử dụng để tái cấu trúc hình ảnh nhiệt với độ phân giải cao hơn, giúp cải thiện chất lượng và chi tiết của hình ảnh.

2. Deblurring (Làm mịn hình ảnh)

CNN-based Deblurring Models: Các mô hình CNN được sử dụng để giảm nhiễu và làm mịn hình ảnh nhiệt, giúp khắc phục vấn đề blurring do chuyển động hoặc nhiễu trong hình ảnh.

3. Object Detection (Phát hiện đối tượng)

Mask R-CNN: Một kiến trúc mạng CNN được sử dụng để phát hiện và phân loại đối tượng trong hình ảnh nhiệt. Nó có khả năng phát hiện và định vị các đối tượng với độ chính xác cao.

4. Image Restoration (Khôi phục hình ảnh)

Autoencoders: Mô hình autoencoder được sử dụng để khôi phục hình ảnh nhiệt từ dữ liệu gốc, giúp giảm nhiễu và tái tạo lại hình ảnh một cách chính xác.

5. Generative Adversarial Networks (GANs)

Deblur-SRRGAN: Một mô hình GAN đặc biệt được thiết kế để cùng lúc tái cấu trúc siêu độ phân giải và làm mịn hình ảnh nhiệt độ.

6. Video Processing (Xử lý video)

RNNs hoặc LSTM: Sử dụng các mạng nơ-ron tái phát để xử lý dữ liệu video nhiệt, giúp theo dõi và phân tích các chuỗi dữ liệu hình ảnh theo thời gian.

7. Multi-channel Conversion (Chuyển đổi đa kênh)

Phương pháp chuyển đổi từ hình ảnh nhiệt 1 kênh sang 3 kênh: Đôi khi, việc chuyển đổi dữ liệu hình ảnh nhiệt từ định dạng 1 kênh sang định dạng 3 kênh có thể cải thiện khả năng xử lý của mô hình.

## **Các công cụ và thư viện giúp xừ lý và giải quyết**

* + **Ngôn ngữ lập trình:**

- Python: Ngôn ngữ lập trình phổ biến cho deep learning và xử lý hình ảnh.

* + **Thư viện Deep Learning:**

- TensorFlow và Keras: TensorFlow là một thư viện mạnh mẽ cho deep learning, và Keras là một API dễ sử dụng được tích hợp trong TensorFlow giúp xây dựng mô hình nhanh chóng.

- PyTorch: Một thư viện deep learning linh hoạt, đặc biệt được ưa chuộng trong cộng đồng nghiên cứu.

* + **Thư viện Xử lý Hình ảnh:**

- OpenCV: Thư viện xử lý hình ảnh mạnh mẽ, hỗ trợ nhiều chức năng như xử lý và phân tích hình ảnh, video.

- Pillow: Thư viện hỗ trợ xử lý hình ảnh đơn giản, thường được sử dụng để đọc và hiển thị ảnh.

* + **Thư viện Xử lý Dữ liệu Nhiệt:**

- Flir Thermal SDK: Thư viện cung cấp các công cụ để xử lý và phân tích dữ liệu hình ảnh nhiệt từ các thiết bị của FLIR.

* + **Mô hình Pre-trained:**

- ImageNet Pre-trained Models: Có sẵn nhiều mô hình đã được huấn luyện trước trên ImageNet, có thể được fine-tune cho mục đích của bạn.

# **CÁC BÀI TOÁN LIÊN QUAN**

Trong mạng nơ-ron, mạng nơ-ron tích chập (ConvNets hay CNNs, Convolutional Neural Network) là một trong những phương pháp chính để thực hiện nhận dạng hình ảnh, phân loại hình ảnh. CNN được sử dụng rộng rãi trong một số lĩnh vực như phát hiện đối tượng, nhận dạng khuôn mặt

## **Convolutional Neural Network**

Như trình bày ở trên, Convolutional Neural Network là một trong những phương pháp chính khi sử dụng dữ liệu về ảnh. Kiến trúc mạng này xuất hiện do các phương pháp xử lý dữ liệu ảnh thường sử dụng giá trị của từng pixel. Vậy nên với một ảnh có giá trị kích thước 100x100 sử dụng kênh RGB ta có tổng cộng ta có 100 \* 100 \* 3 bằng 30000 nút ở lớp đầu vào. Điều đó kéo theo việc có một số lượng lớn weight và bias dẫn đến mạng nơ-ron trở nên quá đồ sộ, gây khó khăn cho việc tính toán. Hơn nữa, chúng ta có thể thấy rằng thông tin của các pixel thường chỉ chịu tác động bởi các pixel ngay gần nó, vậy nên việc bỏ qua một số nút ở tầng đầu vào trong mỗi lần huấn luyện sẽ không làm giảm độ chính xác của mô hình. Vậy nên người ta sử dụng cửa số tích chập nhằm giải quyết vấn đề số lượng tham số lớn mà vẫn trích xuất được đặc trưng của ảnh.

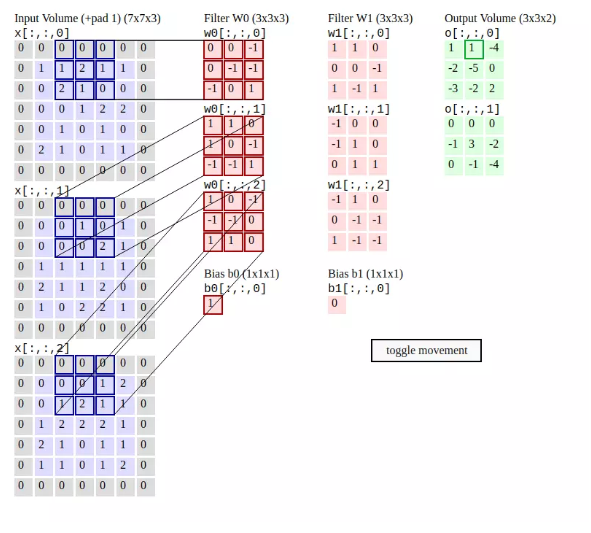
Về mặt kỹ thuật, trong mô hình học sâu CNN, mô hình ảnh đầu vào sẽ chuyển nó qua một loạt các lớp tích chập với các bộ lọc, sau đó đến lớp Pooling, rồi tiếp theo là các lớp được kết nối đầy đủ (FC — fully connected layers) và cuối cùng áp dụng hàm softmax để phân loại một đối tượng dựa trên giá trị xác suất trong khoản từ 0 đến 1.

## **Convolution Layer**

Convolution (lớp tích chập) là lớp đầu tiên trích xuất các đặc tính từ hình ảnh. Tham số lớp này bao gồm một tập hợp các bộ lọc có thể học được. Các bộ lọc đều nhỏ thường có kích cỡ hai chiều đầu tiên khoảng 3x3 hoặc 5x5, .... và có độ sâu bằng với độ sâu của đầu vào đầu vào. Bằng cách trượt dần bộ lọc theo chiều ngang và dọc trên ảnh, chúng thu được một Feature Map chứa các đặc trưng được trích xuất từ trên hình ảnh đầu vào.

**Quá trình trượt các bộ lọc thường có các giá trị được quy định bao gồm:**

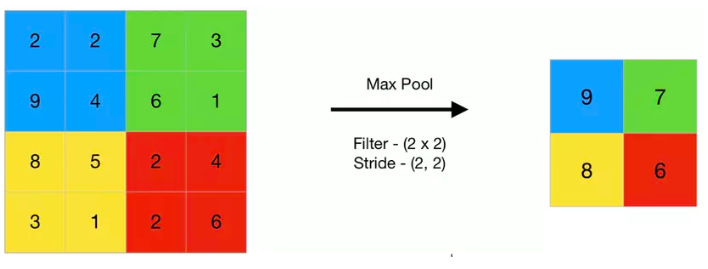
* padding: quy định bộ đệm của bộ lọc hay chính là phần màu xám được thêm vào ảnh
* stride: quy định bước nhảy trong quá trình thực hiện.



Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được nhiều thuộc tính của ảnh. Vì mỗi kernel cho ra output là 1 matrix nên k kernel sẽ cho ra k output matrix. Ta kết hợp k output matrix này lại thành 1 tensor 3 chiều có chiều sâu k. Output của convolutional layer sẽ qua hàm activation function trước khi trở thành input của convolutional layer tiếp theo.

