

Super VIP Cheatsheet: Học sâu

Afshine AMIDI và Shervine AMIDI

Ngày 17 tháng 5 năm 2020

Mục lục

1 Mạng nơ ron tích chập

1.1	Tổng quan	1
1.2	Các kiểu tầng	1
1.3	Các siêu tham số của bộ lọc	2
1.4	Điều chỉnh siêu tham số	2
1.5	Các hàm kích hoạt thường gặp	3
1.6	Phát hiện vật thể (object detection)	3
1.6.1	Xác nhận khuôn mặt và nhận diện khuôn mặt	5
1.6.2	Neural style transfer	5
1.6.3	Những kiến trúc sử dụng computational tricks	6

2 Mạng nơ-ron hồi quy

2.1	Tổng quan	7
2.2	Xử lý phụ thuộc dài hạn	7
2.3	Học từ đại diện	8
2.3.1	Giải thích và các kí hiệu	9
2.3.2	Word embeddings	9
2.4	So sánh các từ	10
2.5	Mô hình ngôn ngữ	10
2.6	Dịch máy	10
2.7	Chú ý	11

3 Mẹo và thủ thuật

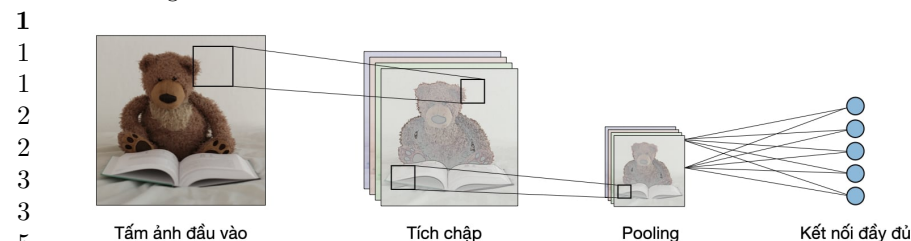
3.1	Xử lý dữ liệu	11
3.2	Huấn luyện mạng neural	12
3.2.1	Định nghĩa	12
3.2.2	Tìm trọng số tối ưu	12
3.3	Tinh chỉnh tham số	12
3.3.1	Khởi tạo trọng số	12
3.3.2	Tối ưu hội tụ	13
3.4	Chính quy	13
3.5	Thói quen tốt	13

1 Mạng nơ ron tích chập

Dịch bởi Phạm Hồng Vinh và Đàm Minh Tiến

1.1 Tổng quan

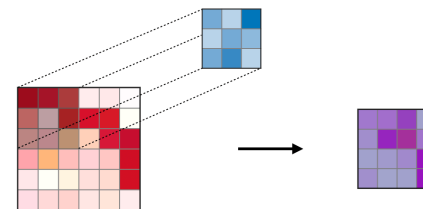
□ **Kiến trúc truyền thống của một mạng CNN** – Mạng neural tích chập (Convolutional neural networks), còn được biết đến với tên CNNs, là một dạng mạng neural được cấu thành bởi các tầng sau:



Tầng tích chập và tầng pooling có thể được hiệu chỉnh theo các siêu tham số (hyperparameters) được mô tả ở những phần tiếp theo.

1.2 Các kiểu tầng

□ **Tầng tích chập (CONV)** – Tầng tích chập (CONV) sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào I theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc F và độ trượt (stride) S . Kết quả đầu ra O được gọi là feature map hay activation map.

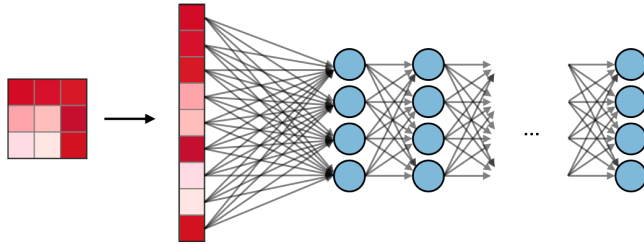


Lưu ý: Bước tích chập cũng có thể được khái quát hóa cả với trường hợp một chiều (1D) và ba chiều (3D).

□ **Pooling (POOL)** – Tầng pooling (POOL) là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

	Max pooling	Average pooling
Kiểu	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng
Minh họa		
Nhận xét	<ul style="list-style-type: none"> - Bảo toàn các đặc trưng đã phát hiện - Được sử dụng thường xuyên 	<ul style="list-style-type: none"> - Giảm kích thước feature map - Được sử dụng trong mạng LeNet

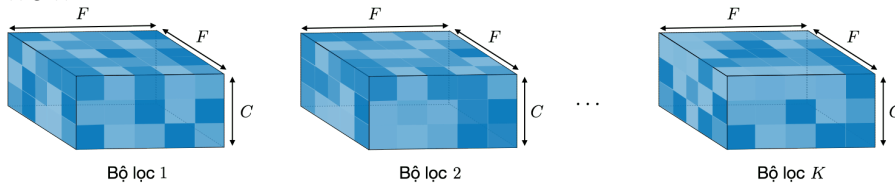
□ **Fully Connected (FC)** – Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



1.3 Các siêu tham số của bộ lọc

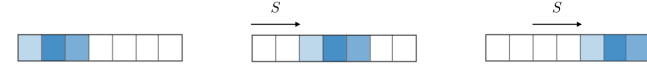
Tầng tích chập chứa các bộ lọc mà rất quan trọng cho ta khi biết ý nghĩa đằng sau các siêu tham số của chúng.

□ **Các chiều của một bộ lọc** – Một bộ lọc kích thước $F \times F$ áp dụng lên đầu vào chứa C kênh (channels) thì có kích thước tổng thể là $F \times F \times C$ thực hiện phép tích chập trên đầu vào kích thước $I \times I \times C$ và cho ra một feature map (hay còn gọi là activation map) có kích thước $O \times O \times 1$.



Lưu ý: Việc áp dụng K bộ lọc có kích thước $F \times F$ cho ra một feature map có kích thước $O \times O \times K$.

□ **Stride** – Đối với phép tích chập hoặc phép pooling, độ trượt S ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.



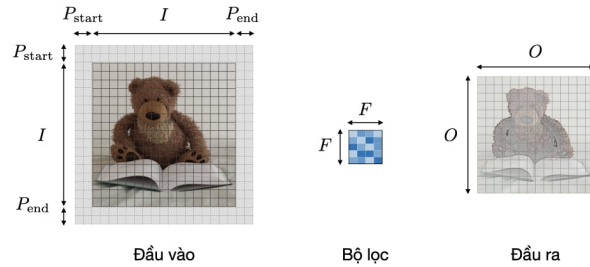
□ **Zero-padding** – Zero-padding là tên gọi của quá trình thêm P số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp mô tả bên dưới:

	Valid	Same	Full
Giá trị	$P = 0$	$P_{\text{start}} = \left\lfloor \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rfloor$ $P_{\text{end}} = \left\lceil \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$	$P_{\text{start}} \in [0, F - 1]$ $P_{\text{end}} = F - 1$
Minh họa			
Mục đích	<ul style="list-style-type: none"> - Không sử dụng padding - Bỏ phép tích chập cuối nếu số chiều không khớp 	<ul style="list-style-type: none"> - Sử dụng padding để làm cho feature map có kích thước $\lceil \frac{I}{S} \rceil$ - Kích thước đầu ra thuận lợi về mặt toán học - Còn được gọi là 'half' padding 	<ul style="list-style-type: none"> - Padding tối đa sao cho các phép tích chập có thể được sử dụng tại các rìa của đầu vào - Bộ lọc 'thấy' được đầu vào từ đầu đến cuối

1.4 Điều chỉnh siêu tham số

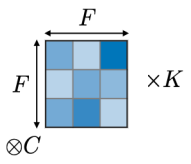
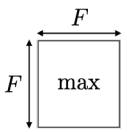
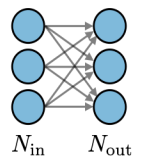
□ **Tính tương thích của tham số trong tầng tích chập** – Bằng cách ký hiệu I là độ dài kích thước đầu vào, F là độ dài của bộ lọc, P là số lượng zero padding, S là độ trượt, ta có thể tính được độ dài O của feature map theo một chiều bằng công thức:

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



Lưu ý: Trong một số trường hợp, $P_{start} = P_{end} \triangleq P$, ta có thể thay thế $P_{start} + P_{end}$ bằng $2P$ trong công thức trên.

