



BÁO CÁO

MÔN ĐIỆN TOÁN ĐÁM MÂY

Đề tài: “Triển khai mô hình học sâu dự đoán giá sim điện thoại trên nền tảng Amazon Web Services”

Giảng viên hướng dẫn : T.S Nguyễn Đức Bình

Thành viên nhóm : Nguyễn Thái Học

Lý Quốc Huy

Nông Minh Đức

Nhóm thực hiện : Nhóm 5

Lớp : CNTT K19 CLC

Thái Nguyên, năm 2024

MỤC LỤC

DANH SÁCH HÌNH VẼ	1
LỜI NÓI ĐẦU	2
1 TỔNG QUAN VỀ NỀN TẢNG ĐIỆN TOÁN ĐÁM MÂY AMAZON WEB SERVICES	3
1.1 Khái niệm về Amazon Web Services	3
1.2 Dịch vụ điện toán đám mây AWS	3
1.2.1 Compute	3
1.2.2 Storage and Content Delivery	3
1.2.3 Database	4
1.2.4 Networking	4
1.2.5 Developer tools	4
1.2.6 Management tools	5
1.2.7 Security and Identity	5
1.3 Lợi ích của AWS	5
2 TỔNG QUAN VỀ BÀI TOÁN DỰ ĐOÁN GIÁ SIM	6
2.1 Giới thiệu về bài toán định giá sim	6
2.2 Phương pháp triển khai bài toán	7
2.2.1 Ưu điểm	7
2.2.2 Hạn chế	7
3 CƠ SỞ LÝ THUYẾT VỀ HỌC SÂU VÀ MÔ HÌNH MẠNG LSTM	9
3.1 Tổng quan lý thuyết học sâu	9
3.1.1 Khái niệm về học sâu	9

MỤC LỤC

3.1.2	Cách hoạt động của học sâu	9
3.1.3	Lĩnh vực và ứng dụng trong học sâu	10
3.2	Một số mô hình mạng tiêu biểu trong học sâu	11
3.2.1	Convolutional Neural Network	11
3.2.2	Recurrent Neural Network	14
3.2.3	Long Short Term Memory	16
4	TRIỂN KHAI MÔ HÌNH TRÊN NỀN TẢNG AMAZON WEB SERVICES	19
4.1	Ngôn ngữ và công cụ	19
4.1.1	Ngôn ngữ	19
4.1.2	Công cụ	19
4.2	Bộ dữ liệu	20
4.2.1	Crawl dữ liệu	20
4.2.2	Tập dữ liệu	21
4.3	Tiền xử lý dữ liệu	22
4.3.1	Làm sạch dữ liệu	22
4.3.2	Gán nhãn dữ liệu	22
4.3.3	Chuẩn hoá dữ liệu	23
4.4	Huấn luyện mô hình	24
4.5	Đánh giá và kiểm thử mô hình	25
4.5.1	Đánh giá mô hình	25
4.5.2	Thử nghiệm mô hình	27
4.6	Triển khai mô hình trên nền tảng Amazon Web Services	27
4.6.1	Tạo tài khoản AWS	27
4.6.2	Tạo máy chủ ảo trên AWS	28
4.6.3	Đưa web lên hệ thống máy chủ ảo AWS	28
	KẾT LUẬN	30
	TÀI LIỆU THAM KHẢO	31
	PHỤ LỤC CODE	32

NHẬN XÉT CỦA GIẢNG VIÊN	36
--------------------------------	-----------

DANH SÁCH HÌNH VẼ

3.1	Cấu trúc mô hình mạng nơ ron tích chập	13
3.2	Các dạng bài toán RNN	15
3.3	Mô hình RNN	16
3.4	Cấu trúc mạng LSTM	17
4.1	Giao diện google colab	20
4.2	Website mua bán sim điện thoại	20
4.3	File đường dẫn các trang web con	21
4.4	Tập dữ liệu về sim điện thoại sau khi được crawl từ website	21
4.5	Dữ liệu sau khi được làm sạch	22
4.6	Phân khoảng giá trị tương ứng với từng nhãn	23
4.7	Tập dữ liệu sau khi được gán nhãn	23
4.8	Mô hình mạng LSTM cho bài toán dự đoán giá sim	24
4.9	Đồ thị biểu diễn độ chính xác của mô hình	25
4.10	Đồ thị biểu diễn sai số của mô hình	26
4.11	Thử nghiệm mô hình trên tập kiểm tra	27
4.12	Giao diện Amazon Web Services	28
4.13	Máy chủ ảo trên Amazon Web Services	28
4.14	Mô hình dự đoán giá sim triển khai trên nền tảng AWS	29

LỜI NÓI ĐẦU

Trong thời đại kỹ thuật số ngày nay, điện toán đám mây (cloud computing) đã trở thành một trong những xu hướng công nghệ quan trọng nhất. Nó mang lại nhiều lợi ích đáng kể, bao gồm khả năng mở rộng linh hoạt, tiết kiệm chi phí và khả năng quản lý tài nguyên hiệu quả. Với sự phát triển nhanh chóng của mô hình này, việc hiểu và áp dụng các công cụ và nền tảng quản trị là vô cùng quan trọng.

Trong báo cáo này, nhóm chúng tôi tập trung vào nghiên cứu xây dựng một ứng dụng triển khai trên nền tảng điện toán đám mây. Cụ thể là bài toán "Triển khai mô hình học sâu dự đoán giá sim điện thoại trên nền tảng Amazon Web Services".

CHƯƠNG 1: TỔNG QUAN VỀ NỀN TẢNG ĐIỆN TOÁN ĐÁM MÂY AMAZON WEB SERVICES

1.1 Khái niệm về Amazon Web Services

Amazon Web Services (AWS) là nền tảng dịch vụ đám mây an toàn, mang đến khả năng tính toán, lưu trữ cơ sở dữ liệu, phân phối nội dung và các chức năng khác nhằm giúp các doanh nghiệp mở rộng và phát triển.

1.2 Dịch vụ điện toán đám mây AWS

Dịch vụ điện toán đám mây AWS cung cấp: compute, storage, database, networking, developer tools, management tools, security and identity.

1.2.1 Compute

- Amazon Elastic Computer Cloud (EC2): dịch vụ máy chủ ảo
- Elastic Load Balancing (ELB): dịch vụ cân bằng tải
- AWS Lambda: dịch vụ triển khai code không server (serverless)
- AWS Elastic Beanstalk: triển khai các ứng dụng web
- VM Import/Export: import/export ảnh các máy ảo

1.2.2 Storage and Content Delivery

- Amazon S3: dịch vụ lưu trữ đối tượng

- Amazon Glacier: dịch vụ lưu trữ dữ liệu ít truy cập
- Amazon Elastic Block Store (EBS): dịch vụ lưu trữ dạng
- Amazon Elastic File System (EFS): dịch vụ lưu trữ và chia sẻ file.
- Amazon CloudFront: dịch vụ phân phối nội dung (content delivery).

1.2.3 Database

- Amazon RDS: dịch vụ cơ sở dữ liệu mô hình quan hệ Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL & MariaDB.
- Amazon DynamoDB: dịch vụ cơ sở dữ liệu NoSQL của Amazon
- Amazon Redshift: dịch vụ kho dữ liệu của Amazon
- Amazon ElastiCache: dịch vụ cache dữ liệu của Amazon

1.2.4 Networking

- Amazon VPC: dịch vụ mạng riêng ảo
- Amazon Direct Connect: dịch vụ thiết lập kết nối dành riêng từ AWS đến DataCenter.
- Amazon Route53: dịch vụ quản lý tên miền DNS và định tuyến đến các dịch vụ của AWS.

1.2.5 Developer tools

- Amazon CodeCommit: dịch vụ quản lý code, có thể giao tiếp với git.
- Amazon CodeDeploy: dịch vụ triển khai code tự động lên các máy chủ ảo EC2, Lambda
- Amazon CodePipeline: dịch vụ liên quan đến code như cập nhật, biên dịch, test, ...

1.2.6 Management tools

- Amazon CloudWatch: giám sát các nguồn tài nguyên
- AWS CloudFormation: quản lý các nguồn tài nguyên
- AWS CloudTrail: dịch vụ lưu lại lịch sử hoạt động các dịch vụ
- AWS Config: dịch vụ quản lý cấu hình AWS
- AWS OpsWorks: dịch vụ định nghĩa cấu trúc ứng dụng

1.2.7 Security and Identity

- AWS Identity and Access Management: quản lý người dùng và quyền truy cập dịch vụ AWS
- AWS Key Management Service (KMS): quản lý khóa mã hóa các dịch vụ
- AWS Directory Service: quản lý và truy cập các tài nguyên để dàng
- AWS WAF: dịch vụ tường lửa cho các ứng dụng web
- AWS CloudHSM: dịch vụ bảo mật các mô đun phần cứng. Sinh và quản lý khóa mã hóa.

1.3 Lợi ích của AWS

- Nền tảng AWS đáp ứng hầu hết mọi nhu cầu sử dụng.
- Nhanh chóng.
- Quy mô linh hoạt.
- Tiết kiệm chi phí.
- Triển khai trên toàn cầu chỉ trong vài phút.

