

HỌC SÂU

BÀI 5. MẠNG NƠ-RON HỒI QUY

RECURRENT NEURAL NETWORK - RNN

1. DỮ LIỆU TUẦN TỰ VÀ MÔ HÌNH TUẦN TỰ

1.1 Dữ liệu tuần tự (Sequence data)

Dữ liệu tuần tự được đặc trưng bởi tính chất có thứ tự của nó. Với dữ liệu tuần tự, cách sắp xếp các điểm dữ liệu là rất quan trọng.

Loại dữ liệu này được gặp trong nhiều lĩnh vực.

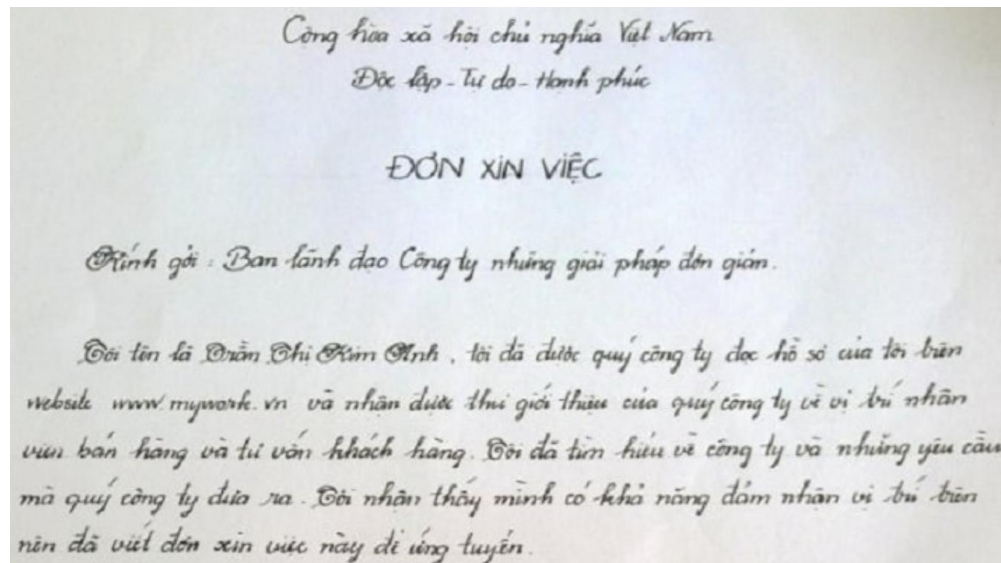
Dữ liệu chuỗi thời gian:

Loại dữ liệu này liên quan đến các dãy giá trị được ghi lại tại các khoảng thời gian đều đặn, như giá cổ phiếu, dữ liệu thời tiết, hoặc các phép đo từ cảm biến. Phân tích chuỗi thời gian được sử dụng để dự báo, phát hiện bất thường, và phân tích xu hướng.

Ngày	Giá mở cửa	Giá cao nhất	Giá thấp nhất	Giá đóng cửa	Thay đổi giá	% thay đổi	Khối lượng
28/03/2025	15,600	15,900	15,550	15,700	+150	0.96%	27,953,000
27/03/2025	15,100	15,650	15,100	15,550	+300	1.97%	22,568,100
26/03/2025	15,250	15,500	15,150	15,250	-100	-0.65%	20,625,300
25/03/2025	15,700	15,750	15,300	15,350	-250	-1.60%	22,181,600
24/03/2025	15,350	15,600	15,300	15,600	+250	1.63%	22,483,700

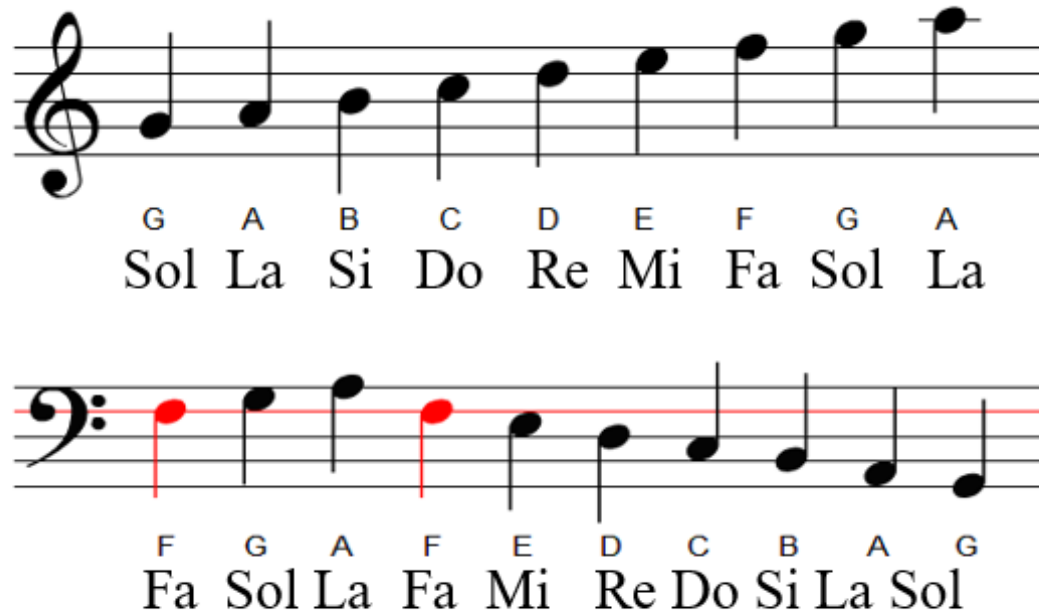
Dữ liệu Văn bản:

Trong xử lý ngôn ngữ tự nhiên (NLP), dữ liệu văn bản vốn có tính tuần tự, vì ý nghĩa của một câu phụ thuộc vào thứ tự của các từ. Các nhiệm vụ như phân tích cảm xúc, dịch máy và tạo văn bản dựa vào việc hiểu dữ liệu tuần tự này.



Dữ liệu âm thanh:

Dữ liệu âm thanh bao gồm các dãy sóng âm theo thời gian. Trong nhận dạng giọng nói hoặc phân tích âm nhạc, thứ tự thời gian của các tín hiệu âm thanh rất quan trọng để xác định các âm vị, từ, hoặc nốt nhạc.



