ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
CHƯƠNG TRÌNH KỸ SƯ CHẤT LƯỢNG CAO VIỆT PHÁP
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG

LUẬN VĂN TỐT NGHIỆP

HỆ THỐNG GIÁM SÁT VIỆC SỬ DỤNG THIẾT BỊ BẢO HỘ CÁ NHÂN ỨNG DỤNG MẠNG HỌC SÂU

NGUYỄN THÁI SƠN - 1512847

GIẢNG VIÊN HƯỚNG DẪN:
PGS. TS. HÀ HOÀNG KHA

Thành phố Hồ Chí Minh, tháng 6 năm 2020

LỜI CẢM ƠN

Khoảng thời gian học tập và rèn luyện tại **Trường Đại học Bách Khoa Thành phố Hồ Chí Minh** đã trang bị cho em rất nhiều kiến thức hữu ích và cần thiết về chuyên môn và xã hội để em có thể trở thành một công dân tốt và một kỹ sư có năng lực. Con đường học tập ở đại học trong suốt 5 năm vừa qua là không hề dễ dàng với muôn vàn thử thách và khó khăn. Để vượt qua những rào cản ấy, bên cạnh sự cố gắng của bản thân em còn là sự ủng hộ và giúp đỡ tận tình của quý **Thầy Cô**, **Gia đình** và **Bạn bè**. Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến quý **Thầy Cô Trường Đại học Bách Khoa Thành phố Hồ Chí Minh**, quý **Thầy Cô Khoa Điện – Điện Tử**, những người đã đi cùng với tri thức và tâm huyết truyền đạt vốn kiến thức quý báu của mình cho chúng em.

Em muốn giành riêng lời cảm ơn đặc biệt cho Thầy hướng dẫn của mình – PGS. TS. Hà Hoàng Kha – người đã luôn tận tình hướng dẫn và giúp đỡ em trong quá trình thực hiện luận văn này.

Ngoài ra, em cũng muốn giành cho gia đình và bạn bè lời cảm ơn chân thành và đặc biệt là ba mẹ và ông bà em, những người đã luôn là chỗ dựa tinh thần vững chắc cho em trong nhưng thời điểm khó khăn.

Trong quá trình thực hiện luận văn chắc chắn không thể tránh khỏi những sai sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô để em có thể học hỏi thêm nhưng điều tốt đẹp và hoàn thiện luận văn của mình. Sau cùng, em xin kính chúc quý **Thầy Cô Trường Đại học Bách Khoa Thành phố Hồ Chí Minh** và **Thầy Hà Hoàng Kha** dồi dào sức khỏe, đạt được nhiều thành công trong cuộc sống và luôn giữ vững niềm đam mê nghiên cứu và giảng dạy để có thể tiếp tục truyền lửa tri thức cho những thế hệ sau.

Thành phố Hồ Chí Minh, tháng 6 năm 2020

Nguyễn Thái Sơn

TÓM TẮT LUẬN VĂN

Ý tưởng về việc kết hợp trí tuệ nhân tạo và thị giác máy tính để ứng dụng vào các bài toán giám sát đã xuất hiện từ lâu. Tuy nhiên chỉ đến những năm gần đây khi các thiết bị phần cứng có thể đáp ứng được yêu cầu về tính kĩ thuật và kinh tế thì các ứng dụng sử dụng các công nghệ này mới dần trở nên phổ biến. Một trong những ứng dụng đang rất được quan tâm là đảm bảo an toàn lao động của công nhân xây dựng thông qua một hệ thống giám sát sử dụng camera và trí tuệ nhân tạo để theo dõi việc sử dụng các thiết bị bảo hộ lao động của những người làm việc trong công trường.

Trong khuôn khổ của luận văn này, hệ thống giám sát việc sử dụng thiết bị bảo hộ cá nhân sẽ tập trung vào ba thiết bị thường gặp trong công trường xây dựng: mũ cứng, áo dạ quang bảo hộ và khẩu trang. Hệ thống sử dụng phương pháp phát hiện và phân loại vật thể YOLO, được xây dựng trên cơ sở mạng tích chập - CNN. Khi hoạt động, hệ thống sẽ có thể phát hiện và phân loại việc sử dụng các thiết bị bảo hộ lao động của công nhân ở công trường. Nếu phát hiện một trường hợp không sử dụng thiết bị bảo hộ lao động đang được theo dõi thì hệ thống sẽ gửi cảnh báo cho quan sát viên để kiểm tra và nhắc nhở. Việc này sẽ hỗ trợ rất nhiều trong công tác đảm bảo an toàn lao động trong xây dựng.

Hệ thống được viết bằng ngôn ngữ Python, mô hình máy học YOLOv3 và thư viện thị giác máy tính OpenCV.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách từ viết tắt

MRC Maximal Ratio Combining

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Ngành xây dựng luôn được coi là một trong những ngành ẩn chưa nhiều rủi ro về tai nạn lao động và khả năng mắc các bệnh nghề nghiệp. Trên thực tế nhiều vụ tai nạn nghiêm trọng đã xảy ra, lấy đi sinh mang hoặc để lại những thương tật nặng nề cho người lao động khiến họ mất khả năng làm việc, sinh hoạt như người bình thương. Kéo theo đó là nỗi đau về tinh thần và gánh nặng về kinh tế cho những thành viên trong gia đình người bị nạn. Do đó vấn đề đảm bảo an toàn vệ sinh lao động luôn là một trong những vấn đề được quan tâm hàng đầu trong ngành xây dựng.

Một trong những nguyên nhân chính gây ra những vụ tai nạn thương tâm là việc người lao động không sử dụng trang thiết bị bảo hộ cá nhân trong quá trình lao động. Vấn đề này không chỉ xuất phát từ sự chủ quan của cá nhân người lao động mà còn ở sự thiếu sót, lỏng lẻo trong quá trình giám sát công trình của nhà thầu và người sử dụng lao động. Đối với người lao động, những điều kiện khắc nghiệt của môi trường làm việc như nhiệt độ ngoài trời cao hay thường xuyên phải vận động mạnh khiến đổ mồ hôi liên tục đã khiến họ chấp nhận đánh đổi sự an toàn của bản thân để đổi lấy sự thoải mái. Còn đối với những người giám sát công trình, họ không thể bao quát được toàn bộ quá trình làm việc tại các nơi làm việc khác nhau, do đó không thể nhắc nhở người lao động kịp thời trước khi xảy ra những tai nan mà hậu quả là có thể tránh khỏi hoặc được giảm nhẹ nếu người lao động có sử dụng trang thiết bị bảo hộ cá nhân.

Để tăng cường năng lực thực hiên đảm bảo an toàn vệ sinh lao động ở các

công trình, nhiều chủ đầu tư và nhà thầu đã tiến hành lắp đặt các hệ thống camera giám sát quá trình làm việc. Các hệ thống này giúp giám sát viên có thể quan sát nhiều vị trí một lúc mà không cần phải di chuyển qua các đia điểm khác nhau trong công trình, giảm thiểu chi phí và thời gian thực hiên các công tác an toàn. Tuy nhiên, khi số lượng các khu vực cần quan sát tăng lên hoặc những người chịu trách nhiệm quan sát không tập trung vào nhiệm vụ thì việc giám sát thông qua màn hình dễ xảy ra sai sót. Việc tích hợp công nghê AI vào các hệ thống giám sát sẽ là sẽ tăng thêm độ tin cậy cho công tác đảm bảo an toàn, giảm thiểu những sai sót không đáng có.