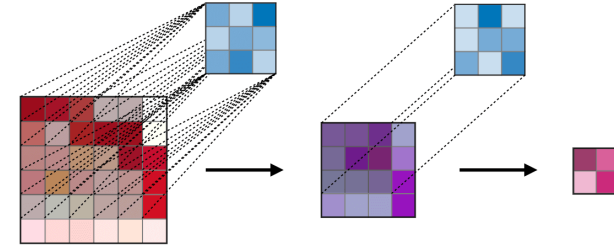
□ **Hiểu về độ phức tạp của mô hình** – Để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng neural tích chập, nó sẽ được tính toán như sau:

	CONV	POOL	FC
Minh họa			
Kích thước đầu vào	$I \times I \times C$	$I \times I \times C$	N_{in}
Kích thước đầu ra	$O \times O \times K$	$O \times O \times C$	N_{out}
Số lượng tham số	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
Lưu ý	<ul style="list-style-type: none"> - Một tham số bias với mỗi bộ lọc - Trong đa số trường hợp, $S < F$ - Một lựa chọn phổ biến cho K là $2C$ 	<ul style="list-style-type: none"> - Phép pooling được áp dụng lên từng kênh (channel-wise) - Trong đa số trường hợp, $S = F$ 	<ul style="list-style-type: none"> - Đầu vào được làm phẳng - Mỗi neuron có một tham số bias - Số neuron trong một tầng FC phụ thuộc vào ràng buộc kết cấu

□ **Trường thụ cảm** – Trường thụ cảm (receptive field) tại tầng k là vùng được ký hiệu $R_k \times R_k$ của đầu vào mà những pixel của activation map thứ k có thể "nhìn thấy". Bằng cách gọi F_j là kích thước bộ lọc của tầng j và S_i là giá trị độ trượt của tầng i và để thuận tiện, ta mặc định $S_0 = 1$, trường thụ cảm của tầng k được tính toán bằng công thức:

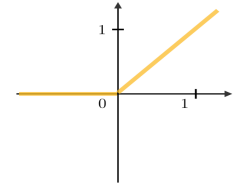
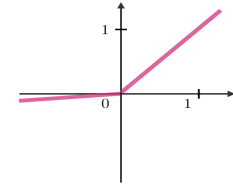
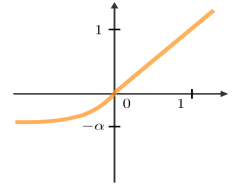
$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i$$

Trong ví dụ bên dưới, ta có $F_1 = F_2 = 3$ và $S_1 = S_2 = 1$, nên cho ra được $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$.



1.5 Các hàm kích hoạt thường gặp

□ **Rectified Linear Unit** – Tầng rectified linear unit (ReLU) là một hàm kích hoạt g được sử dụng trên tất cả các thành phần. Mục đích của nó là tăng tính phi tuyến tính cho mạng. Những biến thể khác của ReLU được tổng hợp ở bảng dưới:




ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ với $\alpha \ll 1$
		
Độ phức tạp phi tuyến tính có thể thông dịch được về mặt sinh học	Gán vấn đề ReLU chết cho những giá trị âm	Khả vi tại mọi nơi

□ **Softmax** – Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị $x \in \mathbb{R}^n$ và cho ra là một vector gồm các xác suất $p \in \mathbb{R}^n$ thông qua một hàm softmax ở cuối kiến trúc. Nó được định nghĩa như sau:

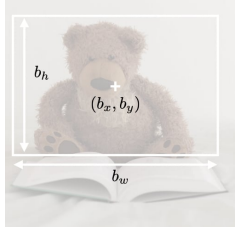
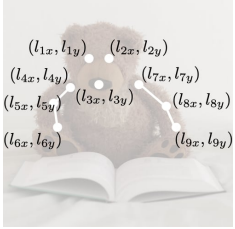
$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{với} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

1.6 Phát hiện vật thể (object detection)

□ **Các kiểu mô hình** – Có 3 kiểu thuật toán nhận diện vật thể chính, vì thế mà bản chất của thứ được dự đoán sẽ khác nhau. Chúng được miêu tả ở bảng dưới:

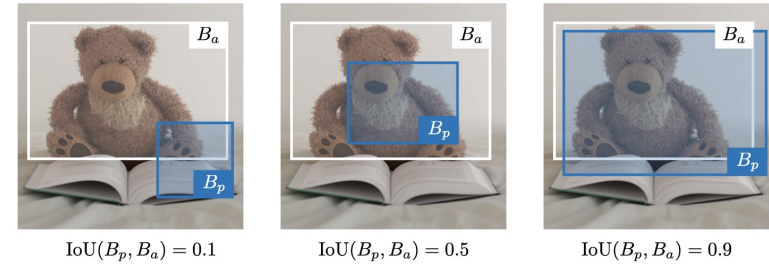
Phân loại hình ảnh	Phân loại cùng với khoanh vùng	Phát hiện
 <p>Gấu bông</p>	 <p>Gấu bông</p>	 <p>Gấu bông</p> <p>Sách</p>
<ul style="list-style-type: none"> - Phân loại một tấm ảnh - Dự đoán xác suất của một vật thể 	<ul style="list-style-type: none"> - Phát hiện một vật thể trong ảnh - Dự đoán xác suất của vật thể và định vị nó 	<ul style="list-style-type: none"> - Phát hiện nhiều vật thể trong cùng một tấm ảnh - Dự đoán xác suất của các vật thể và định vị chúng
CNN cổ điển	YOLO đơn giản hóa, R-CNN	YOLO, R-CNN

□ **Phát hiện** – Trong bối cảnh phát hiện (detection) vật thể, những phương pháp khác nhau được áp dụng tùy thuộc vào liệu chúng ta chỉ muốn định vị vật thể hay phát hiện được những hình dạng phức tạp hơn trong tấm ảnh. Hai phương pháp chính được tổng hợp ở bảng dưới:

Phát hiện hộp giới hạn (bounding box)	Phát hiện landmark
Phát hiện phần trong ảnh mà có sự xuất hiện của vật thể	<ul style="list-style-type: none"> - Phát hiện hình dạng và đặc điểm của một đối tượng (vd: mắt) - Nhiều hạt
	
