

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN**

**HỌC MÁY**

*Người hướng dẫn:* **TS. TRẦN LƯƠNG QUỐC ĐẠI**

*Người thực hiện:* **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

**NGUYỄN QUỐC MẠNH – 52200085**

**PHẠM THỊ THANH BÌNH – 52200104**

Lớp : **22050201**

Khoá : **26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN**

**HỌC MÁY**

Người hướng dẫn: **TS. TRẦN LƯƠNG QUỐC ĐẠI**  
Người thực hiện: **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**  
**NGUYỄN QUỐC MẠNH – 52200085**  
**PHẠM THỊ THANH BÌNH – 52200104**  
Lớp : **22050201**  
Khoá : **26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

## LỜI CẢM ƠN

Để hoàn thành Bài báo cáo cuối kỳ I năm học 2024 - 2025 môn Học máy lần này.

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành của mình đến Ban giám hiệu trường Đại học Tôn Đức Thắng, quý thầy cô giáo giảng viên trong khoa Công nghệ thông tin vì đã tạo điều kiện tốt nhất để có những kiến thức mà hoàn thành bài báo cáo cuối kỳ I lần này. Đây thực sự là một cơ hội tuyệt vời giúp cho nghề nghiệp của em trong tương lai rộng mở hơn khi được tiếp xúc với sự hiện đại, nhiều kiến thức.

Với lòng biết ơn sâu sắc và vô cùng đặc biệt của mình, nhóm chúng em xin gửi lời cảm ơn của mình đến thầy Trần Lương Quốc Đại – Giảng viên lý thuyết Học máy – người đã luôn đồng hành, dẫn dắt và giúp đỡ chúng em trong việc hoàn thành bài báo cáo cuối kỳ. Từ những kiến thức thầy đã giảng dạy trên những giờ học để em có thể áp dụng những kiến thức đó vào bài tập lần này. Một lần nữa nhóm chúng em xin chân thành cảm ơn thầy vì sự hỗ trợ của thầy ạ.

Lời cuối cùng, nhóm chúng em xin gửi lời cảm ơn chân thành và gửi ngàn lời chúc tốt đẹp đến với quý thầy cô khi đã tạo cơ hội cho chúng em nâng cấp kiến thức trong môn học này.

Chúng em xin chân thành cảm ơn ạ!

## **ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Trần Lương Quốc Đại;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 19 tháng 12 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Huỳnh Hoàng Tiến Đạt*

*Nguyễn Quốc Mạnh*

*Phạm Thị Thanh Bình*

## **PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

### **Phần xác nhận của GV hướng dẫn**

---

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(kí và ghi họ tên)

### **Phần đánh giá của GV chấm bài**

---

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(kí và ghi họ tên)

## TÓM TẮT

Trong thời đại công nghệ 4.0, học máy (Machine Learning) đang trở thành một trong những công cụ quan trọng nhất để phân tích dữ liệu và tạo ra các giải pháp dự đoán thông minh. Báo cáo này sẽ giải quyết các vấn đề liên quan đến Học máy đề cập trong đề bài được giao.

Bài báo cáo Cuối kỳ này bao gồm 03 chương:

**Chương 1: Giải quyết vấn đề thứ nhất**

**Chương 2: Giải quyết vấn đề thứ hai**

**Chương 3: Giải quyết vấn đề thứ ba**

## MỤC LỤC

LỜI CẢM ƠN .....	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN .....	iii
TÓM TẮT .....	iv
MỤC LỤC .....	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ .....	4
CHƯƠNG 1 – VẤN ĐỀ THỨ NHẤT .....	6
1.1 Các phương pháp tối ưu hóa .....	6
1.1.1 Gradient Descent – GD .....	6
1.1.1.1 Batch Gradient Descent – BGD .....	7
1.1.1.2 Stochastic Gradient Descent – SGD .....	8
1.1.1.3 Mini-Batch Gradient Descent .....	9
1.1.2 Momentum .....	9
1.1.3 Adaptive Gradient Algorithm – Adagrad .....	10
1.1.4 Root Mean Square Propagation – RMSProp .....	10
1.1.5 Adaptive Moment Estimation – Adam .....	11
1.2 Chương trình thực thi .....	12
1.2.1 Thực thi các phương pháp .....	12
1.2.2 So sánh các phương pháp .....	13
CHƯƠNG 2 – VẤN ĐỀ THỨ HAI .....	15
2.1 Thử nghiệm các phương pháp .....	15
2.1.1 Tiền xử lý dữ liệu .....	16
2.1.2 Feedforward Neural Network – FFNN .....	23
2.1.3 Recurrent Neural Network – RNN .....	25
2.1.4 Linear Regression .....	26
2.1.5 Support Vector Machine – SVM .....	26
2.1.6 Decision Tree .....	27

2.1.7 Random Forest .....	27
2.2 Ngăn chặn overfitting và vẽ đồ thị đào tạo .....	28
2.2.1 Ngăn chặn overfitting .....	28
2.2.2 Đồ thị đào tạo .....	29
2.3 Đánh giá các mô hình, so sánh và vẽ đồ thị kết quả .....	30
CHƯƠNG 3 – VẤN ĐỀ THỨ BA .....	33
3.1 Deep Learning .....	33
3.1.1 Giới thiệu chung .....	33
3.1.2 Thuận lợi và Thách thức .....	33
3.2 Convolutional Neural Network .....	34
3.2.1 Cách hoạt động của mạng neural tích chập .....	35
3.2.1.1 Convolutional layer .....	35
3.2.1.2 Pooling layer .....	36
3.2.1.3 Fully-connected (FC) layer .....	37
3.3 Chương trình thực thi .....	38
3.3.1 Tập dữ liệu và xử lý dữ liệu .....	38
3.3.2 Huấn luyện mô hình và kiểm tra .....	39
3.3.2 Nhận xét .....	42



## **DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

### **CÁC KÝ HIỆU**

### **CÁC CHỮ VIẾT TẮT**

CNN	Convolutional Neural Network
FFNN	Feedforward Neural Network

## **DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

### **DANH MỤC HÌNH**

Hình 1. 1 Tập dữ liệu Wine Quality.....	12
Hình 1. 2 Đặc trưng trong tập dữ liệu Wine Quality .....	13
Hình 1. 3 Biểu đồ Loss theo epochs của các phương pháp tối ưu hóa .....	13
Hình 2. 1 Các đặc trưng của tập dữ liệu trong bài toán .....	15
Hình 2. 2 Sắp xếp các mô hình theo các chỉ số.....	19
Hình 2. 4 Sơ đồ cho mô hình tốt nhất .....	23
Hình 2. 5 Mạng neural Feedforward .....	24
Hình 2. 6 Kết quả chạy FFNN .....	24
Hình 2. 7 Kết quả chạy RNN .....	26
Hình 2. 8 Kết quả chạy Linear Regression .....	26
Hình 2. 9 Kết quả chạy SVM.....	27
Hình 2. 10 Kết quả chạy Decision Tree .....	27
Hình 2. 11 Kết quả chạy Random Forest .....	28
Hình 2. 12 Biểu đồ so sánh hiệu suất các mô hình .....	32
Hình 3. 1 CNN .....	34
Hình 3. 2 Convolution.....	36
Hình 3. 3 Pooling .....	36
Hình 3. 4 Max pooling .....	37
Hình 3. 5 Average pooling .....	37
Hình 3. 6 Các labels trong tập dữ liệu Cifar – 10 .....	38
Hình 3. 7 Cấu trúc tập dữ liệu.....	39
Hình 3. 8 Huấn luyện CNN.....	40

Hình 3. 9 Kết quả CNN.....	41
Hình 3. 10 Accuracy khi huấn luyện bằng CNN .....	42
Hình 3. 11 Loss khi huấn luyện bằng CNN .....	42

## **DANH MỤC BẢNG**

## CHƯƠNG 1 – VẤN ĐỀ THỨ NHẤT

Thuật toán tối ưu hóa là các quy trình tính toán được thiết kế để điều chỉnh các tham số của mô hình, ví dụ: trọng số trong mạng neural, nhằm cải thiện độ chính xác dự đoán của mô hình. Mục tiêu của các thuật toán này là giảm thiểu (hoặc tối đa hóa) giá trị của một hàm mất mát, giúp mô hình đạt được hiệu suất tốt nhất.

