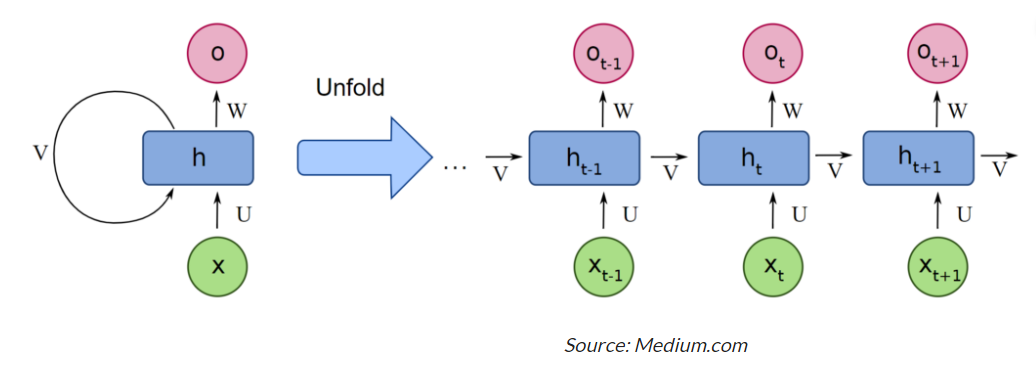
Tổng quan ngắn gọn về các mạng nơ-ron tái phát (RNN)

Nguồn bài viết : <https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/>

# Giới thiệu về mạng nơ-ron tái phát

Một cách tiếp cận học sâu để lập mô hình dữ liệu tuần tự là Mạng thần kinh tái phát (RNN). RNN là gợi ý tiêu chuẩn để làm việc với dữ liệu tuần tự trước sự ra đời của các mô hình chú ý. Các tham số cụ thể cho từng phần tử của chuỗi có thể được yêu cầu bởi một mô hình tiến bộ sâu. Nó cũng có thể không thể khái quát hóa thành các chuỗi có độ dài thay đổi.



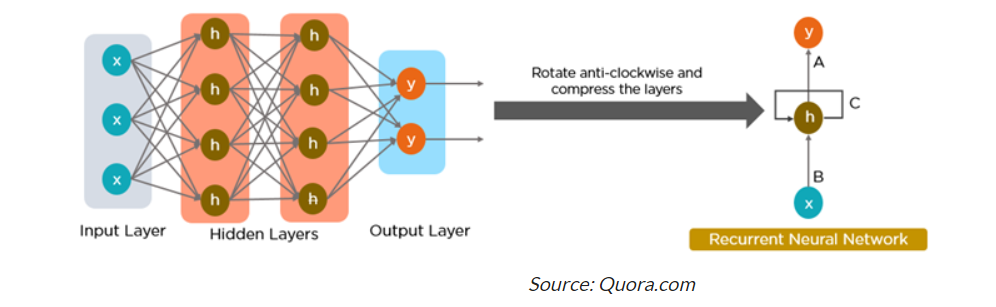
Các mạng nơ-ron tái phát sử dụng cùng một trọng số cho mỗi phần tử của chuỗi, làm giảm số lượng tham số và cho phép mô hình khái quát hóa thành các chuỗi có độ dài khác nhau. RNN khái quát hóa cho dữ liệu có cấu trúc khác với dữ liệu tuần tự, chẳng hạn như dữ liệu địa lý hoặc đồ họa, vì thiết kế của chúng.

Mạng lưới thần kinh tái phát, giống như nhiều kỹ thuật học sâu khác, tương đối cũ. Chúng được phát triển lần đầu tiên vào những năm 1980, nhưng chúng tôi đã không đánh giá cao tiềm năng đầy đủ của chúng cho đến gần đây. Sự ra đời của bộ nhớ dài hạn ngắn hạn (LSTM) vào những năm 1990, kết hợp với sự gia tăng sức mạnh tính toán và lượng lớn dữ liệu mà chúng ta hiện phải xử lý, đã thực sự đẩy RNN lên hàng đầu.

# Mạng nơ-ron tái phát (RNN) là gì?

Mạng nơ-ron bắt chước chức năng của bộ não con người trong các lĩnh vực AI, học máy và học sâu, cho phép các chương trình máy tính nhận ra các mẫu và giải quyết các vấn đề phổ biến.