Hộp có tọa độ trung tâm (b_x, b_y) , chiều cao b_h và chiều rộng b_w	Các điểm tương quan $(l_{1x}, l_{1y}), \dots, (l_{nx}, l_{ny})$

□ **Intersection over Union** – Tỷ lệ vùng giao trên vùng hợp, còn được biết đến là IoU, là một hàm định lượng vị trí B_p của hộp giới hạn dự đoán được định vị đúng như thế nào so với hộp giới hạn thực tế B_a . Nó được định nghĩa:

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

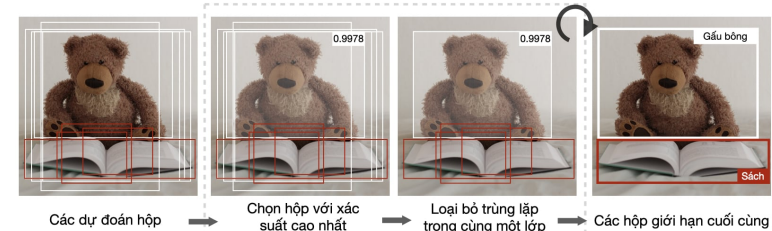


Lưu ý: ta luôn có $\text{IoU} \in [0, 1]$. Để thuận tiện, một hộp giới hạn B_p được cho là khá tốt nếu $\text{IoU}(B_p, B_a) \geq 0.5$.

□ **Anchor boxes** – Hộp mỏ neo là một kỹ thuật được dùng để dự đoán những hộp giới hạn nằm chồng lên nhau. Trong thực nghiệm, mạng được phép dự đoán nhiều hơn một hộp cùng một lúc, trong đó mỗi dự đoán được giới hạn theo một tập những tính chất hình học cho trước. Ví dụ, dự đoán đầu tiên có khả năng là một hộp hình chữ nhật có hình dạng cho trước, trong khi dự đoán thứ hai sẽ là một hộp hình chữ nhật nữa với hình dạng hình học khác.

□ **Non-max suppression** – Kỹ thuật non-max suppression hướng tới việc loại bỏ những hộp giới hạn bị trùng chồng lên nhau của cùng một đối tượng bằng cách chọn chiếc hộp có tính đặc trưng nhất. Sau khi loại bỏ tất cả các hộp có xác suất dự đoán nhỏ hơn 0.6, những bước tiếp theo được lặp lại khi vẫn còn tồn tại những hộp khác.

- Bước 1: Chọn chiếc hộp có xác suất dự đoán lớn nhất.
- Bước 2: Loại bỏ những hộp có $\text{IoU} \geq 0.5$ với hộp đã chọn.



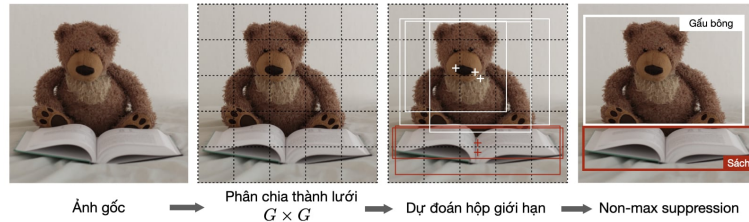
□ **YOLO** – You Only Look Once (YOLO) là một thuật toán phát hiện vật thể thực hiện những bước sau:

- Bước 1: Phân chia tấm ảnh đầu vào thành một lưới $G \times G$.
- Bước 2: Với mỗi lưới, chạy một mạng CNN dự đoán y có dạng sau:

$$y = \underbrace{[p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots]}_{\text{lặp lại } k \text{ lần}}^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

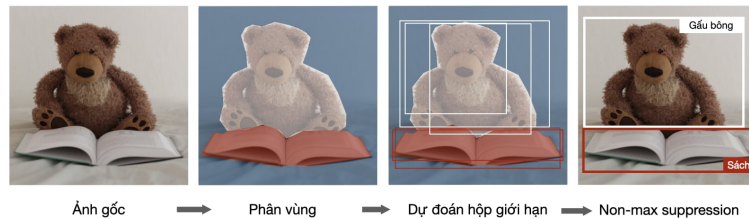
với p_c là xác suất dự đoán được một vật thể, b_x, b_y, b_h, b_w là những thuộc tính của hộp giới hạn được dự đoán, c_1, \dots, c_p là biểu diễn one-hot của việc lớp nào trong p các lớp được dự đoán, và k là số lượng các hộp mỏ neo.

- Bước 3: Chạy thuật toán non-max suppression để loại bỏ bất kỳ hộp giới hạn có khả năng bị trùng lặp.



Lưu ý: khi $p_c = 0$, thì mạng không phát hiện bất kỳ vật thể nào. Trong trường hợp đó, Các dự đoán liên quan b_x, \dots, c_p sẽ bị bỏ đi.

□ **R-CNN** – Region with Convolutional Neural Networks (R-CNN) là một thuật toán phát hiện vật thể mà đầu tiên phân chia ảnh thành các vùng để tìm các hộp giới hạn có khả năng liên quan cao rồi chạy một thuật toán phát hiện để tìm những thứ có khả năng cao là vật thể trong những hộp giới hạn đó.



Lưu ý: mặc dù thuật toán gốc có chi phí tính toán cao và chậm, những kiến trúc mới đã có thể cho phép thuật toán này chạy nhanh hơn, như là Fast R-CNN và Faster R-CNN.

1.6.1 Xác nhận khuôn mặt và nhận diện khuôn mặt

□ **Các kiểu mô hình** – Hai kiểu mô hình chính được tổng hợp trong bảng dưới:

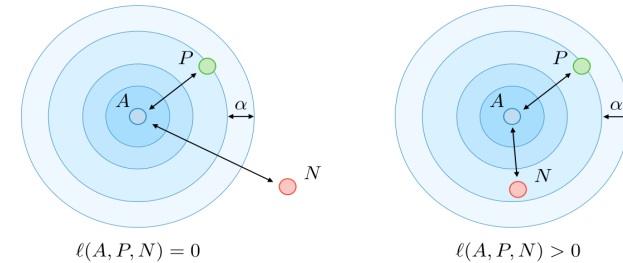
Xác nhận khuôn mặt	Nhận diện khuôn mặt
<ul style="list-style-type: none"> - Có đúng người không? - Tra cứu một-một 	<ul style="list-style-type: none"> - Đây có phải là 1 trong K người trong cơ sở dữ liệu không? - Tra cứu một với tất cả
<p>Truy vấn</p> <p>Tham vấn</p>	<p>Truy vấn</p> <p>Cơ sở dữ liệu</p>

□ **One Shot Learning** – One Shot Learning là một thuật toán xác minh khuôn mặt sử dụng một tập huấn luyện hạn chế để học một hàm similarity nhằm ước lượng sự khác nhau giữa hai tấm hình. Hàm này được áp dụng cho hai tấm ảnh thường được ký hiệu $d(\text{image 1}, \text{image 2})$.

□ **Siamese Network** – Siamese Networks hướng tới việc học cách mã hóa tấm ảnh để rồi định lượng sự khác nhau giữa hai tấm ảnh. Với một tấm ảnh đầu vào $x^{(i)}$, đầu ra được mã hóa thường được ký hiệu là $f(x^{(i)})$.