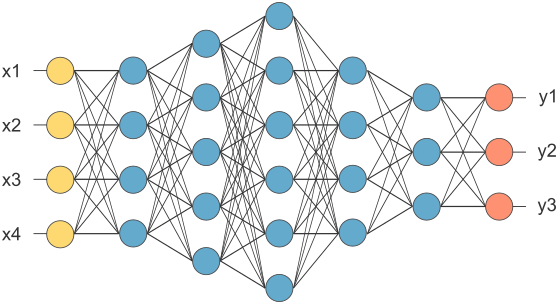
## **Pooling layer**

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model. Trong quá trình này, quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.



## **Fully connected layer**

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra.

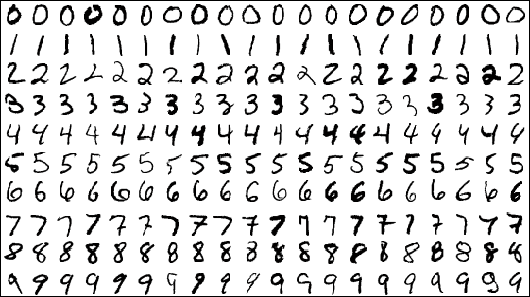


# **ÁP DỤNG VÀO BÀI TOÁN**

## **Mô hinh GAN**

### **Generator**

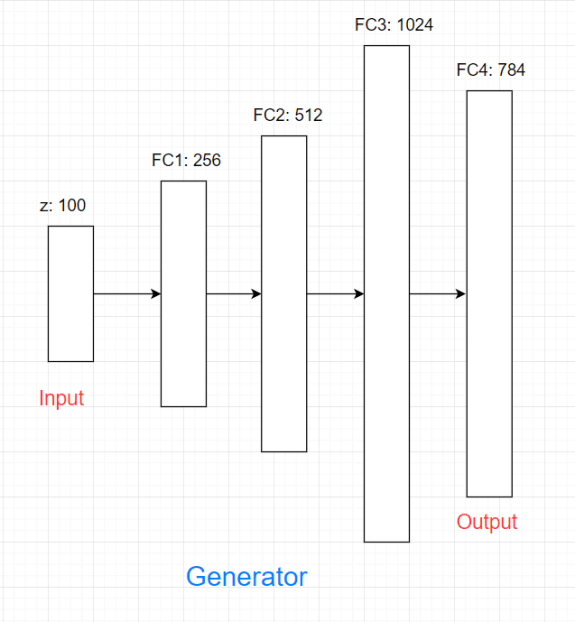
* Generator là mạng sinh ra dữ liệu, tức là sinh ra các chữ số giống với dữ liệu trong MNIST dataset. Generator có input là noise (random vector) là output là chữ số.
* Tại sao input là noise? Vì các chữ số khi viết ra không hoàn toàn giống nhau. Ví dụ số 0 ở hàng đầu tiên có rất nhiều biến dạng nhưng vẫn là số 0. Thế nên input của Generator là noise để khi ta thay đổi noise ngẫu nhiên thì Generator có thể sinh ra một biến dạng khác của chữ viết tay. Noise cho Generator thường được sinh ra từ normal distribution hoặc uniform distribution.



MNIST dataset. Nguồn: <https://syncedreview.com/2019/06/19/mnist-reborn-restored-and-expanded-additional-50k-training-samples/>

Input của Generator là noise vector 100 chiều.

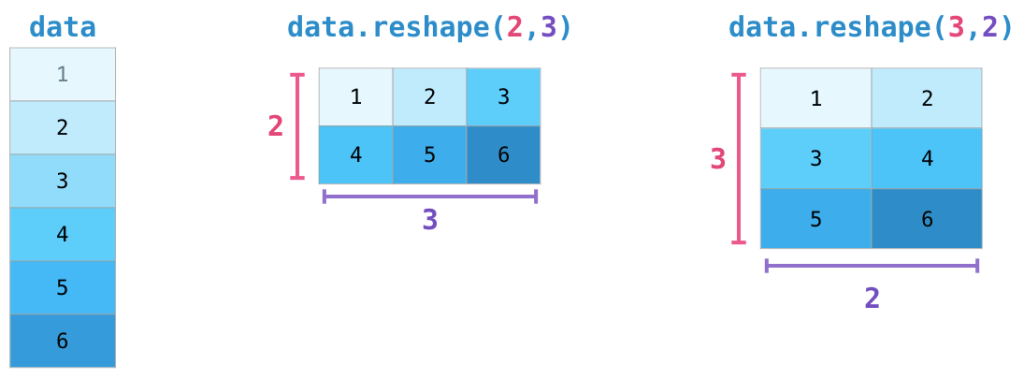
Sau đấy mô hình neural network được áp dụng với số node trong hidden layer lần lượt là 256, 512, 1024.



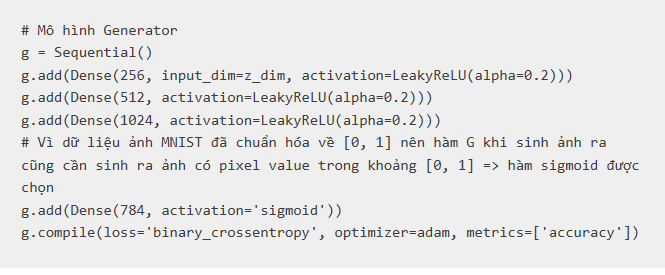
Mô hình generator

Output layer có số chiều là 784, vì output đầu ra là ảnh giống với dữ liệu MNIST, ảnh xám kích thước 28\*28 (784 pixel).

Output là vector kích thước 784\*1 sẽ được reshape về 28\*28 đúng định dạng của dữ liệu MNIST.



Ví dụ về reshape. Nguồn: <http://jalammar.github.io/visual-numpy/>

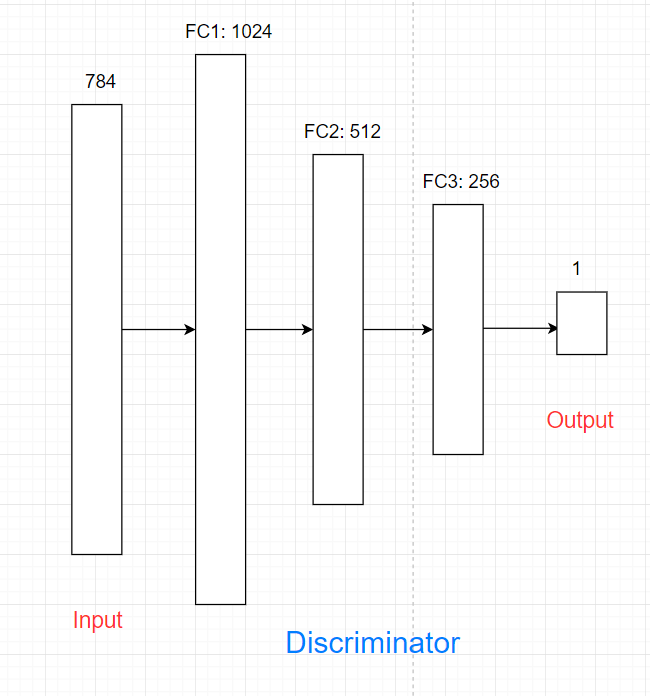


Nguồn: [Bài 1: Giới thiệu về GAN | Deep Learning cơ bản (nttuan8.com)](https://nttuan8.com/bai-1-gioi-thieu-ve-gan/)

### **Discriminator**

Discriminator là mạng để phân biệt xem dữ liệu là thật (dữ liệu từ dataset) hay giả (dữ liệu sinh ra từ Generator). Trong bài toán này thì discriminator dùng để phân biệt chữ số từ bộ MNIST và dữ liệu sinh ra từ Generator. Discriminator có input là ảnh biểu diễn bằng 784 chiều, output là ảnh thật hay ảnh giả.

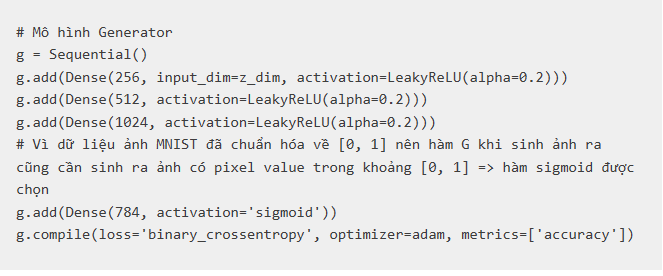
Đây là bài toán binary classification, giống với [logistic regression](https://nttuan8.com/bai-2-logistic-regression/).

Mô hình discriminator

Input của Discriminator là ảnh kích thước 784 chiều.