RNN là một loại mạng nơ-ron có thể được sử dụng để mô hình hóa dữ liệu trình tự. RNN, được hình thành từ mạng lưới thực phẩm, tương tự như bộ não con người trong hành vi của chúng. Nói một cách đơn giản, các mạng nơ-ron định kỳ có thể dự đoán dữ liệu tuần tự theo những cách mà các thuật toán khác không thể làm được.

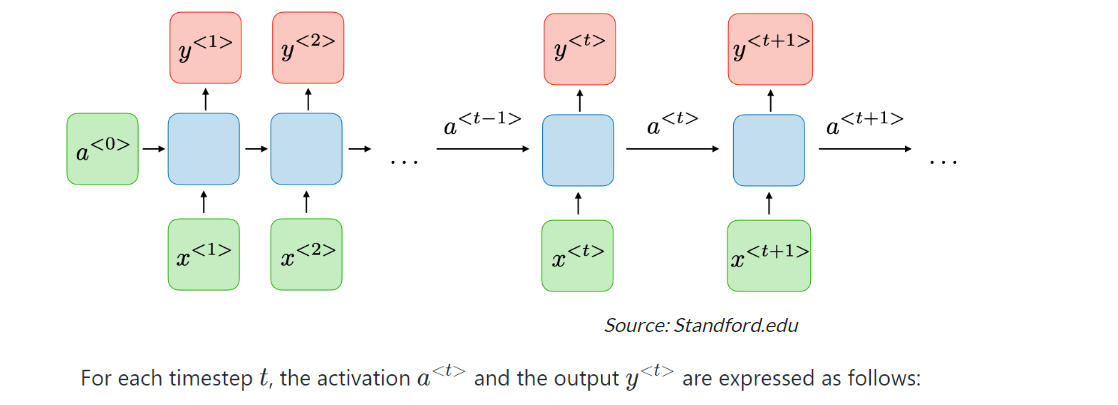


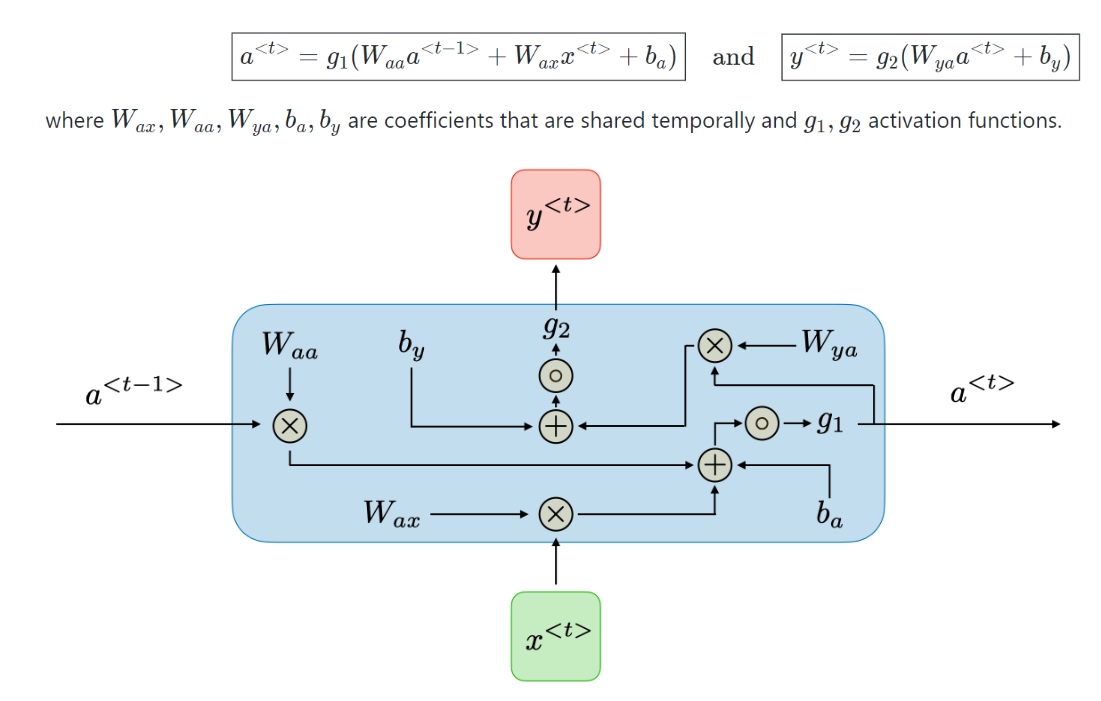
Tất cả các đầu vào và đầu ra trong các mạng nơ-ron tiêu chuẩn là độc lập với nhau, nhưng trong một số trường hợp, chẳng hạn như khi dự đoán từ tiếp theo trong một câu, các từ trước đó là cần thiết, và do đó các từ trước đó phải được ghi nhớ. Kết quả là, RNN đã được tạo ra, sử dụng một lớp ẩn để khắc phục sự cố. Thành phần quan trọng nhất của RNN là trạng thái Ẩn, lưu trữ thông tin cụ thể về một chuỗi.

