ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

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

ĐỒ ÁN CUỐI KỲ

CS115.M13.KHCL

CONVOLUTIONAL NEURAL NETWORK (CNNs) AND BACKPROPAGATION IN CNNs

Giáo viên hướng dẫn

Tiến sĩ Lương Ngọc Hoàng

Nhóm thực hiện

20521938 Trần Phương Thảo

20522015 Nguyễn Ngọc Tín

20521960 Lương Lý Công Thịnh

Hồ Chí Minh, ngày 26 tháng 12 năm 2021

Mục lục

Danh mục hình

Phần 1 – Mạng nơ-ron tích chập (CNNs)

1.1. Lịch sử của CNNs

Năm 1958-1959, David H. Hubel và Torsten Wiesel đưa ra những kiến thức quan trọng về cấu trúc của vỏ não thị giác (giải Nobel – 1981). Và đã truyền cảm hứng cho necognitron – mô hình mạng nơ-ron có cấp bậc được được xuất bởi Fukushima và được phát triển đến nay ta gọi là convolutional neural network.

Năm 1998, mô hình LeNet-5 được đề xuất bởi Yann LeCun và những cộng sự của ông được sử dụng rộng rãi bởi các ngân hàng để nhận dạng các chữ số viết tay – có thể nói đây là cột mốc quan trọng trong lĩnh vực nghiên cứu trên.

Từ những năm 2010, GPU bắt đầu được ứng dụng trong mạng nơ-ron. Năm 2012, một thành tựu lớn về phân loại hình ảnh trên dataset ImageNet của Geoffrey Hinton với việc ứng dụng GPU trên dữ liệu huấn luyện

Từ những công trình đó, hàng loạt mạng ra đời với nhiều tham số, nhiều layer và trên dataset lớn hơn,…

1.2. Giới thiệu CNNs

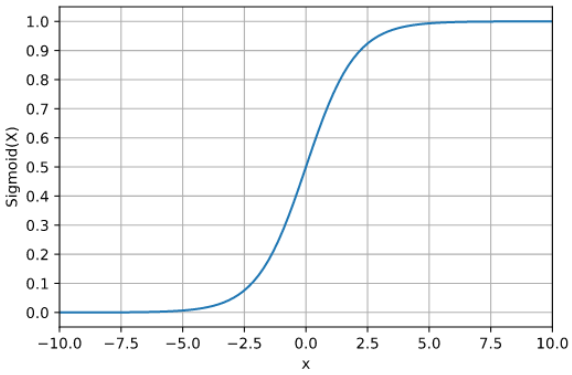
Mạng nơron tích chập là một thuật toán Deep Learning có thể lấy hình ảnh đầu vào, và có thể phân biệt được từng đặc trưng/đối tượng với nhau.

Kiến trúc của nơron tích chập tương tự như mô hình kết nối của các nơron trong bộ não con người và được lấy cảm hứng từ hệ thống vỏ thị giác trong bộ não (visual cort

1.3. Các hàm kích hoạt (Activation function)

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

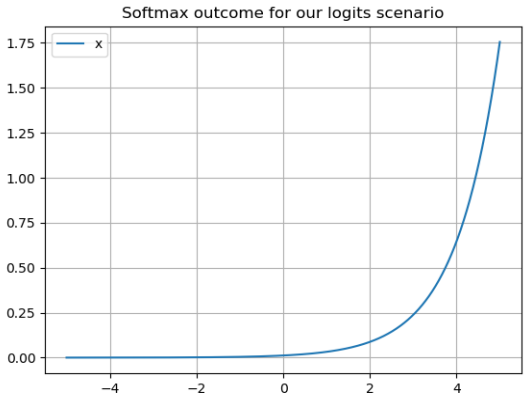


Hình 1.1 Hàm Sigmoid

Nhận bất kỳ giá trị thực nào làm giá trị đầu vào và đầu ra trong phạm vi từ 0 đến 1. Đầu vào càng lớn (càng dương), giá trị đầu ra càng gần 1, ngược lại đầu vào càng nhỏ (càng âm), đầu ra càng gần 0.