Các đặc điểm chính của thuật toán tối ưu hóa:

1. Hàm mục tiêu (Objective Function): Hàm toán học ta muốn tối ưu hóa. Trong học máy, hàm này thường là hàm mất mát (loss function) đo lường sự sai lệch giữa dự đoán của mô hình và giá trị thực tế.
2. Gradient: Gradient: Một vector chứa các đạo hàm bậc nhất của hàm mục tiêu với từng tham số của mô hình. Gradient chỉ ra hướng và tốc độ cần điều chỉnh các tham số để giảm thiểu hàm mục tiêu.
3. Tốc độ học (Learning Rate): Tham số quyết định kích thước của bước đi mỗi lần cập nhật các tham số dựa trên gradient. Tốc độ học quá cao có thể khiến mô hình không hội tụ, trong khi tốc độ học quá thấp có thể làm quá trình huấn luyện kéo dài.

### 1.1 Các phương pháp tối ưu hóa

#### 1.1.1 Gradient Descent – GD

Gradient Descent (GD) là phương pháp cơ bản nhất, sử dụng gradient của hàm mất mát để cập nhật các tham số. Nó được sử dụng để tìm giá trị cực tiểu (hoặc cực đại) của một hàm. Đối với hàm nhiều biến, Gradient Descent cập nhật giá trị của các biến trong mỗi bước để giảm giá trị của hàm mục tiêu.

Gradient Descent cho hàm một biến: Nếu đạo hàm của hàm số tại  $x_t$ :  $f'(x_t) > 0$  thì  $x_t$  nằm về bên phải so với  $x^*$  (và ngược lại). Để điểm tiếp theo  $x_{t+1}$  gần với  $x^*$  hơn,

ta cần di chuyển  $x_t$  về phía bên trái (phía âm), nghĩa là cần di chuyển ngược dấu với đạo hàm:

$$x_{t+1} = x_t + \Delta \quad (1.1)$$

Trong công thức 1.1:

- $x_t$ : điểm tìm được sau vòng lặp thứ  $t$ .
- $x^*$ : điểm local minimum của hàm số, tại đó đạo hàm  $f'(x^*) = 0$ .
- $\Delta$ : đại lượng ngược dấu với đạo hàm  $f'(x_t)$ .

$x_t$  càng xa  $x^*$  về phía bên phải thì  $f'(x_t)$  càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển  $\Delta$  một cách trực quan nhất, là tỉ lệ thuận với  $-f'(x_t)$

Hai nhận xét phía trên cho chúng ta một cách cập nhật đơn giản:

$$x_{t+1} = x_t - \eta f'(x_t) \quad (1.2)$$

Trong công thức 1.2:

- $\eta$ : learning rate (tốc độ học).

Dấu trừ thể hiện việc chúng ta phải *đi ngược* với đạo hàm, đây chính là lý do phương pháp này được gọi là Gradient Descent.

Gradient Descent cho hàm nhiều biến: Giả sử ta cần tìm global minimum cho hàm  $f(\theta)$ . Tương tự như hàm một biến, thuật toán cũng bắt đầu bằng một điểm dự đoán  $\theta_0$ , sau đó, ở vòng lặp thứ  $t$ , ta có quy tắc cập nhật là:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} f(\theta_t) \quad (1.3)$$

Trong công thức 1.3:

- $\theta$ : một vector, ký hiệu tập hợp các tham số của mô hình cần tối ưu.
- $\nabla_{\theta} f(\theta)$ : đạo hàm của hàm số tại điểm  $\theta$  bất kỳ.

#### 1.1.1.1 Batch Gradient Descent – BGD

Batch Gradient Descent sử dụng toàn bộ tập dữ liệu để tính toán và cập nhật gradient. Phương pháp này sẽ tính gradient của hàm mất mát tại  $\theta$  trên toàn bộ tập dữ liệu. Tất cả các điểm dữ liệu đều được sử dụng để tính gradient trước khi cập nhật bộ trọng số  $\theta$ .

BGD có những ưu điểm rất đáng chú ý như độ chính xác cao, vì phương pháp này sử dụng toàn bộ dữ liệu để tính toán gradient nên sẽ giúp cập nhật trọng số chính xác hơn. Đồng thời cũng khá ổn định khi ít bị ảnh hưởng bởi nhiễu vì không dựa vào các mẫu dữ liệu ngẫu nhiên. Tuy nhiên, nó lại gây tốn kém tài nguyên vì đối với các tập dữ liệu lớn, cần rất nhiều bộ nhớ và thời gian tính toán.

#### 1.1.1.2 Stochastic Gradient Descent – SGD

Stochastic Gradient Descent sử dụng từng mẫu dữ liệu ngẫu nhiên để tính toán và cập nhật gradient. Phương pháp này cập nhật trọng số sau mỗi bước dựa trên gradient của một mẫu dữ liệu ngẫu nhiên duy nhất, điều này làm giảm đáng kể thời gian tính toán mỗi lần cập nhật. Công thức cập nhật trọng số như sau:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta_t; x^{(i)}; y^{(i)}) \quad (1.4)$$

Trong công thức 1.4:

- $x^{(i)}; y^{(i)}$ : mẫu dữ liệu ngẫu nhiên.
- $\nabla_{\theta} J(\theta_t; x^{(i)}; y^{(i)})$ : gradient tính toán từ mẫu dữ liệu này.

SGD có những ưu điểm rất đáng chú ý như nhanh, vì phương pháp này cập nhật trọng số ngay sau mỗi mẫu. Đồng thời nó cũng có tính ngẫu nhiên, giúp mô hình dễ dàng vượt qua các điểm tối ưu cục bộ trong quá trình huấn luyện. Tuy nhiên, nó lại bị nhiễu cao vì mỗi lần cập nhật chỉ từ một mẫu ngẫu nhiên, dẫn đến sự biến động lớn trong giá trị hàm mất mát. Cũng chính vì có sự biến động lớn, SGD có thể gặp khó khăn trong việc hội tụ về điểm tối ưu toàn cục nếu tốc độ học không được điều chỉnh đúng.

### 1.1.1.3 Mini-Batch Gradient Descent

Mini-Batch Gradient Descent sử dụng một nhóm nhỏ các mẫu dữ liệu để tính toán và cập nhật gradient. Phương pháp này sẽ cập nhật trọng số dựa trên gradient tính toán từ một nhóm nhỏ (mini-batch) các mẫu dữ liệu, thay vì dùng toàn bộ tập dữ liệu hoặc chỉ một mẫu. Công thức cập nhật trọng số như sau:

$$\theta = \theta - \eta \frac{1}{m} \sum_{i=1}^m \nabla J(\theta; x^{(i)}; y^{(i)}) \quad (1.5)$$

Trong công thức 1.5:

- $x^{(i)}; y^{(i)}$ : các mẫu dữ liệu trong mini-batch.
- $m$ : kích thước của mini-batch.

