TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo

Description automatically generated with low confidence

**BÀI TẬP LỚN**

HỌC PHẦN: HỌC SÂU VÀ ỨNG DỤNG

**ĐỀ TÀI: DỰ ĐOÁN TỐC ĐỘ GIÓ Ở TIBAU BẰNG PHƯƠNG PHÁP RNN, LSTM VÀ GRU**

Giáo viên hướng dẫn: Nguyễn Thị Kim Ngân

Sinh viên/nhóm sinh viên thực hiện:

1. Trần Thanh Thảo, lớp 62TH-NB

2. Nguyễn Hà Linh, lớp 63CNTT2

**Hà Nội, tháng 6 năm 2023**

**MỤC LỤC**

[**Phần 1: Lý thuyết** 3](#_Toc138001943)

[1. Mạng noron hồi tiếp (RNN) 3](#_Toc138001944)

[1.1 Recurrent Neural Network là gì? 3](#_Toc138001945)

[1.2 Dữ liệu dạng sequence 3](#_Toc138001946)

[1.3 Phân loại bài toán RNN 4](#_Toc138001947)

[1.4 Mạng hồi tiếp không có trạng thái ẩn 5](#_Toc138001948)

[1.5 Mạng hồi tiếp có trạng thái ẩn 5](#_Toc138001949)

[1.6 Định nghĩa mô hình RNN 6](#_Toc138001950)

[1.7 Ứng dụng của RNN 7](#_Toc138001951)

[2. Mạng noron hồi tiếp hiện đại 8](#_Toc138001952)

[2.1 Nút hồi tiếp có Cổng (GRU) 8](#_Toc138001953)

[2.2 Bộ nhớ Ngắn hạn Dài (LSTM) 9](#_Toc138001954)

[**Phần 2: Ứng dụng trong thực tế** 11](#_Toc138001955)

[1. Mô tả bài toán 11](#_Toc138001956)

[2. Mô tả tập dữ liệu của bài toán 11](#_Toc138001957)

[3. Thiết kế mô hình 12](#_Toc138001958)

[4. Kết quả thực nghiệm 13](#_Toc138001959)

[**Kết luận** 14](#_Toc138001960)

[**Tài liệu tham khảo** 14](#_Toc138001961)

# **Phần 1: Lý thuyết**

## ***Mạng noron hồi tiếp (RNN)***

### ***1.1 Recurrent Neural Network là gì?***

Bài toán: Cần phân loại hành động của người trong video, input là video 30s, output là phân loại hành động, ví dụ: đứng, ngồi, chạy, đánh nhau, bắn súng,...

Khi xử lý video ta hay gặp khái niệm FPS (frame per second) tức là bao nhiêu frame (ảnh) mỗi giây. Ví dụ 1 FPS với video 30s tức là lấy ra từ video 30 ảnh, mỗi giây một ảnh để xử lý.

Ta dùng 1 FPS cho video input ở bài toán trên, tức là lấy ra 30 ảnh từ video, ảnh 1 ở giây 1, ảnh 2 ở giây 2,... ảnh 30 ở giây 30. Bây giờ input là 30 ảnh: ảnh 1, ảnh 2,... ảnh 30 và output là phân loại hành động. Nhận xét:

* Các ảnh có thứ tự: ảnh 1 xảy ra trước ảnh 2, ảnh 2 xảy ra trước ảnh 3,... Nếu ta đảo lộn các ảnh thì có thể thay đổi nội dung của video. Ví dụ: nội dung video là cảnh bắn nhau, thứ tự đúng là A bắn trúng người B và B chết, nếu ta đảo thứ tự ảnh thành người B chết xong A mới bắn thì rõ ràng bây giờ A không phải là kẻ giết người => nội dung video bị thay đổi
* Ta có thể dùng CNN để phân loại 1 ảnh trong 30 ảnh trên, nhưng rõ ràng là 1 ảnh không thể mô tả được nội dung của cả video. Ví dụ: Cảnh người cướp điện thoại, nếu ta chỉ dùng 1 ảnh là người đấy cầm điện thoại lúc cướp xong thì ta không thể biết được cả hành động cướp.