CHƯƠNG 2: TỔNG QUAN VỀ BÀI TOÁN DỰ ĐOÁN GIÁ SIM

2.1 Giới thiệu về bài toán định giá sim

Sim là từ viết tắt của "Subscriber Identity Module" (mô-đun nhận dạng thuê bao) và là một thiết bị nhỏ có chứa thông tin nhận dạng cá nhân của một thuê bao di động, cho phép người dùng thực hiện các cuộc gọi, gửi tin nhắn và truy cập các dịch vụ di động khác. Tùy thuộc vào từng đặc điểm của sim, loại nhà mạng mà mỗi sim có một giá trị khác nhau. Hiện nay với việc phát triển không ngừng của khoa học dữ liệu đặc biệt là trong các lĩnh vực học máy (Machine Learning) học sâu (Deep Learning), điều đó là tiền đề, để chúng ta ứng dụng các kỹ thuật, phương pháp vào một bài toán thực tế. Bài toán định giá sim điện thoại là quá trình xác định giá trị tài sản của một sim điện thoại di động dựa trên các yếu tố như số lượng, phân loại, tính độc đáo và tính hiếm có của số điện thoại đó. Quá trình này thường được thực hiện bởi các chuyên gia hoặc công ty chuyên về giao dịch sim điện thoại.

Để định giá một sim điện thoại, các yếu tố sau đây thường được đưa vào làm tiêu chuẩn để đánh giá:

- **Số lượng:** Sim điện thoại có giá trị cao thường là những con số ngắn, ví dụ như sim 4 số, 5 số. Sim có số lượng lặp lại, ví dụ như các con số giống nhau hoặc dãy số liên tiếp, cũng có thể có giá trị cao.
- **Phân loại:** Các sim điện thoại được phân thành nhiều loại, bao gồm sim thông thường, sim VIP, sim đẹp, sim tứ quý, sim lộc phát và nhiều hơn nữa. Sim VIP hoặc sim có tính độc đáo đặc biệt thường có giá trị cao hơn.
- **Tính độc đáo:** Sim điện thoại với các con số đặc biệt hoặc dễ nhớ, ví dụ như các con số lặp lại, dãy số đặc biệt, con số may mắn, có thể có giá trị cao hơn. Sự độc đáo của số

điện thoại có thể tăng giá trị của sim.

- Thị trường: Giá trị của sim điện thoại cũng phụ thuộc vào thị trường và nhu cầu của người mua. Các sim điện thoại phổ biến và được ưa chuộng trong cộng đồng người dùng sim có thể có giá trị cao hơn.

Mặc dù Machine Learning và Deep Learning vẫn đang phát triển vô cùng mạnh mẽ, tuy nhiên cũng chưa có một nghiên cứu nào đề cập và ứng dụng đến chủ đề về định giá sim điện thoại.

2.2 Phương pháp triển khai bài toán

Bài toán "Crawl dữ liệu sim điện thoại và ứng dụng mô hình học sâu dự đoán giá sim" nhằm ứng dụng các thuật toán học sâu, triển khai mô hình có khả năng dự đoán giá sim từ một số điện thoại bất kì. Mô hình này được xây dựng và huấn luyện trên tập dữ liệu được crawl từ một trang web sim điện thoại.

2.2.1 Ưu điểm

Về mặt lý thuyết bài toán trên, học máy hoàn toàn cũng có thể được ứng dụng để dự đoán một giá trị của một số điện thoại. Tuy nhiên ta có thể thấy rằng giá trị của sim điện thoại không phải là một giá trị liên tục điều đó sẽ dẫn đến khả năng mô hình bị sai lệch và không thể hoạt động hiệu quả do các thuật toán học máy chỉ hoạt động tốt trên các giá trị biến liên tục. Thêm vào đó các số sim điện thoại bao gồm 10 số từ 0 đến 9. Như đã phân tích ở trên giá sim điện thoại cao hay thấp phụ thuộc rất nhiều vào sự kết hợp, vị trí của các con số với nhau, học máy sẽ gặp rất nhiều khó khăn vì không thể nhận biết ra được sự đặc biệt, hay học được quy luật giữa các con số. Chính vì vậy học sâu được ứng dụng để giải quyết các vấn đề này.

2.2.2 Hạn chế

Giá sim điện thoại có rất nhiều khoảng giá và luôn luôn thay đổi theo thời gian và thị trường. Để một mô hình có hiệu suất, hiệu quả cao thì cần có một tập dữ liệu đủ lớn và luôn luôn được cập nhật. Việc giá sim điện thoại có rất nhiều khoảng giá đồng nghĩa với việc mô

hình sẽ có rất nhiều đầu ra do mô hình phải xác định xem số điện thoại này có giá trị là bao nhiêu. Đối với dữ liệu có quá nhiều đầu ra, mô hình sẽ trở nên phức tạp, học hỏi quá chi tiết từ dữ liệu cũng sẽ làm giảm hiệu suất, hiệu quả của mô hình.

CHƯƠNG 3: CƠ SỞ LÝ THUYẾT VỀ HỌC SÂU VÀ MÔ HÌNH MẠNG LSTM

3.1 Tổng quan lý thuyết học sâu

3.1.1 Khái niệm về học sâu

Học sâu (Deep Learning) là một kỹ thuật học máy tiên tiến dựa trên việc học đại diện. Cách tiếp cận mạnh mẽ này cho phép máy móc tự động học các biểu diễn tính năng cao cấp từ dữ liệu. Do đó, các mô hình học sâu đạt được kết quả hiện đại trong các nhiệm vụ đầy thử thách.

Các thuật toán học sâu sử dụng mạng nơ-ron nhân tạo, một hệ thống máy tính học các tính năng cấp cao từ dữ liệu bằng cách tăng độ sâu (tức số lớp) trong mạng. Mạng lưới thần kinh được lấy cảm hứng từ một phần tử mạng lưới thần kinh sinh học, nơi các tế bào trong hầu hết các bộ não (bao gồm cả chúng ta) kết nối và hoạt động cùng nhau. Mỗi tế bào này trong một mạng lưới thần kinh được gọi là tế bào thần kinh.

3.1.2 Cách hoạt động của học sâu

Mạng nơ-ron học sâu cố gắng bắt chước bộ não con người thông qua sự kết hợp của dữ liệu đầu vào, trọng số và bias (tham số trong quá trình huấn luyện). Các yếu tố này hoạt động cùng nhau để nhận dạng, phân loại và mô tả chính xác các đối tượng trong dữ liệu.

Mạng nơ-ron học sâu bao gồm nhiều lớp được kết nối với nhau, mỗi lớp được xây dựng dựa trên lớp trước đó để tinh chỉnh và tối ưu hóa dự đoán hoặc phân loại. Sự tiến triển của các tính toán thông qua mạng được gọi là lan truyền chuyển tiếp. Các lớp đầu vào và đầu ra là nơi mô hình Deep Learning thu nạp dữ liệu để xử lý và lớp đầu ra là nơi đưa ra dự đoán hoặc phân loại cuối cùng.

Một quá trình khác được gọi là backpropagation (backward propagation of errors) là phương pháp lan truyền ngược được sử dụng các thuật toán như Gradient Descent, để tính toán lỗi trong các dự đoán và sau đó điều chỉnh trọng số và độ lệch của hàm bằng cách di chuyển ngược qua các lớp trong quá trình huấn luyện. Cùng với sự lan truyền về phía trước, lan truyền ngược cho phép một mạng lưới thần kinh đưa ra dự đoán và khắc phục bất kỳ sai số nào để mô hình đạt độ chính xác cao nhất.

3.1.3 Lĩnh vực và ứng dụng trong học sâu

a. Lĩnh vực

Chỉ trong khoảng vài năm gần đây, học sâu đã mang đến nhiều bất ngờ trên quy mô toàn cầu và dẫn đường nhanh chóng trong nhiều lĩnh vực khác nhau:

- Xử lý ngôn ngữ tự nhiên (Natural Language Processing): học sâu được sử dụng để xây dựng các mô hình NLP như dịch máy, hiểu và sản xuất văn bản tự nhiên.
- Thị giác máy tính (Computer Vision): học sâu được sử dụng để phân loại đối tượng trong hình ảnh: như con người, động vật, thiết bị và nhiều hơn nữa.
- Học tăng cường (Reinforcement Learning): là một mô hình trong đó một tác nhân tương tác với môi trường và học từ kinh nghiệm của mình để đưa ra các hành động tối ưu được mục tiêu. Trong học tăng cường, tác nhân phải tìm hiểu cách tương tác với môi trường, tìm kiếm các phương pháp để thu thập thông tin từ môi trường và từ đó tính toán ra các hành động tối ưu.

b. Ứng dụng

Các ứng dụng của Deep Learning rất rộng lớn, dưới đây tôi sẽ đề cập đến những ứng dụng phổ biến và được sử dụng nhiều nhất của các kỹ thuật Deep Learning:

- Phân loại hình ảnh: học sâu hiện nay đã tiên tiến đến mức chúng ta có thể nhận ra các đối tượng khác nhau trong một bức tranh. Ví dụ, một bức ảnh được chụp trong nhà hàng có những đặc điểm khác nhau trong đó, như bàn ghế, thực phẩm, tâm trạng của những người trong ảnh, .v.v. Bằng cách nhìn vào những hình ảnh đó ta có thể phát hiện ra được sở thích của người đó và có thể đưa đề xuất những địa điểm tương tự để mua hoặc ghé thăm, v.v.