Mini-Batch GD có những ưu điểm rất đáng chú ý như ổn định hơn SGD vì sử dụng trung bình gradient của một nhóm nhỏ mẫu dữ liệu, từ đó giảm nhiễu, dẫn đến giảm sự biến động so với SGD. Đồng thời lại nhanh hơn BGD vì không cần tính toán trên toàn bộ tập dữ liệu, nhờ đó tiết kiệm tài nguyên. Tuy nhiên, ta cần điều chỉnh kích thước nhóm nhỏ vì kích thước mini-batch ảnh hưởng đến hiệu quả thuật toán.

### 1.1.2 Momentum

Momentum sử dụng động lực để tăng tốc độ hội tụ của gradient descent, tránh các điểm dừng cục bộ. Phương pháp này cải tiến thuật toán Gradient Descent bằng cách cập nhật trọng số dựa trên cả gradient hiện tại và gradient trước đó, giúp tăng tốc độ hội tụ và vượt qua các điểm dừng cục bộ. Công thức cập nhật trọng số với Momentum:

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla J(\theta) \quad (1.6)$$

$$\theta = \theta - \eta v_t$$

Trong công thức 1.6:

- $v_t$ : vận tốc tại thời điểm  $t$ .
- $\beta$ : hệ số động lực (thường lấy giá trị gần 1).

- $\nabla J(\theta)$ : gradient của hàm mất mát.

Momentum có những ưu điểm rất đáng chú ý như hội tụ nhanh hơn nhờ sử dụng thông tin của gradient trước đó và vận tốc giúp mô hình thoát khỏi những điểm tối ưu cục bộ. Tuy nhiên, phương pháp này yêu cầu điều chỉnh hệ số động lực  $\beta$  nó không thể tự động điều chỉnh tốc độ học.

### 1.1.3 Adaptive Gradient Algorithm – Adagrad

Điều chỉnh tốc độ học cho mỗi tham số dựa trên lịch sử gradient của tham số đó. Công thức cập nhật trọng số với Adagrad:

$$\begin{aligned} G_t &= G_{t-1} + \nabla J(\theta)^2 \\ \theta &= \theta - \frac{\eta}{\sqrt{G_t + \xi}} \nabla J(\theta) \end{aligned} \quad (1.7)$$

Trong công thức 1.7:

- $G_t$ : tổng bình phương gradient tính đến thời điểm  $t$ .
- $\xi$ : hằng số nhỏ để tránh biểu thức chia cho 0.

Adagrad có những ưu điểm rất đáng chú ý như tự động điều chỉnh tốc độ học, tốc độ học sẽ thay đổi theo từng tham số, nhờ đó ta không cần điều chỉnh tốc độ học thủ công. Đồng thời, phương pháp này cũng thích hợp cho các bài toán có gradient hiếm gặp. Tuy nhiên, tốc độ học sẽ giảm nhanh khi gradient tích lũy và nó không hiệu quả cho các mạng nơ-ron sâu.

### 1.1.4 Root Mean Square Propagation – RMSProp

RMSProp là một biến thể của Adagrad, sử dụng trung bình động của bình phương gradient để điều chỉnh tốc độ học, giúp duy trì tốc độ học ổn định hơn. Công thức cập nhật trọng số với RMSProp:



$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) \nabla J(\theta)^2 \quad (1.8)$$

$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \xi}} \nabla J(\theta)$$

Trong công thức 1.8:

- $E[g^2]_t$ : giá trị trung bình động của bình phương gradient.
- $\gamma$ : hệ số suy giảm.

RMSProp có thể duy trì tốc độ học ổn định, tránh sự giảm tốc độ học quá nhanh như trong Adagrad, vậy nên sẽ thích hợp cho các bài toán dài hạn. Tuy nhiên, phương pháp này yêu cầu chọn hệ số suy giảm  $\gamma$  và không thể hoàn toàn giải quyết vấn đề gradient hiếm gặp.

### 1.1.5 Adaptive Moment Estimation – Adam

Adam (Adaptive Moment Estimation) kết hợp các ưu điểm của Momentum và RMSProp bằng cách sử dụng trung bình động của gradient và bình phương gradient để điều chỉnh tốc độ học. Công thức cập nhật trọng số với Adam:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla J(\theta) \quad (1.9)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla J(\theta)^2 \quad (1.10)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla J(\theta) \quad (1.9)$$

Trong công thức 1.9 và 1.10:


- $m_t$ : trung bình động của gradient.
- $v_t$ : trung bình động của bình phương gradient.
- $\beta_1, \beta_2$ : các hệ số động lực.

Adam có thể tự động điều chỉnh tốc độ học bằng phương pháp tối ưu, vậy nên sẽ thích hợp cho các bài toán lớn, đặc biệt là mạng nơ-ron sâu. Tuy nhiên, phương pháp này yêu cầu điều chỉnh nhiều tham số và tiêu tốn nhiều tài nguyên tính toán.

## 1.2 Chương trình thực thi

### 1.2.1 Thực thi các phương pháp

Tập dữ liệu được sử dụng là tập dữ liệu *Wine Quality* từ UCI Machine Learning Repository.



## Wine Quality

Donated on 10/6/2009

Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], <http://www3.dsi.uminho.pt/pcortez/wine/>).

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Business	Classification, Regression

Feature Type	# Instances	# Features
Real	4898	11

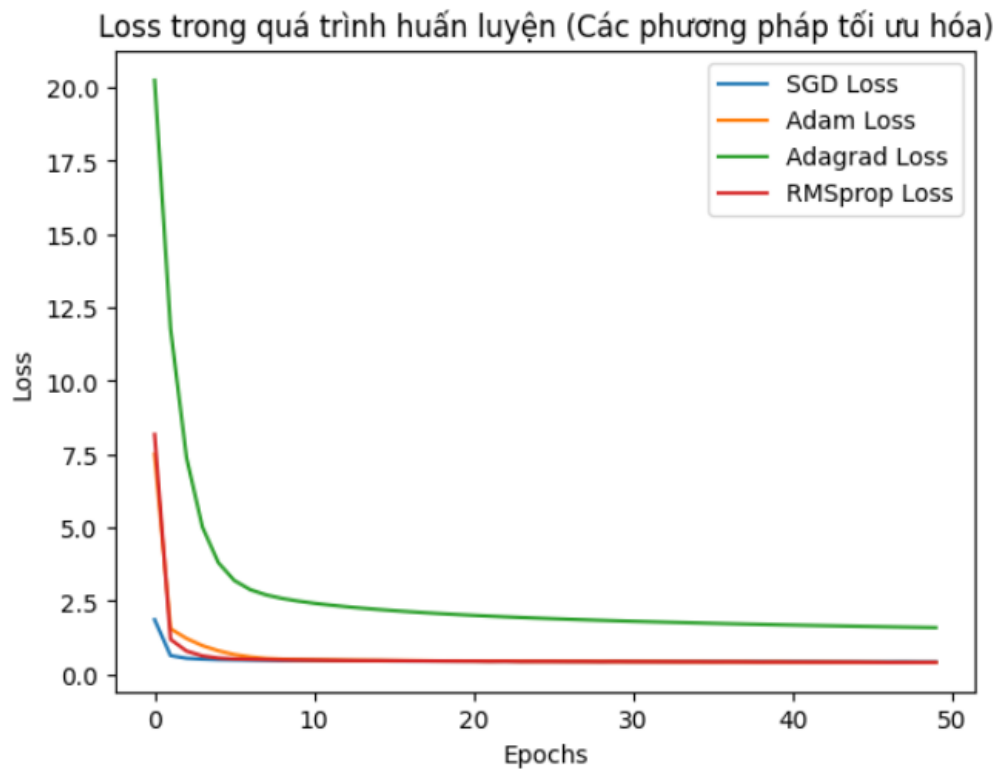
Hình 1. 1 Tập dữ liệu Wine Quality

Variable Name	Role	Type	Description	Units	Missing Values
fixed_acidity	Feature	Continuous			no
volatile_acidity	Feature	Continuous			no
citric_acid	Feature	Continuous			no
residual_sugar	Feature	Continuous			no
chlorides	Feature	Continuous			no
free_sulfur_dioxide	Feature	Continuous			no
total_sulfur_dioxide	Feature	Continuous			no
density	Feature	Continuous			no
pH	Feature	Continuous			no
sulphates	Feature	Continuous			no

alcohol	Feature	Continuous		no
quality	Target	Integer	score between 0 and 10	no
color	Other	Categorical	red or white	no