Softmax

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

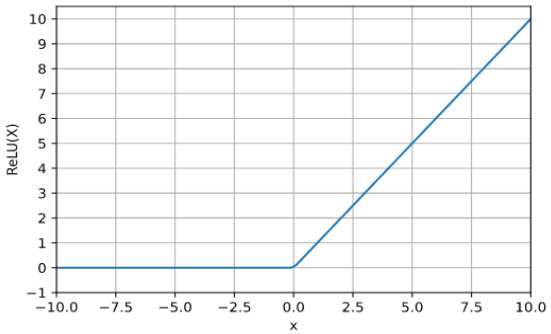


Hình 1.2 Hàm Softmax

Là hàm kích hoạt không những ánh xạ đầu ra của chúng ta tới một phạm vi [0,1] mà còn ánh xạ từng đầu ra theo cách sao cho tổng là 1. Do đó, đầu ra của Softmax là một phân phối xác suất. Với giá trị đầu vào càng lớn thì xác suất dữ liệu rơi vào class i càng cao.

Relu

$$f(x) = \max(0, x)$$



Hình 1.3 Hàm Relu

ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó.

1.4. Lớp tích chập (Convolutional layer)

Lớp tích chập là một lớp ẩn (hidden layer), khác ở chỗ, lớp tích chập là một tập feature map và mỗi feature map này là một bản cắt lớp (scan) của đầu vào (input) ban đầu, nhưng được trích xuất ra các đặc trưng (feature) cụ thể.

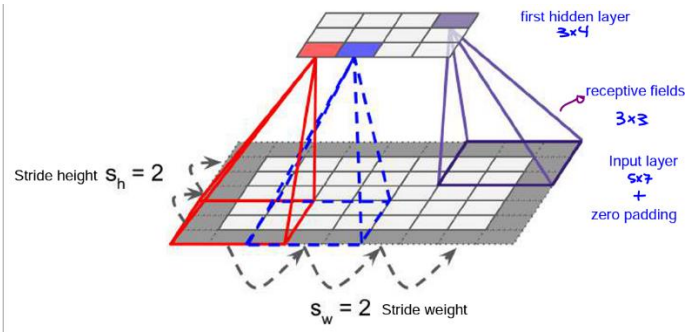
Có thể có nhiều lớp tích chập trong cùng mạng nơ-ron tích chập để lấy được những đặc trưng phức tạp hơn, giúp cho việc phân loại các vật thể khác nhau hiệu quả hơn, với đầu ra của một lớp tích chập cũng chính là đầu vào của của một lớp tích chập kế tiếp.

Filter

Filter hay gọi là kích thước bộ lọc, đề cập đến kích thước để tích chập với feature map tạo đầu vào cho lớp tiếp theo và việc này có tác động lớn đến nhiệm vụ phân loại ảnh.

Stride

Stride là sự chuyển đổi từ 1 trường tiếp nhận đến trường tiếp theo. Và khi bước đi giảm, nhiều đặc trưng được học hơn vì nhiều dữ liệu được trích xuất hơn, điều này cũng dẫn đến các lớp đầu ra lớn hơn và ngược lại. Có một yêu cầu khi chọn giá trị stride là đảm bảo rằng kernel trượt ngang qua hình ảnh đầu vào một cách đối xứng khi triển khai CNN.



Hình 1.4 Giảm số chiều sử dụng stride = 2

Padding

Và zero-padding được sử dụng phổ biến nhất vì hiệu suất, tính đơn giản và hiệu quả tính toán của nó. Kỹ thuật này liên quan đến việc thêm các số 0 một cách đối xứng xung quanh các cạnh của đầu vào.

Same	Valid
Lớp tích chập sử dụng zero-padding nếu cần thiết. Kích thước đầu ra là số nơ-ron đầu chia stride, làm tròn.	Lớp tích chập không sử dụng zero-padding và có thể bỏ qua 1 số hàng, cột ở cuối cùng và bên phải ảnh đầu vào, phụ thuộc và stride.