- Nhận dạng giọng nói: lời nói là phương pháp giao tiếp phổ biến của xã hội loài người. Khi giao tiếp một người nói, người kia nhận ra và hiểu để đưa ra phản hồi phù hợp, giống như cách mô hình học sâu đang nâng cao khả năng để hiểu được con người. Mục tiêu là nhận ra và phản hồi một người nói không xác định thông qua tín hiệu âm thanh của họ.
- Phát hiện gian lận: mô hình Deep Learning sử dụng nhiều nguồn dữ liệu để gán nhãn quyết định xem đâu là hành vi gian lận áp dụng trong các cửa hàng, khách sạn, thi cử.
- Trong y tế: mô hình Deep Learning được áp dụng trong chụp X quang nhận diện ung thư và được đánh giá là một mô hình hiệu quả, ngoài ra mạng nơ-ron sâu giúp điều tra vòng đời tế bào.

3.2 Một số mô hình mạng tiêu biểu trong học sâu

Trong học sâu có hai mô hình mạng lớn là Convolutional Neural Network (CNN) cho bài toán có đầu vào là ảnh và Recurrent Neural Network (RNN) cho bài toán dữ liệu dạng chuỗi (sequence).

3.2.1 Convolutional Neural Network

a. Đặc trưng chung của các mạng nơ ron tích chập

- Sử dụng tích chập: các mạng CNN đều trích xuất đặc trưng dựa trên nguyên lý tích chập.
- Kiến trúc phân tầng: kiến trúc phân tầng giúp mạng CNN học được đặc trưng ở những cấp độ khác nhau, từ cấp độ bậc thấp đến bậc cao. Theo đó mức độ chi tiết của hình ảnh cũng tăng tiến dần từ các đặc trưng chung như các đường chéo, ngang, dọc, rìa, cạnh tới những đặc trưng chi tiết hơn giúp phân biệt vật thể như bánh xe, cánh cửa, mui xe (nếu vật thể là xe), tất cả các chi tiết đó được tổng hợp và ở layer tích chập cuối cùng ta thu được hình ảnh của một chiếc xe.
- Được huấn luyện trên những bộ dữ liệu lớn: khác với phương pháp học máy truyền thống, kiến trúc học sâu nhiều tầng đã cho thấy ưu thế vượt trội về độ chính xác và khả

năng biểu diễn. Điều này dễ hiểu bởi kích thước mạng nơ ron có thể lên tới hàng chục triệu tham số, lớn hơn rất nhiều so với số lượng tham số của các phương pháp học máy dẫn tới khả năng biểu diễn tốt hơn.

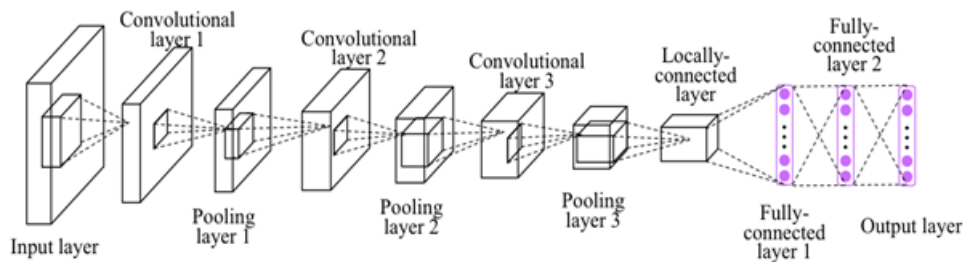
- Kích thước ở mỗi layer giảm dần: kích thước layer giảm dần giúp giảm thiểu số lượng tham số của mô hình đáng kể và giúp tạo ra những mạng có kích thước nhẹ hơn và tốc độ dự báo nhanh hơn, trong khi độ chính xác của mô hình giảm không đáng kể.
- Độ sâu tầng layers tăng dần: độ sâu của các layers tăng dần nhờ tăng số bộ lọc, thường là theo cấp số nhân. Độ sâu tăng sẽ giúp cho mạng CNN học được đa dạng các đặc trưng hơn. Ở những layer đầu tiên là những đặc trưng chung, chúng khá giống nhau về hình dạng, phương hướng, nên không cần quá nhiều bộ lọc để tạo ra chúng với số lượng lớn. Càng ở những layers sau đòi hỏi độ chi tiết cao hơn thì yêu cầu số lượng bộ lọc nhiều hơn để giúp phân biệt được nhiều chi tiết đặc trưng hơn.
- Sử dụng các fully connected layers để phân loại: tích chập từ mạng CNN sẽ tạo ra những đặc trưng 2 chiều. Để sử dụng những đặc trưng này vào quá trình phân loại của mạng CNN thì chúng ta phải chuyển chúng thành đặc trưng 1 chiều bằng phương pháp flatten và lan truyền thuận qua các fully connected layers. Đằng sau mỗi một layer là một hàm kích hoạt phi tuyến nhằm gia tăng khả năng biểu diễn giúp cho kết quả phân loại tốt hơn.

b. Kiến trúc chung của mạng nơ ron tích chập

Tích chập được ứng dụng phổ biến trong lĩnh vực thị giác máy tính. Thông qua các phép tích chập, các đặc trưng chính từ ảnh được trích xuất và truyền vào các tầng tích chập (layer convolution). Mỗi một tầng tích chập sẽ bao gồm nhiều đơn vị mà kết quả ở mỗi đơn vị là một phép biến đổi tích chập từ layer trước đó thông qua phép nhân tích chập với bộ lọc. Về cơ bản thiết kế của một mạng nơ ron tích chập 2 chiều có dạng như sau:

- Input layer: tầng đầu vào
- Convolutional layer: tầng tích chập
- Relu: tầng kích hoạt, thông qua hàm kích hoạt

- Pooling layer: tầng tổng hợp, thông thường là max pooling hoặc có thể là average pooling dùng để giảm chiều của ma trận đầu vào.
- Fully connected layer: tầng kết nối hoàn toàn, thông thường tầng này nằm ở sau cùng và kết nối với các đơn vị đại diện cho nhóm phân loại.



Hình 3.1: Cấu trúc mô hình mạng nơ ron tích chập

Như vậy ta có thể thấy một mạng nơ ron tích chập về cơ bản có 3 quá trình khác nhau:

- Quá trình tích chập (convolution): thông qua các tích chập giữa ma trận đầu vào với bộ lọc để tạo thành các đơn vị trong một tầng mới. Quá trình này có thể diễn ra liên tục ở phần đầu của mạng và thường sử dụng kèm với hàm kích hoạt ReLU. Mục tiêu của tầng này là trích xuất đặc trưng hai chiều.
- Quá trình tổng hợp (max pooling): các tầng càng về sau khi trích xuất đặc trưng sẽ cần số lượng tham số lớn do chiều sâu được quy định bởi số lượng các kênh, ở các tầng sau thường tăng tiến theo cấp số nhân. Điều đó làm tăng số lượng tham số và khối lượng tính toán trong mạng nơ ron. Do đó để giảm tải tính toán chúng ta sẽ cần giảm kích thước các chiều của khối ma trận đầu vào hoặc giảm số đơn vị của tầng. Vì mỗi một đơn vị sẽ là kết quả đại diện của việc áp dụng 1 bộ lọc để tìm ra một đặc trưng cụ thể nên việc giảm số đơn vị sẽ không khả thi. Giảm kích thước khối ma trận đầu vào thông qua việc tìm ra 1 giá trị đại diện cho mỗi một vùng không gian mà bộ lọc đi qua sẽ không làm thay đổi các đường nét chính của bức ảnh nhưng lại giảm được kích thước của ảnh. Do đó quá trình giảm chiều ma trận được áp dụng. Quá trình này gọi là tổng hợp nhằm mục đích giảm kích thước dài, rộng.
- Quá trình kết nối hoàn toàn (fully connected): sau khi đã giảm kích thước đến một mức độ hợp lý, ma trận cần được trải phẳng (flatten) thành một vector và sử dụng các kết nối

hoàn toàn giữa các tầng. Quá trình này sẽ diễn ra cuối mạng nơ ron tích chập và sử dụng hàm kích hoạt là Relu. Tầng kết nối hoàn toàn cuối cùng (fully connected layer) sẽ có số lượng đơn vị bằng với số classes (lớp) và áp dụng hàm kích hoạt là softmax nhằm mục đích tính phân phối xác suất.

c. Hàm kích hoạt trong mạng nơ ron tích chập

Các hàm kích hoạt quyết định một nơ ron có được kích hoạt hay không bằng cách tính toán có trọng số và cộng thêm hệ số điều chỉnh. Các hàm này là toán tử khả vi và hầu hết biến đổi các tín hiệu đầu vào thành các tín hiệu đầu ra theo một cách phi tuyến tính.

Hàm Relu là một phép biến đổi phi tuyến tính đơn giản, hàm này chỉ giữ lại các phần tử có giá trị dương và loại bỏ tất cả các giá trị âm (đặt kích hoạt tương ứng là 0).

$$ReLU(z) = \max(z, 0) \quad (3.1)$$

Hàm sigmoid biến đổi các giá trị đầu vào có một miền giá trị thuộc \mathbb{R} thành các giá trị đầu ra nằm trong khoảng từ $(0, 1)$. Hàm này còn được gọi là hàm ép, chúng ép một giá trị đầu vào bất kì nằm trong khoảng $(-\infty, \infty)$ thành một giá trị đầu ra trong khoảng $(0, 1)$.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

Hàm sigmoid thường được sử dụng rộng rãi khi ta muốn biểu diễn kết quả đầu ra như là xác suất của các bài toán phân loại nhị phân. Tuy nhiên trong các tầng ẩn hàm sigmoid hầu hết bị thay thế bằng hàm ReLu vì nó đơn giản hơn và giúp cho việc huấn luyện trở nên dễ dàng hơn.

Hàm tanh tương tự như hàm sigmoid, hàm tanh cũng ép các biến đầu vào và biến đổi chúng thành các phần tử nằm trong khoảng $(-1, 1)$.