Dữ liệu Video:

Dữ liệu video là một dãy các khung hình hoặc hình ảnh mà khi được phát theo thứ tự, sẽ biểu thị chuyển động. Dữ liệu video được sử dụng trong nhận dạng hành động, tóm tắt video, và theo dõi đối tượng.



Thách thức trong xử lý dữ liệu tuần tự:

- Độ dài biến đổi:** Các dãy dữ liệu có thể có độ dài khác nhau, yêu cầu các mô hình phải linh hoạt xử lý các đầu vào có kích thước khác nhau.
- Phụ thuộc thời gian:** Nắm bắt các phụ thuộc theo thời gian, đặc biệt là các phụ thuộc dài hạn, là điều quan trọng để mô hình hóa chính xác các dãy dữ liệu.
- Nhiều và biến đổi:** Dữ liệu dãy tuần tự trong thực tế có thể bị nhiễu hoặc có sự biến đổi, đòi hỏi các mô hình cần có tính tổng quát hóa tốt.
- Dữ liệu đa phương thức:** Trong một số trường hợp, các dãy dữ liệu có thể liên quan đến nhiều loại dữ liệu, như âm thanh và video, đòi hỏi các mô hình có thể tích hợp các nguồn thông tin đa dạng.

1. DỮ LIỆU TUẦN TỰ VÀ MÔ HÌNH TUẦN TỰ

1.2 Mô hình tuần tự (Sequence models)

Mô hình tuần tự được thiết kế để xử lý và phân tích dữ liệu tuần tự, phát hiện các mẫu và các phụ thuộc theo thời gian. Mô hình tuần tự được ứng dụng trong nhiều lĩnh vực:

- **Xử lý ngôn ngữ tự nhiên (NLP):** Các bài toán như dịch ngôn ngữ, tóm tắt văn bản, và trả lời câu hỏi sử dụng mô hình tuần tự để hiểu và tạo ra ngôn ngữ con người.
- **Nhận dạng giọng nói:** Chuyển đổi ngôn ngữ nói thành văn bản liên quan đến việc xử lý các dãy âm thanh để nhận biết từ và cụm từ.

2. CẤU TRÚC VÀ PHÂN LOẠI MẠNG NEURON HỒI QUY

2.1 Khái niệm cơ bản

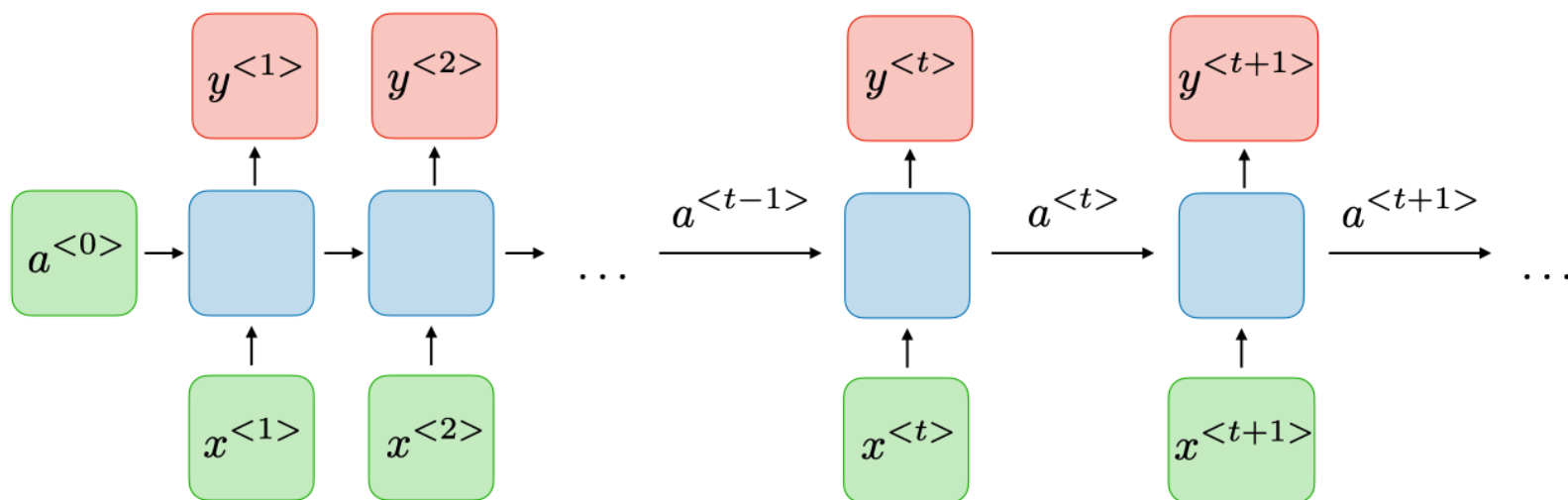
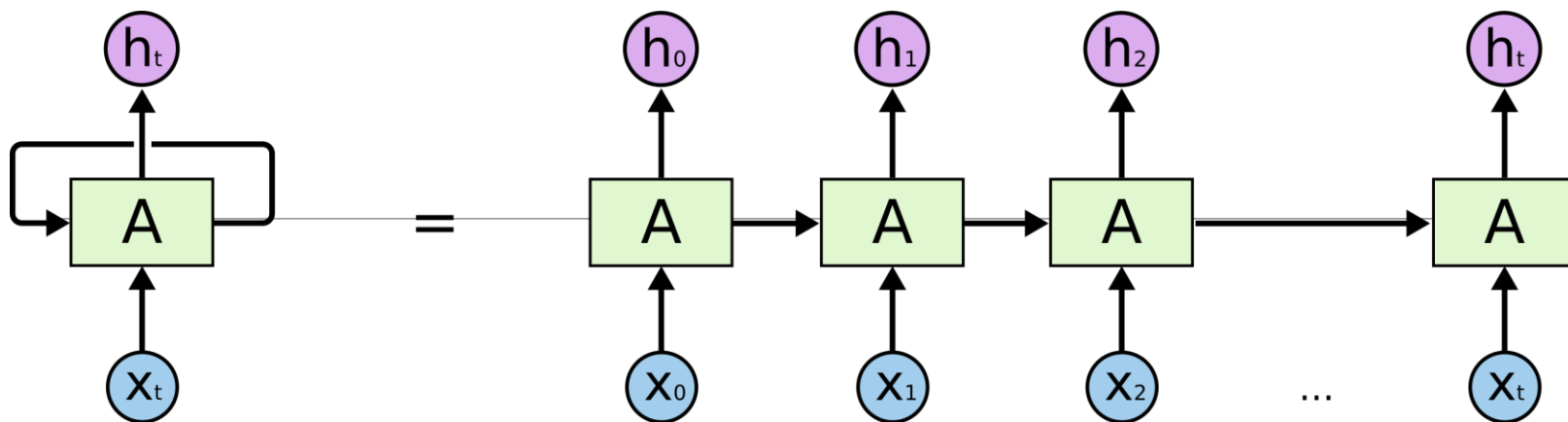
Mạng neuron hồi quy (Recurrent Neural Network - RNN) là một loại mạng neuron nhân tạo được thiết kế để nhận dạng mẫu trong các dữ liệu chuỗi, chẳng hạn như chuỗi thời gian, giọng nói, văn bản, dữ liệu tài chính. RNN đặc biệt hiệu quả cho các bài toán mà ngữ cảnh hoặc thứ tự của đầu vào là quan trọng, nhờ vào khả năng duy trì một dạng bộ nhớ. Chúng ta sẽ tìm hiểu chi tiết về cấu trúc và các loại RNN ở sau đây.