Hình 1. 2 Đặc trưng trong tập dữ liệu Wine Quality

### 1.2.2 So sánh các phương pháp



So sánh MSE giữa các phương pháp tối ưu hóa:

SGD: MSE = 0.5017631305519867

Adam: MSE = 0.4684704579583964

Adagrad: MSE = 1.9226341092271906

RMSprop: MSE = 0.4774079689596685

Hình 1. 3 Biểu đồ Loss theo epochs của các phương pháp tối ưu hóa

Qua đồ thị, có thể thấy rằng cả bốn phương pháp đều có xu hướng giảm loss theo thời gian, nhưng tốc độ và mức độ hội tụ có sự khác biệt rõ rệt.

Phương pháp **RMSprop** thể hiện hiệu suất tốt nhất, với loss giảm nhanh chóng ngay từ những epoch đầu tiên và đạt mức ổn định ở một giá trị rất thấp. Đây là phương pháp có MSE thấp nhất ( $MSE = 0.470$ ), cho thấy nó tối ưu hơn trong việc giảm sai số trên tập dữ liệu kiểm tra. **Adam** cũng giảm loss nhanh và hội tụ tốt, tuy nhiên MSE cuối cùng ( $0.523$ ) cao hơn một chút so với RMSprop. Dù không xuất sắc bằng RMSprop, Adam vẫn là một phương pháp rất mạnh mẽ và phổ biến nhờ sự ổn định và khả năng thích nghi với nhiều bài toán khác nhau.

Phương pháp **SGD** có đặc điểm giảm loss một cách ổn định nhưng chậm hơn so với Adam và RMSprop. Tuy nhiên, kết quả cuối cùng của SGD khá gần với RMSprop, với  $MSE = 0.492$ , cho thấy đây là một lựa chọn tốt khi ưu tiên tính đơn giản hoặc trong các trường hợp cần hiệu quả tính toán trên tập dữ liệu lớn.

Ngược lại, **Adagrad** không đạt được hiệu suất tốt như các phương pháp còn lại. Dù có xu hướng giảm loss mạnh trong giai đoạn đầu, tốc độ giảm dần nhanh chóng chững lại. Điều này dẫn đến MSE cuối cùng là  $2.042$ , cao hơn rất nhiều so với các phương pháp khác.

Nhìn chung, trong bài toán này, RMSprop là phương pháp hiệu quả nhất, tiếp theo là SGD và Adam, tùy vào yêu cầu cụ thể của bài toán. Adagrad không phù hợp trong trường hợp này và có thể bị hạn chế trong các bài toán cần giảm loss đến mức rất thấp hoặc dữ liệu không quá thưa.

## CHƯƠNG 2 – VẤN ĐỀ THỨ HAI

Dự đoán giá cổ phiếu là một bài toán đầy thử thách trong lĩnh vực tài chính và học máy. Thành công trong việc dự đoán giá cổ phiếu không chỉ giúp các nhà đầu tư đưa ra các quyết định chính xác mà còn hỗ trợ xây dựng các chiến lược giao dịch hiệu quả, giảm thiểu rủi ro và tối ưu hóa lợi nhuận.

Trong bài toán này, mục tiêu chính là dự đoán giá mở cửa của cổ phiếu trong ngày tiếp theo bằng cách sử dụng dữ liệu lịch sử, đặc điểm ngành, các yếu tố thời gian, cũng như các chỉ số kinh tế vĩ mô.

### 2.1 Thử nghiệm các phương pháp

Dữ liệu trong bài toán bao gồm nhiều yếu tố quan trọng liên quan đến giá cổ phiếu, được chia thành các nhóm đặc trưng. Những đặc trưng này phản ánh tình trạng chung của nền kinh tế, giúp mô hình có cái nhìn tổng thể và phân tích được các yếu tố vĩ mô ảnh hưởng đến sự biến động giá cổ phiếu.

	date	industry_type	opening_price	closing_price	volume	gdp_growth	\
0	2015-01-01	Energy	190.54	185.42	57565	4.13	
1	2015-01-02	Finance	426.16	421.24	103379	0.57	
2	2015-01-03	Retail	290.08	294.93	576331	0.75	
3	2015-01-04	Finance	356.67	357.05	499847	2.39	
4	2015-01-05	Finance	220.26	222.37	818851	-1.45	

	inflation_rate	interest_rate	unemployment_rate	stock_market_index	\
0	5.44	2.74	7.25	5219.46	
1	4.35	4.86	2.86	5036.66	
2	9.88	2.13	3.47	6293.92	
3	6.19	4.67	9.21	2276.63	
4	7.11	4.25	9.92	3924.55	

	month	day_of_week	quarter	day_of_year
0	1	Thursday	1	1
1	1	Friday	1	2
2	1	Saturday	1	3
3	1	Sunday	1	4
4	1	Monday	1	5

Hình 2. 1 Các đặc trưng của tập dữ liệu trong bài toán

Ngoài ra, ta cũng có thể tải bộ dữ liệu mà chúng ta đã tạo ra về máy và được lưu dưới dạng tệp CSV.

```
[8] from google.colab import files

# Lưu dataset thành file CSV
file_name = "stock_price_dataset.csv"
df.to_csv(file_name, index=False)

# Tải file về
print(f"File '{file_name}' đã được tạo. Bắt đầu tải xuống...")
files.download(file_name)
```

### 2.1.1 Tiền xử lý dữ liệu

Loại bỏ dữ liệu thiếu: Dữ liệu thiếu (null hoặc NaN) có thể ảnh hưởng đến chất lượng của mô hình học máy. Ta xử lý vấn đề này bằng cách loại bỏ các dòng hoặc cột chứa giá trị thiếu. Trong trường hợp dữ liệu quan trọng, có thể thay thế giá trị thiếu bằng các giá trị thống kê như trung bình (mean), trung vị (median), hoặc sử dụng phương pháp gần nhất (nearest neighbor imputation). Trong mã, ta loại bỏ các dòng có giá trị thiếu bằng hàm `dropna()`.

Loại bỏ dữ liệu trùng lặp: Dữ liệu trùng lặp có thể gây ra sự sai lệch trong kết quả phân tích. Ta loại bỏ các dòng trùng lặp bằng hàm `drop_duplicates()` để đảm bảo mỗi bản ghi là duy nhất.

Loại bỏ điểm ngoại lai (outliers): Các điểm ngoại lai có thể gây ảnh hưởng tiêu cực đến mô hình. Ta sẽ loại bỏ các điểm ngoại lai bằng phương pháp Interquartile Range – IQR, khi đó, các điểm có giá trị nằm ngoài khoảng từ  $Q1 - 1.5 * IQR$  đến  $Q3 + 1.5 * IQR$  sẽ bị loại bỏ, điều này giúp loại bỏ các giá trị cực đoan, giữ lại các giá trị hợp lý và tăng độ chính xác của mô hình.

Mã hóa biến phân loại (Categorical Encoding): Các biến phân loại như `industry_type`, `day_of_week`, `month` và `quarter` cần được chuyển đổi thành dạng số để mô hình có thể xử lý, bằng cách dùng hàm `get_dummies()` trong pandas. Đây là một phương pháp one-hot encoding, mỗi giá trị của biến phân loại sẽ được chuyển thành một cột riêng biệt với giá trị 0 hoặc 1.