RNN có một bộ nhớ lưu trữ tất cả thông tin về các phép tính. Nó sử dụng các tham số giống nhau cho mỗi đầu vào vì nó tạo ra cùng một kết quả bằng cách thực hiện cùng một tác vụ trên tất cả các đầu vào hoặc lớp ẩn.

# Kiến trúc của một RNN truyền thống

RNN là một loại mạng nơ-ron có trạng thái ẩn và cho phép các đầu ra trong quá khứ được sử dụng làm đầu vào. Họ thường đi như thế này:





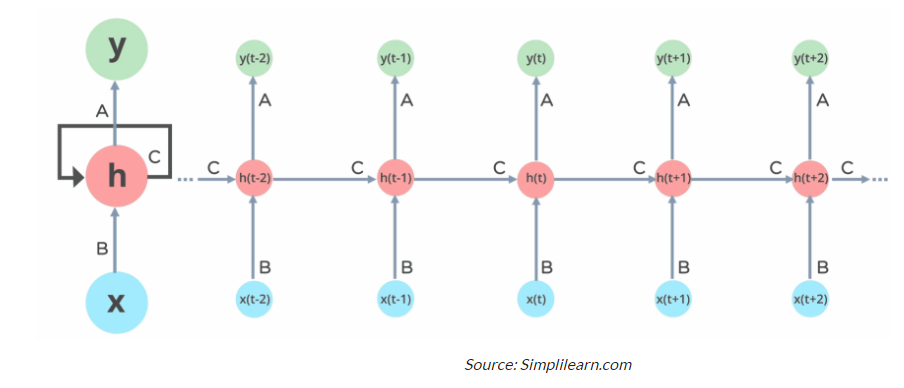
Kiến trúc RNN có thể thay đổi tùy thuộc vào vấn đề bạn đang cố gắng giải quyết. Từ những người có một lối vào và lối ra duy nhất đến những người có nhiều (với các biến thể giữa hai).

Dưới đây là một số ví dụ về kiến trúc RNN có thể giúp bạn hiểu rõ hơn về điều này.

* **Từng cái một:** Chỉ có một cặp ở đây. Kiến trúc một-một được sử dụng trong các mạng nơ-ron truyền thống.
* **Một đến nhiều:** Một lần nhập vào mạng một-nhiều có thể dẫn đến nhiều kết quả đầu ra. Ví dụ, một quá nhiều mạng được sử dụng trong sản xuất âm nhạc.
* **Nhiều đến một:** Trong trường hợp này, một đầu ra duy nhất được tạo ra bằng cách kết hợp nhiều đầu vào từ các bước thời gian riêng biệt. Phân tích cảm xúc và nhận dạng cảm xúc sử dụng các mạng như vậy, trong đó nghi thức lớp được xác định bởi một chuỗi các từ.
* **Nhiều đến nhiều:** Đối với nhiều đến nhiều người, có rất nhiều lựa chọn. Hai đầu vào tạo ra ba đầu ra. Các hệ thống dịch máy, chẳng hạn như hệ thống dịch từ tiếng Anh sang tiếng Pháp hoặc ngược lại, sử dụng nhiều mạng.