Sau đấy mô hình neural network được áp dụng với số node trong hidden layer lần lượt là 1024, 512, 256. Mô hình đối xứng lại với Generator.

Output là 1 node thể hiện xác suất ảnh input là ảnh thật, hàm sigmoid được sử dụng.



Nguồn: [Bài 1: Giới thiệu về GAN | Deep Learning cơ bản (nttuan8.com)](https://nttuan8.com/bai-1-gioi-thieu-ve-gan/)

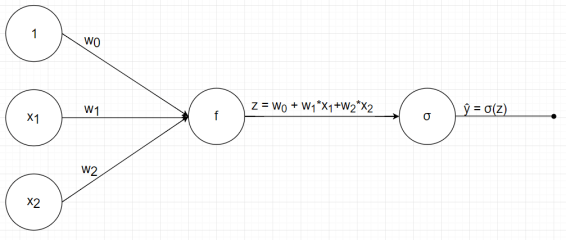
## **Mô hình neural network**

### **Logistic regression**

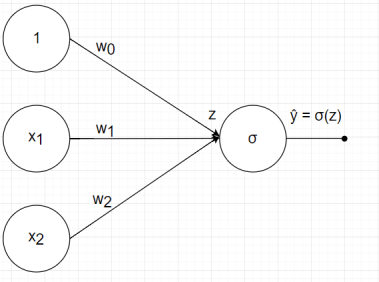
Logistic regression là mô hình neural network đơn giản nhất chỉ với input layer và output layer.

Mô hình của logistic regression từ bài trước là:  *y*^​=*σ*(*w*0​+*w*1​∗*x*1​+*w*2​∗*x*2​). Có 2 bước:

* Tính tổng linear:  2*z*=1∗*w*0​+*x*1​∗*w*1​+*x*2​∗*w*2​
* Áp dụng sigmoid function:  *y*^​=*σ*(*z*)



Để biểu diễn gọn lại ta sẽ gộp hai bước trên thành một trên biểu đồ

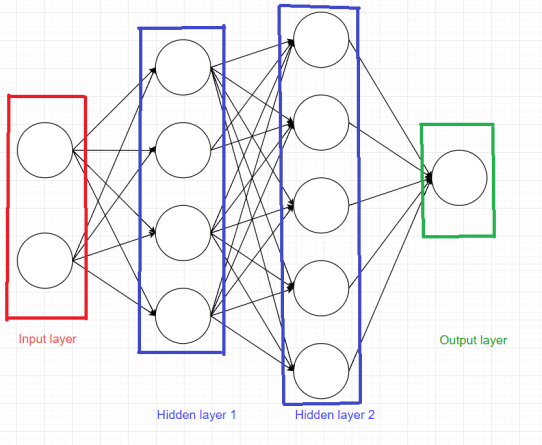


Mô hình logistic regresion

Hệ số *w*0​ được gọi là bias. Để ý từ những bài trước đến giờ dữ liệu khi tính toán luôn được thêm 1 để tính hệ số bias *w*0​ . Tại sao lại cần hệ số bias? Quay lại với bài 1, phương trình đường thẳng sẽ thế nào nếu bỏ *w*0​, phương trình giờ có dạng:  0*w*1​∗*x*+*w*2​∗*y*=0, sẽ luôn đi qua gốc tọa độ và nó không tổng quát hóa phương trình đường thẳng nên có thể không tìm được phương trình mong muốn. => Việc thêm bias (hệ số tự do) là **rất** quan trọng.

Hàm sigmoid ở đây được gọi là **activation function**.

### **Mô hình tổng quát**



Mô hình neural network

Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Các hình tròn được gọi là node.

Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer).

Ví dụ như ở hình trên có 1 input layer, 2 hidden layer và 1 output layer. Số lượng layer của mô hình là 3 layer.

Mỗi node trong hidden layer và output layer:

* Liên kết với tất cả các node ở layer trước đó với các hệ số w riêng.
* Mỗi node có 1 hệ số bias b riêng.
* Diễn ra 2 bước: tính tổng linear và áp dụng activation function.

### **Kí hiệu**

Số node trong hidden layer thứ i là *l*(*i*).

Ma trận *W*(*k*) kích thước *l*(*k*−1)∗*l*(*k*) là ma trận hệ số giữa layer (k-1) và layer k, trong đó *wij*(*k*)​ là hệ số kết nối từ node thứ i của layer k-1 đến node thứ j của layer k.

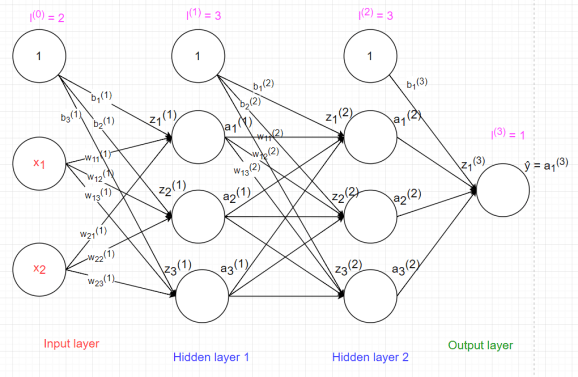
Vector *b*(*k*) kích thước *lk*∗1 là hệ số bias của các node trong layer k, trong đó *bi*(*k*)​ là bias của node thứ i trong layer k.

Với node thứ i trong layer l có bias *bi*(*l*)​thực hiện 2 bước:

* Tính tổng linear:  (*l*)​=∑*j*=1*l*(*l*−1)​*aj*(*l*−1)​∗*wji*(*l*)​+*bi*(*l*)​, là tổng tất cả các node trong layer trước nhân với hệ số w tương ứng, rồi cộng với bias b.
* Áp dụng activation function*ai*(*l*)​=*σ*(*zi*(*l*)​)

Vector (*k*) kích thước (*k*)∗1 là giá trị các node trong layer k sau bước tính tổng linear.

Vector *a*(*k*) kích thước *l*(*k*)∗1 là giá trị của các node trong layer k sau khi áp dụng hàm activation function.



Mô hình neural network trên gồm 3 layer. Input layer có 2 node (2*l*(0)=2), hidden layer 1 có 3 node, hidden layer 2 có 3 node và output layer có 1 node.

Do mỗi node trong hidden layer và output layer đều có bias nên trong input layer và hidden layer cần thêm node 1 để tính bias (nhưng không tính vào tổng số node layer có).

Tại node thứ 2 ở layer 1, ta có:

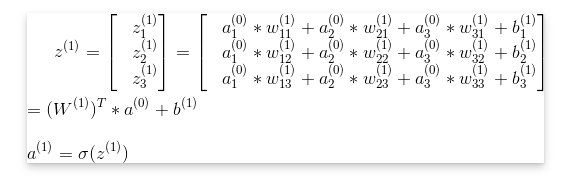
* *z*2(1)​=*x*1​∗*w*12(1)​+*x*2​∗*w*22(1)​+*b*2(1)​
* *a*2(1)​=*σ*(*z*2(1)​)

Hay ở node thứ 3 layer 2, ta có:

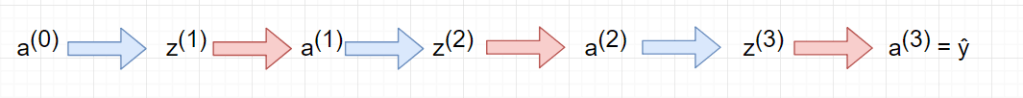
* *z*3(2)​=*a*1(1)​∗*w*13(2)​+*a*2(1)​∗*w*23(2)​+*a*3(1)​∗*w*33(2)​+*b*3(2)​
* *a*3(2)​=*σ*(*z*3(2)​)

## **Feedforward**

Để nhất quán về mặt ký hiệu, gọi input layer *a*(0)(=*x*) kích thước 2\*1.



Tương tự ta có:

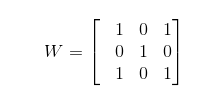
*z*(2)=(*W*(2))*T*∗*a*(1)+*b*(2)*a*(2)=*σ*(*z*(2))*z*(3)=(*W*(3))*T*∗*a*(2)+*b*(3)*y*^​=*a*(3)=*σ*(*z*(3))Feedforward

## **Phép tính convolution**

### **Convolution**

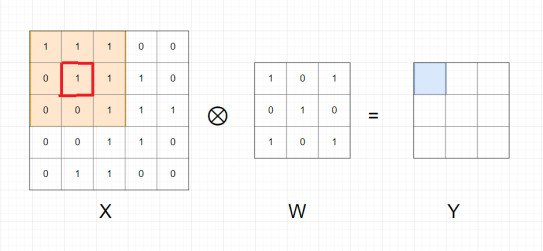
Để cho dễ hình dung mình sẽ lấy ví dụ trên ảnh xám, tức là ảnh được biểu diễn dưới dạng ma trận A kích thước m\*n.