Chuẩn hóa dữ liệu (Data Scaling): Các đặc trưng liên tục như `opening_price`, `closing_price`, `volume` và các chỉ số kinh tế vĩ mô cần được chuẩn hóa để có tầm quan trọng tương đương trong mô hình. Phương pháp chuẩn hóa phổ biến là sử dụng `StandardScaler`, giúp chuẩn hóa các giá trị vào phân phối chuẩn với trung bình bằng 0 và độ lệch chuẩn bằng 1. Việc chuẩn hóa dữ liệu giúp cải thiện tốc độ hội tụ và hiệu suất của mô hình, đồng thời giúp các thuật toán học máy hoạt động tốt hơn khi các đặc trưng có độ lớn khác nhau.

Chia dữ liệu (Data Splitting): Sau khi hoàn tất các bước tiền xử lý dữ liệu, chúng ta sẽ chia dữ liệu thành hai phần: bộ huấn luyện (training set) và bộ kiểm tra (test set). Việc chia dữ liệu giúp mô hình có thể học từ một phần dữ liệu và đánh giá hiệu suất trên phần còn lại, giúp chúng ta đánh giá khả năng tổng quát của mô hình. Một tỷ lệ phổ biến để chia dữ liệu là 80% cho bộ huấn luyện và 20% cho bộ kiểm tra, tuy nhiên tỷ lệ này có thể điều chỉnh tùy theo yêu cầu và đặc thù của bài toán. Trong ví dụ trên, chúng ta đã sử dụng `train_test_split()` từ thư viện `scikit-learn` để thực hiện việc chia dữ liệu.

Chọn mô hình học máy (Model Selection): Khi đã chuẩn bị dữ liệu, bước tiếp theo là chọn mô hình học máy phù hợp để giải quyết bài toán. Trong trường hợp này, vì bài toán của bạn là dự báo giá cổ phiếu (một bài toán hồi quy), chúng ta có thể thử nghiệm với nhiều mô hình hồi quy khác nhau để tìm ra mô hình phù hợp nhất. Để tiết kiệm thời gian và công sức, thư viện `LazyRegressor` có thể giúp chúng ta tự động thử

nghiệm với nhiều mô hình hồi quy khác nhau, bao gồm các mô hình như Linear Regression, Random Forest, XGBoost, Lasso và nhiều mô hình khác. Thư viện này sẽ trả về các chỉ số đánh giá của các mô hình như  $R^2$ , MSE (Mean Squared Error), MAE (Mean Absolute Error), v.v., giúp chúng ta so sánh và lựa chọn mô hình tốt nhất.

**Đánh giá mô hình (Model Evaluation):** Sau khi đã chọn mô hình, chúng ta cần đánh giá hiệu suất của mô hình trên bộ kiểm tra (test set). Các chỉ số thường được sử dụng để đánh giá mô hình hồi quy bao gồm:

- $R^2$  (R-squared): Đo lường tỷ lệ phương sai của biến mục tiêu được giải thích bởi mô hình. Giá trị của  $R^2$  dao động từ 0 đến 1, với giá trị càng gần 1, mô hình càng phù hợp với dữ liệu.
- MSE (Mean Squared Error): Đo lường sai số trung bình của các dự đoán. MSE càng nhỏ, mô hình càng chính xác.
- RMSE (Root Mean Squared Error): Là căn bậc hai của MSE, giúp đo lường độ lệch chuẩn của sai số dự đoán. RMSE nhỏ cho thấy mô hình có độ chính xác cao.
- MAE (Mean Absolute Error): Đo lường sai số tuyệt đối trung bình, cho biết mức độ sai lệch giữa các giá trị dự đoán và giá trị thực tế.

Sau khi sử dụng thư viện LazyRegressor, chúng ta có thể sắp xếp các mô hình theo các chỉ số như  $R^2$  để chọn ra mô hình tốt nhất.



• stock\_price\_dataset.csv(text/csv) - 164397 bytes, last modified: 12/19/2024 - 100% done

Saving stock\_price\_dataset.csv to stock\_price\_dataset (3).csv

100%|██████████| 42/42 [00:13<00:00, 3.66it/s][LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000243 seconds. You can set 'force\_col\_wise=true' to remove the overhead.

[LightGBM] [Info] Total Bins 2023

[LightGBM] [Info] Number of data points in the train set: 1600, number of used features: 8

[LightGBM] [Info] Start training from score 0.013139

100%|██████████| 42/42 [00:13<00:00, 3.00it/s]

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
OrthogonalMatchingPursuitCV	1.00	1.00	0.04	0.03
OrthogonalMatchingPursuit	1.00	1.00	0.04	0.02
LassoLarsIC	1.00	1.00	0.04	0.09
SGDRegressor	1.00	1.00	0.04	0.06
ElasticNetCV	1.00	1.00	0.04	0.29
TransformedTargetRegressor	1.00	1.00	0.04	0.05
LinearRegression	1.00	1.00	0.04	0.07
Lars	1.00	1.00	0.04	0.03
RANSACRegressor	1.00	1.00	0.04	0.06
BayesianRidge	1.00	1.00	0.04	0.09
RidgeCV	1.00	1.00	0.04	0.08
Ridge	1.00	1.00	0.04	0.05
HuberRegressor	1.00	1.00	0.04	0.05
LarsCV	1.00	1.00	0.04	0.09
LassoLarsCV	1.00	1.00	0.04	0.09
LassoCV	1.00	1.00	0.04	0.39
LinearSVR	1.00	1.00	0.04	0.08
KernelRidge	1.00	1.00	0.05	0.34
PassiveAggressiveRegressor	1.00	1.00	0.05	0.02
GradientBoostingRegressor	1.00	1.00	0.05	1.21
ExtraTreesRegressor	1.00	1.00	0.05	1.67
HistGradientBoostingRegressor	1.00	1.00	0.05	0.75
RandomForestRegressor	1.00	1.00	0.05	1.96
LGBMRegressor	1.00	1.00	0.05	0.22
BaggingRegressor	1.00	1.00	0.05	0.56
AdaBoostRegressor	1.00	1.00	0.06	1.00
ExtraTreeRegressor	1.00	1.00	0.06	0.05
DecisionTreeRegressor	1.00	1.00	0.06	0.12
NuSVR	0.99	0.99	0.08	0.89
MLPRegressor	0.99	0.99	0.09	0.74
SVR	0.99	0.99	0.10	0.22

Hình 2. 2 Sắp xếp các mô hình theo các chỉ số

Tiếp theo, chúng ta sẽ lựa chọn mô hình tối ưu nhất: Sau khi tiến hành đánh giá các mô hình và so sánh các chỉ số đánh giá như  $R^2$ , MSE, RMSE, MAE từ thư viện LazyRegressor, bước tiếp theo là lựa chọn mô hình tốt nhất cho bài toán của mình. Việc lựa chọn mô hình tối ưu sẽ phụ thuộc vào việc cân nhắc giữa độ chính xác và khả năng tổng quát của mô hình. Mô hình có giá trị  $R^2$  cao và MSE thấp thường được coi là tốt hơn. Tuy nhiên, trong một số trường hợp, một mô hình có thể cho giá trị  $R^2$  thấp nhưng lại ít bị overfitting và có khả năng tổng quát tốt hơn trên dữ liệu chưa thấy, điều này có thể giúp mô hình hoạt động hiệu quả hơn trong môi trường thực tế.