□ **Triplet loss** – Triplet loss ℓ là một hàm mất mát được tính toán dựa trên biểu diễn nhúng của bộ ba hình ảnh A (mỏ neo), P (dương tính) và N (âm tính). Ảnh mỏ neo và ảnh dương tính đều thuộc một lớp, trong khi đó ảnh âm tính thuộc về một lớp khác. Bằng các gọi $\alpha \in \mathbb{R}^+$ là tham số margin, hàm mất mát này được định nghĩa như sau:

$$\ell(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$



1.6.2 Neural style transfer

□ **Ý tưởng** – Mục tiêu của neural style transfer là tạo ra một ảnh G dựa trên một nội dung C và một phong cách S .



□ **Tầng kích hoạt** – Trong một tầng l cho trước, tầng kích hoạt được ký hiệu $a^{[l]}$ và có các chiều là $n_H \times n_w \times n_c$

□ **Hàm mất mát nội dung** – Hàm mất mát nội dung $J_{\text{content}}(C, G)$ được sử dụng để xác định nội dung của ảnh được tạo G khác biệt với nội dung gốc trong ảnh C . Nó được định nghĩa như dưới đây:

$$J_{\text{content}}(C, G) = \frac{1}{2} \|a^{[l](C)} - a^{[l](G)}\|^2$$

□ **Ma trận phong cách** – Ma trận phong cách $G^{[l]}$ của một tầng cho trước l là một ma trận Gram mà mỗi thành phần $G_{kk'}^{[l]}$ của ma trận xác định sự tương quan giữa kênh k và kênh k' . Nó được định nghĩa theo tầng kích hoạt $a^{[l]}$ như sau:

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{ijk}^{[l]} a_{ijk'}^{[l]}$$

Lưu ý: ma trận phong cách cho ảnh phong cách và ảnh được tạo được ký hiệu tương ứng là $G^{[l](S)}$ và $G^{[l](G)}$.

□ **Hàm mất mát phong cách** – Hàm mất mát phong cách $J_{\text{style}}(S, G)$ được sử dụng để xác định sự khác biệt về phong cách giữa ảnh được tạo G và ảnh phong cách S . Nó được định nghĩa như sau:

$$J_{\text{style}}^{[l]}(S, G) = \frac{1}{(2n_H n_w n_c)^2} \|G^{[l](S)} - G^{[l](G)}\|_F^2 = \frac{1}{(2n_H n_w n_c)^2} \sum_{k, k'=1}^{n_c} \left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \right)^2$$

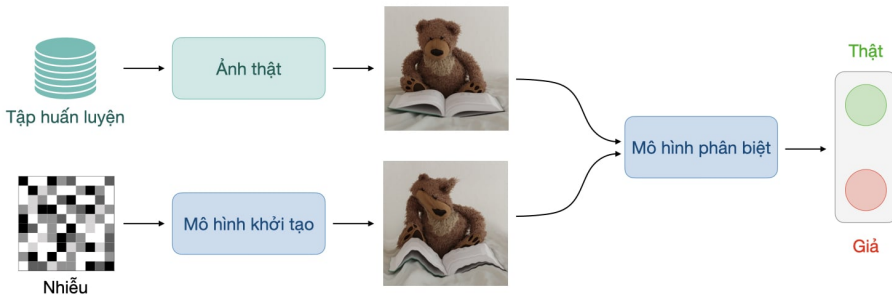
□ **Hàm mất mát tổng quát** – Hàm mất mát tổng quát được định nghĩa là sự kết hợp của hàm mất mát nội dung và hàm mất mát phong cách, độ quan trọng của chúng được xác định bởi hai tham số α, β , như dưới đây:

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

Lưu ý: giá trị của α càng lớn dẫn tới việc mô hình sẽ quan tâm hơn cho nội dung, trong khi đó, giá trị của β càng lớn sẽ khiến nó quan tâm hơn đến phong cách.

1.6.3 Những kiến trúc sử dụng computational tricks

□ **Generative Adversarial Network** – Generative adversarial networks, hay còn được gọi là GAN, là sự kết hợp giữa mô hình khởi tạo và mô hình phân biệt, khi mà mô hình khởi tạo cố gắng tạo ra hình ảnh đầu ra chân thực nhất, sau đó được đưa vào mô hình phân biệt, mà mục tiêu của nó là phân biệt giữa ảnh được tạo và ảnh thật.



Lưu ý: có nhiều loại GAN khác nhau bao gồm từ văn bản thành ảnh, sinh nhạc và tổ hợp.

□ **ResNet** – Kiến trúc Residual Network (hay còn gọi là ResNet) sử dụng những khối residual (residual blocks) cùng với một lượng lớn các tầng để giảm lỗi huấn luyện. Những khối residual có những tính chất sau đây:

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

□ **Inception Network** – Kiến trúc này sử dụng những inception module và hướng tới việc thử các tầng tích chập khác nhau để tăng hiệu suất thông qua sự đa dạng của các feature. Cụ thể, kiến trúc này sử dụng thủ thuật tăng tích chập 1×1 để hạn chế gánh nặng tính toán.

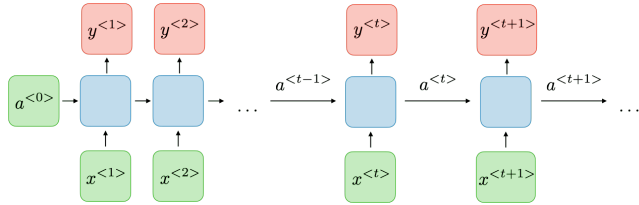
★ ★ ★

2 Mạng nơ-ron hồi quy

Dịch bởi Trần Tuấn Anh, Đàm Minh Tiến, Hung Nguyễn và Nguyễn Trí Minh

2.1 Tổng quan

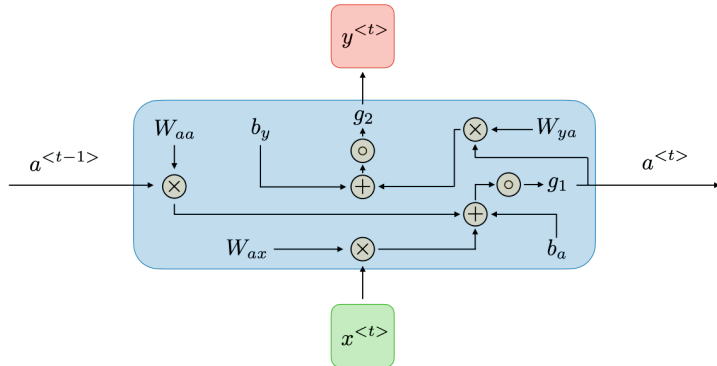
□ **Kiến trúc của một mạng RNN truyền thống** – Các mạng neural hồi quy, còn được biến đến như là RNNs, là một lớp của mạng neural cho phép đầu ra được sử dụng như đầu vào trong khi có các trạng thái ẩn. Thông thường là như sau:



Tại mỗi bước t , giá trị kích hoạt $a^{<t>}$ và đầu ra $y^{<t>}$ được biểu diễn như sau:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{và} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

với $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ là các hệ số được chia sẻ tạm thời và g_1, g_2 là các hàm kích hoạt.