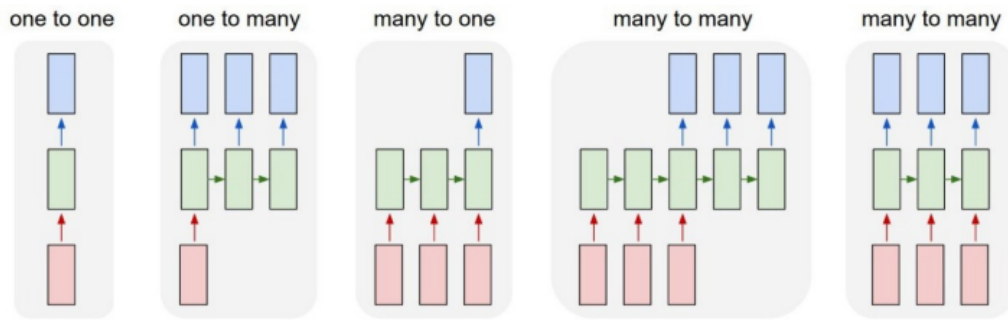
$$tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \quad (3.3)$$

3.2.2 Recurrent Neural Network

Như đã đề cập ở phía trên mạng RNN được dùng cho dữ liệu là dạng chuỗi. Không giống như mạng CNN bao gồm 3 phần chính là input layer hidden layer và output layer là các lớp đầu ra đầu vào độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, dự đoán và phân loại một chuỗi các hành động trong

video... vì những dự đoán tiếp theo thường phụ thuộc vào các vị và những phần tử đằng trước nó. Vì vậy RNN ra đời với ý tưởng chính là sử dụng bộ nhớ để lưu lại thông tin từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại.

a. Phân loại bài toán RNN



Hình 3.2: Các dạng bài toán RNN

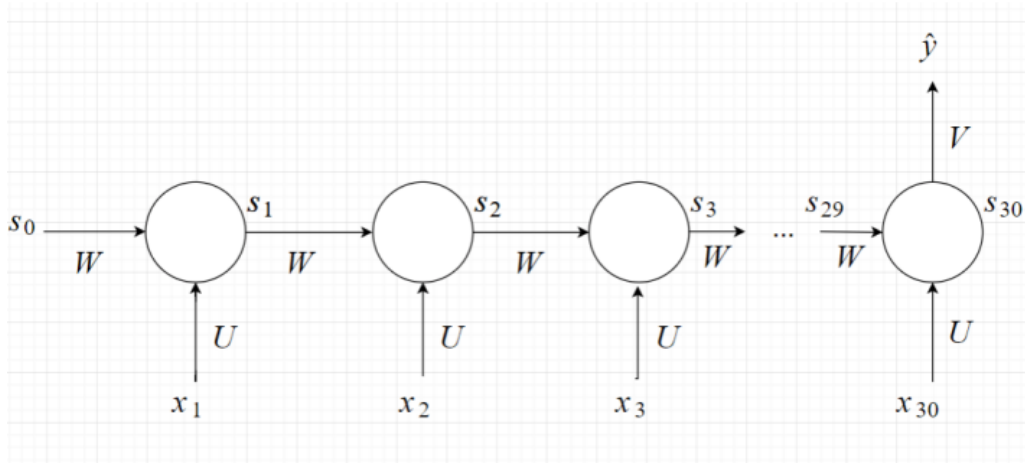
- One to one: mẫu bài toán cho một input và một output.
- One to many: bài toán có một input nhưng nhiều output.
- Many to one: bài toán có nhiều input nhưng chỉ có một output.
- Many to many: bài toán có nhiều input và nhiều output.

b. Mô hình RNN

Giả sử bài toán là nhận diện hành động trong video 30s thuộc dạng bài toán có nhiều output và 1 output. Input sẽ được tách thành 30 ảnh (mỗi giây một ảnh). Các ảnh sẽ được trích lọc đặc trưng thành vector có kích thước $n \times 1$. Vector tương ứng với ảnh ở giây thứ i là x_i . Output là vector có kích thước $d \times 1$ (d là số lượng hành động cần phân loại), kết hợp với các hàm kích hoạt (hình 2.3)

Mỗi hình tròn được gọi là 1 state (trạng thái), state t có input là x_t và s_{t-1} (output của state trước), output là $s_t = f(U * x_t + W * s_{t-1})$. Trong đó f là hàm kích hoạt thường là tanh hoặc relu.

Dễ dàng nhận thấy S_t chính là bộ nhớ, nhớ các đặc điểm của các input từ x_1 đến x_t vì mang cả thông tin từ state trước và input của state hiện tại.



Hình 3.3: Mô hình RNN

Đầu ra sẽ được đặt ở state cuối cùng, khi đó state cuối sẽ học được thông tin từ tất cả các input trước đó và đưa ra dự đoán. Tùy thuộc vào từng bài mà hàm kích hoạt sẽ được sử dụng khác nhau.

Một trong những đặc điểm đặc biệt của RNN đó là nó có khả năng kết nối các thông tin liên trước với thông tin hiện tại. Tuy nhiên có những thông tin cần phải đòi hỏi nhiều sự liên kết, một chuỗi của các thông tin trước đó. Trên thực tế RNN lại không thể giải quyết được vấn đề đó, hay nói nôm na rằng RNN có bộ nhớ rất kém, khi một quyết định của mô hình cần nhiều thông tin hơn để đưa ra dự đoán cuối cùng thì RNN không thể hoạt động tốt.

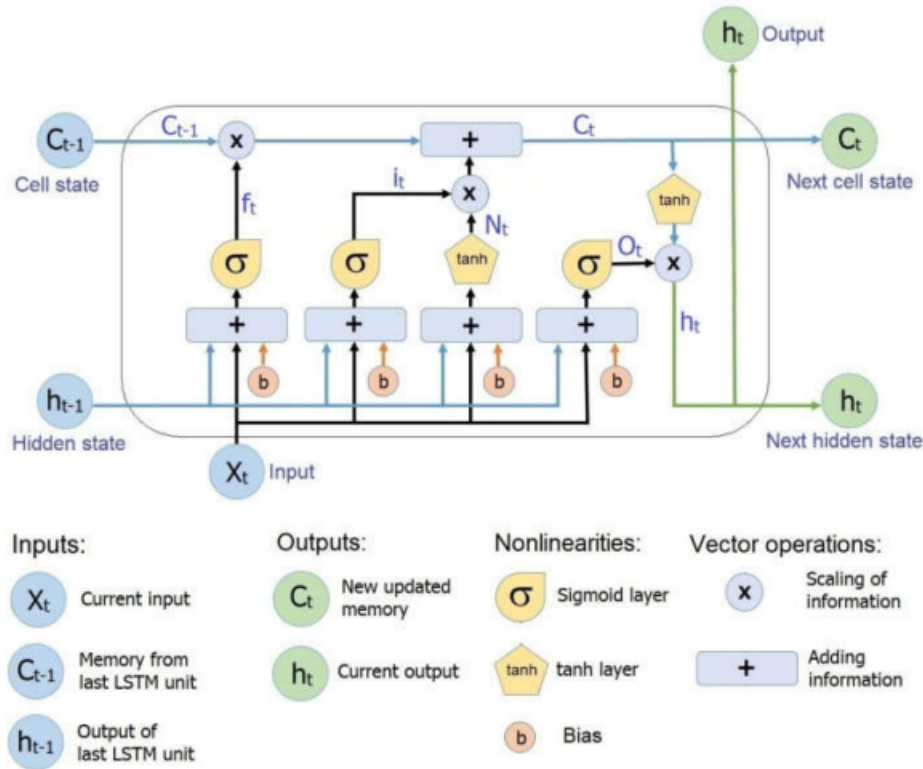
3.2.3 Long Short Term Memory

Long short term memory hay còn gọi là mạng trí nhớ ngắn hạn định hướng dài hạn viết tắt là LSTM. LSTM chính là phiên bản mới của RNN nhằm khắc phục những vấn đề mà RNN đang gặp phải. Tuy nhiên cấu trúc của RNN có phần phức tạp hơn mặc dù vẫn giữ được tư tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi.

Mạng LSTM có kiến trúc dạng chuỗi gồm các mô đun lặp đi lặp lại, nó không chỉ có một tầng nơ ron như RNN chuẩn mà có tới 4 tầng tương tác với nhau một cách đặc biệt (hình 2.4). Mỗi mô đun LSTM gồm có trạng thái tế bào (cell state) và các cổng (gate). Trạng thái tế bào chạy xuyên suốt qua tất cả các mô đun giúp thông tin được truyền đạt dễ dàng, còn cổng là nơi sàng lọc thông tin đi qua nó, có 3 cổng và 4 tầng trong một mô đun LSTM. Đầu tiên là tầng của cổng quên f_t (forget gate layer), nó sẽ quyết định thông tin nào cần bỏ đi từ trạng

thái tế bào. Đầu vào của tầng này là h_{t-1} (giá trị đầu ra tại thời điểm $t - 1$) và x_t (dữ liệu đầu vào hiện tại), đầu ra của nó là f_t , một số trong khoảng từ 0 đến 1 cho mỗi số trong trạng thái tế bào C_{t-1} .