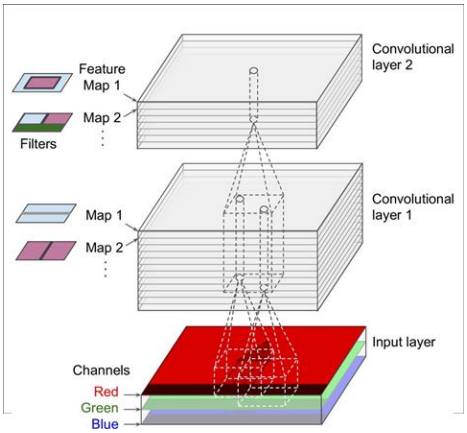
Tính tương thích của tham số trong lớp tích chập

Bằng cách ký hiệu I là độ dài kích thước đầu vào, F là độ dài của bộ lọc, P là số lượng zero padding, S là độ trượt, ta có thể tính được độ dài O của feature map theo một chiều bằng công thức:

$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1$$

Xếp chồng nhiều ma trận đặc trưng

Ảnh đầu vào thường có nhiều lớp con: mỗi lớp 1 channel. Và có 3 loại điển hình: red, green, blue (RGB). Và ảnh xám (grayscale) chỉ có 1 channel, nhưng một số ảnh có thể có nhiều hơn.



Hình 1.5 Lớp tích chập có 3 channels

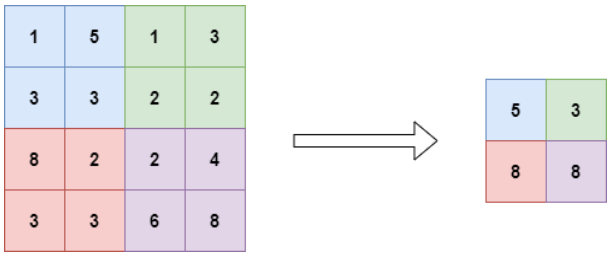
1.5. Lớp tổng hợp (Pooling layer)

Lớp tổng hợp nhằm mục đích giảm tính toán, sử dụng bộ nhớ và số lượng các thông số (hạn chế lỗi overfitting).

Như lớp tích chập, từng nơ-ron của lớp tổng hợp kết nối.Tuy nhiên, nơ-ron ở lớp này sẽ không thay đổi weight mà chỉ tập trung vào tổng hợp đầu vào (input) bằng cách sử dụng hàm tổng hợp như max hoặc average,…

Max pooling

Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng



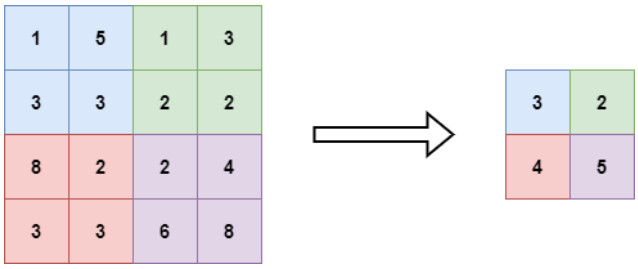
Hình 1.6 Max pooling

Bảo toàn đặc trưng đã phát hiện.

Được sử dụng thường xuyên,

Average pooling

Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng



Hình 1.7 Average pooling

Giảm kích thước feature map,

Được sử dụng trong mạng LeNet.

1.6. Lớp kết nối đầy đủ (Fully-connected)

Lớp kết nối đầy đủ chỉ đơn giản là cung cấp các mạng nơ-ron chuyển tiếp. Các lớp kết nối đầy đủ tạo thành một vài lớp cuối cùng trong mạng.

Đầu vào cho lớp kết nối đầy đủ là đầu ra được làm phẳng (flatten) từ lớp tích chập hoặc lớp tổng hợp cuối cùng. Sau đó, vector này được kết nối với một vài lớp kết nối đầy đủ giống như ANNs và thực hiện các phép toán tương tự .

Sau khi đi qua các lớp được kết nối đầy đủ, lớp cuối cùng sử dụng chức năng kích hoạt softmax (thay vì ReLU) được sử dụng để nhận xác suất đầu vào nằm trong một lớp cụ thể (phân loại). Và cuối cùng, chúng ta có xác suất của đối tượng trong hình ảnh thuộc các lớp khác nhau.

Cách tính kích thước của tensor đầu ra từ tensor đầu vào:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

W_1 là $\frac{\text{chiều dài}}{\text{chiều rộng}}$ của đầu vào.