Ưu và nhược điểm của một kiến trúc RNN thông thường được tổng kết ở bảng dưới đây:

Ưu điểm	Hạn chế
<ul style="list-style-type: none"> - Khả năng xử lý đầu vào với bất kì độ dài nào - Kích cỡ mô hình không tăng theo kích cỡ đầu vào - Quá trình tính toán sử dụng các thông tin cũ - Trọng số được chia sẻ trong suốt thời gian 	<ul style="list-style-type: none"> - Tính toán chậm - Khó để truy cập các thông tin từ một khoảng thời gian dài trước đây - Không thể xem xét bất kì đầu vào sau này nào cho trạng thái hiện tại

□ **Ứng dụng của RNNs** – Các mô hình RNN hầu như được sử dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên và ghi nhận tiếng nói. Các ứng dụng khác được tổng kết trong bảng dưới đây:

Các loại RNN	Hình minh họa	Ví dụ
Một-Một $T_x = T_y = 1$		Mạng neural truyền thống
Một-nhiều $T_x = 1, T_y > 1$		Sinh nhạc
Nhiều-một $T_x > 1, T_y = 1$		Phân loại ý kiến
Nhiều-nhiều $T_x = T_y$		Ghi nhận thực thể tên
Nhiều-nhiều $T_x \neq T_y$		Dịch máy

□ **Hàm mất mát** – Trong trường hợp của mạng neural hồi quy, hàm mất mát \mathcal{L} của tất cả các bước thời gian được định nghĩa dựa theo mất mát ở mọi thời điểm như sau:

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

□ **Lan truyền ngược theo thời gian** – Lan truyền ngược được hoàn thành ở mỗi một thời

điểm cụ thể. Ở bước T , đạo hàm của hàm mất mát \mathcal{L} với ma trận trọng số W được biểu diễn như sau:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(T)}}{\partial W} \Big|_{(t)}$$

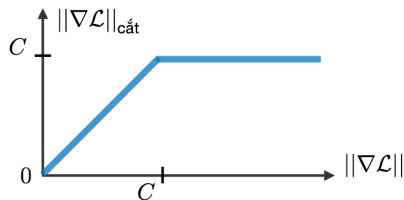
2.2 Xử lý phụ thuộc dài hạn

□ **Các hàm kích hoạt thường dùng** – Các hàm kích hoạt thường dùng trong các modules RNN được miêu tả như sau:

Sigmoid	Tanh	ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$

□ **Vanishing/exploding gradient** – Hiện tượng vanishing và exploding gradient thường gặp trong ngữ cảnh của RNNs. Lí do tại sao chúng thường xảy ra đó là khó để có được sự phụ thuộc dài hạn vì multiplicative gradient có thể tăng/giảm theo hàm mũ tương ứng với số lượng các tầng.

□ **Gradient clipping** – Là một kĩ thuật được sử dụng để giải quyết vấn đề exploding gradient xảy ra khi thực hiện lan truyền ngược. Bằng việc giới hạn giá trị lớn nhất cho gradient, hiện tượng này sẽ được kiểm soát trong thực tế.



□ **Các loại cổng** – Để giải quyết vấn đề vanishing gradient, các cổng cụ thể được sử dụng trong một vài loại RNNs và thường có mục đích rõ ràng. Chúng thường được kí hiệu là Γ và bằng với:

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)$$

Với W, U, b là các hệ số của một cổng và σ là hàm sigmoid. Các loại chính được tổng kết ở bảng dưới đây:

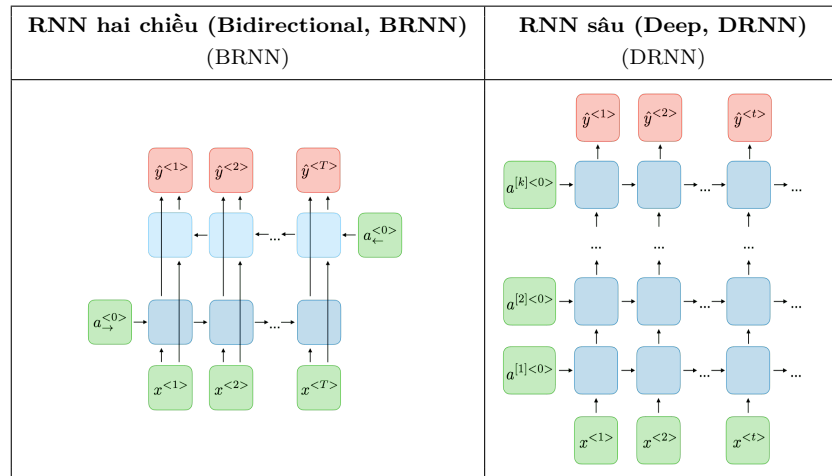
Loại cổng	Vai trò	Được sử dụng trong
Cổng cập nhật Γ_u	Dữ liệu cũ nên có tầm quan trọng như thế nào ở hiện tại?	GRU, LSTM
Cổng relevance Γ_r	Bỏ qua thông tin phía trước?	GRU, LSTM
Cổng quên Γ_f	Xoá ô hay không xoá?	LSTM
Cổng ra Γ_o	Biểu thị một ô ở mức độ bao nhiêu?	LSTM

□ **GRU/LSTM** – Gated Recurrent Unit (GRU) và Các đơn vị bộ nhớ dài-ngắn hạn (LSTM) đối phó với vấn đề vanishing gradient khi gặp phải bằng mạng RNNs truyền thống, với LSTM là sự tổng quát của GRU. Phía dưới là bảng tổng kết các phương trình đặc trưng của mỗi kiến trúc:

	Gated Recurrent Unit (GRU) (GRU)	Bộ nhớ dài-ngắn hạn (LSTM) (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Các phụ thuộc		

Chú ý: kí hiệu \star chỉ phép nhân từng phần tử với nhau giữa hai vectors.

□ **Các biến thể của RNNs** – Bảng dưới đây tổng kết các kiến trúc thường được sử dụng khác của RNN:



2.3 Học từ đại diện

Trong phần này, chúng ta kí hiệu V là tập từ vựng và $|V|$ là kích cỡ của nó.

2.3.1 Giải thích và các kí hiệu

□ **Các kĩ thuật biểu diễn** – Có hai cách chính để biểu diễn từ được tổng kết ở bảng bên dưới:

Biểu diễn 1-hot	Word embedding
<ul style="list-style-type: none"> - Lưu ý o_w - Tiếp cận Naive, không có thông tin chung 	<ul style="list-style-type: none"> - Lưu ý e_w - Xem xét độ tương đồng của các từ

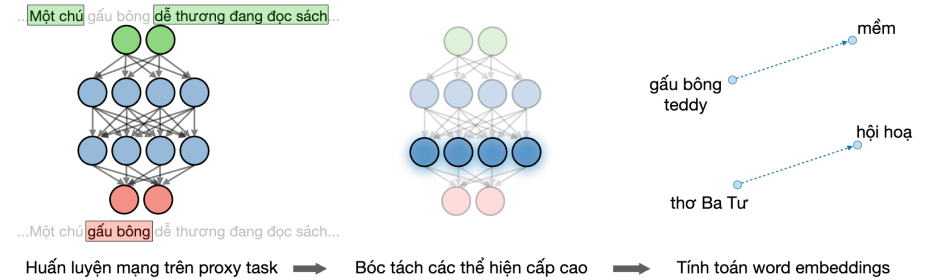
□ **Embedding matrix** – Cho một từ w , embedding matrix E là một ma trận tham chiếu thể hiện 1-hot o_w của nó với embedding e_w của nó như sau:

$$e_w = E o_w$$

Chú ý: học embedding matrix có thể hoàn thành bằng cách sử dụng các mô hình target/context likelihood.