2.2 Cấu trúc của RNN

•**Neuron và lớp:** RNN bao gồm các neuron được tổ chức thành các lớp, tương tự như các mạng neuron khác. Tuy nhiên, không giống như mạng neural truyền thẳng, RNN có các kết nối vòng lặp lại chính nó, cho phép thông tin được duy trì.

•**Các trạng thái ẩn:** Thành phần chính của RNN là trạng thái ẩn, hoạt động như một bộ nhớ để nắm bắt thông tin về các đầu vào trước đó. Tại mỗi bước thời gian, trạng thái ẩn được cập nhật dựa trên đầu vào hiện tại và trạng thái ẩn trước đó.

•**Trọng số:** Các trọng số trong RNN được chia sẻ qua tất cả các bước. Việc chia sẻ trọng số này cho phép RNN tổng quát hóa qua các chuỗi dữ liệu có độ dài khác nhau.



2.3 Biểu diễn toán học

Tại mỗi bước thời gian t , RNN nhận một vector đầu vào \mathbf{x}_t và trạng thái ẩn trước đó \mathbf{h}_{t-1} để tính toán trạng thái ẩn mới \mathbf{h}_t theo công thức (5.1):

$$\mathbf{h}_t = f_1(\mathbf{W}_{hh} \cdot \mathbf{h}_{t-1} + \mathbf{W}_{xh} \cdot \mathbf{x}_t + \mathbf{b}_h) \quad (5.1)$$

Đầu ra \mathbf{y}_t được tính từ trạng thái ẩn theo công thức (5.2):

$$\mathbf{y}_t = f_2(\mathbf{W}_{hy} \cdot \mathbf{h}_t + \mathbf{b}_y) \quad (5.2)$$

Trong đó:

- f_1 và f_2 là các hàm kích hoạt. f_1 có thể là hàm ReLU, Sigmoid, Tanh; f_2 có thể là hàm Sigmoid, Tanh.
- \mathbf{W}_{hh} , \mathbf{W}_{xh} , và \mathbf{W}_{hy} là các ma trận trọng số.
- \mathbf{b}_h và \mathbf{b}_y là các vector bias.

Các công thức này thể hiện rằng trạng thái ẩn hiện tại \mathbf{h}_t phụ thuộc vào cả trạng thái ẩn trước đó \mathbf{h}_{t-1} và đầu vào hiện tại \mathbf{x}_t , tạo nên khả năng "ghi nhớ" thông tin qua các bước thời gian của RNN.

2.4 Lan truyền ngược qua thời gian

Huấn luyện RNN sử dụng một biến thể của lan truyền ngược gọi là lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT).

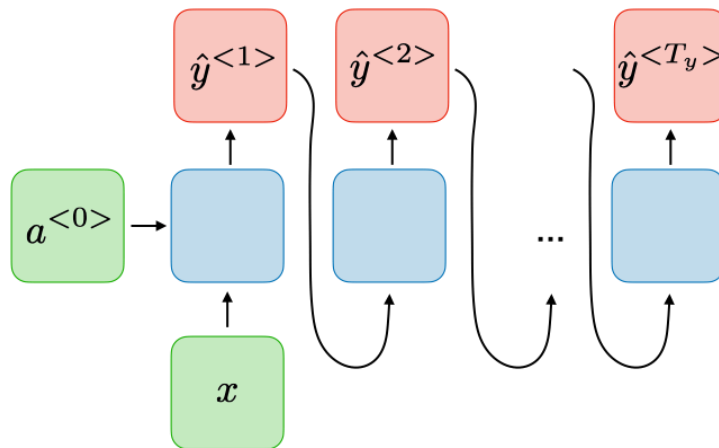
Phương pháp này mở rộng RNN thành dạng chuỗi và tính toán gradient cho mỗi bước thời gian, sau đó sử dụng gradient để cập nhật trọng số.

2.5 Các loại RNN

Các mạng neural hồi quy (RNN) là các mô hình linh hoạt được thiết kế để xử lý dữ liệu tuần tự. Chúng có nhiều kiến trúc khác nhau để phù hợp với các loại nhiệm vụ xử lý chuỗi dữ liệu khác nhau. Dưới đây sẽ mô tả các loại RNN khác nhau dựa trên mối quan hệ giữa chuỗi dữ liệu đầu vào và đầu ra: one to many, many to one, và many to many.

2.5.1 One to many RNN

- RNN loại này có một dữ liệu đầu vào duy nhất và đầu ra là một chuỗi dữ liệu (Hình 5.2).
- One to many RNN thường được sử dụng trong các bài toán mô tả ảnh. Trong bài toán này, một hình ảnh duy nhất (đầu vào) được sử dụng để tạo ra một câu mô tả (chuỗi đầu ra). One to many RNN cũng có thể được sử dụng để viết nhạc (music generation).