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.4)$$



Hình 3.4: Cấu trúc mạng LSTM

Trong đó: σ là hàm sigmoid W_f và b_f lần lượt là trọng số và tham số của tầng cổng quên. Hai tầng tiếp theo sẽ quyết định thông tin lưu vào trạng thái tế bào và cập nhật giá trị cho trạng thái này, đó là tầng cổng vào i_t (input gate layer) và một tầng tanh N_t (tanh layer).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.5)$$

$$N_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3.6)$$

$$C_t = f_t * C_{t-1} + i_t * N_t \quad (3.7)$$

Trong đó C_{t-1} và C_t là trạng thái tế bào lần lượt ở thời điểm $t - 1$ và t ; W_i và b_i lần lượt là trọng số và tham số của tầng cổng vào. W_c và b_c là trọng số và tham số của trạng thái tế bào.

Cuối cùng là tầng cổng ra O_t (output gate), giá trị đầu ra (h_t) sẽ được quyết định bởi trạng thái tế bào muốn xuất ra.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.8)$$

$$h_t = o_t * \tanh(C_t) \quad (3.9)$$

Trong đó W_o và b_o lần lượt là trọng số và tham số của tầng cổng ra. Trong bài này, thư viện phần mềm mã nguồn mở Tensorflow của Google, các thư viện Numpy, Pandas, Keras sẽ được sử dụng để thiết kế mô hình mạng LSTM từ lý thuyết.

CHƯƠNG 4: TRIỂN KHAI MÔ HÌNH TRÊN NỀN TẢNG AMAZON WEB SERVICES

4.1 Ngôn ngữ và công cụ

4.1.1 Ngôn ngữ

Sử dụng Python là ngôn ngữ chính để crawl dữ liệu và triển khai huấn luyện mô hình học sâu.

4.1.2 Công cụ

Báo cáo này sử dụng hai công cụ chính: VSCode để crawl dữ liệu từ website và colab để triển khai xây dựng mô hình học sâu:

- VScode hay còn gọi là Visual Studio Code là trình chỉnh sửa mã nguồn nhẹ nhưng mạnh mẽ chạy trên máy tính và hỗ trợ cho ba hệ điều hành: Windows, macOS và Linux. Công cụ này tích hợp rất nhiều tiện ích mở rộng cho đa dạng các ngôn ngữ và thời gian chạy khác. Truy cập vào <https://code.visualstudio.com/> để tiến hành cài đặt và sử dụng.
- Colab (colaboratory) (hình 3.1) cho phép viết và thực thi Python trong trình duyệt với các lợi ích như: không yêu cầu cấu hình, sử dụng miễn phí GPU, chia sẻ dễ dàng. Truy cập vào: <https://colab.research.google.com/> chọn file -> new notebook -> connect sau đó ở phía góc phải màn hình chọn change runtime type -> T4 GPU -> save.

Bên cạnh đó thư viện selenium cũng được sử dụng để crawl dữ liệu từ website. Selenium là công cụ kiểm thử tự động mã nguồn mở, dành cho các ứng dụng website hoạt động trên nhiều trình duyệt và nền tảng khác nhau như Windows, Mac, Linux để giả lập lại các tương tác trên trình duyệt như một người dùng thực sự một cách tự động.

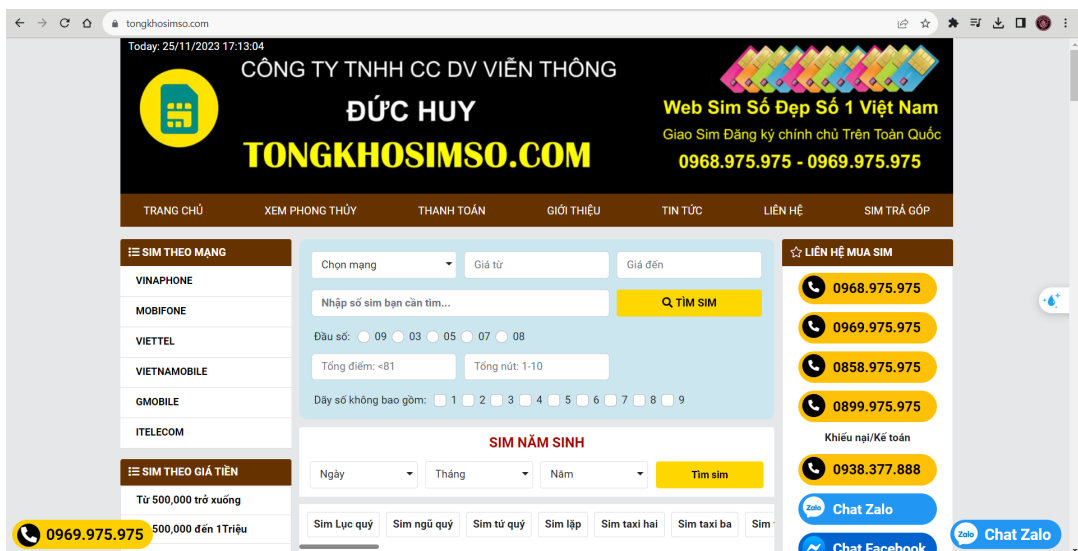


Hình 4.1: Giao diện google colab

4.2 Bộ dữ liệu

4.2.1 Crawl dữ liệu

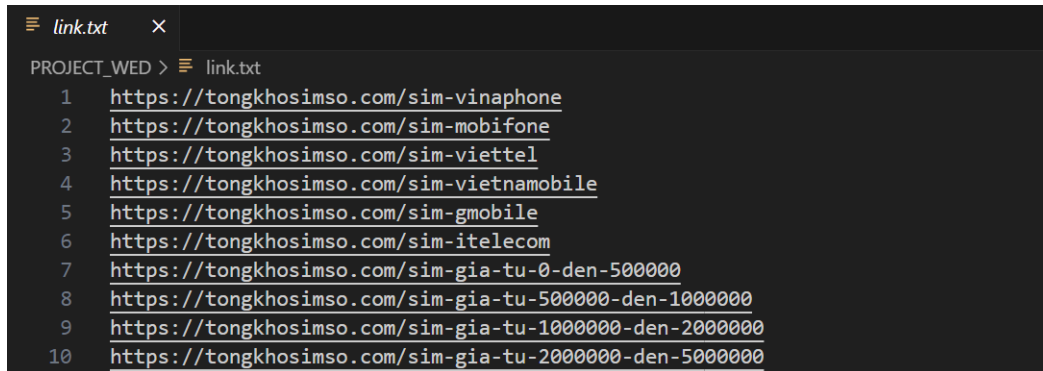
Dữ liệu sim điện thoại được lấy từ trang web <https://tongkhosimso.com/>. Đây là một trang mua bán sim lớn, chứa rất nhiều và đa dạng về các thể loại và giá trị sim.



Hình 4.2: Website mua bán sim điện thoại

Cấu trúc của trang web thường chứa rất nhiều những thẻ a (hay gòn gọi là những trang web con). Những trang web nhỏ này thường đại diện cho một nội dung, thành phần của trang

web cha. Để có được một tập dữ liệu đủ lớn và bao gồm tất cả các thể loại và giá trị sim cần lấy được tất cả các đường dẫn của các trang web con. Sử dụng VScode và selenium, kết quả thu được một file bao gồm các trang web con có dạng như hình dưới đây.



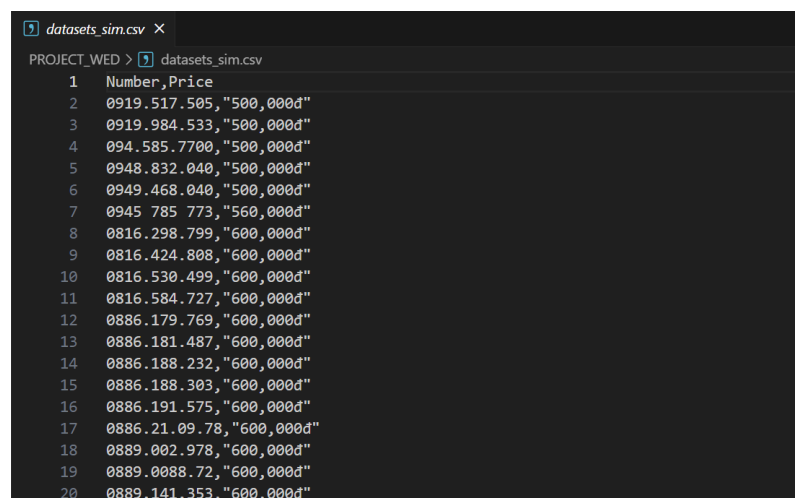
```
link.txt
PROJECT_WED > link.txt
1 https://tongkhosimso.com/sim-vinaphone
2 https://tongkhosimso.com/sim-mobifone
3 https://tongkhosimso.com/sim-viettel
4 https://tongkhosimso.com/sim-vietnamobile
5 https://tongkhosimso.com/sim-gmobile
6 https://tongkhosimso.com/sim-itelecom
7 https://tongkhosimso.com/sim-gia-tu-0-den-500000
8 https://tongkhosimso.com/sim-gia-tu-500000-den-1000000
9 https://tongkhosimso.com/sim-gia-tu-1000000-den-2000000
10 https://tongkhosimso.com/sim-gia-tu-2000000-den-5000000
```

Hình 4.3: File đường dẫn các trang web con

Sau khi đã lấy được đường dẫn của từng trang web con, ta xây dựng một chương trình giả lập thao tác giống như người dùng bằng cách lần lượt truy cập vào từng đường dẫn và lấy ra những thông tin về: số điện thoại, nhà mạng, giá trị.v.v.