🡺 Cần một mô hình mới có thể giải quyết được bài toán với input là sequence (chuỗi ảnh 1->30)

🡺 Recurrent Neural Network (RNN) ra đời.

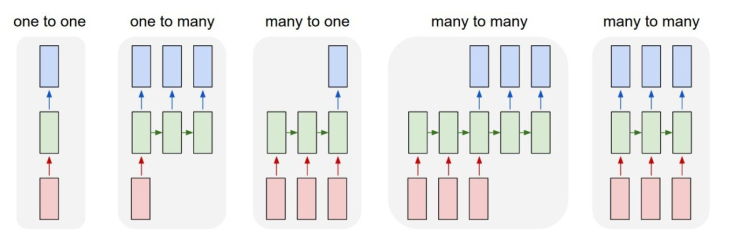
### ***1.2 Dữ liệu dạng sequence***

Dữ liệu có thứ tự như các ảnh tách ra từ video ở trên gọi là sequence, time-series data.

Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trướcđó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2,... i8 là lần khám tháng 8. (i1,i2,..i8) được gọi là sequence data. RNN sẽ học từ input và dự đoán xem bệnh nhân có bị đột quy tim hay không.

Ví dụ khác là trong bài toán dịch tự động với input là 1 câu, ví dụ "tôi yêu Việt Nam" thì vịtrí các từ và sự xắp xếp cực kì quan trọng đến nghĩa của câu và dữ liệu input các từ [’tôi’, ’yêu’,’việt’, ’nam’] được gọi là sequence data. Trong bài toán xử lý ngôn ngữ (NLP) thì không thể xửlý cả câu được và người ta tách ra từng từ (chữ) làm input, giống như trong video người tatách ra các ảnh (frame) làm input.

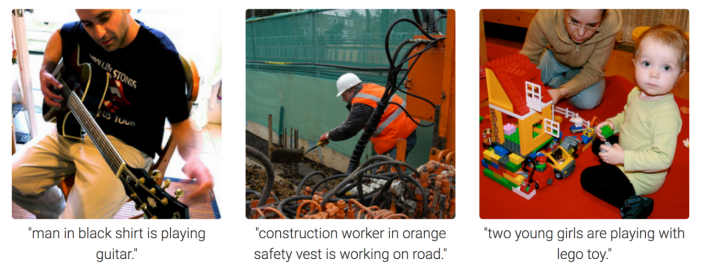
### **1.3 Phân loại bài toán RNN**



Hình 1: Các dạng bài toán RNN

**One to one**: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với bài toán phân loại ảnh MNIST input là ảnh và output là ảnh đấy là số nào

**One to many**: bài toán có 1 input nhưng nhiều output, ví dụ với bài toán caption cho ảnh, input là 1ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.



Hình 2: Ví dụ image captioning

**Many to one**: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video

**Many to many**: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: "I love Vietnam" và output cũng là 1 câu gồm nhiều chữ "Tôi yêu Việt Nam". Để ý là độ dài sequence của input và output có thể khác nhau.

**Ưu và nhược điểm** của một kiến trúc RNN thông thường

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| * Khả năng xử lí 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 | * 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 |

### **1.4 Mạng hồi tiếp không có trạng thái ẩn**

|  |  |
| --- | --- |
| - Xét một perception đa tầng với một tầng ẩn duy nhất   * Giả sử ta có một minibatch với n mẫu và d đầu vào * Tầng ẩn có h nút ẩn, hàm kích hoạt của tầng ẩn là |  |

* Đầu ra của tầng ẩn được tính như sau:

|  |  |
| --- | --- |
|  | : tham số trọng số, : hệ số điều chỉnh |

* Tầng đầu ra được tính toán bởi:

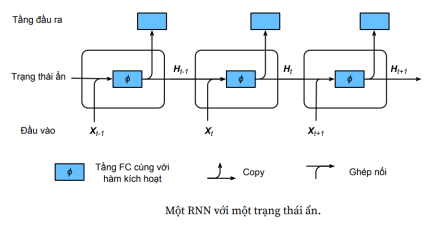
|  |  |
| --- | --- |
|  | : tham số trọng số, : hệ số điều chỉnh,  : hệ số điều chỉnh, |