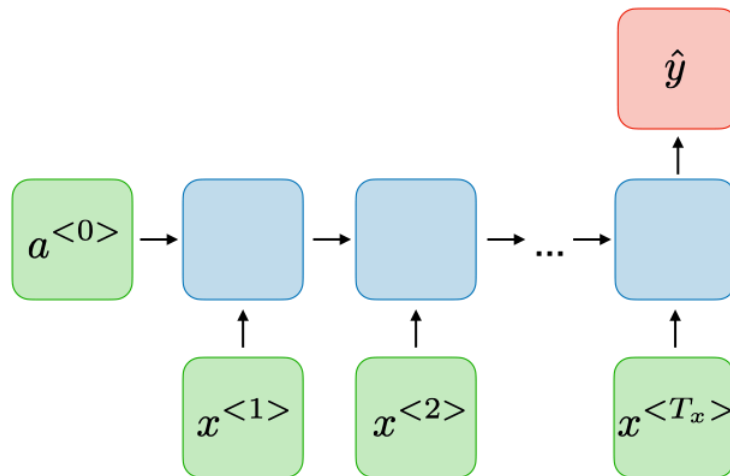


One-to-many
 $T_x = 1, T_y > 1$

Example: Music generation

2.5.2 Many to one RNN

- Ngược lại với one to many RNN, many to one RNN xử lý một chuỗi các đầu vào để tạo ra một đầu ra duy nhất (Hình 5.3).
- Kiến trúc này thường được sử dụng trong phân tích cảm xúc, với dữ liệu đầu vào là một chuỗi các từ và dữ liệu đầu ra là cảm xúc của văn bản. Many to one RNN cũng được sử dụng trong bài toán nhận diện/phân tích hành động, với dữ liệu đầu vào là video và dữ liệu đầu ra là hành động được thực hiện trong đoạn video đó.

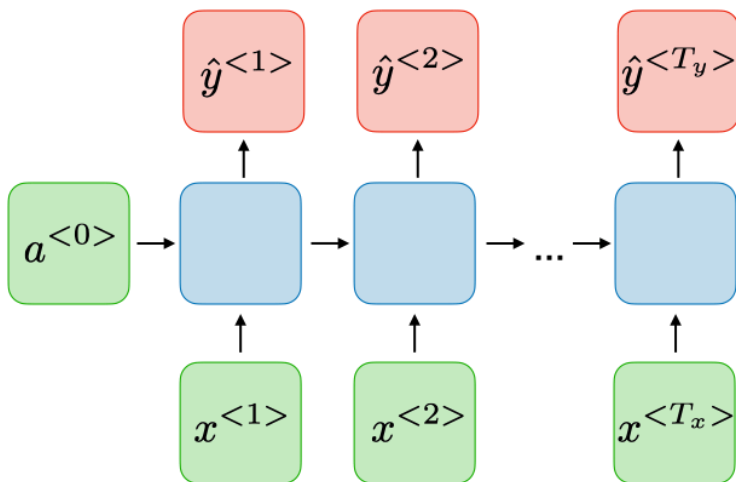


Many-to-one
 $T_x > 1, T_y = 1$
Sentiment classification

2.5.3 Many to many RNN

Dạng 1:

- Trong kiến trúc này, chuỗi đầu vào và đầu ra có cùng độ dài. Mỗi đầu vào tương ứng với một đầu ra tại cùng một bước thời gian (hình 5.4).
- Kiến trúc này được dùng trong bài toán mô tả video (video captioning), phát hiện tên người trong câu (name entity recognition).



Many-to-many

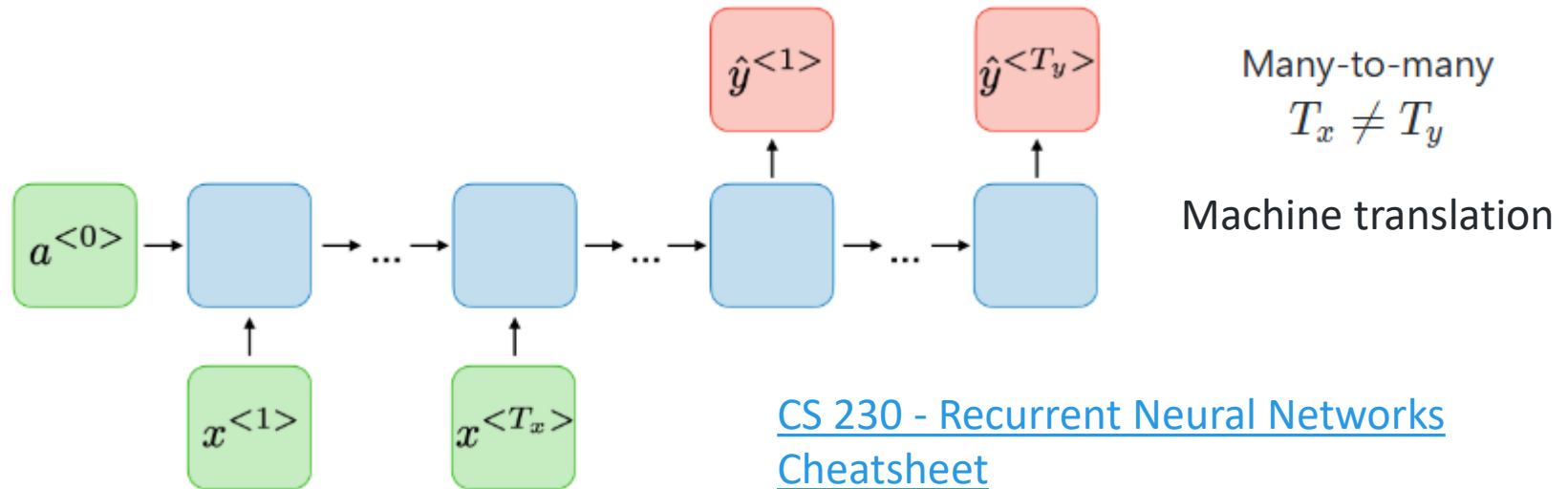
$$T_x = T_y$$

Name entity recognition

2.5.3 Many to many RNN

Dạng 2:

- Đối với kiến trúc many-to-many dạng 2, chuỗi đầu vào và đầu ra có thể khác nhau về độ dài
- RNN xử lý toàn bộ chuỗi đầu vào trước khi tạo ra chuỗi đầu ra
- Kiến trúc này thường được sử dụng trong dịch máy, nơi một câu trong một ngôn ngữ (chuỗi đầu vào) được dịch sang một câu trong một ngôn ngữ khác (chuỗi đầu ra)