4.2.2 Tập dữ liệu

Khi toàn bộ các dữ liệu được lấy hết từ tất cả các trang web con, thu được một tập dữ liệu thô là một dataframe (hình 3.4) với hai thuộc tính là số điện thoại và giá trị. Tập dữ liệu này sẽ được tiền xử lý và sử dụng để huấn luyện và xây dựng mô hình học sâu dự đoán giá sim.



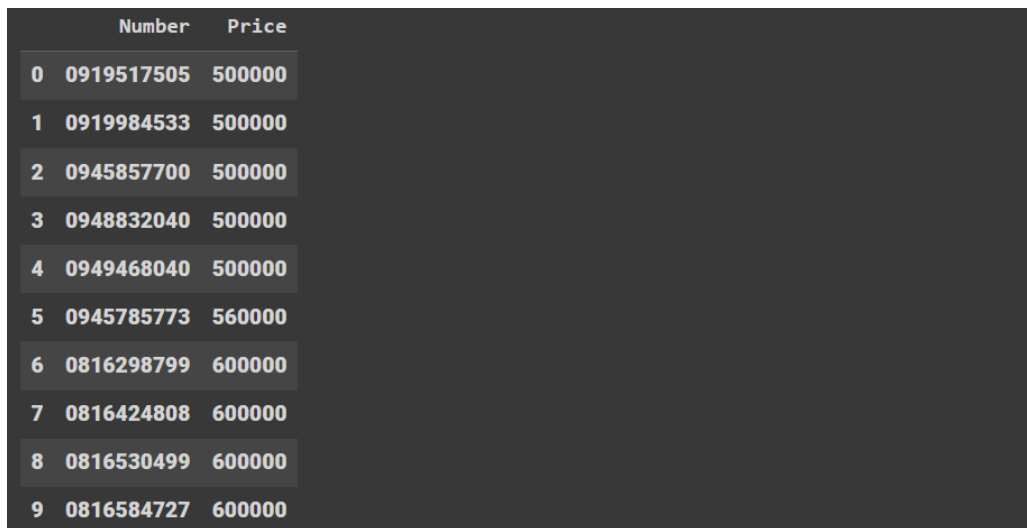
```
datasets_sim.csv
PROJECT_WED > datasets_sim.csv
1 Number,Price
2 0919.517.505,"500,000đ"
3 0919.984.533,"500,000đ"
4 094.585.7700,"500,000đ"
5 0948.832.040,"500,000đ"
6 0949.468.040,"500,000đ"
7 0945.785.773,"560,000đ"
8 0816.298.799,"600,000đ"
9 0816.424.808,"600,000đ"
10 0816.530.499,"600,000đ"
11 0816.584.727,"600,000đ"
12 0886.179.769,"600,000đ"
13 0886.181.487,"600,000đ"
14 0886.188.232,"600,000đ"
15 0886.188.303,"600,000đ"
16 0886.191.575,"600,000đ"
17 0886.21.09.78,"600,000đ"
18 0889.002.978,"600,000đ"
19 0889.0088.72,"600,000đ"
20 0889.141.353,"600,000đ"
```

Hình 4.4: Tập dữ liệu về sim điện thoại sau khi được crawl từ website

4.3 Tiền xử lý dữ liệu

4.3.1 Làm sạch dữ liệu

Dữ liệu thô sau khi được thu thập cần được tiền xử lý, làm sạch để phù hợp với yêu cầu đầu vào của mô hình. Loại bỏ các kí tự đặc biệt.v.v thu được tập dữ liệu hoàn chỉnh làm đầu vào của mô hình học sâu.



	Number	Price
0	0919517505	500000
1	0919984533	500000
2	0945857700	500000
3	0948832040	500000
4	0949468040	500000
5	0945785773	560000
6	0816298799	600000
7	0816424808	600000
8	0816530499	600000
9	0816584727	600000

Hình 4.5: Dữ liệu sau khi được làm sạch

4.3.2 Gán nhãn dữ liệu

Như đã đề cập ở chương 1, sim điện thoại rất đa dạng và có rất nhiều giá trị khác nhau điều đó có nghĩa rằng mô hình sẽ có rất nhiều đầu ra gây ra khó khăn và làm ảnh hưởng đến hiệu quả của mô hình. Vì vậy việc gán nhãn cho dữ liệu là rất quan trọng và là phương pháp được sử dụng trong báo cáo này nhằm hạn chế và khắc phục những điều đó. Quá trình gán nhãn được thực hiện như sau:

- Đầu tiên thực hiện thống kê mô tả với tập dữ liệu để xem phân phối giữa các giá trị. Quan sát và đánh giá xem các mẫu dữ liệu có phân phối đều và cân bằng hay không.
- Tiếp theo thực hiện thống kê số lượng cho từng giá trị và tiến hành phân thành các khoảng giá trị sau đó gán nhãn tương ứng cho từng khoảng giá trị đó. Ở trong báo cáo này các khoảng giá trị được chia và được gán nhãn tương ứng như hình 3.6.

```
def create_cat_sim(x):  
    if x["Price"] <= 500000:  
        return 0  
    elif x["Price"] <= 700000:  
        return 1  
    elif x["Price"] <= 900000:  
        return 2  
    elif x["Price"] <= 1000000:  
        return 3  
    elif x["Price"] <= 1200000:  
        return 4  
    elif x["Price"] <= 1500000:  
        return 5  
    elif x["Price"] <= 3000000:  
        return 6  
    elif x["Price"] <= 6000000:  
        return 7  
    else:  
        return 8
```

Hình 4.6: Phân khoảng giá trị tương ứng với từng nhãn

- Thực hiện gán nhãn cho toàn bộ tập dữ liệu, kết quả thu được như hình 3.7.

	Number	Price	Sim_cat
0	0919517505	500000	0
1	0919984533	500000	0
2	0945857700	500000	0
3	0948832040	500000	0
4	0949468040	500000	0
5	0945785773	560000	1
6	0816298799	600000	1
7	0816424808	600000	1
8	0816530499	600000	1
9	0816584727	600000	1

Hình 4.7: Tập dữ liệu sau khi được gán nhãn

4.3.3 Chuẩn hoá dữ liệu

Tập dữ liệu sau khi được gán nhãn cần được tiếp tục chuẩn hoá bằng các kĩ thuật LabelEncoder và OnehotEncoder. Sau đó chia dữ liệu thành hai phần: dữ liệu huấn luyện (train data) để huấn luyện mô hình và dữ liệu kiểm tra (test data) để đánh giá hiệu suất của mô hình

sau khi được xây dựng.

4.4 Huấn luyện mô hình

Từ tập dữ liệu thô, trải qua các thao tác làm sạch, gán nhãn và chuẩn hoá, phù hợp với yêu cầu đầu vào của mô hình. Tiến hành xây dựng mô hình mạng LSTM với các lớp phù hợp với đầu vào và đầu ra như mong muốn. Mô hình các lớp mạng LSTM được xây dựng trong bài được thể hiện như hình dưới đây.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 9, 512)	1052672
dropout (Dropout)	(None, 9, 512)	0
lstm_1 (LSTM)	(None, 9, 512)	2099200
dropout_1 (Dropout)	(None, 9, 512)	0
lstm_2 (LSTM)	(None, 9, 512)	2099200
dropout_2 (Dropout)	(None, 9, 512)	0
lstm_3 (LSTM)	(None, 128)	328192
dense (Dense)	(None, 512)	66048
dense_1 (Dense)	(None, 9)	4617

```

=====
Total params: 5649929 (21.55 MB)
Trainable params: 5649929 (21.55 MB)
Non-trainable params: 0 (0.00 Byte)

```

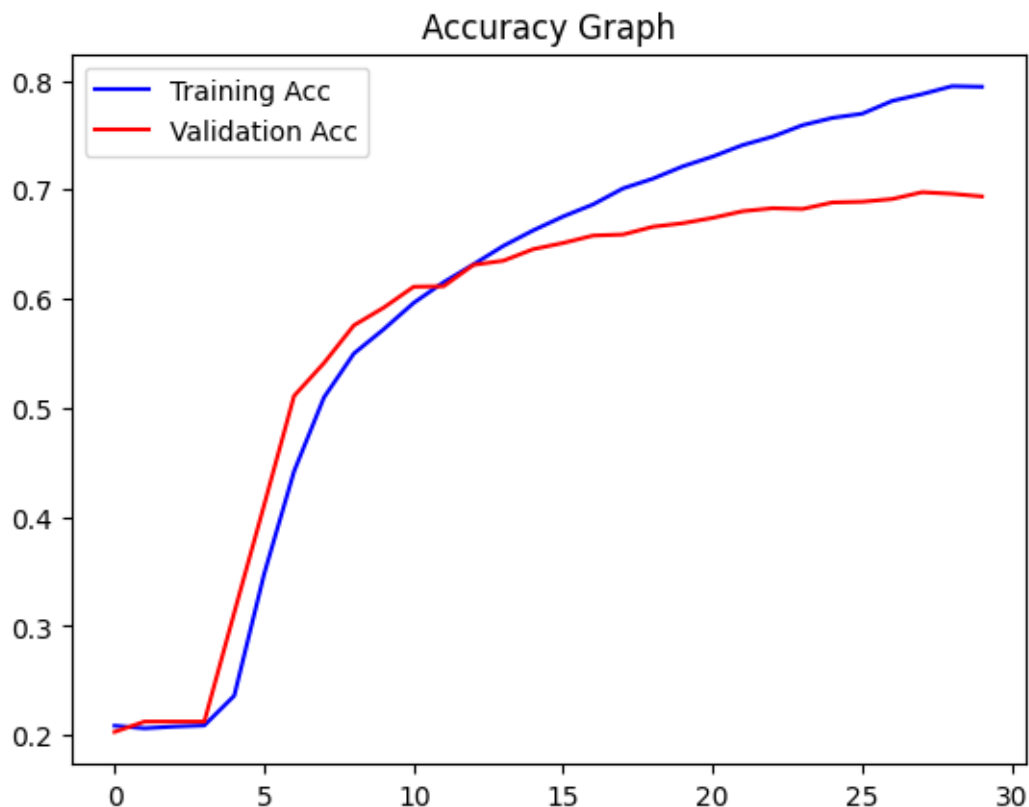
Hình 4.8: Mô hình mạng LSTM cho bài toán dự đoán giá sim

Mô hình gồm 4 lớp mạng LSTM kết hợp với 2 lớp Dense cho đầu ra là 9 (tương ứng với 9 nhãn đã được gán) xen kẽ là các lớp Dropout nhằm làm giảm hiện tượng quá khớp của mô hình.