Ta định nghĩa **kernel**là một ma trận vuông kích thước k\*k trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9,… Ví dụ kernel kích thước 3\*3



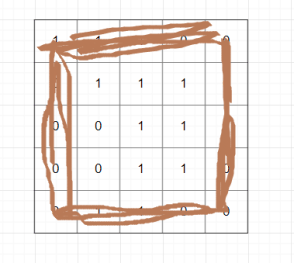
Kí hiệu phép tính convolution (⊗), kí hiệu *Y*=*X*⊗*W*

Với mỗi phần tử *xij*​ trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử *xij*​ làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính element-wise của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.



Ví dụ khi tính tại *x*22 (ô khoanh đỏ trong hình), ma trận A cùng kích thước với W, có *x*22​ làm trung tâm có màu nền da cam như trong hình. Sau đó tính y11= *sum*(*A*⊗*W*)=*x*11​∗*w*11​+*x*12​∗*w*12​+*x*13​∗*w*13​+*x*21​∗*w*21​+*x*22​∗*w*22​+*x*23​∗*w*23​+*x*31​∗*w*31​+*x*32​∗*w*32​+*x*33​∗*w*33​=4. Và làm tương tự với các phần tử còn lại trong ma trận.

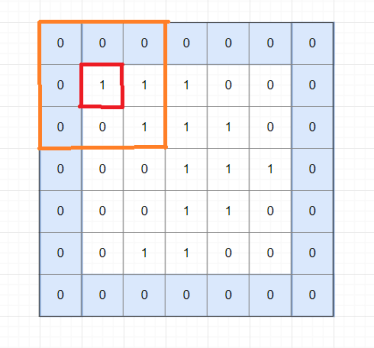
Thế thì sẽ xử lý thế nào với phần tử ở viền ngoài như *x*11​? Bình thường khi tính thì sẽ bỏ qua các phần tử ở viền ngoài, vì không tìm được ma trận A ở trong X.



Nên bạn để ý thấy ma trận Y có kích thước nhỏ hơn ma trận X. Kích thước của ma trận Y là (m-k+1) \* (n-k+1).

### **Padding**

Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên giờ ta muốn ma trận Y thu được có kích thước bằng ma trận X => Tìm cách giải quyết cho các phần tử ở viền => Thêm giá trị 0 ở viền ngoài ma trận X.



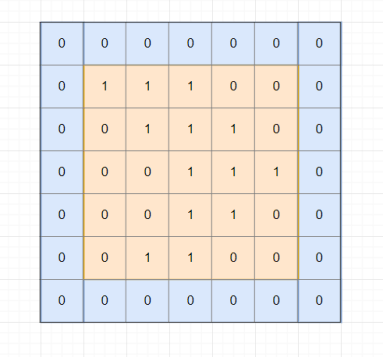
Ma trận X khi thêm viền 0 bên ngoài

Rõ ràng là giờ đã giải quyết được vấn đề tìm A cho phần tử *x*11​ , và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu.

Phép tính này gọi là convolution với **padding=1**. Padding=k nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

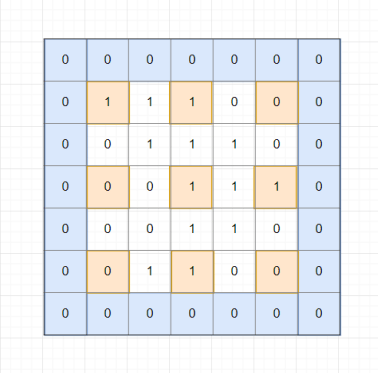
### **Stride**

Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X, thu được ma trận Y cùng kích thước ma trận X, ta gọi là stride=1.



stride=1, padding=1

Tuy nhiên nếu **stride=k** (k > 1) thì ta chỉ thực hiện phép tính convolution trên các phần tử *x*1+*i*∗*k*,1+*j*∗*k*​. Ví dụ k = 2.



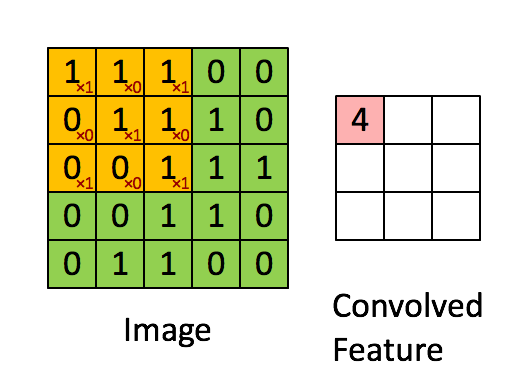
padding=1, stride=2

Hiểu đơn giản là bắt đầu từ vị trí *x*11​ sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X.

Kích thước của ma trận Y là 3\*3 đã giảm đi đáng kể so với ma trận X.

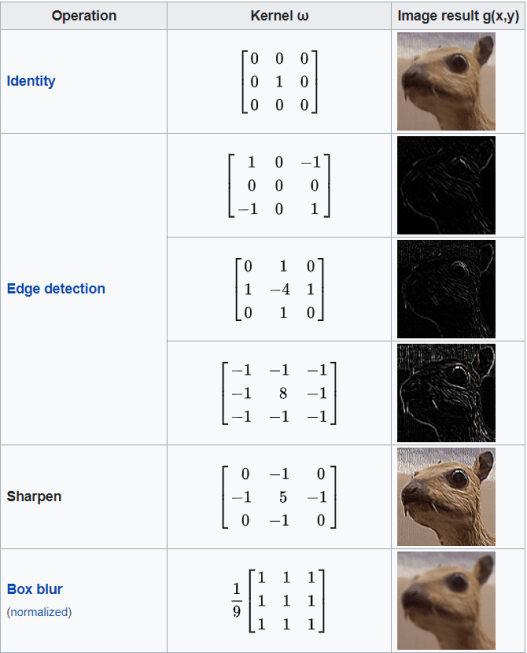
Công thức tổng quát cho phép tính convolution của ma trận X kích thước m\*n với kernel kích thước k\*k, stride = s, padding = p ra ma trận Y kích thước (*sm*−*k*+2*p*​+1)∗(*sn*−*k*+2*p*​+1).

Stride thường dùng để giảm kích thước của ma trận sau phép tính convolution.

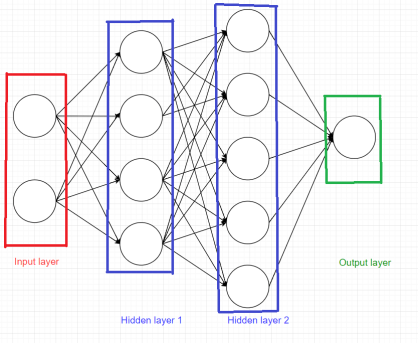


### **Ý nghĩa của phép tính convolution**

Mục đích của phép tính convolution trên ảnh là làm mở, làm nét ảnh; xác định các đường;… Mỗi kernel khác nhau thì sẽ phép tính convolution sẽ có ý nghĩa khác nhau. Ví dụ:



* **Convolutional-Neural-Network(CNN)**
* **Convolutional layer**

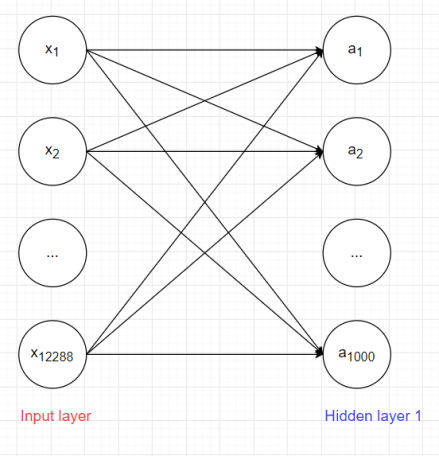


Mô hình neural network.

Mỗi hidden layer được gọi là **fully connected layer**, tên gọi theo đúng ý nghĩa, mỗi node trong hidden layer được kết nối với tất cả các node trong layer trước. Cả mô hình được gọi là **fully connected neural network (FCN)**.