1.2 Mục tiêu nghiên cứu

Mục tiêu của luận văn là xây dựng, đánh giá một hệ thống nhận diện việc sử dụng thiết bị bảo hộ cá nhân của người lao động trong công trường. Khi phát hiện ra các trường hợp không sử dụng các trang thiết bị bảo hộ thì hệ thống sẽ đưa ra cảnh báo.

1.3 Phạm vi nghiên cứu

Phạm vi của luận văn là tiến hành nhận dạng trên các video trích xuất từ camera. Mô hình nhận diện được huấn luyện sử dụng framework được xây dựng sẵn. Tập dữ liệu sử dụng để huấn luyện và đánh giá mô hình nhận diện được thu thập và dán nhãn bởi người làm luận văn. Một phần hình ảnh trong tập dữ liệu này có nguồn gốc từ các tập dữ liệu khác nhưng không sử dụng lại các nhãn của các tập dữ liệu đó. Các thiết bị bảo hộ cá nhân được tích hợp trong hệ thống gồm: mũ cứng, áo bảo hộ và khẩu trang.

1.4 Phương pháp nghiên cứu

Các giai đoạn trong quá trình nghiên cứu và hoàn thiện luận văn:

1. Tìm hiểu các mô hình nhận diện đang được nghiên cứu và sử dung.
2. Chọn mô hình phù hợp, tìm hiểu lý thuyết.
3. Xây dựng tập dữ liệu phù hợp cho bài toán đặt ra.

4. Huấn luyện mô hình nhận diện.

5. Đánh giá các tham số hiệu năng của mô hình nhận diên đã xây dựng.

6. Xây dựng hệ thống sử dụng mô hình nhận diện để đưa ra cảnh báo.

7. Đánh giá khả năng hoạt động của hệ thống.

Trong đó các giai đoạn 3, 4 và 5 được thực hiện luân phiên và nhiều lần để cải thiện hiệu năng của mô hình nhận diện. YOLOv3 được chọn để làm mô hình nhận diện vì đây là một trong số các bộ nhận diện có hiệu năng cao trong thời gian thực, đã được sử dụng và đánh giá trên nhiều tập dữ liệu khác nhau.

1.5 Cấu trúc luận văn

Luận văn này bao gồm 5 chương. Chương 1 là chương mở đầu, giới thiệu bao quát về vấn đề, mục tiêu, phạm vi và phương pháp nghiên cứu luận văn. Chương 2 sẽ cung cấp những lý thuyết về các khái niệm được sử dụng trong luận văn. Chương 3 cho người đọc biết về cách tập dữ liệu được xây dựng, cách mô hình YOLOv3 được huấn luyện sử dụng framework darknet và cách xây dựng hệ thống sử dụng mô hình nhận diện. Chương 4 sẽ chứa những thông số đánh giá hiệu năng của bộ nhận diện và hệ thống trong các trường hợp khác nhau. Cuối cùng, trong chương 5 sẽ là các nhận xét về các kết quả đạt được và kết luận.

Chương 2

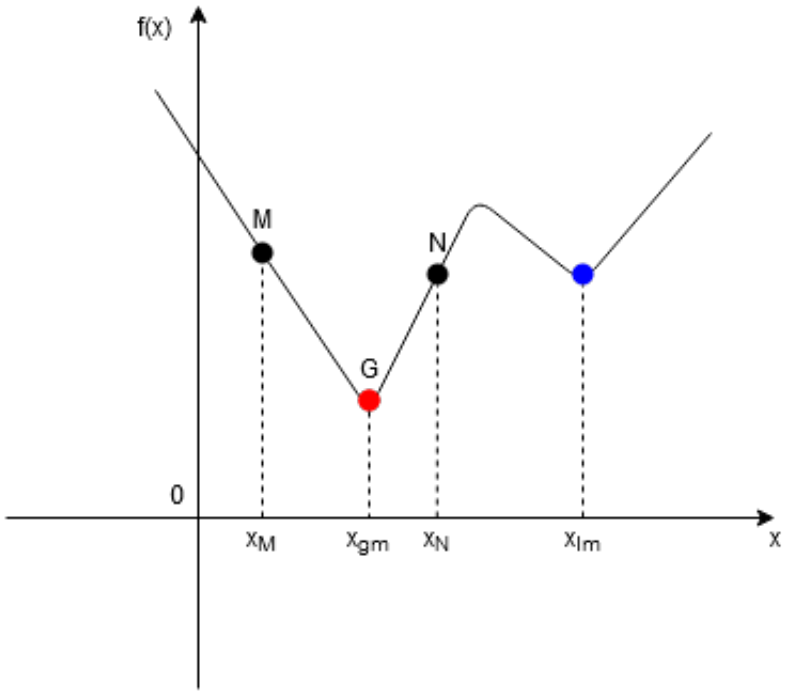
Cơ sở lý thuyết

2.1 Gradient Descent

Phần lớn các mô hình máy học được xây dựng dựa trên việc tối ưu hóa hàm mất mát hay nói cách khác là tìm cực tiểu của một hàm số biết trước. Đối với các hàm số đơn giản, việc xác đinh các điểm cực tiểu có thể được giải quyết thông qua việc tính toán đạo hàm cấp 1 và cấp 2. Tuy nhiên hàm mất mát của các mô hình máy học hay học sâu thường có số chiều lớn và đạo hàm phức tạp, do đó khó có thể áp dụng các phương pháp truyền thống để tìm các giá trị cực tiểu. Thay vào đó các mô hình này sử dụng giải thuật Gradient Descent để tìm các điểm cực tiểu của hàm mất mát.

Một hàm số có thể có nhiều điểm cực tiểu địa phương (local minimum) và cực tiểu toàn cục (global minimum). Ta có thể thấy trên hình 2.1 là đồ thị của một hàm số đơn biến, điểm màu xanh là cực tiểu địa phương, điểm màu đỏ là cực tiểu toàn cục. Giả sử ta có hai điểm M tại x_M và N tại x_N trên đồ thị hình 2.1, ta gọi điểm cực tiểu toàn cục là G. Lúc này ta muốn đưa điểm M và N về xấp xỉ hoặc trùng với vị trí của G bằng giải thuật Gradient Descent. Ta nhận thấy M nằm bên trái G và $f^{'}(x_M) < 0$, nếu M muốn di chuyển về phía G thì $x_{M_{k+1}} = x_{M_k} + \delta$ tại bước thứ $k+1$. Ngược lại nếu ta muốn N có $f^{'}(x_N) > 0$ tiến về phía G tại bước tình toán thứ $k+1$ thì $x_{N_{k+1}} = x_{N_k} - \delta$. Như vậy để một điểm bất kì $(x, f(x))$ lân cận G trên đồ thị tiến về G thì vị trí của điểm đó phải được cập nhật sau mỗi bước tính toán bằng cách cộng với một lượng δ với $sign(\delta) = -sign(f^{'}(x))$. Trong thực thế, công thức được sử dụng có dạng:

$$x_{k+1} = x_k - \mu f^{'}(x_k) \tag{2.1}$$



Hình 2.1: Cực tiểu địa phương (màu xanh) và cực tiểu toàn cục (màu đỏ)