# Mạng lưới thần kinh tái phát hoạt động như thế nào?

Thông tin trong các mạng nơ-ron tái phát đi qua một vòng lặp đến lớp ẩn của giữa.



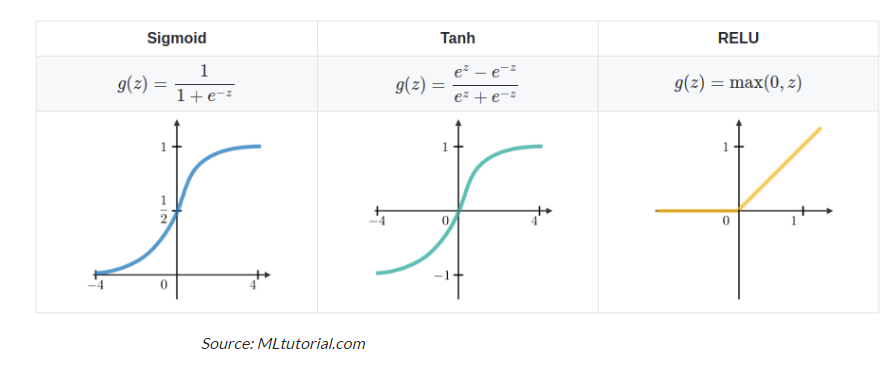
Lớp đầu vào x nhận và xử lý đầu vào từ mạng nơ-ron trước khi truyền nó đến lớp giữa.

Một số lớp ẩn có thể được tìm thấy trong lớp giữa **h**, mỗi lớp có chức năng kích hoạt, trọng lượng và độ lệch riêng. Bạn có thể sử dụng mạng nơ-ron định kỳ nếu các tham số khác nhau của các lớp ẩn khác nhau không bị ảnh hưởng bởi lớp trước đó, nghĩa là không có bộ nhớ trong mạng nơ-ron.

Các chức năng kích hoạt, trọng lượng và độ lệch khác nhau sẽ được chuẩn hóa bởi mạng nơ-ron tái phát, đảm bảo rằng mỗi lớp ẩn có cùng đặc điểm. Thay vì xây dựng nhiều lớp ẩn, nó sẽ chỉ tạo ra một và lặp lại nó nhiều lần nếu cần thiết.

# Các tính năng kích hoạt phổ biến

Chức năng kích hoạt của tế bào thần kinh quyết định nên bật hay tắt nó. Các hàm phi tuyến thường biến đổi đầu ra của một tế bào thần kinh thành một số từ 0 đến 1 hoặc -1 và 1.



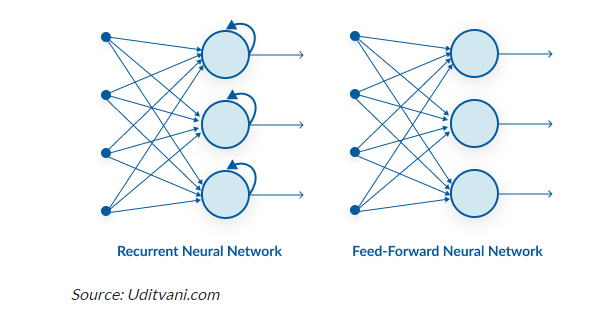
Dưới đây là một số chức năng được sử dụng phổ biến nhất:

* **Sigmoid:** Công thức **g(z) = 1/(1 + e^-z)** được sử dụng để diễn đạt điều này.
* **Tanh:** Công thức **g(z) = (e^-z – e^-z)/(e^-z + e^-z)** được sử dụng để diễn đạt điều này.
* **Đọc lại:** Công thức **g(z) = max(0 , z)** được sử dụng để thể hiện điều này.

# Mạng thần kinh tái phát so với Mạng thần kinh Feedforward

Một mạng nơ-ron trước chỉ có một cách luồng thông tin: từ lớp đầu vào đến lớp đầu ra, đi qua các lớp ẩn. Dữ liệu chảy qua mạng theo một tuyến đường thẳng, mà không bao giờ đi qua cùng một nút hai lần.

Luồng thông tin giữa RNN và mạng nơ-ron công suất trực tiếp được minh họa trong hai hình dưới đây.