#### **Vấn đề của fully connected neural network với xử lý ảnh**

Như vấn đề trên về xử lý ảnh, thì ảnh màu 64\*64 được biểu diễn dưới dạng 1 tensor 64\*64\*3. Nên để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel (64\*64\*3 = 12288). Nghĩa là input layer giờ có 12288 nodes.



Input layer và hidden layer 1

Giả sử số lượng node trong hidden layer 1 là 1000. Số lượng weight W giữa input layer và hidden layer 1 là 12288\*1000 = 12288000, số lượng bias là 1000 => tổng số parameter là: 12289000. Đấy mới chỉ là số parameter giữa input layer và hidden layer 1, trong model còn nhiều layer nữa, và nếu kích thước ảnh tăng, ví dụ 512\*512 thì số lượng parameter tăng cực kì nhanh => Cần giải pháp tốt hơn !!!

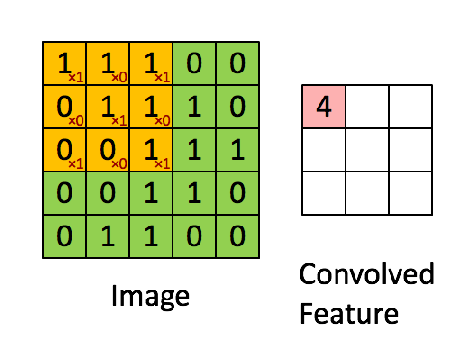
Nhận xét:

* Trong ảnh các pixel ở cạnh nhau thường có **liên kết với nhau hơn** là những pixel ở xa. Ví dụ như phép tính convolution trên ảnh ở bài trước. Để tìm các đường trong ảnh, ta áp dụng sobel kernel trên mỗi vùng kích thước 3\*3. Hay làm nét ảnh ta áp dụng sharpen kernel cũng trên vùng có kích thước 3\*3.
* Trong phép tính convolution trong ảnh, chỉ 1 kernel được dùng trên toàn bộ bức ảnh. Hay nói cách khác là các pixel ảnh **chia sẻ** hệ số với nhau.

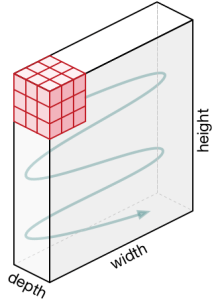
=> Áp dụng phép tính convolution vào layer trong neural network ta có thể giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.

#### **Convolutional layer đầu tiên**

Bài trước phép tính convolution thực hiện trên ảnh xám với biểu diễn ảnh dạng ma trận

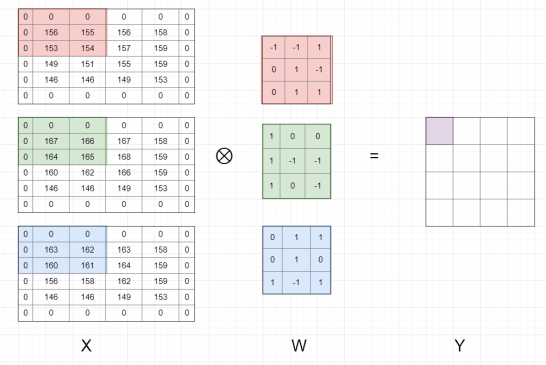


Tuy nhiên ảnh màu có tới 3 channels red, green, blue nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Nên ta cũng sẽ định nghĩa kernel là 1 tensor 3 chiều kích thước k\*k\*3.



Phép tính convolution trên ảnh màu với k=3.

Ta định nghĩa kernel có cùng độ sâu (depth) với biểu diễn ảnh, rồi sau đó thực hiện di chuyển khối kernel tương tự như khi thực hiện trên ảnh xám.



Tensor X, W 3 chiều được viết dưới dạng 3 matrix.

Khi biểu diễn ma trận ta cần 2 chỉ số hàng và cột: i và j, thì khi biểu diễn ở dạng tensor 3 chiều cần thêm chỉ số độ sâu k. Nên chỉ số mỗi phần tử trong tensor là *xijk*​.

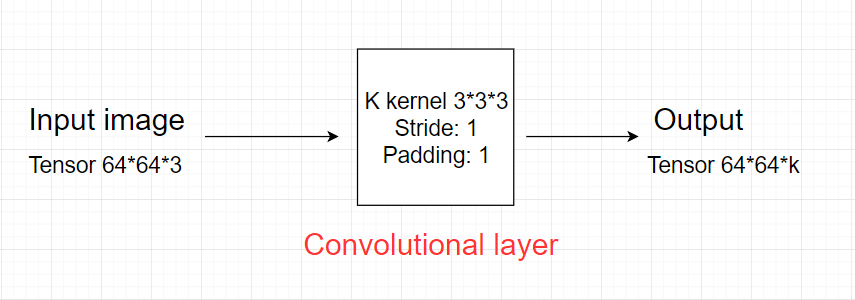
​*b*+(*x*111​∗*w*111​+*x*121​∗*w*121​+*x*131​∗*w*131​+*x*211​∗*w*211​+*x*221​∗*w*221​+*x*231​∗*w*231​+*x*311​∗*w*311​+*x*321​∗*w*321​+*x*331​∗*w*331​)+(*x*112​∗*w*112​+*x*122​∗*w*122​+*x*132​∗*w*132​+*x*212​∗*w*212​+*x*222​∗*w*222​+*x*232​∗*w*232​+*x*312​∗*w*312​+*x*322​∗*w*322​+*x*332​∗*w*332​)+(*x*113​∗*w*113​+*x*123​∗*w*123​+*x*133​∗*w*133​+*x*213​∗*w*213​+*x*223​∗*w*223​+*x*233​∗*w*233​+*x*313​∗*w*313​+*x*323​∗*w*323​+*x*333​∗*w*333​)=−25

Nhận xét:

* Output Y của phép tính convolution trên ảnh màu là 1 matrix.
* Có 1 **hệ số bias** được cộng vào sau bước tính tổng các phần tử của phép tính element-wise

**Các quy tắc đối với**[padding](https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/#Padding)**và**[stride](https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/#Stride)**toàn hoàn tương tự phần số 2**

Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được **nhiều thuộc tính** của ảnh. Vì mỗi kernel cho ra output là 1 matrix nên k kernel sẽ cho ra k output matrix. Ta kết hợp k output matrix này lại thành 1 tensor 3 chiều có chiều sâu k.

Convolutional layer đầu tiên

Output của convolutional layer đầu tiên sẽ thành input của convolutional layer tiếp theo.

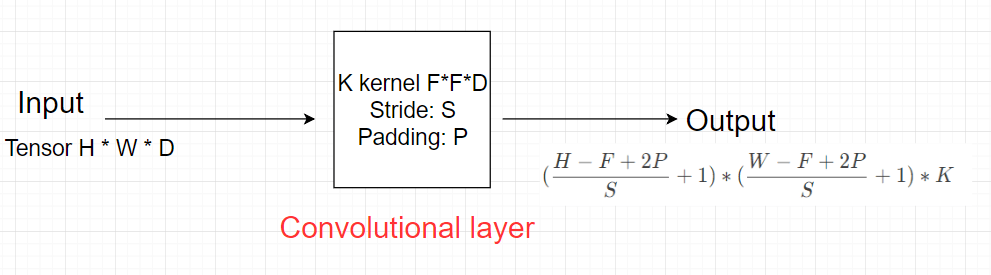
#### Convolutional layer tổng quát

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H \* W \* D.

Kernel có kích thước F \* F \* D (kernel luôn có depth bằng depth của input và F là số lẻ), stride: S, padding: P.

Convolutional layer áp dụng K kernel.

=> Output của layer là tensor 3 chiều có kích thước (*SH*−*F*+2*P*​+1)∗(*SW*−*F*+2*P*​+1)∗*K*



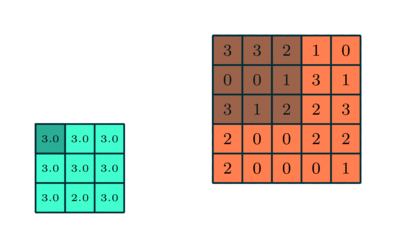
Lưu ý:

* Output của convolutional layer sẽ qua hàm **activation function** trước khi trở thành input của convolutional layer tiếp theo.
* Tổng số parameter của layer: Mỗi kernel có kích thước F\*F\*D và có 1 hệ số bias, nên tổng parameter của 1 kernel là F\*F\*D + 1. Mà convolutional layer áp dụng K kernel => Tổng số parameter trong layer này là K \* (F\*F\*D + 1).