### **1.5 Mạng hồi tiếp có trạng thái ẩn**

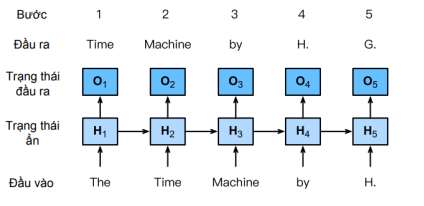
* Thời điểm t tương ứng với bước t trong một vòng lặp. giả sử, tại bước thời gian t ta có:
* Đầu vào Xt € , t = 1, …., T
* Ht € là biến ẩn tại bước thời gian t của chuỗi



* Đầu ra tại bước thời gian t: **Ot =**



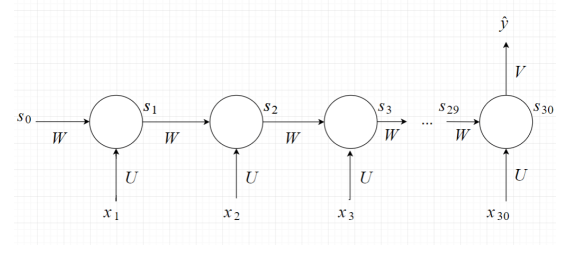
* Minh họa cách ước lượng từ tiếp theo dựa trên từ hiện tại và các từ trước đó của chuỗi “the time machine by H. G. Wells”



Mô hình ngôn ngữ mở mức từ ngữ RNN. Đầu vào và chuỗi nhãn lần lượt là the time machine by H. và time machine by H. G.

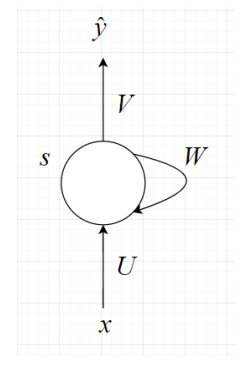
### **1.6 Định nghĩa mô hình RNN**

* Bài toán: Nhận diện hành động trong video 30s. Đây là dạng bài toán many to one trong RNN, tức nhiều input và 1 output.
* Input ta sẽ tách video thành 30 ảnh (mỗi giây một ảnh). Các ảnh sẽ được cho qua pretrained model CNN để lấy ra các feature (feature extraction) vector có kích thước n\*1. Vector tương ứng với ảnh ở giây thứ i là xi.
* Output là vector có kích thước d\*1 (d là số lượng hành động cần phân loại), softmax function được sử dụng như trong bài phân loại ảnh.