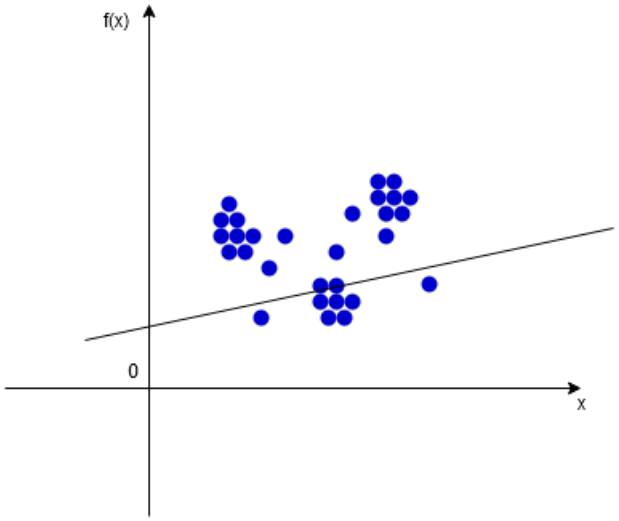
Với μ là *tốc độ học*, $\mu \in \mathbb{R}$, $\mu > 0$. Nếu ta chọn μ lớn thì ta sẽ cần ít số bước tính toán hơn để đến gần vị trí cực tiểu mong muốn nhưng trong nhiều trường hợp độ sai lệch giữa vị trí của điểm tính toán được sau cùng và vị trí của điểm cực tiểu sẽ tương đối cao. Ngược lại, nếu ta chọn μ nhỏ thì ta sẽ cần nhiều hơn số bước tính toán, bù lại khoảng cách giữa vị trí điểm tính toán được sau cùng và vị trí điểm cực tiểu sẽ có thể rất nhỏ.

Việc áp dụng giả thuật Gradient Descent lên làm đa biến là một sự mở rộng của ví dụ hàm đơn biến ở trên. Cho hàm số $f : \mathbb{R}^n \to \mathbb{R}, n \in \mathbb{N}^*$, ta cần tìm cực tiểu cho $f(X)$ với $X = \begin{pmatrix} x_0 & \dots & x_{n-1} \end{pmatrix}, n \in \mathbb{N}^*$ từ một điểm khởi đầu X_0 bằng giải thuật Gradient Descent. Công thức để tính toán cho mỗi bước là:

$$X_{k+1} = X_k - \mu \nabla_X f(X_k) \tag{2.2}$$

2.1.1 Batch Gradient Descent

Giải thuật Batch Gradient Descent sử dụng tất cả các điểm đầu vào để cập nhật lại vector trọng số tại mỗi bước. Giả sử ta cần tối ưu hàm mất mát của một bài toán hồi quy tuyến tính gồm 30 điểm đầu vào với mỗi điểm gồm 2 tham số $(x, f(x))$ ở hình 2.2. Để tìm gradient cho mỗi điểm ta cần thực hiện 2 phép toán theo toán tử ∇, đồng thời ta phải tìm gradient cho cả 30 điểm tại mỗi bước lặp và lấy trung bình của các kết quả này để cập nhật trọng số. Tổng số phép toán mà ta phải thực hiện cho tại mỗi bước là $30 \times 2 = 60$. Con số này sẽ tăng lên gấp nhiều lần đối với các bài toán thực tế khi số điểm và số tham số là vài triệu hoặc vài tỷ. Nói cách khác thuật toán này không hiệu quả về mặt tính toán với các bài toán máy học với dữ liệu lớn và phải cập nhật liên tục. Ngoài ra sau khi đã tìm được nghiêm tối

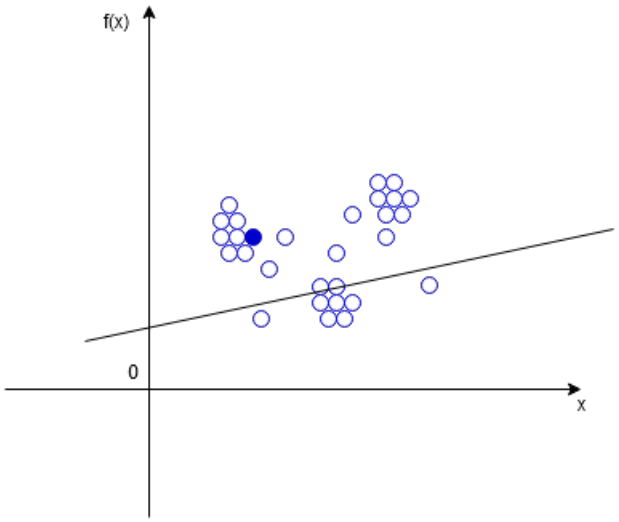


Hình 2.2: Batch Gradient Descent với bài toán hồi quy tuyến tính. Toàn bộ số điểm đầu vào đều được dùng để cập nhật các vector trọng số (a, b) cho đường hồi quy tại mỗi bước, với a là độ dốc và b độ sai lệch.

ưu của bài toán. Nếu ta thêm một điểm đầu vào mới vào tập dữ liệu cũ thì việc tính toán phải thực hiện lại từ đầu với toàn bộ điểm đầu vào bao gồm tập điểm đầu vào cũ và điểm mới thêm vào.

2.1.2 Stochastic Gradient Descent

Khác với Batch Gradient Descent giải thuật Stochastic Gradient Descent chỉ dùng gradient của một điểm ngẫu nhiên để cập nhật lại vector trọng số tại mỗi bước. Sau khi đi qua hết tất cả các điểm của tập đầu vào, thứ tự các điểm sẽ được xáo trộn và giải thuật lại tiếp tục với từng điểm. Mỗi một lần giải thuật Stochastic Gradient Descent tính toán xong với một điểm được gọi là một *iteration* còn với toàn bộ tập điểm thì gọi là một *epoch*. Cũng bài toán hồi quy tuyến tính ở trên nhưng với giải thuật Stochastic Gradient Descent (hình 2.3), ta có thể thấy số iteration mà giải thuật Stochastic Gradient Descent phải thực hiện trong một epoch là 30. Số phép tính của một lần tính toán là 2.



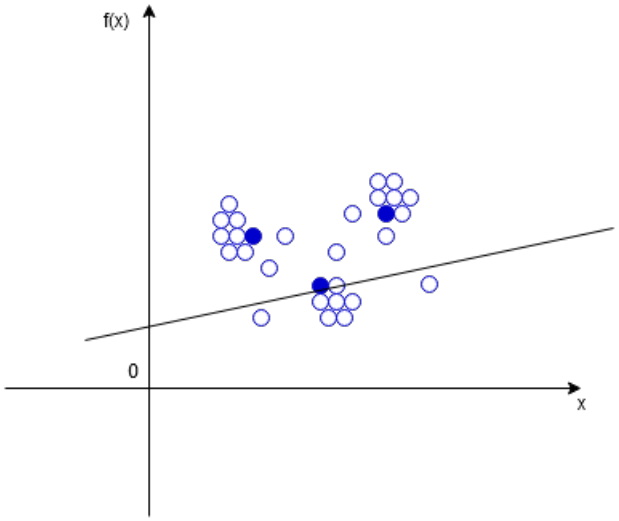
Hình 2.3: Stochastic Gradient Descent với bài toán hồi quy tuyến tính. Một điểm đầu vào được chọn ngẫu nhiên để cập nhật các vector trọng số (a, b) cho đường hồi quy tại mỗi iteration, với a là độ dốc và b độ sai lệch.