### Pooling layer

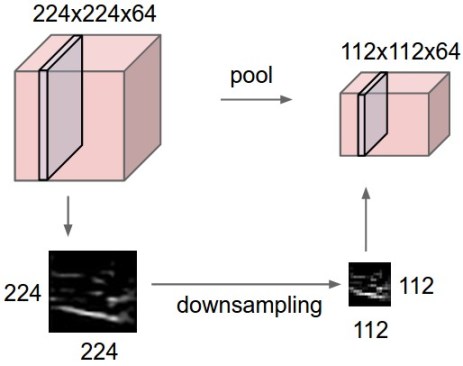
Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

Gọi pooling size kích thước K\*K. Input của pooling layer có kích thước H\*W\*D, ta tách ra làm D ma trận kích thước H\*W. Với mỗi ma trận, trên vùng kích thước K\*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

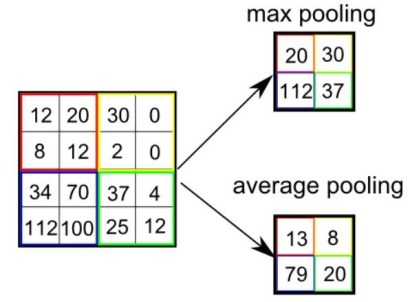


max pooling layer với size=(3,3), stride=1, padding=0

Nhưng **hầu hế**t khi dùng pooling layer thì sẽ dùng size=(2,2), stride=2, padding=0. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên .

  
Sau pooling layer (2\*2).

Có 2 loại pooling layer phổ biến là: max pooling và average pooling.

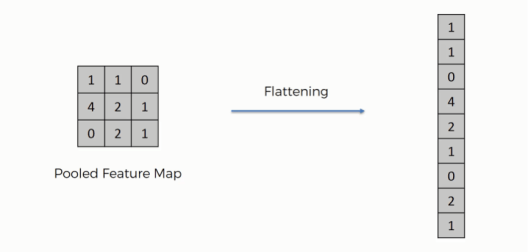


Ví dụ về pooling layer

**Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.**

### **Fully connected layer**

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,…) thì tensor của output của layer cuối cùng, kích thước H\*W\*D, sẽ được chuyển về 1 vector kích thước (H\*W\*D)

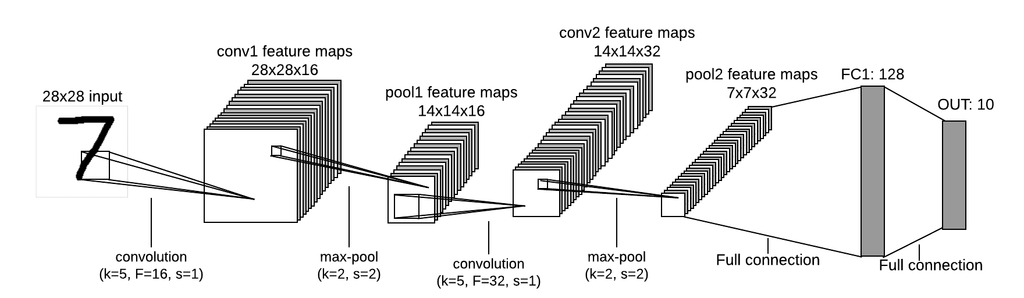


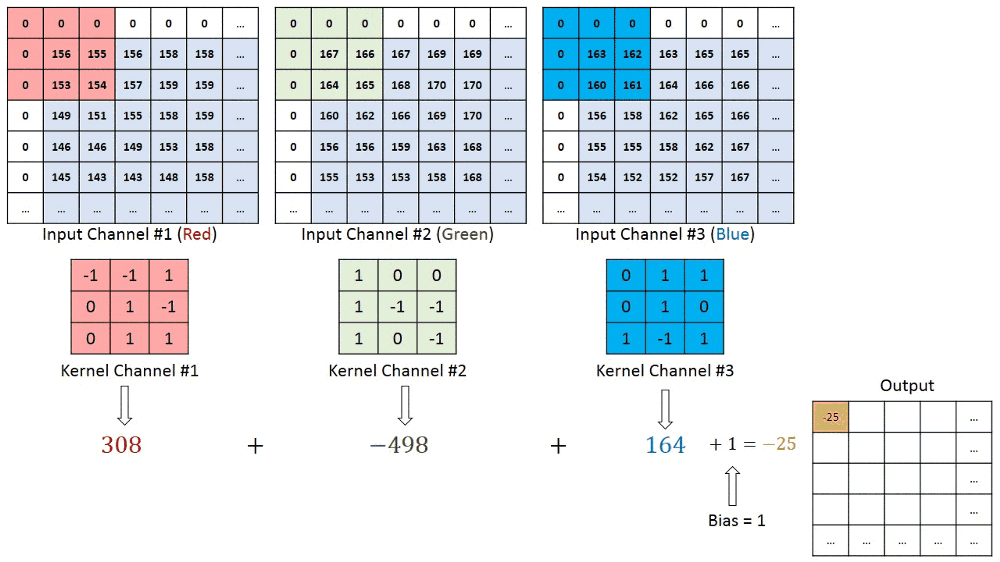
Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.

### **Visualise convolutional neural network**

Mô hình convolutional neural network:

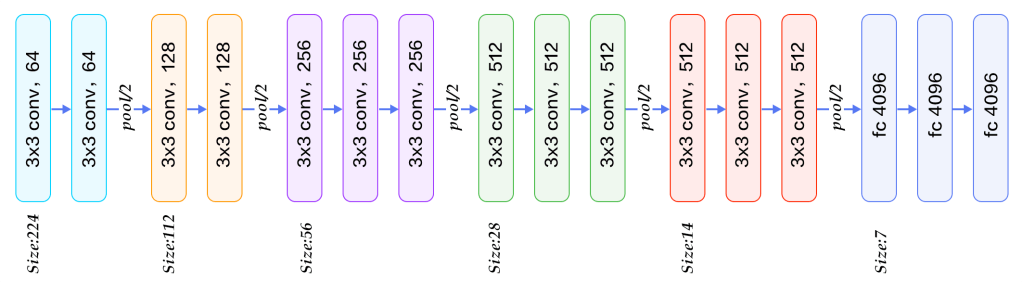
Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.



Thực hiện phép tính convolution trên ảnh màu

## **Mạng VGG 16**

VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu [ImageNet](http://www.image-net.org/)gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ áp dụng kiến thức ở trên để phân tích mạng VGG 16.

Kiến trúc vgg16  
conv: convolutional layer, pool: pooling layer, fc: fully connected layer

Phân tích:

* Convolutional layer: kích thước 3\*3, padding=1, stride=1. Tại sao không ghi stride, padding mà vẫn biết? Vì mặc định sẽ là stride=1 và padding để cho output cùng width và height với input.
* Pool/2 : max pooling layer với size 2\*2
* 3\*3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
* Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
* Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

# **KẾT LUẬN VÀ CÔNG VIỆC TƯƠNG LAI**

**Hiểu:**

Hiểu sâu hơn vể cơ bản lí thuyết của Deep learning và thuật toán convolutional-neural-network(CNN) và áp dụng trong việc xử lí các hình ảnh có màu cũng như cấu tạo của mô hình neural-network(NN)

**Hướng phát triển**

* Trong tương lai, mô hình Convolutional Neural Networks (CNNs) và deep learning dự kiến sẽ tiếp tục đóng vai trò quan trọng và có những tiến bộ đáng kể trong nhiều lĩnh vực khác nhau. Dưới đây là một số kết luận về tương lai của chúng:

1. Mở Rộng Ứng Dụng:CNNs và deep learning sẽ tiếp tục mở rộng và có ứng dụng rộng rãi trong các lĩnh vực như y học, ô tô tự hành, thị giác máy tính, xử lý ngôn ngữ tự nhiên, và nhiều lĩnh vực khác.

2. Cải Tiến Mô Hình:Mô hình CNNs sẽ phát triển với các cải tiến mới như mô hình tự học, mạng nơ-ron chất lượng cao hơn, và khả năng học tập đa nhiệm giữa nhiều tác vụ.

3. Học Sâu và Tự Học:Deep learning sẽ tiếp tục chú trọng vào việc hiểu sâu hơn về cơ chế hoạt động của mô hình và khả năng tự học của chúng để tạo ra những mô hình hiệu quả hơn và phức tạp hơn.

4. Tăng Cường Hiệu Suất và Hiệu Quả:Tiến bộ trong phần cứng và các thuật toán sẽ cải thiện hiệu suất và hiệu quả của deep learning, giúp tăng tốc quá trình huấn luyện và triển khai các mô hình lớn.