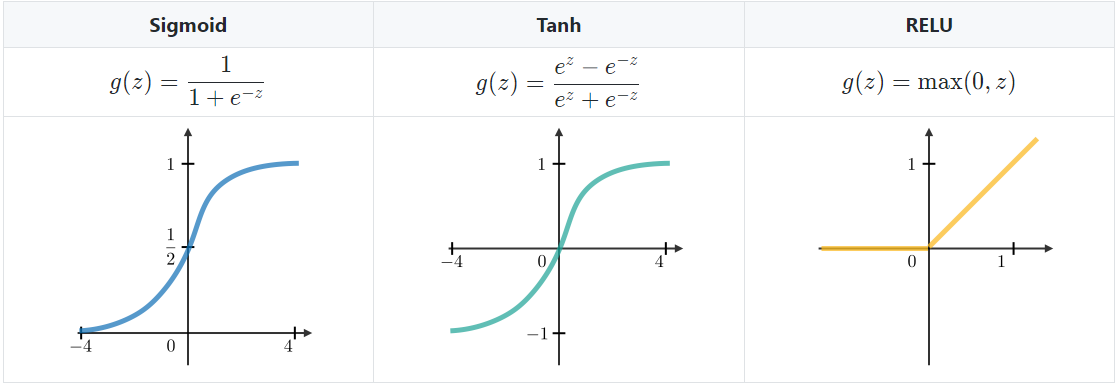
Mô hình RNN cho bài toán

* Ta có:
* Mô hình có 30 input và 1 output, các input được cho vào model đúng với thứ tự ảnh trong video x1;x2;…;x30
* Mỗi hình tròn được gọi là 1 state, state t có input là xt và st-1 (output của state trước); output là st = f(U ∗xt +W ∗st-1). f là activation function thường là Tanh hoặc ReLU.
* Có thể thấy st mang cả thông tin từ state trước ( st-1) và input của state hiện tại => st giống như memory nhớ các đặc điểm của các input từ x1 đến xt
* s0 được thêm vào chỉ cho chuẩn công thức nên thường được gán bằng 0 hoặc giá trị ngẫu nhiên. Có thể hiểu là ban đầu chưa có dữ liệu gì để học thì memory rỗng.
* Do ta chỉ có 1 output, nên sẽ được đặt ở state cuối cùng, khi đó s30 học được thông tin từ tất cả các input. yˆ = g(V ∗ s30). g là activation function, trong bài này là bài toán phân loại nên sẽ dùng softmax.
* Ta thấy là ở mỗi state các hệ số W, U là giống nhau nên model có thể được viết lại thành:



Mô hình RNN rút gọn

* Tóm lại:
* xi là vector có kích thước n\*1, si là vector có kích thước m\*1, yi là vector có kích thước d\*1. U là ma trận có kích thước m\*n, W là ma trận có kích thước m\*m và V là ma trận có kích thước d\*m.
* s0 = 0;st = f(U ∗xt +W ∗st-1) với t >= 1
* yˆ = g(V ∗s30)
* **Các hàm kích hoạt thường dùng -** Các hàm kích hoạt thường dùng trong các modules RNN được miêu tả như sau:



### **1.7 Ứng dụng của RNN**

Về cơ bản nếu bạn thấy sequence data hay time-series data và bạn muốn áp dụng deep learning thì bạn nghĩ ngay đến RNN. Dưới đây là một số ứng dụng của RNN:

* Speech to text: Chuyển giọng nói sang text.
* Sentiment classification: Phân loại bình luận của người dùng, tích cực hay tiêu cực.
* Machine translation: Bài toán dịch tự động giữa các ngôn ngữ.
* Video recognition: Nhận diện hành động trong video.
* Heart attack: Dự đoán đột quỵ tim.

## **Mạng noron hồi tiếp hiện đại**

### **Nút hồi tiếp có Cổng (GRU)**

#### **So sánh RNN và GRU**

Sự khác biệt chính giữa RNN thông thường và GRU là GRU hỗ trợ việc kiểm soát trạng thái ẩn. GRU có các cơ chế được học để quyết định khi nào nên cập nhật và khi nào nên xóa trạng thái ẩn

* Nếu ký tự đầu tiên có mức độ quan trọng cao, mô hình sẽ học để không cập nhật trạng thái ẩn sau lần quan sát đầu tiên
* Học cách bỏ qua những quan sát tạm thời không liên quan, cũng như cách xóa trạng thái ẩn khi cần thiết
* Một ô nhớ GRU có 2 cổng: Cổng xóa, cổng cập nhật

#### **2.1.2 Cổng xóa và cổng cập nhật**

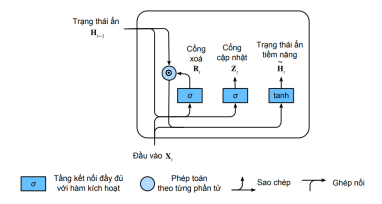
|  |  |
| --- | --- |
| - Cổng xóa cho phép kiểm soát bao nhiêu  phần của trạng thái trước đây được giữ lại    - Cổng cập nhật cho phép kiểm soát bao  nhiêu phần của trạng thái mới sẽ giống  trạng thái cũ |  |

#### **2.1.3 Hoạt động của Cổng Xóa**

Để giảm ảnh hưởng của các trạng thái trước đó, ta nhân Ht-1 với Rt theo từng phần tử