Do gradient của 1 điểm chỉ là xấp xỉ gần đúng của trung bình gradient của cả tập điểm nên việc cập nhật tại mỗi iteration sẽ có sai số nhật định, đồng thời các giá trị gradient tính toán được có thể có sư dao động lớn do tập điểm đầu vào thường bị tác động bởi nhiễu. Trên thực tế thì kết quả của giải thuật này có mức độ tối ưu khá tốt và hiệu quả tính toán cao. Sau khi đã hoàn thành tính toán trên tập dữ liệu cũ, nếu như có những điểm mới

được thêm vào thì ta chỉ cần chạy giải thuật với các điểm mới mà không cần phải chạy lại giải thuật với toàn bộ các điểm như Batch Gradient Descent.

2.1.3 Mini-batch Gradient Descent

Mini-batch Gradient Descent là sự kết hợp của Batch Gradient Descent và Stochastic Gradient Descent. Một mini-batch sẽ có n điểm với $1 < n \leq N$, N là tổng số điểm của tập dữ liệu đầu vào. Việc chia tập điểm ban đầu thành các batch sẽ được thực hiện một cách ngẫu nhiên. Mỗi một lần giải thuật xử lý xong một batch sẽ là một iteration và sau khi tất cả các batch được xử lý thì sẽ là một epoch. Như vậy $no\_batch = \frac{N}{n}$. Phương pháp này cho kết quả gần với Batch Gradient Descent nhưng không dùng nhiều tài nguyên tính toán như Batch Gradient Descent và không cần phải lặp lại nhiều lần như Stochastic Gradient Descent.



Hình 2.4: Mini-batch Gradient Descent với bài toán hồi quy tuyến tính. Một batch sẽ gồm ba điểm đầu vào được chọn ngẫu nhiên để cập nhật các vector trọng số (a, b) cho đường hồi quy tại mỗi iteration, với a là độ dốc và b độ sai lệch. Một epoch sẽ gồm mười batch.

2.1.4 Điều kiện dừng của giải thuật

Ta đã biết các giải thuật Gradient Descent sẽ cần phải thực hiện rất nhiều vòng lặp tính toán để có thể hội tụ. Tuy nhiên rất khó để nói được khi nào có thể dừng được giải thuật. Trong thực tế có nhiều cách khác nhau được dùng để chọn số bước tính toán:

1. Chọn một số lương vòng lặp nhất định dựa vào một số tiêu chí như số lượng dữ liệu đầu vào. Cách làm này có thể cho kết quả không đủ tốt, có thể nghiệm tối ưu nằm ở các bước trước hoặc sau điểm kết thúc.

2. Kiểm tra sự thay đổi của hàm mất mát giữa hai lần cập nhật liên tiếp, nếu sự sai lêch đạt tới ngưỡng đủ nhỏ thì ngưng giải thuật. Tuy nhiên nếu trên đồ thị của hàm mất mát có một vùng bằng phẳng nhưng không phải là cực tiểu thì giải thuật sẽ dừng tại điểm này mà không đạt được cực tiểu.

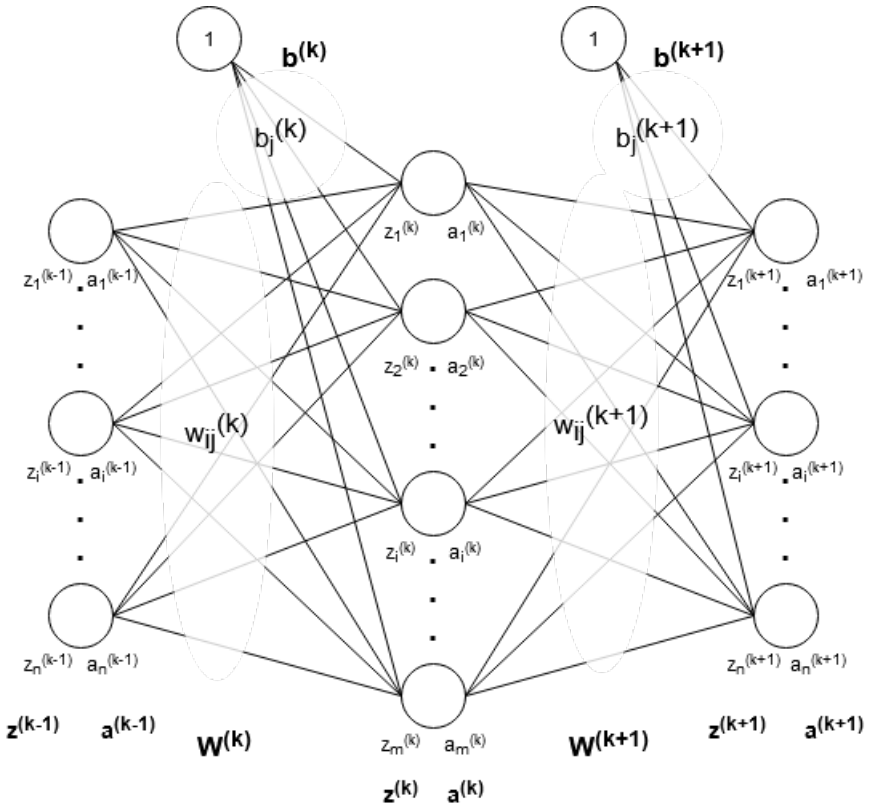
3. Kiểm tra sự thay đổi của gradient giữa hai lần cập nhật liên tiếp, nếu sự sai lêch đạt tới ngưỡng đủ nhỏ thì ngưng giải thuật. Nhược điểm của phương pháp này là việc tính gradient của các hàm phức tạp khó có thể thực hiện được.

4. Kiểm tra kết quả của giải thuật để ngừng việc lặp. Việc này cần người thực hiện việc huấn luyên mô hình phải thường xuyên kiểm tra các tham số hiệu năng của giải thuật lên một tập dữ liệu kiểm tra - *validation set* để xem tại thời điểm nào giải thuật có hiệu năng tốt nhất.

2.2 Backpropagation

Xét một mô hình mạng neuron (hình 2.5) Quá trình dữ liệu được đưa vào lớp đầu tiên cho đến khi có kết quả ở lớp sau cùng được gọi là quá trình *feed-forward*.

$$
\begin{aligned}
a^{(0)} &= x \\
z^{(k)} &= \boldsymbol{W}^{(k)T} a^{(k-1)} + \boldsymbol{b}^k, k = 1, 2, .., N \\
a^{(l)} &= f^{(k)}\left(z^{(k)}\right), k = 1, 2, .., N \\
\widehat{y} &= a^{(N)}
\end{aligned}
$$



Hình 2.5: Mô hình mạng neuron đơn giản.