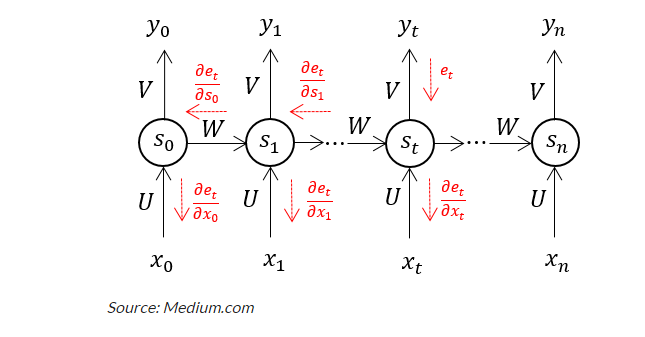
Mạng nơ-ron chuyển tiếp nguồn cấp dữ liệu là những dự đoán tồi về những gì sẽ xảy ra tiếp theo vì chúng không có bộ nhớ về thông tin chúng nhận được. Bởi vì nó chỉ đơn giản là phân tích đầu vào hiện tại, một mạng chuyển tiếp nguồn cấp dữ liệu không có ý tưởng về thứ tự thời gian. Ngoài việc tập luyện, anh không còn nhớ gì về những gì đã xảy ra trong quá khứ.

Thông tin nằm trong chu kỳ RNN thông qua một vòng lặp. Trước khi đưa ra đánh giá, anh đánh giá các bài dự thi hiện tại cũng như những gì anh đã học được từ các bài dự thi trong quá khứ. Mặt khác, một mạng lưới thần kinh tái phát có thể nhớ lại do bộ nhớ trong. Nó tạo ra đầu ra, sao chép nó, và sau đó gửi nó trở lại mạng.

# Backpropagation theo thời gian (BPTT)

Khi chúng tôi áp dụng thuật toán backpropagation cho mạng nơ-ron định kỳ với dữ liệu chuỗi thời gian đầu vào, chúng tôi gọi đó là backpropagation theo thời gian.

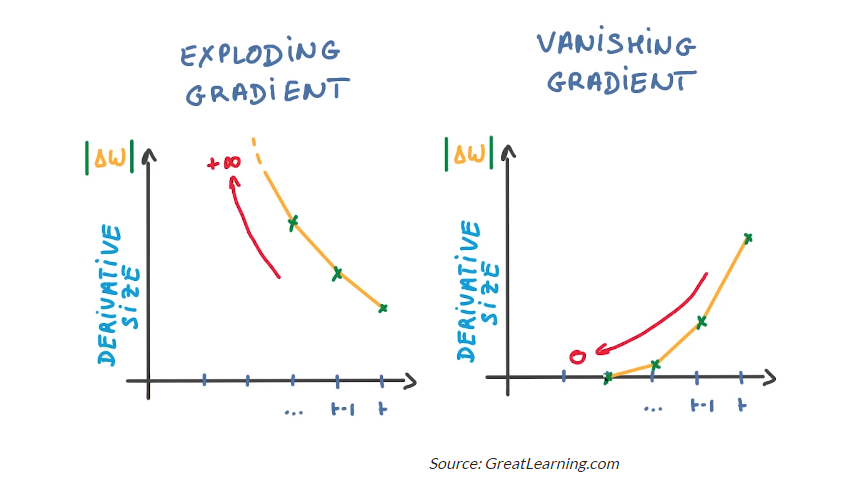
Chỉ có một đầu vào được gửi vào mạng tại một thời điểm trong RNN bình thường và chỉ thu được một đầu ra. Mặt khác, backpropagation sử dụng cả mục nhập hiện tại và trước đó làm đầu vào. Đây được gọi là bước thời gian và một bước thời gian sẽ bao gồm nhiều điểm dữ liệu chuỗi thời gian nhập RNN cùng một lúc.



Đầu ra mạng nơ-ron được sử dụng để tính toán và thu thập lỗi sau khi nó đã được đào tạo qua một tập hợp thời gian và cung cấp cho bạn đầu ra. Mạng sau đó được lắp ráp lại, và trọng lượng được tính toán lại và điều chỉnh để giải thích cho các lỗi.