2.3.2 Word embeddings

□ **Word2vec** – Word2vec là một framework tập trung vào việc học word embeddings bằng cách ước lượng khả năng mà một từ cho trước được bao quanh bởi các từ khác. Các mô hình phổ biến bao gồm skip-gram, negative sampling và CBOW.



□ **Skip-gram** – Mô hình skip-gram word2vec là một task học có giám sát, nó học các word embeddings bằng cách đánh giá khả năng của bất kì target word t cho trước nào xảy ra với context word c . Bằng việc kí hiệu θ_t là tham số đi kèm với t , xác suất $P(t|c)$ được tính như sau:

$$P(t|c) = \frac{\exp(\theta_t^T e_c)}{\sum_{j=1}^{|V|} \exp(\theta_j^T e_c)}$$

Chú ý: Cộng tổng tất cả các từ vựng trong mẫu số của phần softmax khiến mô hình này tốn nhiều chi phí tính toán. CBOW là một mô hình word2vec khác sử dụng các từ xung quanh để dự đoán một từ cho trước.

□ **Negative sampling** – Nó là một tập của các bộ phân loại nhị phân sử dụng logistic regressions với mục tiêu là đánh giá khả năng mà một ngữ cảnh cho trước và các target words cho trước có thể xuất hiện đồng thời, với các mô hình đang được huấn luyện trên các tập của k negative examples và 1 positive example. Cho trước context word c và target word t , dự đoán được thể hiện bởi:

$$P(y = 1|c, t) = \sigma(\theta_t^T e_c)$$

Chú ý: phương thức này tốn ít chi phí tính toán hơn mô hình skip-gram.

□ **GloVe** – Mô hình GloVe, viết tắt của global vectors for word representation, nó là một kĩ thuật word embedding sử dụng ma trận đồng xuất hiện X với mỗi $X_{i,j}$ là số lần mà từ đích (target) i xuất hiện tại ngữ cảnh j . Cost function J của nó như sau:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{|V|} f(X_{i,j})(\theta_i^T e_j + b_i + b'_j - \log(X_{i,j}))^2$$

f là hàm trong số với $X_{i,j} = 0 \implies f(X_{i,j}) = 0$. Với tính đối xứng mà e và θ có được trong mô hình này, word embedding cuối cùng $e_w^{(\text{final})}$ được định nghĩa như sau:

$$e_w^{(\text{final})} = \frac{e_w + \theta_w}{2}$$

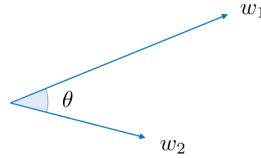
Chú ý: Các phần tử riêng của các word embedding học được không nhất thiết là phải thông dịch được.

2.4 So sánh các từ

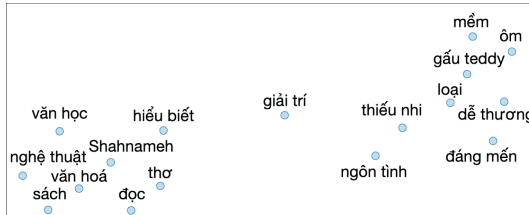
□ **Độ tương đồng cosine** – Độ tương đồng cosine giữa các từ w_1 và w_2 được trình bày như sau:

$$\text{similarity} = \frac{w_1 \cdot w_2}{\|w_1\| \|w_2\|} = \cos(\theta)$$

Chú ý: θ là góc giữa các từ w_1 và w_2 .



□ **t-SNE** – t-SNE (t-distributed Stochastic Neighbor Embedding) là một kĩ thuật nhằm giảm đi số chiều của không gian embedding. Trong thực tế, nó thường được sử dụng để trực quan hoá các word vectors trong không gian 2 chiều (2D).



2.5 Mô hình ngôn ngữ

□ **Tổng quan** – Một mô hình ngôn ngữ sẽ dự đoán xác suất của một câu $P(y)$.

□ **Mô hình n-gram** – Mô hình này là cách tiếp cận naive với mục đích định lượng xác suất mà một biểu hiện xuất hiện trong văn bản bằng cách đếm số lần xuất hiện của nó trong tập dữ liệu huấn luyện.

□ **Độ hỗn tạp** – Các mô hình ngôn ngữ thường được đánh giá dựa theo độ đo hỗn tạp, cũng được biết đến là PP, có thể được hiểu như là nghịch đảo xác suất của tập dữ liệu được chuẩn hoá bởi số lượng các từ T . Độ hỗn tạp càng thấp thì càng tốt và được định nghĩa như sau:

$$PP = \prod_{t=1}^T \left(\frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}} \right)^{\frac{1}{T}}$$

Chú ý: PP thường được sử dụng trong t-SNE.

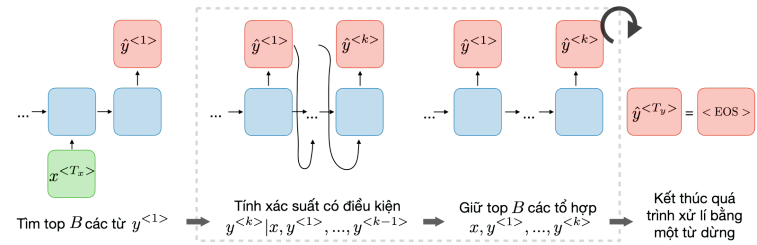
2.6 Dịch máy

□ **Tổng quan** – Một mô hình dịch máy tương tự với mô hình ngôn ngữ ngoại trừ nó có một mạng encoder được đặt phía trước. Vì lý do này, đôi khi nó còn được biết đến là mô hình ngôn ngữ có điều kiện. Mục tiêu là tìm một câu văn y như sau:

$$y = \arg \max_{y^{<1>, \dots, y^{<T_y>}} P(y^{<1>, \dots, y^{<T_y>} | x)$$

□ **Tìm kiếm Beam** – Nó là một giải thuật tìm kiếm heuristic được sử dụng trong dịch máy và ghi nhận tiếng nói để tìm câu văn y đúng nhất tương ứng với đầu vào x .

- Bước 1: Tìm top B các từ $y^{<1>}$
- Bước 2: Tính xác suất có điều kiện $y^{<k>} | x, y^{<1>, \dots, y^{<k-1>}}$
- Bước 3: Giữ top B các tổ hợp $x, y^{<1>, \dots, y^{<k>}}$



Chú ý: nếu độ rộng của beam được thiết lập là 1, thì nó tương đương với tìm kiếm tham lam naive.

□ **Độ rộng Beam** – Độ rộng beam B là một tham số của giải thuật tìm kiếm beam. Các giá trị lớn của B tạo ra kết quả tốt hơn nhưng với hiệu năng thấp hơn và lượng bộ nhớ sử dụng sẽ tăng.

□ **Chuẩn hoá độ dài** – Để cải thiện tính ổn định, beam search thường được áp dụng mục tiêu chuẩn hoá sau, thường được gọi là mục tiêu chuẩn hoá log-likelihood, được định nghĩa như sau:

$$\text{Objective} = \frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log \left[p(y^{<t>} | x, y^{<1>, \dots, y^{<t-1>}}) \right]$$

Chú ý: tham số α có thể được xem như là softener, và giá trị của nó thường nằm trong đoạn 0.5 và 1.