Ta thấy với mô hình này, hàm mất mát $J(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{X}, \boldsymbol{Y})$ sẽ phụ thuộc vào tập các ma trận trọng số $\boldsymbol{W}$ và tập các vector bias của mỗi lớp $\boldsymbol{b}$. Việc tính gradient của làm mất mát phụ thuộc vào việc tính các đạo hàm riêng $\frac{\partial J}{\partial \boldsymbol{W}^{(k)}}$; $\frac{\partial J}{\partial \boldsymbol{b}^{(k)}}$, $\forall k = 1, 2, .., N$. Đối với bài toán hồi quy tuyến tính thì hàm mất mát là hàm trung bình bình phương sai số (Mean Square Error - MSE), lúc này

$$J(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{M}\sum_{i=1}^{(M)} \|y_i - \widehat{y_i}\|_2^{\,2} = \frac{1}{M}\sum_{i=1}^{(M)} \|y_i - a_i{}^{(N)}\|_2^{\,2} \tag{2.3}$$

Với N là số điểm trong tập điểm đầu vào. Ta nhận thấy để tìm các đạo hàm riêng của J với $\boldsymbol{W}$ và $\boldsymbol{b}$ trong trường hợp này là rất khó vì phương trình của J không phụ thuộc trực tiếp vào $\boldsymbol{W}$ và $\boldsymbol{b}$. Để có thể hiện thực các giải thuật thuộc họ Gradient Descent thì phương pháp thường được sử dụng là

Backpropagation. Phương pháp này sẽ cập nhật các trọng số theo chiều từ layer cuối cùng đến layer đầu tiên. Đầu tiên giải thuật sẽ tính đạo hàm của hàm mất mát theo ma trận trọng số của lớp cuối cùng.

$$\begin{aligned}\frac{\partial J}{\partial w_{ij}^{(N)}} &= \frac{\partial J}{\partial z_j^{(N)}} \cdot \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \\ &= e_j^{(N)} \frac{\partial \left(w_{ij}^{(N)T} a^{(N-1)} + b_j^{(N)} \right)}{\partial w_{ij}^{(N)}} \\ &= e_j^{(N)} a_i^{(N-1)}\end{aligned}$$

Với $e_j^{(N)} = \frac{\partial J}{\partial z_j^{(N)}}$ có thể tính được tương đối dễ dàng. Tương tự ta có đạo hàm riêng của J với bias ở lớp cuối cùng.

$$\begin{aligned}\frac{\partial J}{\partial b_j^{(N)}} &= \frac{\partial J}{\partial z_j^{(N)}} \cdot \frac{\partial z_j^{(N)}}{\partial b_j^{(N)}} \\ &= e_j^{(N)}\end{aligned}$$

Các công thức trên cũng đúng với một lớp bất kỳ trong mạng neuron. Ta lấy mô hình hai lớp liên tiếp của một mạng neuron ở hình 2.5 để đưa ra công thức tổng quát như sau:

$$\begin{aligned}\frac{\partial J}{\partial w_{ij}^{(k)}} &= \frac{\partial J}{\partial z_j^{(k)}} \cdot \frac{\partial z_j^{(k)}}{\partial w_{ij}^{(k)}} \\ &= e_j^{(k)} \frac{\partial \left(w_{ij}^{(k)T} a^{(k-1)} + b_j^{(k)} \right)}{\partial w_{ij}^{(N)}} \\ &= e_j^{(k)} a_i^{(k-1)}\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial b_j^{(k)}} &= \frac{\partial J}{\partial z_j^{(k)}} \cdot \frac{\partial z_j^{(k)}}{\partial b_j^{(k)}} \\ &= e_j^{(k)}\end{aligned}$$

Ta sẽ tính $e_j^{(k)}$ như sau:

$$\begin{aligned} e_j^{(k)} &= \frac{\partial J}{\partial z_j^{(k)}} = \frac{\partial J}{\partial a_j^{(k)}} \cdot \frac{\partial a_j^{(k)}}{\partial z_j^{(k)}} \\ &= \left(\sum_{l=1}^{d^{(k+1)}} \frac{\partial J}{\partial z_l^{k+1}} \cdot \frac{\partial z_l^{(k+1)}}{\partial a_j^{(k)}} \right) f^{(k)'} \left(z_j^{(k)} \right) \\ &= \left(\sum_{l=1}^{d^{(k+1)}} e_l^{(k+1)} \cdot w_{jl}^{(k+1)} \right) f^{(k)'} \left(z_j^{(k)} \right) \end{aligned}$$

Ta có $f : \mathbb{R} \to [0, 1]$ là hàm kích (activation function) hay còn gọi là hàm bao tại một node trong mạng neuron, $a_j^k = f\left(z_j^k\right)$, do đó ta có đạo hàm riêng của a_j^k theo z_j^k chính là đạo hàm của f. Ngoài ra do a_j^k trực tiếp tham gia vào việc tính các $z_l^{k+1}, l = 1, 2, .., d^{(k+1)}$ nên $\frac{\partial J}{\partial a_j^k}$ có thể tách ra thành tổng của tích các đạo hàm riêng như dòng thứ hai. Tương tự như vậy ta có thể tính

$$\frac{\partial J}{\partial b_j^{(k)}} = e_j^{(k)} \tag{2.4}$$

Việc tính e_j^k sẽ phụ thuộc vào kết quả của e_j^{k+1} do đó phương pháp này được gọi là Backpropagation. Các bước để thực hiện giải thuật Backpropagation cho một mạng neuron nhân tạo gồm:

1. Feedforward: Với mỗi giá trị đầu vào của x, tính giá trị đầu ra của mạng neuron, đồng thời lưu lại các kết quả $\boldsymbol{a}^{(k)}$ tại mỗi lớp.

2. Với mỗi node thứ j ở lớp ngoài cùng tính

$$e_j^{(N)} = \frac{\partial J}{\partial z_j^{(N)}} \tag{2.5}$$

3. Từ đó suy ra:

$$\begin{aligned} \frac{\partial J}{\partial w_{ij}^{(N)}} &= a_i^{(N-1)} e_j^{(N)} \\ \frac{\partial J}{\partial b_j^{(N)}} &= e_j^{(N)} \end{aligned}$$

4. Với $k = N-1, N-2, ..., 1$ tìm $e_j^{(k)}$

$$e_j^{(k)} = \left(\sum_{l=1}^{d^{(k+1)}} e_l^{(k+1)} \cdot w_{jl}^{(k+1)} \right) f^{(k)'} \left(z_j^{(k)} \right) \tag{2.6}$$

5. Cập nhật đạo hàm cho từng trọng số và bias:

$$\frac{\partial J}{\partial w_{ij}^{(k)}} = a_i^{(k-1)} e_j^{(k)}$$
$$\frac{\partial J}{\partial b_j^{(k)}} = e_j^{(k)}$$

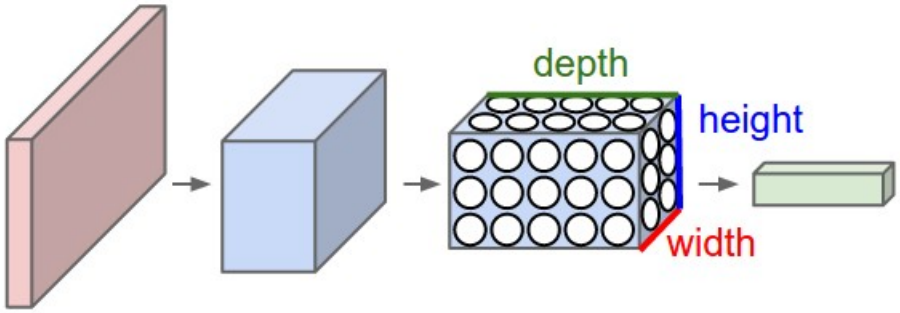
2.3 Mạng neuron tích chập

Mạng neuron tích chập (tiếng Anh: Convolutional Neural Network - CNN) là một loại mạng neuron dùng riêng cho các bài toán về hình ảnh. Bên trong mạng neuron tích chập vẫn là các neuron có các trọng số và bias có thể cập nhật được để học các đặc trưng của hình ảnh.