# Hai số RNN tiêu chuẩn

Có hai thách thức chính mà NRNs đã phải vượt qua, nhưng để hiểu chúng, trước tiên người ta phải nắm bắt gradient là gì.



Đối với đầu vào của nó, một gradient là một đạo hàm một phần. Nếu bạn không chắc chắn điều này đòi hỏi điều gì, hãy xem xét điều này: một gradient định lượng đầu ra của một hàm thay đổi bao nhiêu khi các đầu vào được sửa đổi một chút.

Độ dốc của một hàm còn được gọi là gradient. Độ dốc càng dốc, mô hình có thể học càng nhanh thì độ dốc càng cao. Mặt khác, mô hình sẽ ngừng học nếu độ dốc bằng không. Một gradient được sử dụng để đo sự thay đổi trong tất cả các trọng số liên quan đến sự thay đổi sai số.

* **Độ dốc nổ:** Gradient nổ xảy ra khi thuật toán đưa ra trọng số ưu tiên cao một cách vô lý mà không có lý do rõ ràng. May mắn thay, cắt ngắn hoặc ghi đè gradient là một giải pháp đơn giản cho vấn đề này.
* **Biến mất gradient: Gradient biến mất** xảy ra khi các giá trị gradient quá nhỏ, khiến mô hình ngừng học hoặc mất quá nhiều thời gian. Đây là một vấn đề lớn trong những năm 1990, và nó khó giải quyết hơn nhiều so với việc phát nổ gradient. May mắn thay, khái niệm LSTM của Sepp Hochreiter và Juergen Schmidhuber đã giải quyết được vấn đề.

# Ứng dụng RNN

Mạng nơ-ron tái phát được sử dụng để giải quyết nhiều vấn đề liên quan đến dữ liệu trình tự. Có nhiều loại dữ liệu trình tự, nhưng sau đây là những loại phổ biến nhất: Âm thanh, Văn bản, Video, Trình tự sinh học.

Bằng cách sử dụng mô hình RNN và bộ dữ liệu trình tự, bạn có thể khắc phục nhiều sự cố khác nhau, bao gồm:

* Nhận dạng giọng nói
* Thế hệ âm nhạc
* Dịch máy
* Phân tích hành động video
* Nghiên cứu tuần tự về bộ gen và DNA

# Kết thúc

* Mạng lưới thần kinh tái phát là một công cụ linh hoạt có thể được sử dụng trong nhiều tình huống khác nhau. Chúng được sử dụng trong nhiều phương pháp khác nhau để mô hình hóa ngôn ngữ và trình tạo văn bản. Chúng cũng được sử dụng trong nhận dạng giọng nói.
* Loại mạng nơ-ron này được sử dụng để tạo nhãn cho hình ảnh không được đánh dấu khi được liên kết với mạng nơ-ron phức tạp. Thật đáng kinh ngạc khi sự kết hợp này hoạt động đến mức nào.
* Tuy nhiên, có một lỗ hổng với các mạng lưới thần kinh tái phát. Họ phải vật lộn để tìm hiểu các phụ thuộc tầm xa, có nghĩa là họ không hiểu mối quan hệ giữa các dữ liệu được phân tách bằng nhiều bước.
* Ví dụ, khi chúng ta dự đoán các từ, chúng ta có thể cần nhiều ngữ cảnh hơn là chỉ một từ trước đó. Đây được gọi là vấn đề gradient biến mất và nó được giải quyết bằng cách sử dụng một loại mạng thần kinh tái phát đặc biệt được gọi là mạng bộ nhớ dài hạn và ngắn hạn (LSTM), là một chủ đề rộng hơn sẽ được thảo luận trong các bài viết trong tương lai.

Code : [implémentation Python assignment\_internship/Basic (RNN avec Keras).ipynb à main · Hntam812 /assignment\_internship (github.com)](https://github.com/Hntam812/assignment_internship/blob/main/Basic%20Python%20Implementation%20(RNN%20with%20Keras).ipynb)

LSTM và GRU trong mạng lưới thần kinh tái phát: một nghiên cứu so sánh

Bộ nhớ dài hạn trong LSTM ngắn hạn là một loại RNN đặc biệt có khả năng học các chuỗi dài hạn. Chúng được schmidhuber và Hochreiter giới thiệu vào năm 1997. Nó được thiết kế rõ ràng để tránh các vấn đề nghiện lâu dài. Ghi nhớ các chuỗi dài trong một thời gian dài là cách làm việc của anh ấy.