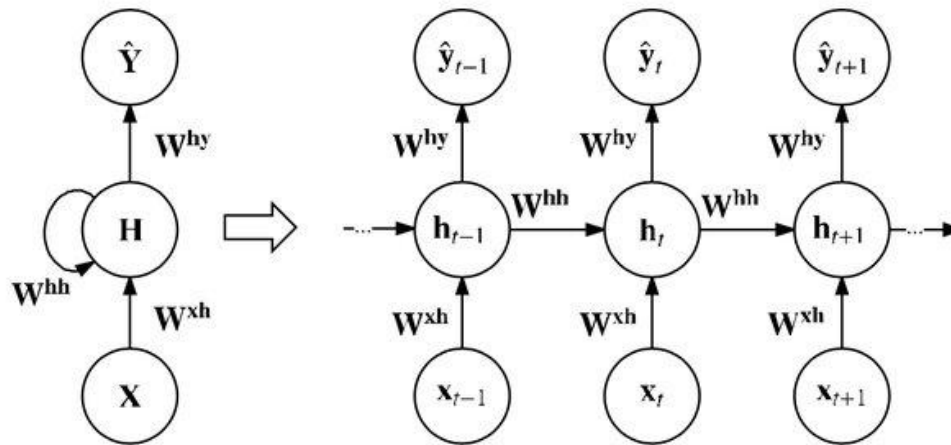


5.3 HUẤN LUYỆN MẠNG NEURON HỒI QUY

Huấn luyện mạng hồi quy RNN sử dụng biến thể của thuật toán lan truyền ngược, đó là lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT). Thuật toán **BPTT** giải quyết các vấn đề gặp phải do sự phụ thuộc theo thời gian trong dữ liệu tuần tự.

RNN Unrolling:

- RNN xử lý các chuỗi dữ liệu bằng cách duy trì một trạng thái ẩn được cập nhật tại mỗi bước thời gian
- Để áp dụng lan truyền ngược, RNN được "mở rộng" theo thời gian, tạo ra một loạt các lớp, mỗi lớp tương ứng với một bước thời gian
- Việc mở rộng này biến RNN thành một mạng truyền thẳng (feedforward network), trong đó mỗi lớp chia sẻ trọng số với các lớp khác



Ví dụ

- Chúng ta có một chuỗi văn bản gồm 3 từ: "Tôi yêu Việt_Nam"
- **Trước khi unroll (mở rộng):** RNN là một khối duy nhất có vòng lặp (loop) bên trong
- **Sau khi unroll:** RNN được mở rộng thành 3 bản sao của cùng một mạng nơ-ron (như 3 tầng)
 - Bước 1: Từ "Tôi" đi vào, ra trạng thái ẩn h_1
 - Bước 2: Từ "yêu" đi vào, kết hợp với h_1 để tạo ra h_2
 - Bước 3: Từ "Việt_Nam" đi vào, kết hợp với h_2 để tạo ra h_3
- Điểm quan trọng: Mặc dù trông như 3 mạng riêng biệt, nhưng chúng chia sẻ cùng một bộ trọng số W

Trọng số chia sẻ:

- Các trọng số giữa lớp đầu vào và lớp ẩn, cũng như giữa các lớp ẩn qua các bước thời gian, được chia sẻ

Lan truyền tiến (forward pass):

- Tại mỗi bước thời gian, các gradient được tính toán theo từng trọng số và bias
- Các gradient này được tích lũy qua tất cả các bước thời gian để cập nhật trọng số
- Sau khi tính toán các gradient, trọng số được cập nhật bằng cách sử dụng một thuật toán tối ưu hóa như SGD hoặc Adam

Lan truyền ngược (backward pass):

- Tại mỗi bước thời gian, các gradient được tính toán theo từng trọng số và bias
- Các gradient này được tích lũy qua tất cả các bước thời gian để cập nhật trọng số
- Sau khi tính toán các gradient, trọng số được cập nhật bằng cách sử dụng một thuật toán tối ưu hóa như SGD hoặc Adam

Các vấn đề có thể gặp khi huấn luyện RNN:

Gradient biến mất và bùng nổ:

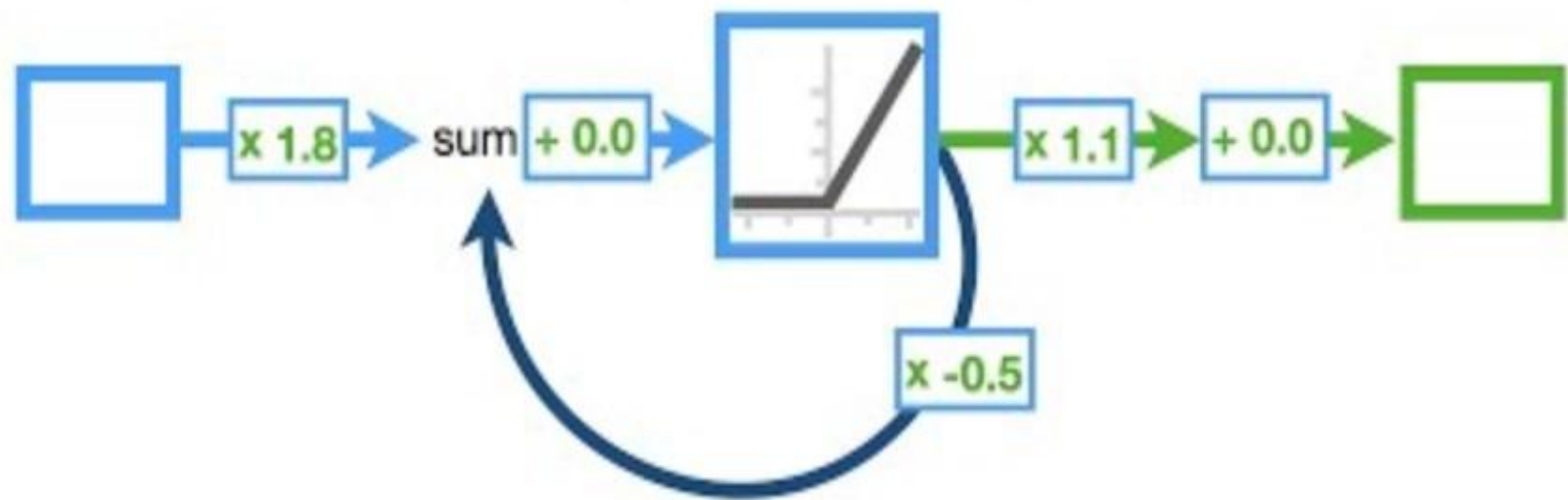
- Do việc nhân liên tục các gradient qua nhiều bước thời gian, gradient có thể trở nên rất nhỏ (biến mất) hoặc rất lớn (bùng nổ)
- Điều này làm cho việc huấn luyện các deep RNN trở nên khó khăn
- Các giải pháp phổ biến để giải quyết hai vấn đề này là kỹ thuật cắt gradient để xử lý gradient bùng nổ, và sử dụng các kiến trúc như LSTM hoặc GRU để giảm thiểu gradient biến mất