□ **Phân tích lỗi** – Khi có được một bản dịch tối \hat{y} , chúng ta có thể tự hỏi rằng tại sao chúng ta không có được một kết quả dịch tốt y^* bằng việc thực hiện việc phân tích lỗi như sau:

Trường hợp	$P(y^* x) > P(\hat{y} x)$	$P(y^* x) \leq P(\hat{y} x)$
Nguyên nhân sâu xa	Lỗi Beam search	Lỗi RNN
Biện pháp khắc phục	Tăng beam width	- Thử kiến trúc khác - Chính quy - Lấy nhiều dữ liệu hơn

□ **Điểm Bleu** – Bilingual evaluation understudy (bleu) score định lượng mức độ tốt của dịch máy bằng cách tính một độ tương đồng dựa trên dự đoán n -gram. Nó được định nghĩa như sau:

$$\text{bleu score} = \exp \left(\frac{1}{n} \sum_{k=1}^n p_k \right)$$

với p_n là bleu score chỉ trên n -gram được định nghĩa như sau:

$$p_n = \frac{\sum_{n\text{-gram} \in \hat{y}} \text{count}_{\text{clip}}(n\text{-gram})}{\sum_{n\text{-gram} \in \hat{y}} \text{count}(n\text{-gram})}$$

Chú ý: một mức phạt ngắn có thể được áp dụng với các dự đoán dịch ngắn để tránh việc làm thổi phồng giá trị bleu score.

2.7 Chú ý

□ **Attention model** – Mô hình này cho phép một RNN tập trung vào các phần cụ thể của đầu vào được xem xét là quan trọng, nó giúp cải thiện hiệu năng của mô hình kết quả trong thực tế. Bằng việc kí hiệu $\alpha^{<t,t'>}$ là mức độ chú ý mà đầu ra $y^{<t>}$ nên có đối với hàm kích hoạt $a^{<t'>}$ và $c^{<t>}$ là ngữ cảnh ở thời điểm t , chúng ta có:

$$c^{<t>} = \sum_{t'} \alpha^{<t,t'>} a^{<t'>} \quad \text{với} \quad \sum_{t'} \alpha^{<t,t'>} = 1$$

Chú ý: Các attention scores thường được sử dụng trong chú thích ảnh và dịch máy.



Một chú gấu bông dễ thương đang đọc bài văn Ba Tư



Một chú gấu bông dễ thương đang đọc bài văn Ba Tư

□ **Attention weight** – Sự chú ý mà đầu ra $y^{<t>}$ nên có với hàm kích hoạt $a^{<t'>}$ với $\alpha^{<t,t'>}$ được tính như sau:

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t''=1}^{T_x} \exp(e^{<t,t''>})}$$

Chú ý: độ phức tạp tính toán là một phương trình bậc hai đối với T_x .

* * *

3 Mẹo và thủ thuật

Dịch bởi Hoàng Minh Tuấn, Trần Tuấn Anh và Đàm Minh Tiến

3.1 Xử lí dữ liệu

□ **Data augmentation** – Các mô hình học sâu thường cần rất nhiều dữ liệu để có thể được huấn luyện đúng cách. Việc sử dụng các kỹ thuật Data augmentation là khá hữu ích để có thêm nhiều dữ liệu hơn từ tập dữ liệu hiện thời. Những kĩ thuật chính được tóm tắt trong bảng dưới đây. Chính xác hơn, với hình ảnh đầu vào sau đây, đây là những kỹ thuật mà chúng ta có thể áp dụng:

Hình gốc	Lật	Xoay	Cắt ngẫu nhiên
- Hình ảnh không có bất kỳ sửa đổi nào	- Lật đối với một trục mà ý nghĩa của hình ảnh được giữ nguyên	- Xoay với một góc nhỏ - Mô phỏng hiệu chỉnh đường chân trời không chính xác	- Lấy nét ngẫu nhiên trên một phần của hình ảnh - Một số cách cắt ngẫu nhiên có thể được thực hiện trên một hàng

Dịch chuyển màu	Thêm nhiễu	Mất mát thông tin	Thay đổi độ tương phản
- Các sắc thái của RGB bị thay đổi một chút - Captures noise có thể xảy ra khi tiếp xúc với ánh sáng nhẹ	- Bổ sung nhiễu - Chịu được sự thay đổi chất lượng của các yếu tố đầu vào	- Các phần của hình ảnh bị bỏ qua - Mô phỏng khả năng mất của các phần trong hình ảnh	- Thay đổi độ sáng - Kiểm soát sự khác biệt do phơi sáng theo thời gian trong ngày

□ **Chuẩn hóa batch** – Đây là một bước của hyperparameter γ, β chuẩn hóa tập dữ liệu $\{x_i\}$. Bằng việc kí hiệu μ_B, σ_B^2 là trung bình và phương sai của tập dữ liệu ta muốn chuẩn hóa, nó được thực hiện như sau:

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Thường hoàn thành sau một lớp fully connected/nhân chập và trước lớp phi tuyến tính và mục đích cho phép tốc độ học cao hơn và giảm thiểu sự phụ thuộc vào khởi tạo

3.2 Huấn luyện mạng neural

3.2.1 Định nghĩa

□ **Epoch** – Trong ngữ cảnh huấn luyện mô hình, epoch là một thuật ngữ chỉ một vòng lặp mà mô hình sẽ duyệt toàn bộ tập dữ liệu huấn luyện để cập nhật trọng số của nó.

□ **Mini-batch gradient descent** – Trong quá trình huấn luyện, việc cập nhật trọng số thường không dựa trên toàn bộ tập huấn luyện cùng một lúc do độ phức tạp tính toán hoặc một điểm dữ liệu nhiều. Thay vào đó, bước cập nhật được thực hiện trên các lô nhỏ (mini-batch), trong đó số lượng điểm dữ liệu trong một lô (batch) là một siêu tham số (hyperparameter) mà chúng ta có thể điều chỉnh.

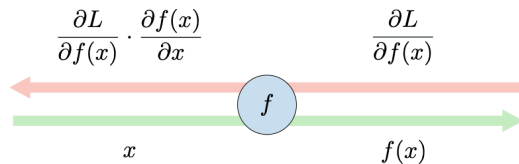
□ **Hàm mất mát** – Để định lượng cách thức một mô hình nhất định thực hiện, hàm mất mát L thường được sử dụng để đánh giá mức độ đầu ra thực tế y được dự đoán chính xác bởi đầu ra của mô hình là z .

□ **Cross-entropy loss** – Khi áp dụng phân loại nhị phân (binary classification) trong các mạng neural, cross-entropy loss $L(z, y)$ thường được sử dụng và được định nghĩa như sau:

$$L(z, y) = - \left[y \log(z) + (1 - y) \log(1 - z) \right]$$

3.2.2 Tìm trọng số tối ưu

□ **Lan truyền ngược** – Lan truyền ngược (backpropagation) là một phương thức để cập nhật các trọng số trong mạng neural bằng cách tính toán đầu ra thực tế và đầu ra mong muốn. Đạo hàm tương ứng với từng trọng số w được tính bằng quy tắc chuỗi.