F là $\frac{\text{chiều dài}}{\text{chiều rộng}}$ của bộ lọc.

P là padding.

S là stride.

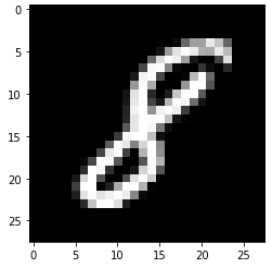
W_2 là $\frac{\text{chiều dài}}{\text{chiều rộng}}$ của đầu ra.

1.7. Ứng dụng mạng nơ-ron tích chập

MNIST là một trong những bộ dữ liệu phổ biến nhất được sử dụng phân loại hình ảnh và có thể truy cập từ nhiều nguồn dữ liệu khác nhau.

MNIST chứa 60000 hình ảnh huấn luyện có kích thước 28×28 pixel và 10000 hình ảnh kiểm tra lấy từ nhân viên American Census Bureau và học sinh trung học Hoa Kỳ.

Đầu tiên, ta load dữ liệu thông qua Keras. Và cũng tách bộ dữ liệu thanh 2 nhóm là train và test; tách nhãn và hình ảnh.



Hình 1.8 Ảnh số viết tay với index = 7777

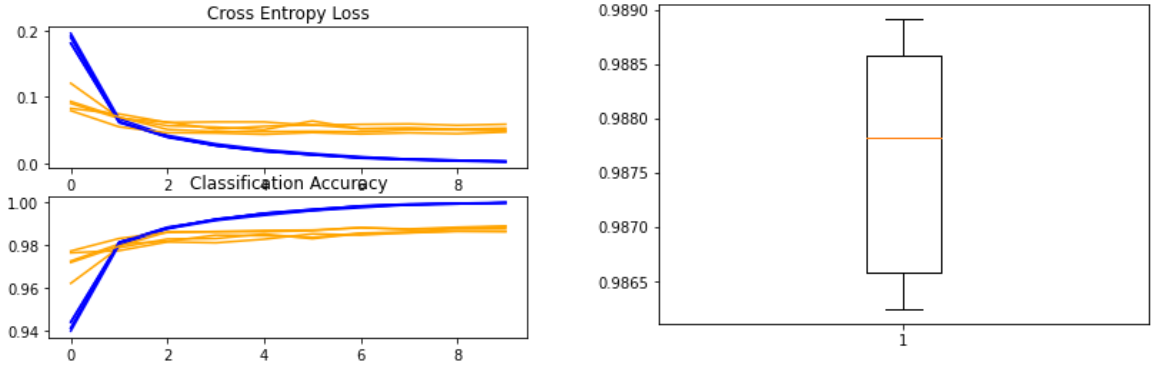
Vì mảng của ta chỉ có 3 chiều nhưng mà để sử dụng tập dữ liệu trên Keras ta cần mảng 4 chiều nên ta thêm 1 channel màu (grayscale). Và sử dụng *one hot encoding* cho từng lớp của mỗi mẫu do tập dữ liệu có mỗi lớp đại diện 1 số nguyên duy nhất.

Tiếp theo, vì pixels trong từng ảnh trong tập dữ liệu là các số nguyên không dấu trong khoảng giữa 0 và 255, nên sẽ chuẩn hóa bằng các chia chúng trong đoạn [0, 1]

Mô hình có 2 vấn đề chính là lớp tích chập và lớp tổng hợp, phần phân loại để dự đoán. Ở phần này ta sử dụng max pooling với kích thước bộ lọc 3×3 có số lượng 32, softmax và giữa 2 lớp tên ta thêm lớp Dense.

Sau đó , ta sẽ đánh giá mô hình bằng cách sử dụng K-fold cross-validation. Ta sẽ chọn $k = 5$ để đánh giá nhiều lần và thời gian không quá dài cùng với *10 epoch* và *batch_size = 32*. Vì thế tập huấn luyện sẽ được xáo trộn trước khi chia và thực hiện mỗi lần nhằm tang sự so sánh rõ rệt về mô hình.

Cuối cùng ta tạo 2 đồ thị về độ lỗi và độ chính xác.