4.5 Đánh giá và kiểm thử mô hình

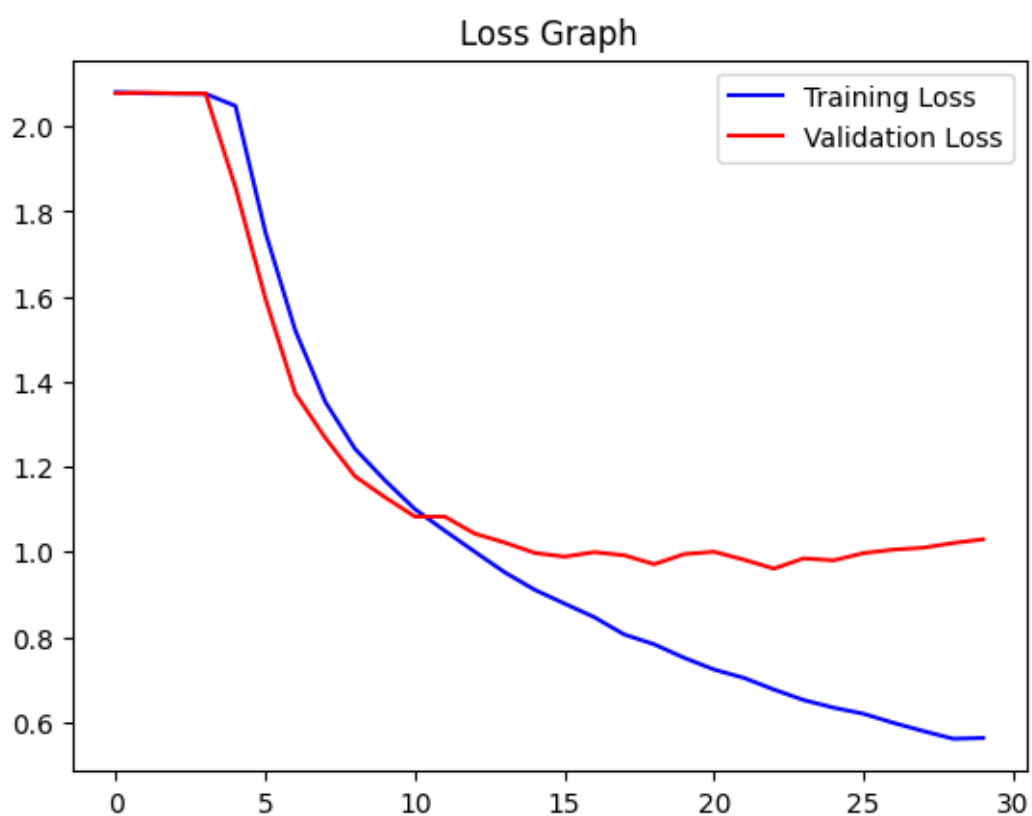
4.5.1 Đánh giá mô hình

Sau khi mô hình được huấn luyện trên một số epochs nhất định thu được độ chính xác (hình 3.9) và sai số (hình 3.10).



Hình 4.9: Đồ thị biểu diễn độ chính xác của mô hình

Quan sát ta thấy độ chính xác của mô hình ở cả hai tập dữ liệu huấn luyện (training accuracy) và kiểm định (validation accuracy) đều tăng đáng kể. Đối với tập dữ liệu huấn luyện độ chính xác tăng từ 0.2 đến 0.9, còn với tập dữ liệu kiểm định độ chính xác tăng từ 0.35 đến 0.7. Trong khi đó sai số của mô hình ở cả hai tập dữ liệu và kiểm định đều giảm, tuy nhiên từ epochs thứ 15 sai số ở tập kiểm định lại tăng lên. Điều đó cho thấy mô hình đã gặp vấn đề có thể do hiện tượng quá khớp gây nên, mô hình học quá chi tiết nhưng lại không khái quát hoá tốt hoặc do dữ liệu chưa đủ tốt, chưa đủ đại diện cho phân phối của tập dữ liệu.



Hình 4.10: Đồ thị biểu diễn sai số của mô hình

4.5.2 Thử nghiệm mô hình

Sử dụng tập dữ liệu kiểm tra đã được phân chia trước đó kiểm thử mô hình ngẫu nhiên với một số giá trị trong tập kiểm tra sau đó đối chiếu giá trị sim mà mô hình dự đoán với giá trị sim thực tế thu được như hình 3.11. Dễ dàng nhận thấy mô hình đã dự đoán đúng 8/11 mẫu. Mặc dù mô hình vẫn chưa đạt được hiệu suất một cách tối ưu nhưng cũng có kết quả khả thi và có thể tiếp tục được nghiên cứu và phát triển để đưa vào sử dụng trong thực tế.

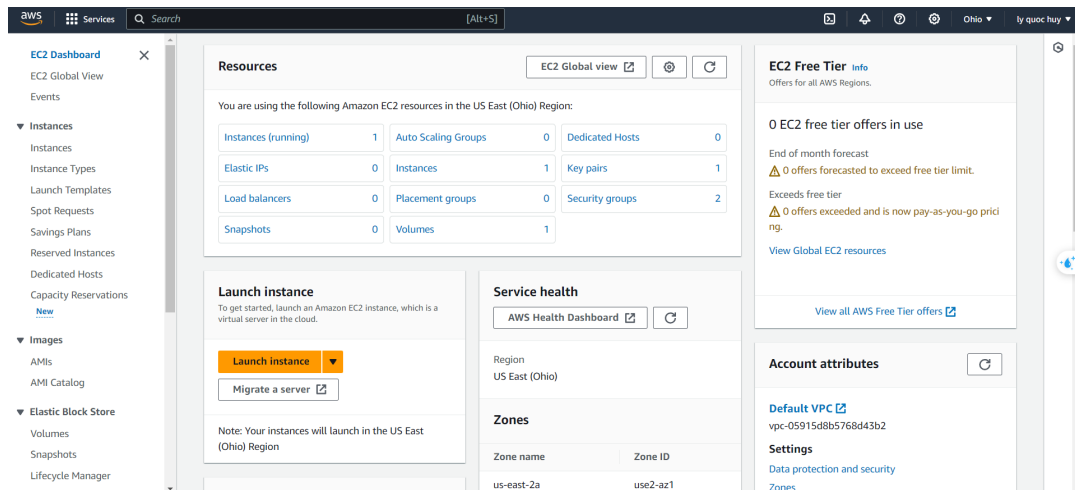
	Number	Result
0	772181368	True
1	942760686	True
2	773791959	True
3	329332014	False
4	797114777	True
5	828848686	False
6	938604444	True
7	909601284	True
8	931323243	False
9	823971555	True
10	818371994	True

Hình 4.11: Thử nghiệm mô hình trên tập kiểm tra

4.6 Triển khai mô hình trên nền tảng Amazon Web Services

4.6.1 Tạo tài khoản AWS

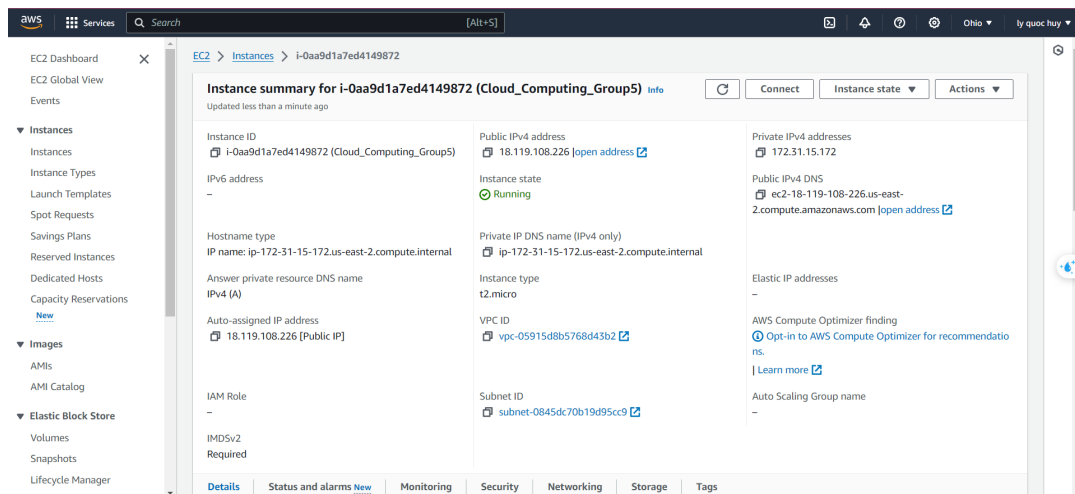
Tạo tài khoản trên trang chủ của AWS sau đó tiến hành đăng nhập. Giao diện người dùng sẽ hiện ra (hình 4.12).