Tối ưu hóa mô hình (Model Optimization): Sau khi chọn được mô hình tốt nhất, bước tiếp theo là tối ưu hóa mô hình để cải thiện hiệu suất dự đoán. Một trong những phương pháp tối ưu hóa phổ biến là sử dụng GridSearchCV hoặc RandomizedSearchCV. Trong GridSearchCV, ta thực hiện việc tìm kiếm các kết hợp tham số của mô hình thông

qua việc thử nghiệm các giá trị tham số khác nhau, giúp tìm ra bộ tham số tốt nhất cho mô hình.

Trong đoạn mã đã cung cấp, GridSearchCV được áp dụng cho một số mô hình hồi quy như RandomForestRegressor, SVR, DecisionTreeRegressor, KNeighborsRegressor, GradientBoostingRegressor và AdaBoostRegressor. Các tham số được tối ưu hóa có thể bao gồm số lượng cây trong rừng (`n_estimators`), độ sâu tối đa của cây quyết định (`max_depth`), số lượng lá tối thiểu trong một nhánh cây (`min_samples_leaf`) và các tham số khác tùy thuộc vào từng mô hình cụ thể. Việc sử dụng GridSearchCV giúp tìm ra mô hình tối ưu với các tham số phù hợp nhất để cải thiện độ chính xác và khả năng tổng quát.

Đánh giá và lựa chọn mô hình tối ưu: Kết quả từ GridSearchCV sẽ cung cấp mô hình với bộ tham số tối ưu (`best_estimator_`), giúp chúng ta có thể đánh giá mô hình này trên bộ dữ liệu kiểm tra (test set). Các chỉ số như MSE,  $R^2$  và RMSE sẽ được tính toán để đánh giá mức độ phù hợp của mô hình. Việc sử dụng các chỉ số này cho phép chúng ta kiểm tra xem mô hình có thể dự đoán giá trị mục tiêu một cách chính xác hay không, đồng thời giúp xác định các vấn đề như overfitting (quá khớp) hoặc underfitting (chưa đủ khớp).

Chẩn đoán và điều chỉnh mô hình (Model Diagnosis and Adjustment): Khi mô hình đã được tối ưu và đánh giá, chúng ta cần kiểm tra và chẩn đoán các vấn đề tiềm ẩn. Một trong những vấn đề phổ biến là overfitting, khi mô hình học quá kỹ dữ liệu huấn luyện, dẫn đến kết quả kém khi dự đoán dữ liệu mới. Để giải quyết vấn đề này, có thể áp dụng các phương pháp như điều chỉnh độ phức tạp của mô hình (ví dụ: giới hạn độ sâu cây quyết định hoặc số lượng cây trong rừng), sử dụng phương pháp regularization như

Ridge hoặc Lasso, hoặc áp dụng ensemble methods (như Random Forest hoặc Gradient Boosting).

Ngược lại, underfitting xảy ra khi mô hình quá đơn giản và không thể nắm bắt được các mô hình phức tạp trong dữ liệu. Để khắc phục tình trạng này, có thể thử các mô hình phức tạp hơn hoặc điều chỉnh các tham số để tạo ra một mô hình mạnh mẽ hơn.

```
Đang sử dụng mô hình: OrthogonalMatchingPursuitCV
Fitting 3 folds for each of 4 candidates, totalling 12 fits
Mô hình tối ưu hóa: OrthogonalMatchingPursuit()
Mean Squared Error (MSE): 0.0019618440256418215
R-squared (R2): 0.998001179537008
```

Sau khi xác định được mô hình tốt nhất và thực hiện các bước tối ưu hóa cơ bản, sau đó thì bắt đầu tính learning curve bằng cross validation của mô hình được tối ưu hóa tham số và vẽ biểu đồ.

Learning curve là một công cụ hữu ích để đánh giá hiệu suất mô hình khi kích thước tập huấn luyện tăng lên. Để tính toán learning curve, bạn cần sử dụng cross-validation để tính toán độ chính xác của mô hình trên các tập huấn luyện khác nhau.

Giả sử bạn đã tối ưu hóa mô hình của mình bằng GridSearchCV và có mô hình best\_model (mô hình tốt nhất sau khi tối ưu hóa tham số). Tiếp theo, bạn sẽ tính toán learning curve bằng cách sử dụng nhiều kích thước tập huấn luyện khác nhau và tính toán các chỉ số như training score và test score.

Đầu tiên, cần chia dữ liệu của mình thành hai phần: tập huấn luyện và tập kiểm tra. Sau đó, bạn sẽ tính toán learning curve cho mô hình đã được tối ưu hóa. Điều này có thể thực hiện bằng cách sử dụng hàm learning\_curve từ thư viện scikit-learn, giúp tính toán các điểm số huấn luyện và kiểm tra trên các kích thước tập huấn luyện khác nhau.

Đoạn mã này sẽ trả về các giá trị như `train_sizes`, `train_scores` và `test_scores`. Sau khi có các điểm số này thì tiếp theo cần tính toán trung bình và độ lệch chuẩn cho mỗi tập.

Biểu đồ learning curve sẽ dễ hình dung sự thay đổi của điểm số huấn luyện và kiểm tra khi kích thước tập huấn luyện tăng lên. Biểu đồ này thường bao gồm hai đường: đường training score (thường có màu xanh) và đường test score (thường có màu đỏ). Cả hai đường này sẽ được tính toán cho mỗi kích thước tập huấn luyện khác nhau.

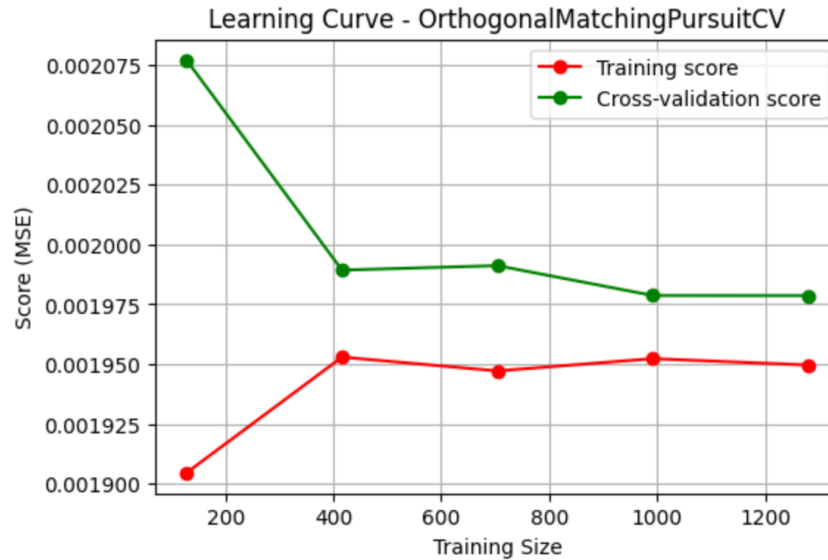
Overfitting và Underfitting:

- Nếu training score rất cao và test score thấp, mô hình có thể đang bị overfitting, nghĩa là mô hình học quá chi tiết từ dữ liệu huấn luyện nhưng không tổng quát tốt trên dữ liệu mới.
- Nếu cả training score và test score đều thấp, mô hình có thể bị underfitting, nghĩa là mô hình chưa học đủ từ dữ liệu huấn luyện.

Hiệu suất mô hình: Nếu cả hai đường (training và test) đều có xu hướng hội tụ khi kích thước tập huấn luyện tăng lên, điều này cho thấy mô hình đang học tốt và có khả năng tổng quát tốt.

Lựa chọn mô hình: Mục tiêu là tìm một mô hình có test score ổn định và gần với training score để đảm bảo khả năng tổng quát của mô hình.