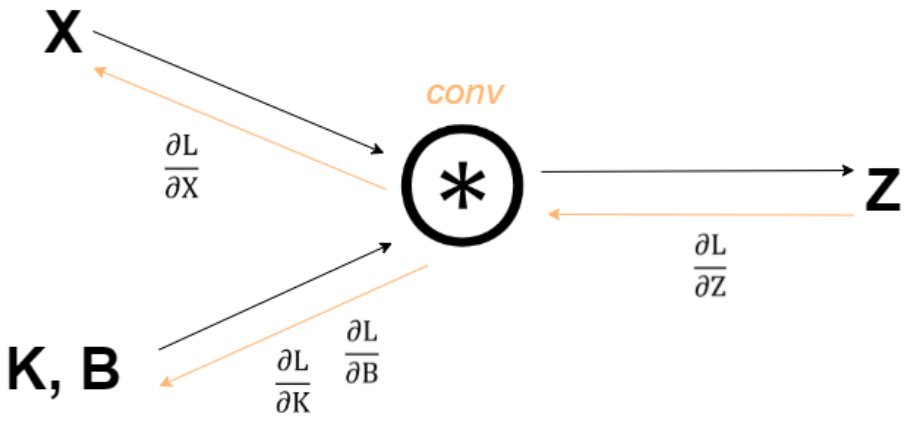
Hình bên trái, với hai đường đại diện cho hiệu suất của từng tập với màu cam – tập huấn luyện và màu xanh – tập kiểm tra để đánh giá mô hình đang ở dạng overfitting, under fitting hay good fit.. Và hình bên phải cho ta thấy được hiệu suất trung bình trên mô hình huấn luyện.

Ta có thể thấy mô hình đạt được kết quả rất đẹp qua từng phần – good fit trên của 2 tập huấn luyện và kiểm tra.

Phần 2 – Lan truyền ngược trong CNNs

Truyền ngược (hay còn gọi là lan truyền ngược, Tiếng Anh: back-propagation), là một từ viết tắt cho "backward propagation of errors" tức là "truyền ngược của sai số", là một phương pháp phổ biến để huấn luyện các mạng nơ ron nhân tạo được sử dụng kết hợp với một phương pháp tối ưu hóa như gradient descent. Phương pháp này tính toán gradient của hàm tổn thất với tất cả các trọng số có liên quan trong mạng nơ ron đó. Gradient này được đưa vào phương pháp tối ưu hóa, sử dụng nó để cập nhật các trọng số, để cực tiểu hóa hàm tổn thất.

2.1. Lớp tích chập



Hình 2.1 Lan truyền xuôi và lan truyền ngược

Tương tự với trọng số trong các thuật toán ở phía trước, thì ở trong CNN Layer ta sẽ bắt đầu với các trọng số là những pixel kernel và hệ số bias ngẫu nhiên.

Chúng ta sẽ làm cho các tham số (trọng số và bias) dần dần tốt hơn bằng cách cập nhật chúng trong mỗi lần lặp:

$$\mathbf{K = K - \alpha \frac{\partial L}{\partial K}}$$

$$\mathbf{B = B - \alpha \frac{\partial L}{\partial B}}$$

Để tính $\frac{\partial L}{\partial K}$ và $\frac{\partial L}{\partial B}$, chúng ta cần thực hiện backpropagation. Với ví dụ sau đây:

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \circledast \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} + B = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

Chúng ta sẽ tính toán từng phần tử của gradient riêng biệt bằng chain rule.

Đầu tiên ta sẽ tính $\frac{\partial L}{\partial K}$

$$\frac{\partial L}{\partial K_{mn}} = \sum \frac{\partial L}{\partial Z_{ij}} * \frac{\partial Z_{ij}}{\partial K_{mn}}$$

Ví dụ với: $\frac{\partial L}{\partial K_{12}} = \frac{\partial L}{\partial Z_{11}} * \frac{\partial Z_{11}}{\partial K_{12}} + \frac{\partial L}{\partial Z_{12}} * \frac{\partial Z_{12}}{\partial K_{12}} + \frac{\partial L}{\partial Z_{21}} * \frac{\partial Z_{21}}{\partial K_{12}} + \frac{\partial L}{\partial Z_{22}} * \frac{\partial Z_{22}}{\partial K_{12}}$

$$Z_{11} = X_{11}K_{11} + X_{12}K_{12} + X_{21}K_{21} + X_{22}K_{22} + B$$
$$Z_{12} = X_{12}K_{11} + X_{13}K_{12} + X_{22}K_{21} + X_{23}K_{22} + B$$
$$Z_{21} = X_{21}K_{11} + X_{22}K_{12} + X_{31}K_{21} + X_{32}K_{22} + B$$
$$Z_{22} = X_{22}K_{11} + X_{23}K_{12} + X_{32}K_{21} + X_{33}K_{22} + B$$