Các vấn đề có thể gặp khi huấn luyện RNN:

BPTT có độ phức tạp thời gian và bộ nhớ lớn:

- BPTT yêu cầu lưu trữ các gradient cho mỗi bước thời gian trong quá trình truyền tới
- Để giảm chi phí tính toán và giải quyết các vấn đề của các chuỗi dữ liệu dài, BPTT rút gọn thường được sử dụng
- Trong BPTT rút gọn, thay vì lan truyền lỗi qua toàn bộ chuỗi, chuỗi được chia thành các đoạn nhỏ hơn, và BPTT được áp dụng cho từng đoạn

Recurrent Neural Networks (RNNs)...



...Clearly Explained!!!

Đọc thêm

- <https://nttuan8.com/bai-13-recurrent-neural-network/>
- <https://www.youtube.com/watch?v=t0EoeTYU-fc&t=1321s>
- <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

Tham khảo

[Teaching - CS 221](#)

[Teaching - CS 229](#)

[Teaching - CS 230](#)

5.4 Ứng Dụng Mạng Neuron Hồi Quy vào Bài Toán Mô Tả Hình Ảnh (Image Captioning)

Mô tả hình ảnh (Image captioning) là một bài toán kết hợp thị giác máy tính và xử lý ngôn ngữ tự nhiên để tạo ra các văn bản mô tả cho hình ảnh. Một phương pháp hiệu quả để tạo ra mô tả hình ảnh là sử dụng kết hợp mạng neuron tích chập (CNN) để trích xuất đặc trưng từ hình ảnh và mạng neuron hồi quy (RNN) để tạo câu/đoạn văn bản mô tả.

5.4 Ứng Dụng Mạng Neuron Hồi Quy vào Bài Toán Mô Tả Hình Ảnh (Image Captioning)

Tổng quan về kiến trúc:

- Quá trình mô tả hình ảnh bắt đầu với một CNN, thường là một mô hình đã được huấn luyện trước như VGG16, ResNet, hoặc Inception, để trích xuất đặc trưng từ hình ảnh đầu vào.
- CNN hoạt động như một bộ mã hóa chuyển đổi hình ảnh thành một vector đặc trưng có độ dài cố định.
- Vector đặc trưng được trích xuất từ CNN được đưa vào một RNN. RNN này hoạt động như một bộ giải mã để tạo ra một chuỗi các từ mô tả hình ảnh đầu vào.
- RNN tạo ra từng từ một, sử dụng từ đầu vào hiện tại và trạng thái ẩn từ bước thời gian trước để dự đoán từ tiếp theo trong dãy.
- Quá trình tiếp tục cho đến khi một ký hiệu kết thúc chuỗi được tạo ra hoặc đạt đến độ dài chú thích tối đa.

Quy trình chi tiết:

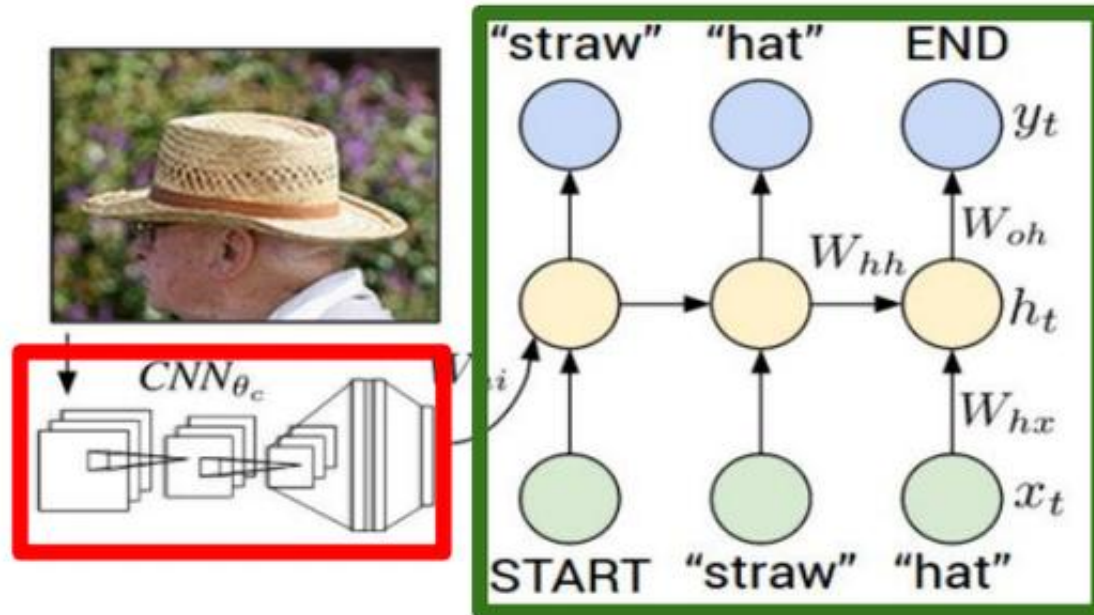
- **Trích xuất đặc trưng với CNN:**

- Vector đặc trưng được trích xuất từ CNN được đưa vào một phép biến đổi tuyến tính để giảm chiều của chuỗi vector đặc trưng để phù hợp với kích thước đầu vào của RNN.