* Nếu các phần tử của Rt có giá trị gần với 1, kết quả sẽ giống RNN thông thường
* Nếu tất cả các phần tử của Rt gần với 0, trạng thái ẩn sẽ là đầu ra của một perceptron đa tầng với đầu vào là Xt. Bất kỳ trạng thái ẩn nào tồn tại trước đó đều được đặt lại về giá trị mặc định
* Tại đây, nó được gọi là trạng thái ẩn tiềm năng



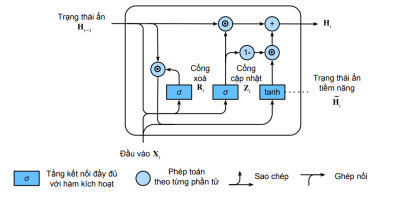


#### **2.1.4 Hoạt động của Cổng Cập nhật**

Cổng Cập nhật xác định mức độ giống nhau giữa trạng thái mới Ht và trạng thái cũ Ht-1, cũng như mức độ trạng thái ẩn tiềm năng được sử dụng



* Nếu Zt = 1, chúng ta giữ lại trạng thái cũ. Thông tin từ Xt được bỏ qua, tương đương với việc bỏ qua bước thời gian t trong chuỗi phụ thuộc
* Nếu Zt gần 0, trạng thái ẩn Ht sẽ gần với trạng thái ẩn tiềm năng



#### **2.1.5 Tính chất của GRU**

🡺 Các mạng GRU có 2 tính chất nổi bật sau:

* Cổng xóa giúp nắm bắt các phụ thuộc ngắn hạn trong chuỗi thời gian.
* Cổng cập nhật giúp nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian

### **2.2 Bộ nhớ Ngắn hạn Dài (LSTM)**

Mạng LSTM được thiết kế gồm 3 cổng:

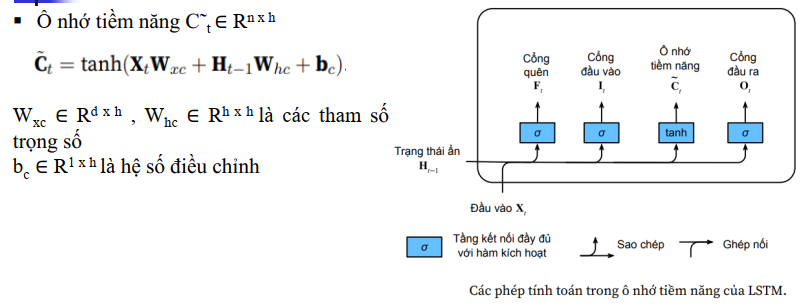
|  |  |
| --- | --- |
| * Cổng đầu ra (ouput gate): để đọc các thông tin từ ô nhớ * Cổng đầu vào (input gate): để quyết định khi nào cần ghi dữ liệu vào ô nhớ * Cổng quên (forget gate): để thiết lập lại   nội dung chứa trong ô nhớ   * Xt, và Ht-1 được xử lý bởi một tầng kết nối đầy đủ và một hàm kích hoạt sigmoid để tính toán các giá trị của các cổng |  |

#### **2.2.1 Cổng đầu vào, Cổng quên, Cổng đầu ra**

Giả sử có h nút ẩn, mỗi minibatch có kích thước n và kích thước đầu vào là d

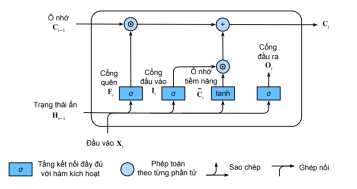
|  |  |
| --- | --- |
| - Đầu vào:  - Trạng thái ẩn: | - Cổng đầu vào:  - Cổng quên: |
| - Cổng đầu ra: |  |