Bởi vì với việc pixel K_{12} thay đổi dẫn tới sự thay đổi của Z_{11}, Z_{12}, Z_{21}, Z_{22} và tất cả dẫn tới L thay đổi. Do đó, chúng ta đang tính toán ảnh hưởng của việc thay đổi K_{12} ảnh hưởng đến L như thế nào qua 4 con đường khác nhau. Sau đó, chúng tôi tổng hợp tất cả các đường dẫn lại với nhau để tính $\frac{\partial L}{\partial K_{12}}$.

Chúng ta có thể dễ dàng tính toán các đạo hàm sau: $\frac{\partial Z_{11}}{\partial K_{12}} = X_{12}$, $\frac{\partial Z_{12}}{\partial K_{12}}$ X_{13}, $\frac{\partial Z_{21}}{\partial K_{1}} = X_{22}$,…

Từ đó, ta tính được:

$$\frac{\partial L}{\partial K_{11}} = \frac{\partial L}{\partial Z_{11}} * X_{11} + \frac{\partial L}{\partial Z_{12}} * X_{12} + \frac{\partial L}{\partial Z_{21}} * X_{21} + \frac{\partial L}{\partial Z_{22}} * X_{22}$$
$$\frac{\partial L}{\partial K_{12}} = \frac{\partial L}{\partial Z_{11}} * X_{12} + \frac{\partial L}{\partial Z_{12}} * X_{13} + \frac{\partial L}{\partial Z_{21}} * X_{22} + \frac{\partial L}{\partial Z_{22}} * X_{23}$$
$$\frac{\partial L}{\partial K_{11}} = \frac{\partial L}{\partial Z_{11}} * X_{21} + \frac{\partial L}{\partial Z_{12}} * X_{22} + \frac{\partial L}{\partial Z_{21}} * X_{31} + \frac{\partial L}{\partial Z_{22}} * X_{32}$$
$$\frac{\partial L}{\partial K_{11}} = \frac{\partial L}{\partial Z_{11}} * X_{22} + \frac{\partial L}{\partial Z_{12}} * X_{23} + \frac{\partial L}{\partial Z_{21}} * X_{32} + \frac{\partial L}{\partial Z_{22}} * X_{33}$$

$$\Leftrightarrow \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \circledast \begin{bmatrix} \frac{\partial L}{\partial Z_{11}} & \frac{\partial L}{\partial Z_{12}} \\ \frac{\partial L}{\partial Z_{21}} & \frac{\partial L}{\partial Z_{22}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial K_{11}} & \frac{\partial L}{\partial K_{12}} \\ \frac{\partial L}{\partial K_{21}} & \frac{\partial L}{\partial K_{22}} \end{bmatrix} = \frac{\partial L}{\partial K}$$

Vậy ta tính được $\frac{\partial L}{\partial K} = \text{conv}\left(X, \frac{\partial L}{\partial Z}\right)$

Tương tự cho $\frac{\partial L}{\partial B}$ ta có: $\frac{\partial L}{\partial B} = \sum \frac{\partial L}{\partial Z_{ij}} * \frac{\partial Z_{ij}}{\partial B}$ (chain rule)

Dễ dàng tính được $\frac{\partial Z_{ij}}{\partial B} = 1 \Longrightarrow \frac{\partial L}{\partial B} = \frac{\partial L}{\partial Z_{11}} + \frac{\partial L}{\partial Z_{12}} + \frac{\partial L}{\partial Z_{21}} + \frac{\partial L}{\partial Z_{22}} \Leftrightarrow \frac{\partial L}{\partial B} = sum\left(\frac{\partial L}{\partial Z}\right)$