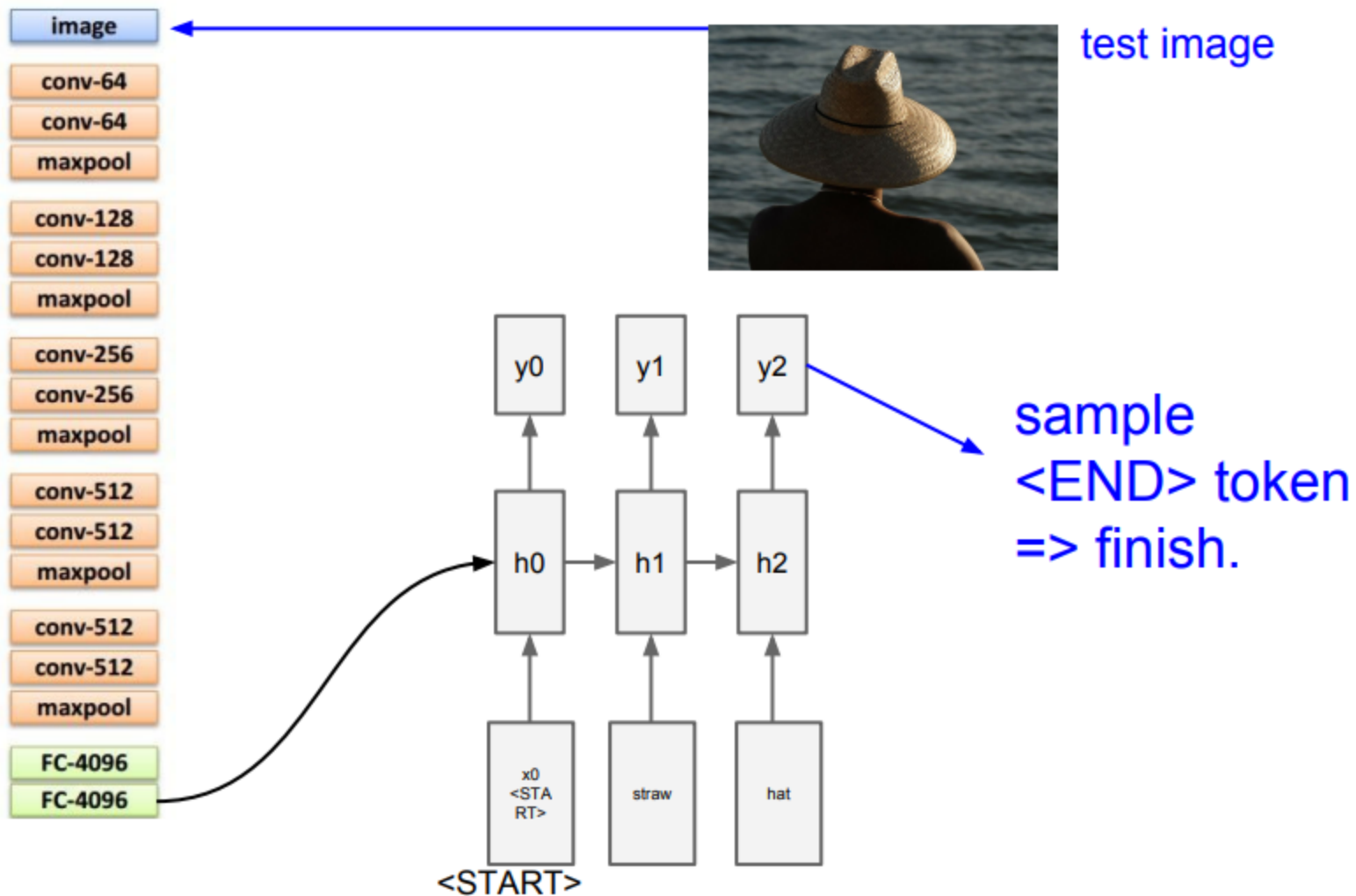
- **Tạo chuỗi từ với RNN:**

- Vector đặc trưng được đưa vào một RNN để tạo ra một chuỗi các từ mô tả hình ảnh đầu vào.
- RNN tạo ra từng từ một, sử dụng từ đầu vào hiện tại và trạng thái ẩn từ bước thời gian trước để dự đoán từ tiếp theo trong dãy.
- Quá trình tiếp tục cho đến khi một ký hiệu kết thúc chuỗi được tạo ra hoặc đạt đến độ dài chú thích tối đa.

Recurrent Neural Network



Convolutional Neural Network



Huấn luyện mô hình:

- **Bộ dữ liệu:** Các mô hình mô tả hình ảnh có thể được huấn luyện trên các tập dữ liệu như MS COCO hoặc Flickr8k/30k, bao gồm các hình ảnh được ghép đôi với nhiều chú thích do con người tạo ra.
- **Hàm mất mát:** Mô hình thường được huấn luyện sử dụng hàm mất mát cross-entropy, đo lường sự khác biệt giữa các dãy từ dự đoán và thực tế.
- **Tối ưu hóa:** Thuật toán tối ưu hóa như Adam hoặc RMSprop được sử dụng để giảm thiểu hàm mất mát. Quá trình huấn luyện bao gồm lan truyền ngược qua thời gian (BPTT) để cập nhật trọng số của RNN như mô tả ở mục trên.