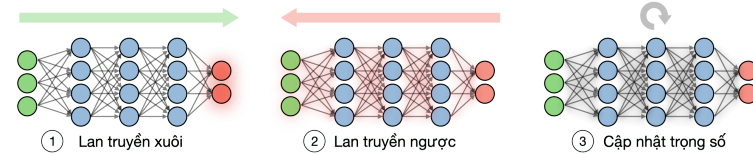


Sử dụng phương thức này, mỗi trọng số được cập nhật theo quy luật:

$$w \leftarrow w - \alpha \frac{\partial L(z, y)}{\partial w}$$

□ **Cập nhật trọng số** – Trong một mạng neural, các trọng số được cập nhật như sau:

- Bước 1: Lấy một loạt dữ liệu huấn luyện và thực hiện lan truyền xuôi (forward propagation) để tính toán mất mát
- Bước 2: Lan truyền ngược mất mát để có được độ dốc (gradient) của mất mát theo từng trọng số
- Bước 3: Sử dụng độ dốc để cập nhật trọng số của mạng.



3.3 Tinh chỉnh tham số

3.3.1 Khởi tạo trọng số

□ **Khởi tạo Xavier** – Thay vì khởi tạo trọng số một cách ngẫu nhiên, khởi tạo Xavier cho chúng ta một cách khởi tạo trọng số dựa trên một đặc tính độc nhất của kiến trúc mô hình.

□ **Transfer learning** – Huấn luyện một mô hình deep learning đòi hỏi nhiều dữ liệu và quan trọng hơn là rất nhiều thời gian. Sẽ rất hữu ích để tận dụng các trọng số đã được huấn luyện trước trên các bộ dữ liệu rất lớn mất vài ngày / tuần để huấn luyện và tận dụng nó cho trường hợp của chúng ta. Tùy thuộc vào lượng dữ liệu chúng ta có trong tay, đây là các cách khác nhau để tận dụng điều này:

Kích thước tập huấn luyện	Mô phỏng	Giải thích
Nhỏ		Cố định các tầng
Trung bình		huấn luyện trọng số trên hàm softmax
Lớn		Cố định hầu hết các tầng

3.3.2 Tối ưu hội tụ

□ **Tốc độ học** – Tốc độ học (learning rate), thường được kí hiệu là α hoặc đôi khi là η , cho biết mức độ thay đổi của các trọng số sau mỗi lần được cập nhật. Nó có thể được cố định hoặc thay đổi thích ứng. Phương thức phổ biến nhất hiện nay là Adam, đây là phương thức thích nghi với tốc độ học.

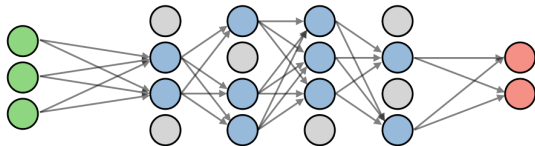
□ **Tốc độ học thích nghi** – Để cho tốc độ học thay đổi khi huấn luyện một mô hình có thể giảm thời gian huấn luyện và cải thiện giải pháp tối ưu số. Trong khi tối ưu hóa Adam (Adam optimizer) là kỹ thuật được sử dụng phổ biến nhất, nhưng những phương pháp khác cũng có thể hữu ích. Chúng được tổng kết trong bảng dưới đây:

Phương thức	Giải thích	Cập nhật của w	Cập nhật của b
Momentum	- Làm giảm dao động - Cải thiện SGD - 2 tham số để tinh chỉnh	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	- lan truyền Root Mean Square - Thuật toán tăng tốc độ học bằng kiểm soát dao động	$w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \leftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	- Ước lượng Adaptive Moment - Các phương pháp phổ biến - 4 tham số để tinh chỉnh	$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \leftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

Chú ý: những phương pháp khác bao gồm Adadelat, Adagrad và SGD.

3.4 Chính quy

□ **Dropout** – Dropout là một kỹ thuật được sử dụng trong các mạng neural để tránh overfitting trên tập huấn luyện bằng cách loại bỏ các nơ-ron (neural) với xác suất $p > 0$. Nó giúp mô hình không bị phụ thuộc quá nhiều vào một tập thuộc tính nào đó.

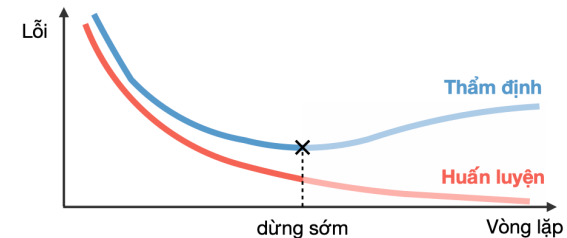


Ghi chú: hầu hết các frameworks deep learning đều có thiết lập dropout thông qua biến tham số 'keep' $1 - p$.

□ **Weight regularization** – Để đảm bảo rằng các trọng số không quá lớn và mô hình không bị overfitting trên tập huấn luyện, các kỹ thuật chính quy (regularization) thường được thực hiện trên các trọng số của mô hình. Những kỹ thuật chính được tổng kết trong bảng dưới đây:

LASSO	Ridge	Elastic Net
- Giảm hệ số về 0 - Tốt cho việc lựa chọn biến	Làm cho hệ số nhỏ hơn	Đánh đổi giữa việc lựa chọn biến và hệ số nhỏ
$\dots + \lambda \ \theta\ _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda \ \theta\ _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda \left[(1 - \alpha) \ \theta\ _1 + \alpha \ \theta\ _2^2 \right]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

□ **Dừng sớm** – Kỹ thuật chính quy này sẽ dừng quá trình huấn luyện một khi mất mát trên tập kiểm định (validation) đạt đến một ngưỡng nào đó hoặc bắt đầu tăng.



3.5 Thói quen tốt

□ **Overfitting batch nhỏ** – Khi gỡ lỗi một mô hình, khá hữu ích khi thực hiện các kiểm tra nhanh để xem liệu có bất kỳ vấn đề lớn nào với kiến trúc của mô hình đó không. Đặc biệt, để đảm bảo rằng mô hình có thể được huấn luyện đúng cách, một batch nhỏ (mini-batch) được truyền vào bên trong mạng để xem liệu nó có thể overfit không. Nếu không, điều đó có nghĩa là mô hình quá phức tạp hoặc không đủ phức tạp để thậm chí overfit trên batch nhỏ (mini-batch), chứ đừng nói đến một tập huấn luyện có kích thước bình thường.

□ **Kiểm tra gradient** – Kiểm tra gradient là một phương thức được sử dụng trong quá trình thực hiện lan truyền ngược của mạng neural. Nó so sánh giá trị của gradient phân tích (analytical gradient) với gradient số (numerical gradient) tại các điểm đã cho và đóng vai trò kiểm tra độ chính xác.

	Gradient số	Gradient phân tích
Công thức	$\frac{df}{dx}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$	$\frac{df}{dx}(x) = f'(x)$
Bình luận	<ul style="list-style-type: none"> - Đắt, Mất mát phải được tính hai lần cho mỗi chiều - Được sử dụng để xác minh tính chính xác của việc triển khai phân tích - Đánh đổi trong việc chọn h không quá nhỏ (mất ổn định số) cũng không quá lớn (xấp xỉ độ dốc kém) 	<ul style="list-style-type: none"> - Kết quả 'Chính xác' - Tính toán trực tiếp - Được sử dụng trong quá trình triển khai cuối cùng

★ ★ ★