Và cũng ở lớp này chúng ta cũng phải tính $\frac{\partial L}{\partial X}$, lí do bởi vì $\frac{\partial L}{\partial X}$ ở đây có thể sẽ trở thành gradient loss cho lớp trước trong lan truyền ngược.

$$\boldsymbol{\frac{\partial L}{\partial X_{mn}} = \sum \frac{\partial L}{\partial Z_{ij}} * \frac{\partial Z_{ij}}{\partial X_{mn}}}$$

Ví dụ: $\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial Z_{11}} * \frac{\partial Z_{11}}{\partial X_{11}} = \frac{\partial L}{\partial Z_{11}} * K_{11}$ với $\frac{\partial Z_{12}}{\partial X_{11}}, \frac{\partial Z_{21}}{\partial X_{11}}, \frac{\partial Z_{22}}{\partial X_{11}} = 0$

Bởi vì mọi thay đổi X_{11} không ảnh hưởng tới Z_{12}, Z_{21}, Z_{22}. Do đó, chúng không được xem xét trong phương trình đó. Từ đó ta tính được như sau:

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial Z_{11}} * K_{11}$$
$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial Z_{11}} * K_{12} + \frac{\partial L}{\partial Z_{12}} * K_{11}$$
$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial Z_{12}} * K_{12}$$
$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial Z_{11}} * K_{21} + \frac{\partial L}{\partial Z_{21}} * K_{11}$$
$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial Z_{11}} * K_{22} + \frac{\partial L}{\partial Z_{12}} * K_{21} + \frac{\partial L}{\partial Z_{21}} * K_{12} + \frac{\partial L}{\partial Z_{22}} * K_{11}$$
$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial Z_{12}} * K_{22} + \frac{\partial L}{\partial Z_{22}} * K_{12}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}_{31}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{21}} * \mathrm{K}_{21}$$
$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}_{32}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{21}} * \mathrm{K}_{22} + \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{22}} * \mathrm{K}_{21}$$
$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}_{33}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{22}} * \mathrm{K}_{22}$$

Từ những phương trình trên ta nhận thấy được là lật filter 180 độ rồi tích chập với ma trận gồm những $\frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{\mathrm{ij}}}$ theo thứ tự từ trái qua phải, trên xuống dưới và được thêm padding để bằng kích thước input ban đầu. Từ đó, ta thu được $\frac{\partial L}{\partial X}$ như bên dưới.

$$\text{Từ } \begin{bmatrix} \mathrm{K}_{11} & \mathrm{K}_{12} \\ \mathrm{K}_{21} & \mathrm{K}_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathrm{K}_{22} & \mathrm{K}_{21} \\ \mathrm{K}_{12} & \mathrm{K}_{11} \end{bmatrix} \text{ với } \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{11}} & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{12}} & 0 \\ 0 & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{21}} & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{22}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{11}} & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{12}} & 0 \\ 0 & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{21}} & \frac{\partial \mathrm{L}}{\partial \mathrm{Z}_{22}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} \mathrm{K}_{22} & \mathrm{K}_{21} \\ \mathrm{K}_{12} & \mathrm{K}_{11} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{11}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{12}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{13}} \\ \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{21}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{22}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{23}} \\ \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{31}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{32}} & \frac{\partial \mathrm{L}}{\partial \mathrm{X}_{33}} \end{bmatrix}$$

$$\Rightarrow \frac{\partial L}{\partial X} = \mathrm{conv}\left(\text{padded}\left(\frac{\partial L}{\partial Z}\right), 180^\circ \text{ rotated filter K}\right)$$

Tổng kết lại, ta có :

$$\frac{\partial L}{\partial K} = \mathrm{conv}\left(X, \frac{\partial L}{\partial Z}\right)$$

$$\frac{\partial L}{\partial B} = \mathrm{sum}\left(\frac{\partial L}{\partial Z}\right)$$

$$\frac{\partial L}{\partial X} = \mathrm{conv}\left(\text{padded}\left(\frac{\partial L}{\partial Z}\right), 180^\circ \text{ rotated filter K}\right)$$