5. Xử Lý Dữ Liệu Lớn và Thời Gian Thực:Deep learning sẽ tiếp tục phát triển để xử lý các tập dữ liệu lớn và thời gian thực, mở ra nhiều cơ hội mới trong việc phân tích và đưa ra dự đoán từ dữ liệu phức tạp.

6. Tiềm Năng và Thách Thức:Mặc dù có tiềm năng lớn, việc xử lý các vấn đề về tính minh bạch, đạo đức và bảo mật dữ liệu cũng là những thách thức mà deep learning sẽ phải đối mặt.

* Trong tương lai, lĩnh vực deep learning và CNNs (Convolutional Neural Networks) dự kiến sẽ phát triển theo nhiều hướng khác nhau để cải thiện hiệu suất, ứng dụng và tiềm năng. Dưới đây là một số hướng phát triển chính:

1.Mô hình Deep Learning Chất Lượng Cao Hơn:

Mô Hình Tự Học (Self-Supervised Learning): Phát triển các mô hình có khả năng tự học từ dữ liệu không giám sát để cải thiện việc trích xuất đặc trưng và học biểu diễn.

2.Mô Hình Tích Hợp Thêm Kiến Thức Từ Người Chuyên Gia (Expert-in-the-Loop): Kết hợp thông tin từ người chuyên gia để cải thiện hiệu suất của mô hình và giảm độ phức tạp trong việc huấn luyện.

3.Mở Rộng Ứng Dụng:Tăng Cường Trong Y Học và Y Sinh Học: Sử dụng deep learning để phát triển công nghệ y học tiên tiến, từ chuẩn đoán bệnh bị covid 19

# **TÀI LIỆU THAM KHẢO**

[1] L. St-Laurent, D. Prévost, and X. Maldague, ‘‘Thermal imaging for

enhanced foreground-background segmentation,’’ in Proc. Int. Conf.

Quant. Infr. Thermogr., Padova, Italy, Jun. 2006, pp. 1–10.

[2] B. Oswald-Tranta, ‘‘Temperature reconstruction of infrared images with

motion deblurring,’’ J. Sensors Sensor Syst., vol. 7, no. 1, pp. 13–20,

Jan. 2018.

[3] FLIR Systems. (2020). FLIR Tau 2. [Online]. Available:

<https://www.flir.com/products/tau-2/>

[4] G. Batchuluun, Y. W. Lee, D. T. Nguyen, T. D. Pham, and K. R. Park,

‘‘Thermal image reconstruction using deep learning,’’ IEEE Access, vol. 8,

pp. 126839–126858, 2020.

[5] G. Batchuluun, H. S. Yoon, D. T. Nguyen, T. D. Pham, and K. R. Park,

‘‘A study on the elimination of thermal reflections,’’ IEEE Access, vol. 7,

pp. 174597–174611, 2019.

[6] Digital Media Lab. (2020). Dongguk Single Model Both for Thermal

Image Super-Resolution Reconstruction and Deblurring, and Detec-

tion Model of Object and Thermal Reflection. [Online]. Available:

<http://dm.dgu.edu/link.html>

[7] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, ‘‘Fast and robust

multiframe super resolution,’’ IEEE Trans. Image Process., vol. 13, no. 10,

pp. 1327–1344, Oct. 2004.

[8] Y. Matsushita, H. Kawasaki, S. Ono, and K. Ikeuchi, ‘‘Simultaneous deblur

and super-resolution technique for video sequence captured by hand-held

video camera,’’ in Proc. IEEE Int. Conf. Image Process. (ICIP), Paris,

France, Oct. 2014, pp. 4562–4566.

[9] L. He, G. Li, and J. Liu, ‘‘Joint motion deblurring and superresolution from

single blurry image,’’ Math. Problems Eng., vol. 2015, pp. 1–10, Jan. 2015.

[10] H. Park and K. M. Lee, ‘‘Joint estimation of camera pose, depth, deblur-

ring, and super-resolution from a blurred image sequence,’’ Sep. 2017,

arXiv:1709.05745. [Online]. Available: http://arxiv.org/abs/1709.05745

[11] B. Bascle, A. Blake, and A. Zisserman, ‘‘Motion deblurring and super-

resolution from an image sequence,’’ in Proc. Eur. Conf. Comput. Vis.,

Cambridge, U.K., Apr. 1996, pp. 571–582.

[12] B. Du, X. Ren, and J. Ren, ‘‘CNN-based image super-resolution and

deblurring,’’ in Proc. Int. Conf. Video, Signal Image Process. Wuhan,

China: Association for Computing Machinery, Oct. 2019, pp. 70–74.

[13] X. Zhang, F. Wang, H. Dong, and Y. Guo, ‘‘A deep encoder-decoder

networks for joint deblurring and super-resolution,’’ in Proc. IEEE Int.

Conf. Acoust., Speech Signal Process. (ICASSP), Calgary, AB, Canada,

Apr. 2018, pp. 1448–1452.

[14] X. Zhang, H. Dong, Z. Hu, W.-S. Lai, F. Wang, and M.-H. Yang, ‘‘Gated

fusion network for joint image deblurring and super-resolution,’’ Jul. 2018,

arXiv:1807.10806. [Online]. Available: http://arxiv.org/abs/1807.10806

[15] F. Albluwi, V. A. Krylov, and R. Dahyot, ‘‘Image deblurring and super-

resolution using deep convolutional neural networks,’’ in Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP), Aalborg, Denmark,

Sep. 2018, pp. 1–6.

[16] J. U. Yun and I. K. Park, ‘‘Joint face super-resolution and deblurring using a generative adversarial network,’’ Dec. 2019, arXiv:1912.10427. [Online].

Available: <http://arxiv.org/abs/1912.10427>

[17] F. Vankawala, A. Ganatra, and A. Patel, ‘‘A survey on different image

deblurring techniques,’’ Int. J. Comput. Appl., vol. 116, no. 13, pp. 15–18,

Apr. 2015.

[18] S. F. Hamood, M. S. M. Rahim, O. Farook, and D. Kasmuni, ‘‘A survey

on various image deblurring methods,’’ J. Eng. Appl. Sci., vol. 11, no. 3,

pp. 561–569, 2016.

[19] D. Singh and R. K. Sahu, ‘‘A survey on various image deblurring

techniques,’’ Int. J. Adv. Res. Comput. Commun. Eng., vol. 2, no. 12,

pp. 4736–4739, 2013.

[20] N. Patel and K. N. Jariwala, ‘‘A survey on image enhancement using image

super-resolution and deblurring methods,’’ J. Emerg. Technol. Innov. Res.,

vol. 1, no. 5, pp. 359–370, 2014.

[21] S. Sahu, M. K. Lenka, and P. K. Sa, ‘‘Blind deblurring using deep

learning: A survey,’’ Jul. 2019, arXiv:1907.10128. [Online]. Available:

<http://arxiv.org/abs/1907.10128>

[22] K. Nasrollahi and T. B. Moeslund, ‘‘Super-resolution: A comprehensive

survey,’’ Mach. Vis. Appl., vol. 25, no. 6, pp. 1423–1468, Aug. 2014.

[23] Z. Wang, J. Chen, and S. C. H. Hoi, ‘‘Deep learning for image super-

resolution: A survey,’’ Feb. 2019, arXiv:1902.06068. [Online]. Available:

http://arxiv.org/abs/1902.06068

[24] S. Anwar, S. Khan, and N. Barnes, ‘‘A deep journey into super-

resolution: A survey,’’ Mar. 2019, arXiv:1904.07523. [Online]. Available:

<http://arxiv.org/abs/1904.07523>

[25] S. H. Umale and A. M. Sahu, ‘‘A review on various techniques for

image deblurring,’’ Int. J. Comput. Sci. Mobile Comput., vol. 3, no. 4,

pp. 263–268, 2014.

[26] M. M. Sada and M. M. Goyani, ‘‘Image deblurring techniques—A detail

review,’’ Int. J. Sci. Res. Sci. Eng. Technol., vol. 4, pp. 176–188, Jan. 2018.

[27] R. Wang and D. Tao, ‘‘Recent progress in image deblurring,’’ Sep. 2014,

arXiv:1409.6838. [Online]. Available: http://arxiv.org/abs/1409.6838

[28] W. Yang, X. Zhang, Y. Tian, W. Wang, and J.-H. Xue, ‘‘Deep learn-

ing for single image super-resolution: A brief review,’’ Jul. 2018,

arXiv:1808.03344. [Online]. Available: <http://arxiv.org/abs/1808.03344>

[29] S. Borman and R. L. Stevenson, ‘‘Super-resolution from image

sequences—A review,’’ in Proc. Midwest Symp. Circuits Syst.,

Notre Dame, IN, USA, Aug. 1998, pp. 374–378.