Hình 4.12: Giao diện Amazon Web Services

4.6.2 Tạo máy chủ ảo trên AWS

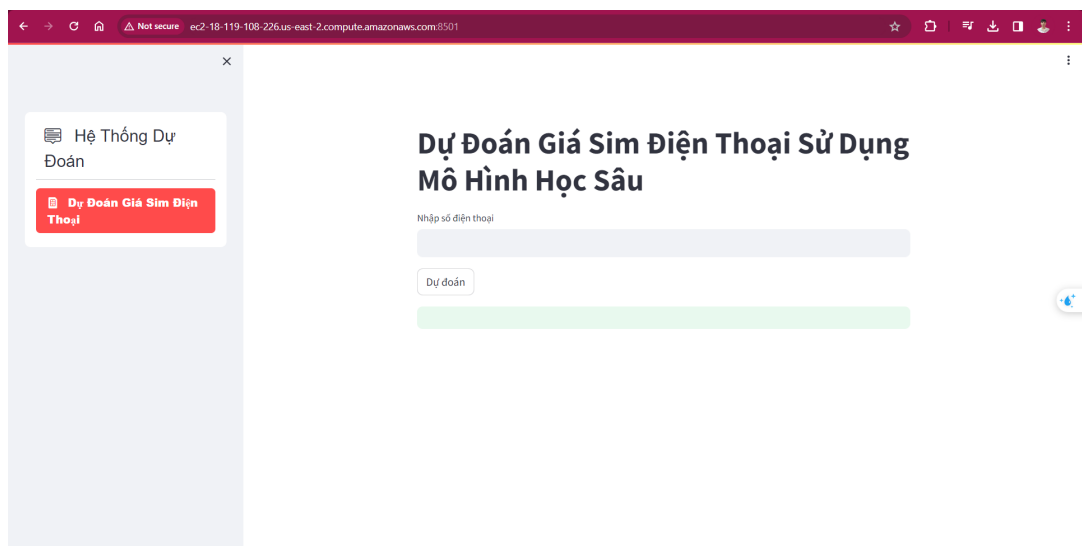
Tiến hành tạo máy chủ ảo EC2. Tại góc phải màn hình bấm chọn launch instances. Sau đó tiến hành cài đặt các thông số cần thiết. Cài đặt thành công hiển thị như hình 4.13.



Hình 4.13: Máy chủ ảo trên Amazon Web Services

4.6.3 Đưa web lên hệ thống máy chủ ảo AWS

Sau khi tạo được máy chủ ảo. Cài đặt hai ứng dụng Putty và WinSCP để kết nối và đưa các file cấu hình lên trên máy chủ ảo. Giao diện webserver được đưa thành công lên hệ thống điện toán đám mây AWS.



Hình 4.14: Mô hình dự đoán giá sim triển khai trên nền tảng AWS

KẾT LUẬN

Bài báo cáo đã nghiên cứu, xây dựng và triển khai một mô hình học sâu cho bài toán dự đoán giá sim điện thoại trên nền tảng điện toán đám mây Amazon Web Services. Trong suốt quá trình nghiên cứu nhóm chúng em đã:

- Nắm bắt được tổng quan về nền tảng dịch vụ điện toán đám mây nói chung và Amazon Web Services nói riêng.
- Xây dựng và huấn luyện được mô hình học sâu sử dụng mạng LSTM.
- Triển khai thành công mô hình học sâu trên nền tảng Amazon Web Services.

Mặc dù đã rất nỗ lực, xong với kiến thức và thời gian còn hạn chế nên không thể tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự cảm thông, nhận xét, đánh giá từ thầy giáo T.S Nguyễn Đức Bình và toàn thể các bạn lớp CNTT K19 CLC.

TÀI LIỆU THAM KHẢO

- [1] Vũ Hữu Tiệp. [Machine Learning cơ bản]. Nhà xuất bản Hà Nội. 2020.
- [2] Vũ Hữu Tiệp và cộng sự [Đắm mình vào học sâu]. Bản dịch của cuốn sách Drive to Deep Learning. 2018.
- [3] Nguyễn Thanh Tuấn. [Deep Learning cơ bản] 2020.
- [4] Mughal, Muhammd Jawad Hamid. "Data mining: Web data mining techniques, tools and algorithms: An overview." International Journal of Advanced Computer Science and Applications 9.6 (2018).
- [5] Medsker, Larry, and Lakhmi C. Jain, eds. Recurrent neural networks: design and applications. CRC press, 1999.
- [6] Li, Zewen, et al. "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE transactions on neural networks and learning systems (2021).
- [7] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).
- [8] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." 2017 international conference on engineering and technology (ICET). Ieee, 2017.
- [9] Yamashita, Rikiya, et al. "Convolutional neural networks: an overview and application in radiology." Insights into imaging 9 (2018): 611-629.
- [10] Memory, Long Short-Term. "Long short-term memory." Neural computation 9.8 (2010): 1735-1780.
- [11] Madria, Sanjay Kumar, et al. "Research issues in web data mining." DataWarehousing and Knowledge Discovery: First International Conference, DaWaK'99 Florence, Italy, August 30–September 1, 1999 Proceedings 1. Springer Berlin Heidelberg, 1999.

PHỤ LỤC CODE

```
1      # ket noi toi google drive
2      from google.colab import drive
3      drive.mount("/content/drive")
4
5      # import thu vien can thiet
6      import pandas as pd
7      import numpy as np
8
9      # ham gan nhan cho du lieu
10     def create_cat_sim(x):
11         if x["Price"] <= 500000:
12             return 0
13         elif x["Price"] <= 700000:
14             return 1
15         elif x["Price"] <= 900000:
16             return 2
17         elif x["Price"] <= 1000000:
18             return 3
19         elif x["Price"] <= 1200000:
20             return 4
21         elif x["Price"] <= 1500000:
22             return 5
23         elif x["Price"] <= 3000000:
24             return 6
25         elif x["Price"] <= 6000000:
26             return 7
27         else:
28             return 8
29
```

```
30     # gan nhan cho toan bo du lieu
31     sim_data["Sim_cat"] = sim_data.apply(create_cat_sim, axis=1)
32
33     # bien doi cot number ve dang ma tran
34     X = []
35     y = []
36     for index, row in sim_data.iterrows():
37         X.append([int(c) for c in str(row["Number"])])
38         y.append(row["Sim_cat"])
39     X = np.array(X)
40     y = np.array(y)
41
42     # chuan hoa du lieu
43     from sklearn.preprocessing import LabelEncoder, OneHotEncoder
44     lb_encoder = LabelEncoder()
45     y_lb_encoder = lb_encoder.fit_transform(y)
46     y_lb_encoder = y_lb_encoder.reshape(len(y_lb_encoder), 1)
47     lb_onehot = OneHotEncoder(sparse=False)
48     y_lb_onehot = lb_onehot.fit_transform(y_lb_encoder)
49
50     # chia du lieu train, test
51     from sklearn.model_selection import train_test_split
52     X_train, X_test, y_train, y_test = train_test_split(X,
53     y_lb_onehot, test_size=0.2, random_state=42)
54     X_train.shape, y_train.shape, X_test.shape, y_test.shape
55
56     # xay dung mo hinh mang LSTM
57     from tensorflow.keras.models import Sequential
58     from tensorflow.keras.layers import Dense, LSTM, Dropout
59     from tensorflow.keras.optimizers import Adam
60     model = Sequential()
61     model.add (LSTM(units=512, return_sequences=True,
62     input_shape = (X_train.shape[1], 1)))
63     model.add (Dropout(0.2))
64     model.add (LSTM(units=512, return_sequences=True))
65     model.add (Dropout(0.2))
66     model.add (LSTM(units=512, return_sequences=True))
```

```

67     model.add (Dropout(0.2))
68     model.add (LSTM(units=128, return_sequences=False))
69     model.add (Dense(units=512))
70     model.add (Dense(units=9, activation="softmax"))
71     optimizer = Adam()
72     model.compile(loss = "categorical_crossentropy",
73     metrics=["accuracy"], optimizer=optimizer)
74
75     # huan luyen mo hinh
76     from tensorflow.keras.callbacks import ModelCheckpoint
77     check_point = ModelCheckpoint(
78     filepath = "best.hdf5",
79     save_weights_only = True,
80     monitor = "val_accuracy",
81     save_best_only = True,
82     verbose = 1
83     )
84     history = model.fit(X_train, y_train, epochs=30, batch_size=64,
85     validation_data=(X_test, y_test), callbacks=[check_point])
86
87     # du doan mo hinh voi tap test
88     df = pd.DataFrame()
89     number = []
90     res = []
91     for i in range(19, 30):
92         # du doan
93         model_pred = model.predict(np.expand_dims(X_test[i], axis=0)
94     )
95         # lay phan tu max du doan
96         pred_y = np.argmax(model_pred)
97         # lay phan tu max gia tri thuc te
98         real_y = np.argmax(y_test[i])
99         # so sim
100        number.append(X_test[i])
101        # so sanh du doan va thuc te
102        a = pred_y == real_y
103        res.append(a)

```

```
103     df["Number"] = number
104     df["Result"] = res
105     df["Number"] = df["Number"].astype(str).apply(lambda x: ''.join(
        filter(str.isdigit, x)))
```

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thái nguyên, ngày ... tháng ... năm 2024

Giảng viên