ReLU layer

ReLU (x) = max (x, 0). Khi x > 0, nó trả về x vì vậy nó là tuyến tính trong vùng này của hàm (gradient là 1) và khi x < 0 thì nó luôn là 0 (vì vậy một giá trị không đổi), do đó gradient là 0. Với the gradient ở chính xác 0 về mặt kỹ thuật là không xác định nhưng trong thực tế, chúng tôi chỉ đặt nó về 0. Vì thế ta có :

Forward :

$$y_l = \max(0, y_{l-1}) \longrightarrow$$

Backward:

$$\frac{\partial L}{\partial y_{l-1}} = \begin{cases} 0, \text{ nếu } y_l < 0 \\ 1, \text{ còn lại} \end{cases}$$

2.2. Lớp tổng hợp (Pooling layer)

Đối với lớp pooling, chúng ta không cần tính toán các đạo hàm riêng một cách rõ ràng, vì không có tham số nào. Thay vào đó, chúng tôi quan tâm đến cách phân phối gradient cho từng giá trị trong cửa sổ đầu vào tương ứng.

Max Pooling

Theo trực giác, việc thay đổi các giá trị không phải tối đa của mỗi cửa sổ sẽ không ảnh hưởng đến đầu ra, vì đầu ra chỉ quan tâm đến giá trị tối đa trong cửa sổ. Do đó, các giá trị không phải max có gradient là 0.

Đối với giá trị tối đa trong mỗi cửa sổ, vì đầu ra chỉ là giá trị đó, mỗi lần thay đổi trong giá trị tối đa sẽ có một lần thay đổi tương ứng trong đầu ra. Do đó, gradient bằng với gradient từ phía trên. Vì vậy, một cách hiệu quả, chúng tôi chỉ định tuyến gradient chỉ thông qua giá trị tối đa của bản cửa sổ đầu vào tương ứng đó.

Forward:

$$\text{for } (p = 0; p < k; \text{p} + +)$$
$$\quad \text{for } (q = 0; q < k; q + +)$$
$$\quad\quad y_l(x, y) = max(y_l(x, y), y_{l-1}(x + p, y + q));$$

Backward:

$$\frac{\partial L}{\partial y_{l-1}}(x+p, y+q) = \begin{cases} 0 & \text{,nếu } y_n(x,y) \neq y_{n-1}(x+p, y+q) \\ \frac{\partial L}{\partial y_l}(x,y) & \text{,còn lại} \end{cases}$$

2.3. Lớp kết nối đầy đủ (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 làm phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron.

Forward

$y_l = W_l * y_{l-1}$ + B

Backward

$$\begin{cases} \frac{\partial E}{\partial y_{l-1}} = \frac{\partial E}{\partial y_l} * W_l^T \\ \frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial y_l} * y_{l-1} \\ \frac{\partial L}{\partial B} = \frac{\partial L}{\partial y_l} \end{cases}$$

Phần 3 – Kết luận

Đây được xem là một trong những mô hình của học sâu (Deep Learning) – tập hợp các thuật toán trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý cấu trúc phức tạp. CNNs được thiết kế với mục đích xử lý dữ liệu thông qua nhiều mảng. Và hiện nay, CNNs được áp dụng phổ biến nhất trong phân tích hình ảnh trực quan.

Đồng thời, mạng nơ-ron sử dụng chiến lược chia sẻ trọng số dẫn đến giảm đáng kể số lượng các tham số phải học. Sự hiện diện của các kích thước trường tiếp nhận lớn hơn của nơ-ron trong các lớp chập liên tiếp cùng với sự hiện diện của các lớp tổng hợp cũng dẫn đến sự dịch chuyển bất biến. Như chúng ta đã quan sát, nguồn gốc của quá trình lan truyền tiến và ngược sẽ khác nhau tùy thuộc vào lớp mà chúng ta đang lan truyền qua

Tham khảo

[1] Hands-on machine learning with Scikit-learn, Keras & TensorFlow – Aurélien Géron

[2] Convolutional neural networks – Standford University School of Engineering

https://youtu.be/bNb2fEVKeEo

[3] Does CNN have backpropagation – Linkedin

https://bitly.com.vn/7wsnwp

[4] Backpropagation in CNN – Coding Lane

https://youtu.be/Pn7RK7tofPg

https://youtu.be/vbUozbkMhI0