Mô hình tốt nhất: OrthogonalMatchingPursuitCV



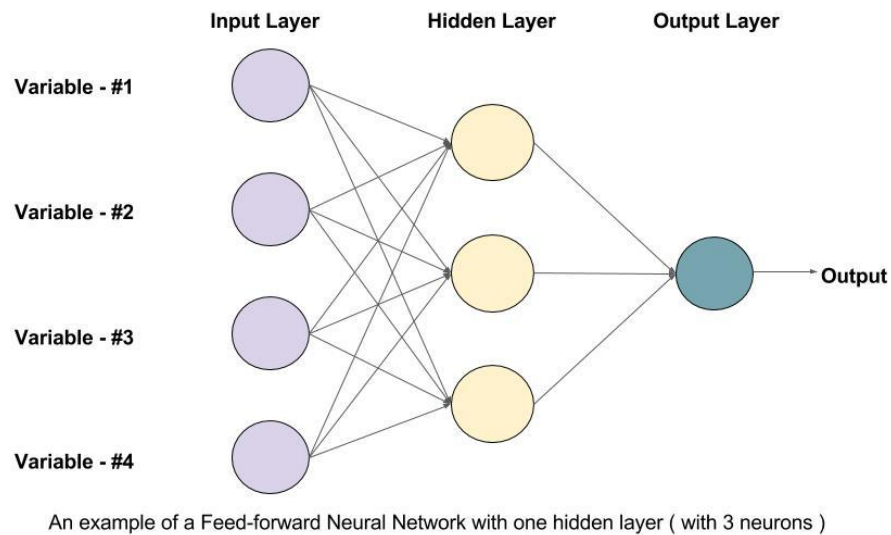
Hình 2. 3 Sơ đồ cho mô hình tốt nhất

### 2.1.2 Feedforward Neural Network – FFNN

Mạng Neural Feedforward là một loại mạng neural nhân tạo đơn giản, trong đó các lớp được kết nối đầy đủ từ đầu vào đến đầu ra. Mỗi lớp trong mạng được liên kết với lớp kế tiếp thông qua các trọng số và tín hiệu đầu vào được truyền qua các lớp theo một chiều duy nhất, từ đầu vào đến đầu ra.

FFNN có những ưu điểm đáng chú ý như:

1. Khả năng học phi tuyến tốt: FFNN có thể học được các mối quan hệ phi tuyến giữa các đặc trưng trong dữ liệu, giúp nó áp dụng hiệu quả cho nhiều loại bài toán khác nhau.
2. Dễ triển khai và phù hợp với dữ liệu không có mối quan hệ thời gian phức tạp: Mạng FFNN đơn giản và dễ triển khai, rất phù hợp với các bài toán mà dữ liệu không có tính chất chuỗi thời gian hoặc các mối quan hệ thời gian phức tạp.



Hình 2. 4 Mạng neural Feedforward

Nhược điểm của FFNN: Hiệu suất không cao với dữ liệu chuỗi thời gian hoặc dữ liệu phức tạp, vì không có cơ chế lưu trữ trạng thái hay thông tin về chuỗi thời gian trước đó, nghĩa là các tín hiệu không được ghi nhớ giữa các lần tính toán nên FFNN không thể học tốt các mối quan hệ phụ thuộc vào thời gian hoặc các cấu trúc dữ liệu phức tạp. Mạng FFNN không thể tận dụng thông tin từ các trạng thái trước đó để dự đoán kết quả trong tương lai.

```
Epoch 29/100
40/40 — 0s 3ms/step - loss: 0.0418 - mse: 0.0418 - val_loss: 0.0155 - val_mse: 0.0155
Epoch 30/100
40/40 — 0s 3ms/step - loss: 0.0270 - mse: 0.0270 - val_loss: 0.0055 - val_mse: 0.0055
Epoch 31/100
40/40 — 0s 3ms/step - loss: 0.0083 - mse: 0.0083 - val_loss: 0.0920 - val_mse: 0.0920
Epoch 32/100
40/40 — 0s 3ms/step - loss: 0.0772 - mse: 0.0772 - val_loss: 0.0134 - val_mse: 0.0134
13/13 — 0s 5ms/step
FFNN Metrics:
MSE: 0.00359460003600107
RMSE: 0.05995498341256605
MAE: 0.047288457575108
R²: 0.998001179537008
Explained Variance: 0.9967190446632551
```

Hình 2. 5 Kết quả chạy FFNN

### 2.1.3 Recurrent Neural Network – RNN

Mạng Neural Hồi tiếp là một loại mạng neural đặc biệt có khả năng ghi nhớ trạng thái trước đó thông qua cơ chế hồi tiếp, cho phép chúng lưu trữ và sử dụng thông tin từ các bước trước để tác động đến kết quả dự đoán của các bước sau. Điều này khiến RNN đặc biệt hiệu quả trong việc xử lý dữ liệu chuỗi thời gian, chẳng hạn như dự đoán văn bản, âm thanh, hoặc các tín hiệu thời gian.

Các biến thể phổ biến của RNN bao gồm Long Short-Term Memory (LSTM) và Gated Recurrent Unit (GRU), cả hai đều cải thiện khả năng học hỏi của RNN trong các tác vụ xử lý chuỗi thời gian.

RNN có những ưu điểm như:

1. Khả năng xử lý dữ liệu phụ thuộc thời gian: RNN có khả năng học và ghi nhớ các mối quan hệ phụ thuộc theo thời gian trong dữ liệu chuỗi, từ đó giúp mô hình nắm bắt được các thông tin quan trọng từ quá khứ để dự đoán các giá trị trong tương lai.
2. Ghi nhớ các thông tin quan trọng trong chuỗi: Các biến thể của RNN như LSTM và GRU giúp cải thiện khả năng ghi nhớ và lưu trữ các thông tin quan trọng trong dữ liệu chuỗi thời gian, đặc biệt là khi chuỗi có sự phụ thuộc dài hạn.

Nhược điểm:

1. Vấn đề Gradient Vanishing/Exploding: RNN dễ gặp phải vấn đề gradient vanishing (biến mất gradient) hoặc gradient exploding (nổ gradient), khiến cho quá trình huấn luyện trở nên khó khăn và không ổn định khi làm việc với chuỗi dài.
2. Thời gian huấn luyện dài và yêu cầu tài nguyên tính toán lớn: Việc huấn luyện RNN, đặc biệt là các mô hình lớn và phức tạp như LSTM

và GRU, yêu cầu nhiều tài nguyên tính toán và có thể tốn nhiều thời gian, có thể làm giảm hiệu suất trong các ứng dụng thực tế.

```
Epoch 22/100
40/40 ————— 0s 8ms/step - loss: 0.0393 - mse: 0.0393 - val_loss: 0.0068 - val_mse: 0.0068
Epoch 23/100
40/40 ————— 0s 7ms/step - loss: 0.0350 - mse: 0.0350 - val_loss: 0.0090 - val_mse: 0.0090
Epoch 24/100
40/40 ————— 0s 6ms/step - loss: 0.0363 - mse: 0.0363 - val_loss: 0.0075 - val_mse: 0.0075
Epoch 25/100
40/40 ————— 0s 7ms/step - loss: 0.0351 - mse: 0.0351 - val_loss: 0.0092 - val_mse: 0.0092
Epoch 26/100
40/40 ————— 0s 6ms/step - loss: 0.0343 - mse: 0.0343 - val_loss: 0.0147 - val_mse: 0.0147
Epoch 27/100
40/40 ————— 0s 7ms/step - loss: 0.0340 - mse: 0.0340 - val_loss: 0.0066 - val_mse: 0.0066
13/13 ————— 1s 27ms/step
RNN Metrics:
MSE: 0.006657045060049902
RMSE: 0.08159071675166178
MAE: 0.06428410423740806
R²: 0.9932174843080429
Explained Variance: 0.9932870372167524
```