[Mạng nơ-ron định kỳ](https://analyticsindiamag.com/implementing-a-recurrent-neural-network-rnn-from-scratch/) là một loại ANN được sử dụng khi người dùng muốn thực hiện các hoạt động dự đoán trên dữ liệu tuần tự hoặc dựa trên chuỗi thời gian . Các lớp học sâu này thường được sử dụng cho các vấn đề thứ tự hoặc thời gian như [xử lý ngôn ngữ tự nhiên](https://analyticsindiamag.com/how-to-identify-entities-in-nlp/), dịch máy thần kinh, [các tác vụ phụ đề hình ảnh](https://analyticsindiamag.com/hands-on-guide-to-effective-image-captioning-using-attention-mechanism/) tự động và các tác vụ khác. Các thiết bị hỗ trợ giọng nói hiện đại ngày nay như Google Assistance, Alexa, Siri được tích hợp vào các lớp này để mang đến trải nghiệm không rắc rối cho người dùng.

Trong RNN để tạo thành các mạng, chúng tôi yêu cầu theo thời gian và tại mỗi bước thời gian hoặc vòng lặp, gradient được tính toán và gradient được sử dụng để cập nhật trọng số trong mạng. Bây giờ, nếu ảnh hưởng của chuỗi trước đó lên lớp nhỏ, gradient tương đối được tính nhỏ. Sau đó, nếu gradient của lớp trước nhỏ hơn, nó sẽ làm giảm trọng số được gán cho ngữ cảnh và hiệu ứng này được quan sát thấy khi chúng ta xử lý các chuỗi dài hơn. Do mạng này không tìm hiểu tác dụng của các mục trước đó và do đó gây ra vấn đề về bộ nhớ ngắn hạn.

Để khắc phục vấn đề này, các phiên bản chuyên dụng của RNN được tạo ra như LSTM, GRU,...

# ****Cách thức hoạt động của LSTM****

**Bộ nhớ dài hạn hoặc LSTM là gì?**

Bộ nhớ dài hạn trong LSTM ngắn hạn là một loại RNN đặc biệt có khả năng học các chuỗi dài hạn. Chúng được schmidhuber và Hochreiter giới thiệu vào năm 1997. Nó được thiết kế rõ ràng để tránh các vấn đề nghiện lâu dài. Ghi nhớ các chuỗi dài trong một thời gian dài là cách làm việc của anh ấy.

Sự phổ biến của LSTM là do cơ chế Getting liên quan đến mỗi tế bào LSTM. Trong một ô RNN bình thường, đầu vào dấu thời gian và trạng thái ẩn của bước thời gian trước đó được chuyển qua lớp kích hoạt để có được trạng thái mới. Trong khi ở LSTM, quá trình này hơi phức tạp, như bạn có thể thấy trong kiến trúc ở trên mỗi lần, nó cần đầu vào của ba trạng thái khác nhau như trạng thái đầu vào hiện tại, bộ nhớ ngắn hạn của ô trước đó và cuối cùng là bộ nhớ dài hạn.

Các ô này sử dụng các cổng để điều chỉnh thông tin nào cần giữ hoặc loại bỏ trong quá trình hoạt động vòng lặp trước khi truyền thông tin dài hạn và ngắn hạn đến ô tiếp theo. Chúng ta có thể tưởng tượng những cánh cửa này như những bộ lọc loại bỏ thông tin không mong muốn đã chọn và không liên quan. Có tổng cộng ba cửa mà LSTM sử dụng làm cửa trước, cửa oblivion và cửa thoát hiểm.

Cửa trước

Cửa trước quyết định thông tin nào sẽ được lưu trữ trong bộ nhớ dài hạn. Nó chỉ hoạt động với thông tin từ đầu vào hiện tại và bộ nhớ ngắn hạn từ bước trước. Tại cổng này, nó lọc thông tin từ các biến không hữu ích.