Các lớp của mạng neuron tích chập được bố trí theo ba chiều: chiều rộng (tiếng Anh: width), chiều cao (tiếng Anh: height), chiều sâu (tiếng Anh: depth). Chiều sâu ở đây muốn nói tời chiều sâu của miền các neuron kích hoạt (tiếng Anh: activation volume) chứ không phải là chiều sâu của cả mạng neuron. Các neuron ở lớp sau sẽ chỉ được kết nối với một phần nhỏ các neuron ở lớp trước chứ không phải là toàn bộ như trong các mạng neuron thông thường. Ta lấy ví dụ mạng CIFAR-10 (hình 2.6), miền các neuron kích hoạt ở mạng neuron này có kích thước các chiều là 32×32×3 (*rộng* × *cao* × *sâu*). Lớp cuối cùng của CIFAR-10 sẽ có kích thước các chiều là 1×10×10 ứng với vector điểm cho các nhãn cần được phân loại (tiếng Anh: class scores). Một mạng neuron tích chập thông thường sẽ được cấu tạo từ ba loại lớp neuron: lớp tích chập (tiếng Anh: convolutional layer), lớp pooling (tiếng Anh: pooling layer) và lớp đầy đủ kết nối (tiếng Anh: fully-connected layer).

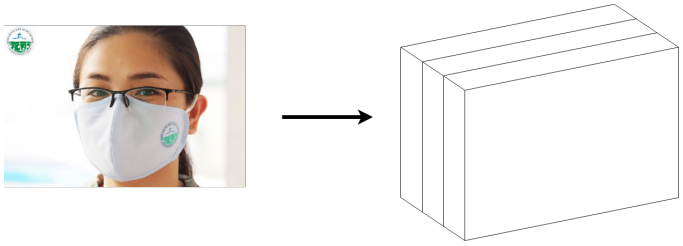
2.3.1 Lớp tích chập

Trong lớp này đầu vào lớp đầu tiên sẽ là một ảnh màu có ba kênh màu: đỏ, xanh lá, xanh dương (hình 2.7). Đầu ra của các lớp trước sẽ là đầu vào



Hình 2.6: Mô hình mạng neuron tích chập đơn giản. Lớp nhận hình ảnh vào màu đỏ là một lớp có cấu trúc ba chiều với chiều rộng và chiều cao là chiều rộng và chiều cao của hình ảnh đầu vào, chiều sâu bằng ba ứng với ba kênh màu đỏ, xanh lá và xanh dương. Các lớp của mạng neuron tích chập sẽ chuyển đổi một nhóm các ma trận thành một nhóm các ma trận khác. Lớp ngoài cùng là lớp phân loại, có kích thước các chiều tương ứng với một vector.

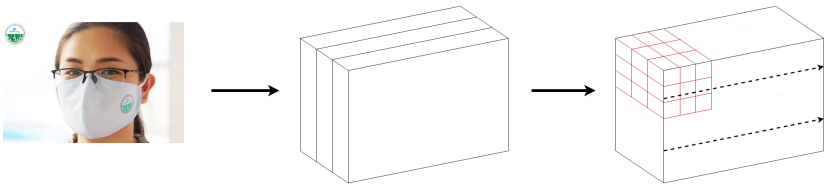
của các lớp sau. Các tensor trong mạng tích chập được gọi là các tensor. Sau



Hình 2.7: Hình ảnh đầu vào gồm ba kênh màu được mô hình hóa thành tensor với chiều cao và chiều rộng là chiều cao và chiều rộng của ảnh, chiều sâu là ba.

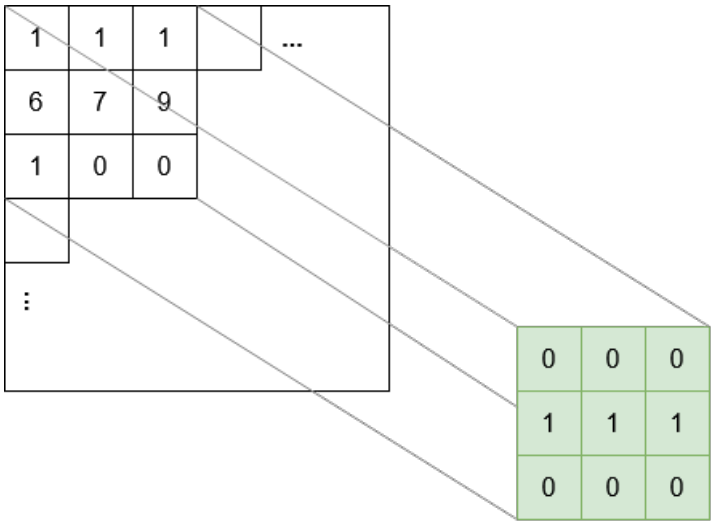
đó một bộ lọc có kích thước $m \times n \times 3$ (tiếng Anh: kernel) sẽ được trượt qua tensor của ảnh đầu vào. Ở mỗi kênh màu, lớp tương ứng của kernel sẽ hoạt động như một cửa sổ trượt (tiếng Anh: sliding window). Nhắc lại một chút về phép toán của cửa sổ trượt trên ảnh trắng đen. Giả sử ta có một cửa sổ trượt có kích thước 3×3 đang quét qua một hình trắng đen (hình 2.9), tại vị trí như trên hình việc tính toán giá trị đầu ra được thực hiện như sau

$$1 \times 0 + 1 \times 0 + 1 \times 0 + 6 \times 1 + 7 \times 1 + 9 \times 1 + 1 \times 0 + 0 \times 0 + 0 \times 0 = 22 \quad (2.7)$$



Hình 2.8: Hình ảnh sau khi được đưa qua đầu vào và chuyển đổi thành dữ liệu ba chiều sẽ được đưa vào lớp convolution đầu tiên. Một kernel có kích thước $3 \times 3 \times 3$ (góc trên bên trái của mô hình ngoài cùng bên phải) được trượt qua hình đầu vào.