#### **2.2.2 Ô nhớ Tiềm năng**



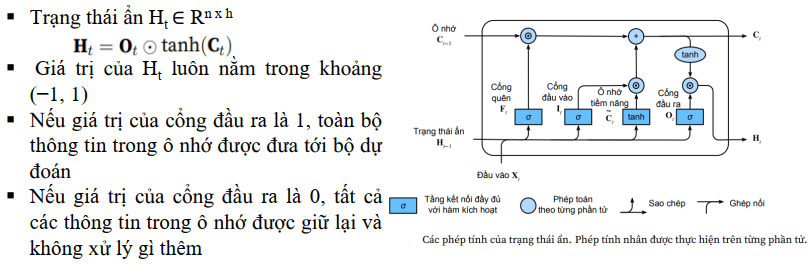
#### **2.2.3 Ô nhớ**

* + điều chỉnh lượng dữ liệu mới được lấy vào thông qua
  + chỉ định lượng thông tin cũ cần giữ lại trong ô nhớ
  + Nếu Ft xấp xỉ bằng 1 và It xấp xỉ bằng 0, thì giá trị ô nhớ trong quá khứ Ct-1 sẽ  
    được lưu lại qua thời gian và truyền tới bước thời gian hiện tại
  + Thiết kế này nhằm giảm bớt sự tiêu biến gradient, nắm bắt các phụ thuộc dài hạn  
    trong chuỗi thời gian tốt hơn

**

Các phép tính toán trong ô nhớ của LSTM. Ở đây. Ta sử dụng phép nhân theo từng phần tử

#### **2.2.4 Các trạng thái ẩn**



# **Phần 2: Ứng dụng trong thực tế**

## ***Mô tả bài toán***

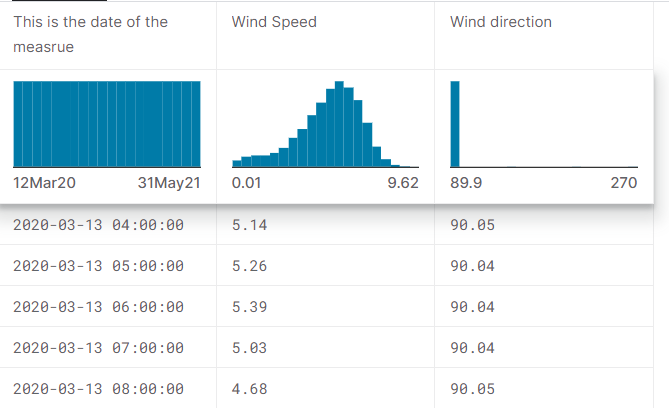
* **Tên bài toán**: Dự đoán tốc độ gió ở Tibau
* **Mục đích của bài toán**: Mục tiêu của bài toán nhằm xây dựng một mô hình học máy sử dụng phương pháp mạng nơ-ron hồi tiếp (RNN), nút hồi tiếp có cổng (GRU), bộ nhớ ngắn hạn dài (LSTM) để đạt được chất lượng dự đoán tốt nhất
* **Input**: tập dữ liệu gồm các thuộc tính và đã được gán nhán
* **Output**: kết quả dự đoán tốc độ gió
* **Tóm tắt công việc thực hiện của bài toán**:
* Xử lí dữ liệu:
* Đọc dữ liệu từ file
* Chia tập dữ liệu thành 3 phần:
* Training data: 70% dùng để huấn luyện mô hình
* Validattion data: 10% dùng để kiểm thử mô hình
* Test data: 20% dùng để đánh giá mô hình
* Chuẩn hóa dữ liệu
* Xây dựng mô hình dự đoán: Sử dụng các phương pháp RNN, GRU, LSTM để thực hiện bài toán (kết hợp thêm các kỹ thuật BatchNorm, Dropout)
* Huấn luyện mô hình: Đưa dữ liệu huấn luyện vào mô hình và điều chỉnh các tham số để mô hình học được từ dữ liệu và tối ưu hóa hiệu suất dự đoán.
* Đánh giá mô hình: Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra độc lập. Sử dụng các độ đo cho bài toán hồi quy: NSE, R2, MAE, RMSE

## ***Mô tả tập dữ liệu của bài toán***

* **Link download dữ liệu của bài toán:**

https://www.kaggle.com/datasets/ramontanoeiro/tibau-wind-speed?fbclid=IwAR2Qrd87VX93By5BIAHTCKGRuNrQhz3Ryhz\_iRAc21HIoJgLtY6cLsk-Jlw