### **Quên cửa**

Quên quyết định thông tin nào trong bộ nhớ dài hạn nên được giữ lại hoặc từ chối và điều này được thực hiện bằng cách nhân bộ nhớ dài hạn đến với một vectơ quên được tạo ra bởi đầu vào hiện tại và bộ nhớ ngắn đến.

### **Cửa thoát hiểm**

Cửa thoát sẽ lấy đầu vào hiện tại, bộ nhớ ngắn hạn trước đó và bộ nhớ dài hạn mới được tính toán để tạo ra bộ nhớ ngắn hạn mới sẽ được truyền đến tế bào trong bước thời gian tiếp theo. Lối ra từ bước thời gian hiện tại cũng có thể được rút ra từ trạng thái ẩn này.

Vì vậy, đây là cơ chế của LSTM để đạt được điều này với việc thực hiện thực tế. [Ở đây](https://colab.research.google.com/drive/1-c36k29ptuMUXUrQzH5SrQ-24t5pFZZL?usp=sharing) tôi đã trình bày trường hợp sử dụng LSTM trong đó bạn có thể kiểm tra các chuỗi đầu vào và đầu ra với hình dạng của chúng.

# ****Cách thức hoạt động của GRU****

**Đơn vị định kỳ đã đóng hoặc GRU là gì?**

Quy trình làm việc của đơn vị định kỳ đóng, viết tắt GRU, giống như quy trình làm việc của RNN, nhưng sự khác biệt nằm ở hoạt động và các cửa được liên kết với mỗi đơn vị GRU. Để giải quyết vấn đề mà RNN tiêu chuẩn gặp phải , GRU tích hợp hai cơ chế hoạt động của cửa được gọi là cửa cập nhật và cửa đặt lại.

### **Cập nhật cửa**

Cổng cập nhật chịu trách nhiệm xác định lượng thông tin trước đó sẽ được truyền đến trạng thái tiếp theo. Điều này thực sự mạnh mẽ vì mô hình có thể quyết định sao chép tất cả thông tin từ quá khứ và loại bỏ nguy cơ biến mất gradient.

### **Đặt lại cửa**

Cửa đặt lại được sử dụng từ mô hình để quyết định lượng thông tin trong quá khứ cần bỏ qua; nói tóm lại, nó quyết định xem trạng thái của ô trước đó có quan trọng hay không.

Trước hết, cửa đặt lại đi vào hoạt động, nó lưu trữ thông tin liên quan của giai đoạn dành cho thời gian trong một nội dung bộ nhớ mới. Sau đó, nó nhân vectơ đầu vào và trạng thái ẩn với trọng số của chúng. Tiếp theo, nó tính toán phép nhân trên mỗi mục giữa cửa đặt lại và bội số trạng thái ẩn trước đó. Sau khi tóm tắt các bước trên, chức năng kích hoạt phi tuyến được áp dụng và trình tự tiếp theo được tạo.

Đây là chức năng của GRU, các ví dụ thực tế được bao gồm trong máy tính xách tay.

# ****Sự khác biệt giữa GRU và LSTM là gì?****

Một vài điểm khác biệt như sau:   
GRU có hai cửa, LSTM có ba cửa  
Gru không có bộ nhớ trong, chúng không có cửa thoát có trong LSTM Trong LSTM  
, cửa trước và cửa mục tiêu được kết hợp bởi một cửa cập nhật và trong cửa đặt lại GRU được áp dụng trực tiếp vào trạng thái ẩn trước đó. Trong LSTM, trách nhiệm đối với cửa đặt lại được đảm nhận bởi cả hai cửa, tức là lối vào và mục tiêu.

# Kết thúc

Nhờ bài viết này, chúng tôi đã hiểu được sự khác biệt cơ bản giữa các đơn vị RNN, LSTM và GRU. Từ công việc của hai lớp, tức là LSTM và GRU, GRU sử dụng ít tham số ổ đĩa hơn và do đó sử dụng ít bộ nhớ hơn và chạy nhanh hơn LSTM trong khi LSTM chính xác hơn trên tập dữ liệu lớn hơn. Người ta có thể chọn LSTM nếu bạn đang xử lý các chuỗi lớn và liên quan đến độ chính xác, GRU được sử dụng khi bạn có ít bộ nhớ hơn và muốn có kết quả nhanh hơn.