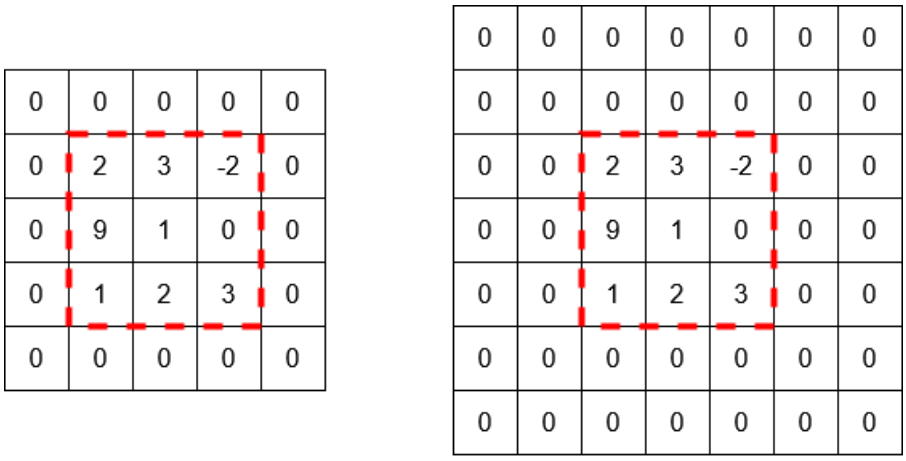
Ngoài ra, ta còn hai khái niệm cần nhắc tới là stride và padding.



Hình 2.9: Ví dụ về phép toán cửa sổ trượt với kích thước 3×3.

- Với stride bằng một thì cửa sổ trượt sẽ di chuyển tuần tự qua tất cả các ô của ma trận. Tổng quát với $stride = k$ thì các điểm ảnh được cửa sổ trượt đi qua của một ma trận có kích thước $m \times n$ sẽ là $x_{1+i\times k,1+j\times k}$ với $i, j \in \mathbb{N}; 1 + i \times k \leq m; 1 + j \times k \leq n$.
- Đối với các điểm ảnh ở gần biên, nếu như trong vùng cửa sổ không có những chỗ không tồn tại giá trị điểm ảnh thì các phương pháp chèn giá trị (tiếng Anh: padding) sẽ được sử dụng để thay làm các giá trị tính

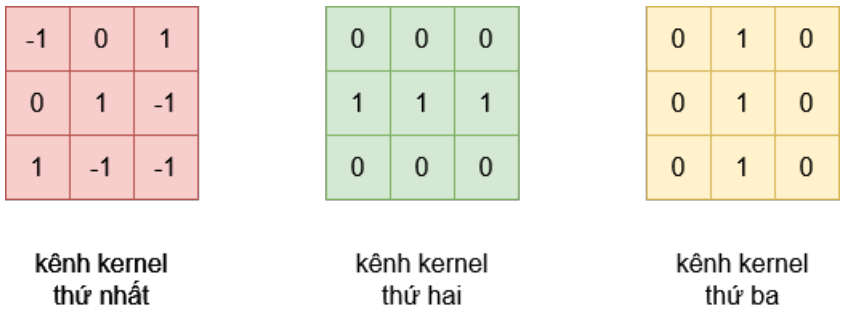
toán. Một trong các cách padding phổ biến là dùng các giá trị bằng không (tiếng Anh: zero padding).



Hình 2.10: Bên trái, ma trận 3×3 được zero padding với $padding = 1$. Bên phải, ma trận 3×3 được zero padding với $padding = 2$

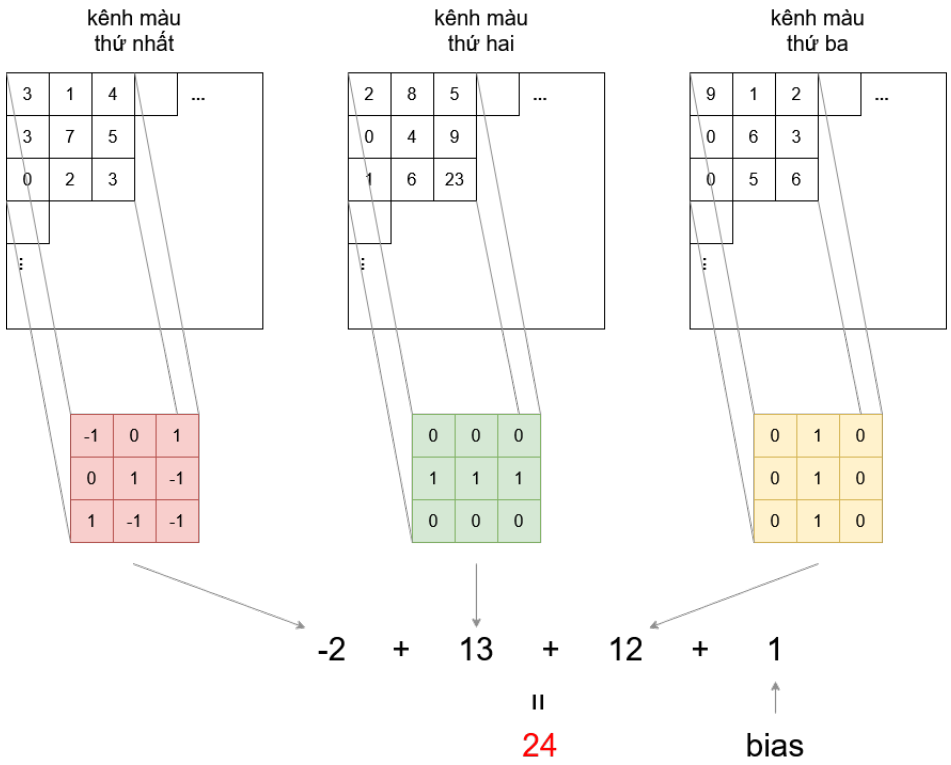
Như vậy, nếu đầu vào của phép tính tích chập là ma trận X có kích thước $m \times n$ với cửa sổ trượt có kích thước $k \times k, stride = s, padding = p$ thì đầu ra sẽ là một ma trận Y có kích thước $\left(\frac{m-k+2p}{8} + 1\right) \times \left(\frac{n-k+2p}{8} + 1\right)$

Việc tính toán tại mỗi kênh màu của hình khi kernel đi qua cũng gần tương tự với cửa sổ trượt. Kết quả phép toán của ba kênh màu và một bias sẽ được cộng lại và đưa vào ma trận kết quả. Giả sử ta có một kernel có kích thước $3 \times 3 \times 3$ như hình 2.11. Dữ liệu một ảnh đầu vào gồm ba kênh màu,



Hình 2.11: Ví dụ về một kernel có kích thước $3 \times 3 \times 3$.