[30] J. W. Davis and V. Sharma, ‘‘Background-subtraction using contour-based

fusion of thermal and visible imagery,’’ Comput. Vis. Image Understand.,

vol. 106, nos. 2–3, pp. 162–182, May 2007.

[31] J. W. Davis and V. Sharma, ‘‘Robust detection of people in thermal

imagery,’’ in Proc. 17th Int. Conf. Pattern Recognit., Cambridge, U.K.,

Aug. 2004, pp. 713–716.

[32] W. K. Wong, H. L. Lim, C. K. Loo, and W. S. Lim, ‘‘Home alone

faint detection surveillance system using thermal camera,’’ in Proc. 2nd

Int. Conf. Comput. Res. Develop., Kuala Lumpur, Malaysia, May 2010,

pp. 747–751.

[33] P. Kumar, A. Mittal, and P. Kumar, ‘‘Fusion of thermal infrared and visible

spectrum video for robust surveillance,’’ in Proc. Int. Conf. Comput. Vis.,

Graph. Image Process., Madurai, India, Dec. 2006, pp. 528–539.

[34] D. Gangodkar, P. Kumar, and A. Mittal, ‘‘Segmentation of moving objects

in visible and thermal videos,’’ in Proc. Int. Conf. Comput. Commun.

Informat., Coimbatore, India, Jan. 2012, pp. 1–5.

[35] J. Lee, J.-S. Choi, E. Jeon, Y. Kim, T. Le, K. Shin, H. Lee, and K. Park,

‘‘Robust pedestrian detection by combining visible and thermal infrared

cameras,’’ Sensors, vol. 15, no. 5, pp. 10580–10615, May 2015.

[36] E. S. Jeon, J. H. Kim, H. G. Hong, G. Batchuluun, and K. R. Park, ‘‘Human

detection based on the generation of a background image and fuzzy system

by using a thermal camera,’’ Sensors, vol. 16, no. 4, pp. 1–31, 2016.

[37] P. Kumar, A. Mittal, and P. Kumar, ‘‘Study of robust and intelligent

surveillance in visible and multimodal framework,’’ Informatica, vol. 32,

pp. 63–77, Apr. 2008.

[38] G. Batchuluun, N. R. Baek, D. T. Nguyen, T. D. Pham, and K. R. Park,

‘‘Region-based removal of thermal reflection using pruned fully convolu-

tional network,’’ IEEE Access, vol. 8, pp. 75741–75760, 2020.

[39] G. Batchuluun, Y. G. Kim, J. H. Kim, H. G. Hong, and K. R. Park, ‘‘Robust behavior recognition in intelligent surveillance environments,’’ Sensors, vol. 16, no. 7, pp. 1–23, 2016.

[40] G. Batchuluun, J. H. Kim, H. G. Hong, J. K. Kang, and K. R. Park,

‘‘Fuzzy system based human behavior recognition by combining behavior

prediction and recognition,’’ Expert Syst. Appl., vol. 81, pp. 108–133,

Sep. 2017.

[41] G. Batchuluun, H. S. Yoon, J. K. Kang, and K. R. Park, ‘‘Gait-based humanidentification by combining shallow convolutional neural network-stacked

long short-term memory and deep convolutional neural network,’’ IEEE

Access, vol. 6, pp. 63164–63186, 2018.

[42] G. Batchuluun, R. A. Naqvi, W. Kim, and K. R. Park, ‘‘Body-movement-

based human identification using convolutional neural network,’’ Expert

Syst. Appl., vol. 101, pp. 56–77, Jul. 2018.

[43] G. Batchuluun, D. T. Nguyen, T. D. Pham, C. Park, and K. R. Park,

‘‘Action recognition from thermal videos,’’ IEEE Access, vol. 7,

pp. 103893–103917, Aug. 2019.

[44] Mathworks. (2020). Colormap. [Online]. Available:

<https://www.mathworks.com/help/matlab/ref/jet.html>

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, ‘‘Focal loss for

dense object detection,’’ in Proc. IEEE Int. Conf. Comput. Vis., Venice,

Italy, Oct. 2017, pp. 2999–3007.

[46] K. He, G. Gkioxari, P. Dollár, and R. Girshick, ‘‘Mask R-CNN,’’ Mar. 2017, arXiv:1703.06870. [Online]. Available: <http://arxiv.org/abs/1703.06870>

[47] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image

recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR),

Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[48] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie,

‘‘Feature pyramid networks for object detection,’’ in Proc. IEEE Conf.

Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017,

pp. 936–944.

[49] J. Long, E. Shelhamer, and T. Darrell, ‘‘Fully convolutional networks for

semantic segmentation,’’ Mar. 2014, arXiv:1411.4038. [Online]. Avail-

able: http://arxiv.org/abs/1411.4038

[50] S. Ren, K. He, R. Girshick, and J. Sun, ‘‘Faster R-CNN: Towards

real-time object detection with region proposal networks,’’ Jan. 2015,

arXiv:1506.01497. [Online]. Available: <http://arxiv.org/abs/1506.01497>

[51] Digital Media Lab. (2020). Dongguk Thermal Image Database (DTh-DB)

and Dongguk Items & Vehicles Database (DI&V-DB). [Online]. Available:

<http://dm.dgu.edu/link.html>

[52] R. Gade and T. B. Moeslund, ‘‘Constrained multi-target tracking for team

sports activities,’’ IPSJ Trans. Comput. Vis. Appl., vol. 10, no. 1, pp. 1–11,

Dec. 2018.

[53] J. Davis and M. Keck, ‘‘A two-stage approach to person detection in

thermal imagery,’’ in Proc. Workshop Appl. Comput. Vis., Breckenridge,

CO, USA, Jan. 2005, pp. 6–111.

[54] R. Miezianko, ‘‘Terravic research infrared database-terravic motion

infrared database,’’ in Proc. IEEE OTCBVS WS Ser. Bench, to be published.

[55] R. Miezianko, ‘‘Terravic research infrared database-terravic weapon

infrared database,’’ in Proc. IEEE OTCBVS WS Ser. Bench, to be published.

[56] R. Bergevin, P.-L. St-Charles, and G.-A. Bilodeau, ‘‘Mutual foreground

segmentation with multispectral stereo pairs,’’ in Proc. IEEE Int. Conf.

Comput. Vis. Workshops (ICCVW), Venice, Italy, Oct. 2017, pp. 375–384.

[57] K. Van Beeck, K. Van Engeland, J. Vennekens, and T. Goedeme, ‘‘Abnor-

mal behavior detection in LWIR surveillance of railway platforms,’’ in

Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS), Lecce,

Italy, Aug./Sep. 2017, pp. 1–6.

[58] D. Tan, K. Huang, S. Yu, and T. Tan, ‘‘Efficient night gait recognition based on template matching,’’ in Proc. 18th Int. Conf. Pattern Recognit. (ICPR),

Hong Kong, Aug. 2006, pp. 1000–1003.

[59] Z. Wu, N. Fuller, D. Theriault, and M. Betke, ‘‘A thermal infrared video

benchmark for visual analysis,’’ in Proc. IEEE Conf. Comput. Vis. Pattern

Recognit. Workshops, Columbus, OH, USA, Jun. 2014, pp. 201–208.

[60] (2020). ASL Datasets. [Online]. Available: https://projects.asl.

ethz.ch/datasets/doku.php?id=ir:iricra2014.q11

[61] NVIDIA Corporation. (2019). NVIDIA Titan X. [Online]. Available:

https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/

[62] Keras. (2019). Keras: The Python Deep Learning Library. [Online]. Avail-

able: https://keras.io/

[63] OpenCV. (2019). OpenCV: Open Source Computer Vision. [Online]. Avail-

able: http://opencv.org/

[64] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta,

A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, ‘‘Photo-realistic single

image super-resolution using a generative adversarial network,’’ in Proc.

IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA,

Jul. 2017, pp. 105–114.

[65] C. Dong, C. C. Loy, K. He, and X. Tang, ‘‘Image super-resolution using

deep convolutional networks,’’ IEEE Trans. Pattern Anal. Mach. Intell.,

vol. 38, no. 2, pp. 295–307, Feb. 2016.