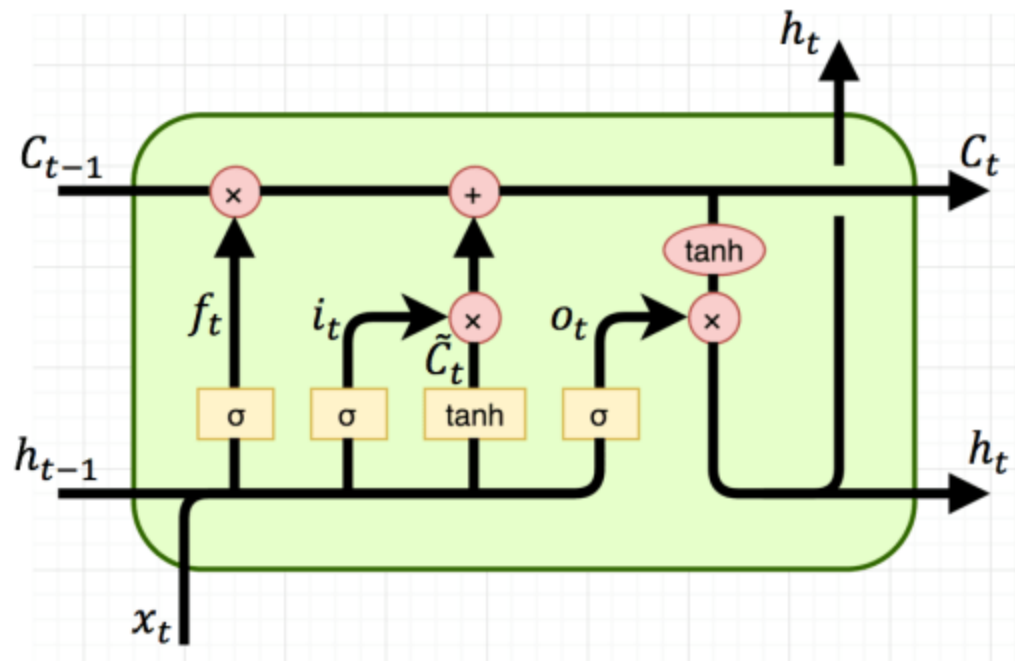
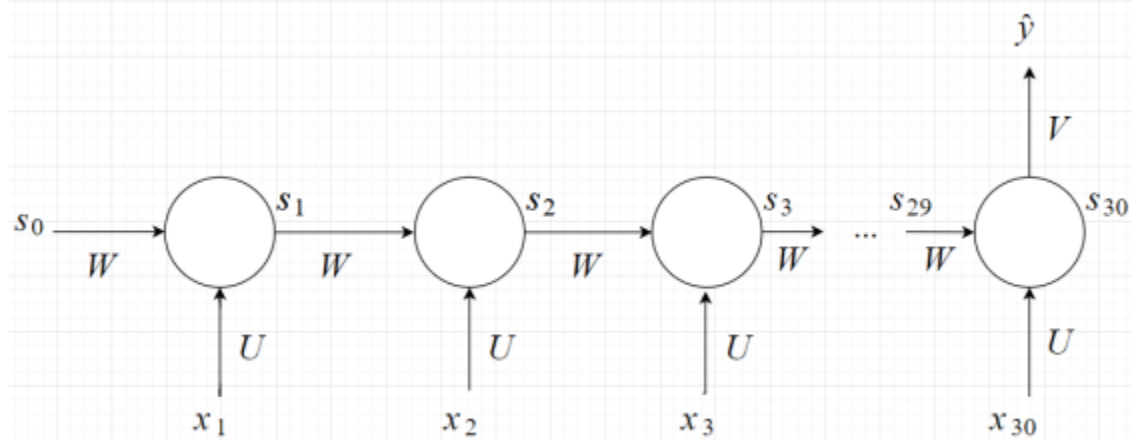
Đánh giá mô hình sau huấn luyện

- Các mô hình chú thích hình ảnh được đánh giá bằng các chỉ số như BLEU, METEOR, ROUGE, và CIDEr, so sánh các chú thích được tạo ra với các chú thích tham khảo.
- Kiểm tra trực quan các chú thích được tạo ra có thể cung cấp thông tin chi tiết về điểm mạnh và điểm yếu của mô hình.

5.5 Mô Hình Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) là một loại kiến trúc mạng neuron hồi quy (RNN) được thiết kế để mô hình hóa các chuỗi thời gian và các phụ thuộc dài hạn hiệu quả hơn so với các RNN tiêu chuẩn. LSTM được giới thiệu bởi Sepp Hochreiter và Jürgen Schmidhuber vào năm 1997 để giải quyết vấn đề gradient biến mất có thể xảy ra trong quá trình huấn luyện các RNN truyền thống.

LSTM được thiết kế để ghi nhớ thông tin trong thời gian dài. LSTM đạt được điều này bằng cách sử dụng một cấu trúc đặc biệt gọi là "ô nhớ" và ba loại cổng (cổng đầu vào - input gate, cổng đầu ra - output gate, và cổng quên - forgetting gate) để kiểm soát luồng thông tin vào và ra khỏi ô.



Ưu điểm

- **Xử lý các phụ thuộc dài hạn:** LSTM được thiết kế để xử lý các phụ thuộc dài hạn, làm cho LSTM phù hợp cho các nhiệm vụ như mô hình hóa ngôn ngữ, dịch thuật, và dự đoán chuỗi thời gian.
- **Tránh gradient biến mất (vanishing gradient):** Kiến trúc của LSTM giúp giảm thiểu vấn đề gradient biến mất, cho phép chúng học và giữ lại thông tin qua các chuỗi dài.
- **Cơ chế cổng cho phép LSTM linh hoạt quên, giữ lại, hoặc thêm thông tin,** cung cấp sự linh hoạt trong quản lý bộ nhớ.

Ứng dụng

- **Xử lý ngôn ngữ tự nhiên (NLP):** LSTM được sử dụng rộng rãi trong các nhiệm vụ NLP như dịch ngôn ngữ, phân tích cảm xúc, và tạo văn bản.
- **Dự đoán chuỗi thời gian:** Chúng hiệu quả trong việc dự đoán dữ liệu chuỗi thời gian, như giá cổ phiếu, dự báo thời tiết, và tiêu thụ năng lượng.
- **Nhận dạng giọng nói:** LSTM được sử dụng trong các hệ thống nhận dạng giọng nói để mô hình hóa các phụ thuộc thời gian trong dữ liệu âm thanh.
- **Phát hiện dị thường:** Chúng có thể được sử dụng để phát hiện các dị thường trong chuỗi, như lưu lượng mạng hoặc dữ liệu cảm biến.

Các vấn đề cần xem xét:

- Độ phức tạp tính toán:** LSTM phức tạp hơn về mặt tính toán so với các RNN tiêu chuẩn, điều này có thể dẫn đến thời gian huấn luyện dài hơn.
- Điều chỉnh siêu tham số:** Điều chỉnh đúng các siêu tham số như tốc độ học, kích thước batch là rất quan trọng để có hiệu suất tốt.
- Kiến trúc thay thế:** Mặc dù LSTM hoạt động có hiệu quả, các kiến trúc mới hơn như Gated Recurrent Units (GRUs) và Transformers đã được phát triển để giải quyết một số hạn chế của LSTM, như hiệu suất tính toán và khả năng song song hóa.