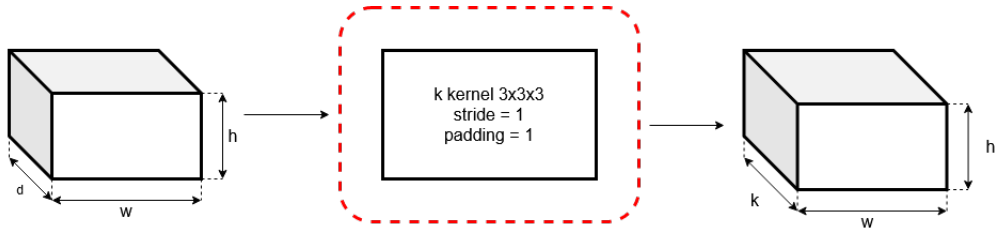
khi đi qua lớp tích chập đầu tiên sẽ được tính toán như hình 2.12 Sau khi



Hình 2.12: Ví dụ về phép toán của một kernel lên một vị trí của ảnh trong lớp tích chập.

kernel đã quét qua hết các điểm ảnh mong muốn thì kết quả nhận được sẽ là một ma trận. Mỗi một kernel khác nhau sẽ trích xuất ra được một đặc trưng khác nhau của ảnh. Do đó một lớp tích chập sẽ có nhiều kernel để lấy các đặc trưng khác nhau. Lúc này đầu ra sẽ là một tensor gồm nhiều ma trận. Nếu như có k kernel được dùng tại một lớp tích chập thì đầu ra sẽ có chiều sâu bằng k, chiều rộng và chiều cao sẽ bằng chiều rộng và chiều cao của ảnh đầu vào. Đầu ra của lớp tích chập trước sẽ là đầu vào của lớp tích chập sau. Tổng quát hóa, với một lớp tích chập với K kernel có kích thước $N \times N \times D$ (với D là chiều sâu của đầu vào và là số lẻ), $stride = S$, $padding = D$. Đầu vào là một tensor có kích thước $H \times W \times D$ thì kích thước của tensor đầu ra sẽ là $\left(\frac{H-F+2P}{S} + 1\right) \times \left(\frac{W-F+2P}{S} + 1\right) \times K$

Đầu ra của lớp tích chập sẽ đi qua hàm kích hoạt trước khi được đưa vào lớp tích chập tiệp theo. Mỗi kernel với kích thước $N \times N \times D$ sẽ có một hệ số bias tương ứng với tổng số các tham số của môt kernel là $N \times N \times D + 1$,



Hình 2.13: Một lớp tích chập có k kernel với kích thước $3 \times 3 \times 3$, $stride = 1$, $padding = 1$. Đầu vào là một tensor có kích thước $h \times w \times d$ đầu ra của phép tích chập lên tensor này khi khối tích chập có các thông số ở trên là một tensor có kích thước $h \times w \times k$

với K kernel thì số tham số sẽ là $K \times (N \times N \times D + 1)$.

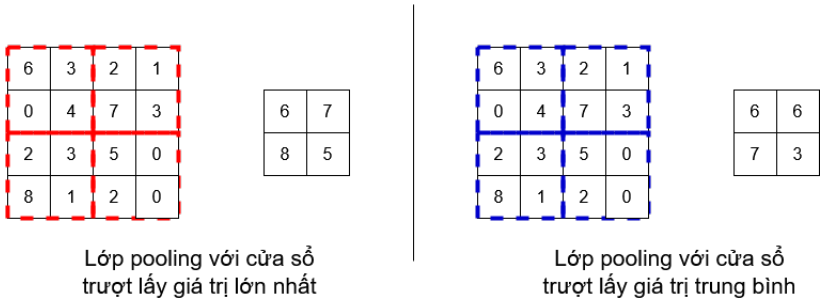
2.3.2 Lớp pooling

Việc tính toán với toàn bộ dữ liệu đầu vào của ảnh có độ phân giải lớn và kích thước lớn trên mạng neuron tích chập thường không hiệu quả về mặt tính toán do sẽ có nhiều điểm ảnh miêu tả cùng một đặc trưng. Do đó lớp pooling được dùng ở giữa các lớp tích chập để giảm kích thước của các tensor nhưng vẫn không làm mất đi các đặc trưng của dữ liệu.

Cho một lớp pooling có kích thước cửa sổ trượt là $N \times N$, đầu vào là một tensor có kích thước $H \times W \times D$. Ta chia tensor này thành D ma trận $H \times W$. Với mỗi ma trận ta lần lượt trượt cửa sổ trượt của lớp pooling lên tưng điểm ảnh. Trong vùng dữ liệu của cửa sổ trượt ta sẽ tìm giá trị lớn nhất hoặc trung bình của các giá trị để đưa vào ma trận mới. Một số mô hình mạng neuron tích chập sẽ dùng $stride > 1$ trong lớp tích chập để làm giảm kích thước dữ liệu thay vì dùng lớp pooling. Ngoài ra, trong thực tế lớp pooling thường được sử dụng với kích thước cửa sổ trượt $2 \times 2, stride = 2, padding = 0$. Chiều cao và chiều rộng của tensor đầu ra sẽ giảm đi một nửa còn chiều sâu vẫn giữ nguyện.

2.3.3 Lớp đầy đủ kết nối

Hình ảnh sau khi qua các lớp tích chập và pooling thì đầu ra sẽ là một tensor chứa các đặc trưng mà mô hình trích xuất được. Tensor có kích thước $H \times W \times D$ tại lớp tích chập cuối cùng sẽ được chuyển thành một vector có

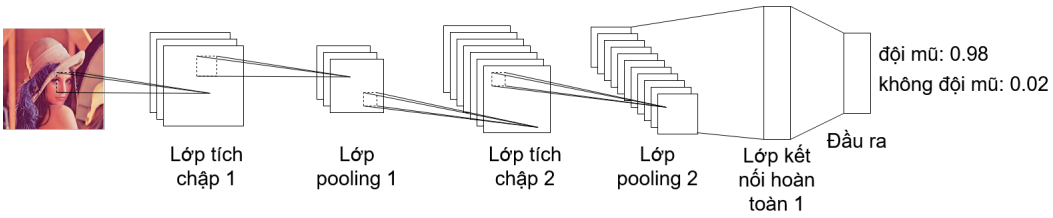


Hình 2.14: Bên trái, lớp pooling với cửa sổ trượt lấy giá trị lớn nhất với kích thước cửa sổ $2 \times 2, stride = 1, padding = 0$. Bên phải, lớp pooling với cửa sổ trượt lấy giá trị trung bình với kích thước cửa sổ $2 \times 2, stride = 2, padding = 0$.

chiều dài $H \times W \times D$. Sau đó vector này sẽ được đưa vào các lớp đầy đủ kết nối để đưa ra kết quả dự đoán cho ảnh.

2.3.4 Mô hình mạng neuron tích chập

Ảnh đầu vào $\rightarrow$ [Lớp tích chập $\rightarrow$ Lớp pooling] $\times n \rightarrow$ [Lớp liên kết hoàn toàn] $\times m \rightarrow$ Đầu ra, với $m, n \in \mathbb{N}^*$.



Hình 2.15: Mạng neuron tích chập gồm hai lớp tích chập và pooling, một lớp kết nối đầy đủ.

Column 1	*Column 2*	*Column 3*	*Column 4*
Row 1.1	Row 1.2	Row 1.3	Row 1.4
Row 2.1	Row 2.2	Row 2.3	Row 2.4

Bảng 2.1: Caption of this table.

Citing to the table above 2.1

Chương 3

Phương pháp tiếp cận

Some mathematics formula

$$F(x) = \arg \max_{y \in GEN(x)} w \cdot f(x, y)$$

Chương 4

Phân tích kết quả

In this chapter, ...

Chương 5

Kết luận

In this chapter, ...