* **Mô tả tập dữ liệu của bài toán:**
* Tập dữ liệu có 10681 vector gồm các thuộc tính và nhãn
* Mỗi cột tương ứng với một thuộc tính lần lượt là: Wind Speed (tốc độ gió), Wind direction (hướng gió)



## **Thiết kế mô hình**

- **Mô hình RNN:**

|  |  |
| --- | --- |
| Mạng noron được thiết kế:  + RNN: 100 node, hàm kích hoạt ‘relu’  + Dense: 100 node, hàm kích hoạt ‘relu’  + Dense: 50 node, hàm kích hoạt ‘relu’  + Dense: 1 node (kết quả dự báo đầu ra) | Mô hình thiết kế: |

**- Mô hình LSTM:**

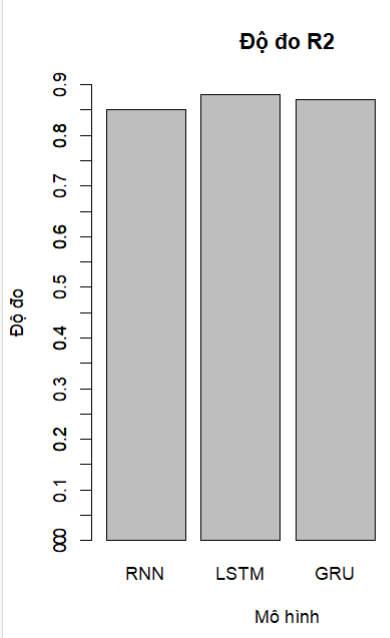
|  |  |
| --- | --- |
| Mạng noron được thiết kế:  + LSTM: 100 node, hàm kích hoạt ‘relu’  + Dense: 100 node, hàm kích hoạt ‘relu’  + Dense: 50 node, hàm kích hoạt ‘relu’  + Dense: 1 node (kết quả dự báo đầu ra) | Mô hình thiết kế: |

- **Mô hình GRU:**

|  |  |
| --- | --- |
| Mạng noron được thiết kế:  + GRU: 100 node, hàm kích hoạt ‘relu’  + Dense: 100 node, hàm kích hoạt ‘relu’  + Dense: 50 node, hàm kích hoạt ‘relu’  + Dense: 1 node (kết quả dự báo đầu ra) | Mô hình thiết kế: |

## ***Kết quả thực nghiệm***

* Biểu đồ so sánh chất lượng của các mô hình RNN, GRU, LSTM trên tập test bằng độ đo R2:



* Các độ đo đánh giá chất lượng mô hình trên tập test:

|  |  |  |  |
| --- | --- | --- | --- |
|  | RNN | LSTM | GRU |
| MSE | 0.32 | 0.25 | 0.29 |
| RMSE | 0.56 | 0.50 | 0.53 |
| MAE | 0.45 | 0.38 | 0.42 |
| R2 | 0.85 | 0.88 | 0.84 |

# **Kết luận**

Qua bài làm về đề tài: Ứng dụng mạng noron hồi tiếp (RNN) và mạng noron Hồi tiếp hiện đại để dự đoán tốc độ gió ở Tibuan, nhóm chúng em hi vọng mô hình này có thể phát triển hơn nữa. Đồng thời thông qua bài làm giúp nhóm em có những kiến thức về môn học máy cũng như áp dụng kiến thức đó vào phân tích các bài toán thực tế.

Mặc dù đã có nhiều cố gắng trong suốt quá trình làm bài tập, nhưng kiến thức còn hạn chế, cũng như chưa có kinh nghiệm trong thực tế nên không thể tránh khỏi những sai sót. Vì vậy, nhóm em mong thầy, cô góp ý để có thể chỉnh sửa, hoàn thiện bài làm tốt hơn.

# **Tài liệu tham khảo**

1. Đắm mình vào Học sâu:

<https://d2l.aivivn.com/chapter_recurrent-neural-networks/rnn_vn.html>

1. Hai’s Blog:

<https://dominhhai.github.io/vi/2017/10/what-is-rnn/>

1. Shervine Amidi:

<https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-recurrent-neural-networks>

1. Slide bài giảng