Hình 2. 6 Kết quả chạy RNN

### 2.1.4 Linear Regression

Hồi quy tuyến tính là một mô hình cơ bản trong học máy, được sử dụng để dự đoán một giá trị liên tục dựa trên mối quan hệ tuyến tính giữa các đặc trưng và biến mục tiêu. Tuy nhiên, mô hình này gặp phải hạn chế khi dữ liệu có mối quan hệ phi tuyến, bởi nó chỉ hoạt động tốt khi các đặc trưng có mối quan hệ tuyến tính với nhau.

```
➡ Linear Regression Metrics:
MSE: 0.0019877793220660573
RMSE: 0.04458451886099094
MAE: 0.03877742402652215
R²: 0.9979747554173893
Explained Variance: 0.9979814351705155
```


Hình 2. 7 Kết quả chạy Linear Regression

### 2.1.5 Support Vector Machine – SVM

Máy Vector Hỗ Trợ là một thuật toán học máy mạnh mẽ, tìm kiếm siêu phẳng tối ưu để phân loại dữ liệu hoặc dự đoán giá trị liên tục. Mục tiêu của nó là tìm ra siêu phẳng phân chia các lớp dữ liệu sao cho khoảng cách giữa các điểm dữ liệu và siêu phẳng là



lớn nhất. SVM thích hợp cho các bài toán với bộ dữ liệu có số lượng đặc trưng nhỏ và không có quá nhiều điểm dữ liệu.




```
SVM Metrics:
MSE: 0.8988112095639821
RMSE: 0.9480565434424163
MAE: 0.80925861492889
R²: 0.0842481794874701
Explained Variance: 0.09353707835243397
```

Hình 2. 8 Kết quả chạy SVM

### 2.1.6 Decision Tree

Cây quyết định là một mô hình phân loại hoặc hồi quy sử dụng cấu trúc cây để đưa ra các quyết định. Mỗi nút trong cây đại diện cho một câu hỏi về một đặc trưng và các nhánh tương ứng đại diện cho các kết quả của câu hỏi đó. Cây quyết định khá dễ triển khai và giải thích kết quả, tuy nhiên nó dễ bị overfitting khi dữ liệu có sự biến động lớn hoặc có quá nhiều đặc trưng.



```
Decision Tree Metrics:
MSE: 0.004208843878478506
RMSE: 0.06487560310685755
MAE: 0.05232540954613809
R²: 0.9957118286877625
Explained Variance: 0.9957123648265628
```

Hình 2. 9 Kết quả chạy Decision Tree

### 2.1.7 Random Forest

Rừng ngẫu nhiên là một thuật toán học máy mạnh mẽ dựa trên tập hợp của nhiều cây quyết định. Mỗi cây trong rừng được huấn luyện trên một mẫu ngẫu nhiên khác nhau của dữ liệu huấn luyện và sau đó các kết quả của các cây này được kết hợp lại để đưa ra dự đoán cuối cùng. Việc sử dụng nhiều cây giúp giảm bớt độ nhiễu và tăng độ chính xác của mô hình so với cây quyết định đơn lẻ, đồng thời cũng giảm thiểu nguy cơ overfitting.

```

➡ Random Forest Metrics:
MSE: 0.002290128454685767
RMSE: 0.047855286590780824
MAE: 0.040093777241897464
R²: 0.9976667076697859
Explained Variance: 0.9976672200635351

```

Hình 2. 10 Kết quả chạy Random Forest

## 2.2 Ngăn chặn overfitting và vẽ đồ thị đào tạo

### 2.2.1 Ngăn chặn overfitting

Regularization: Kỹ thuật được sử dụng để giảm sự phức tạp của mô hình và ngăn ngừa hiện tượng overfitting. Hai phương pháp chuẩn hóa phổ biến là L1 Regularization và L2 Regularization. Cả hai phương pháp đều áp dụng một hình phạt lên các trọng số của mô hình, đặc biệt là các trọng số có giá trị quá lớn, nhằm hạn chế mô hình quá khớp với dữ liệu huấn luyện. L1 regularization có thể dẫn đến việc loại bỏ các trọng số không quan trọng, trong khi L2 regularization giúp làm giảm sự thay đổi của các trọng số quá mức, từ đó cải thiện khả năng tổng quát hóa của mô hình.

Dropout: Phương pháp đơn giản và hiệu quả để giảm overfitting trong mạng nơ-ron. Trong quá trình huấn luyện, nó ngẫu nhiên loại bỏ một tỷ lệ nơ-ron trong mỗi lần cập nhật, giúp tránh sự phụ thuộc quá mức vào một nhóm nơ-ron hoặc đặc trưng cụ thể. Điều này tạo ra các mạng con khác nhau trong quá trình huấn luyện, làm cho mô hình học được các đặc trưng đa dạng hơn và trở nên mạnh mẽ hơn trong việc tổng quát hóa khi áp dụng vào dữ liệu chưa thấy.

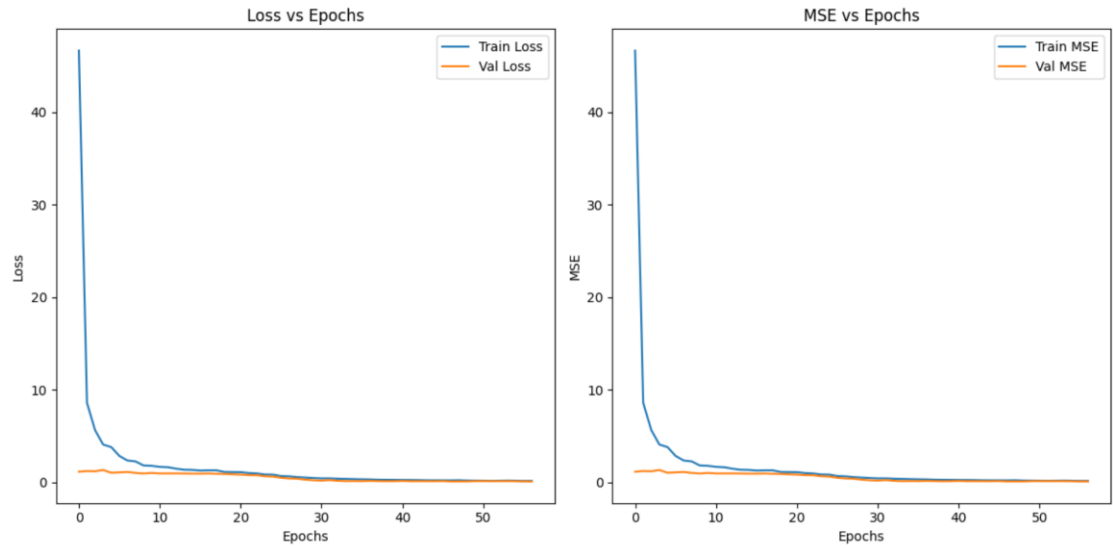
Early Stopping: Kỹ thuật dừng sớm quá trình huấn luyện khi độ lỗi trên tập kiểm tra không còn giảm nữa. Khi mô hình học quá lâu và bắt đầu overfitting, độ lỗi trên tập kiểm tra sẽ bắt đầu tăng, trong khi độ lỗi trên tập huấn luyện vẫn tiếp tục giảm. Để ngừng quá trình huấn luyện trước khi mô hình học quá sâu vào dữ liệu huấn luyện, chúng ta sẽ

giám sát độ lỗi trên tập kiểm tra và dừng huấn luyện khi không thấy cải thiện trong một số vòng lặp liên tiếp. Điều này giúp giảm thiểu overfitting và tối ưu hóa độ chính xác của mô hình.

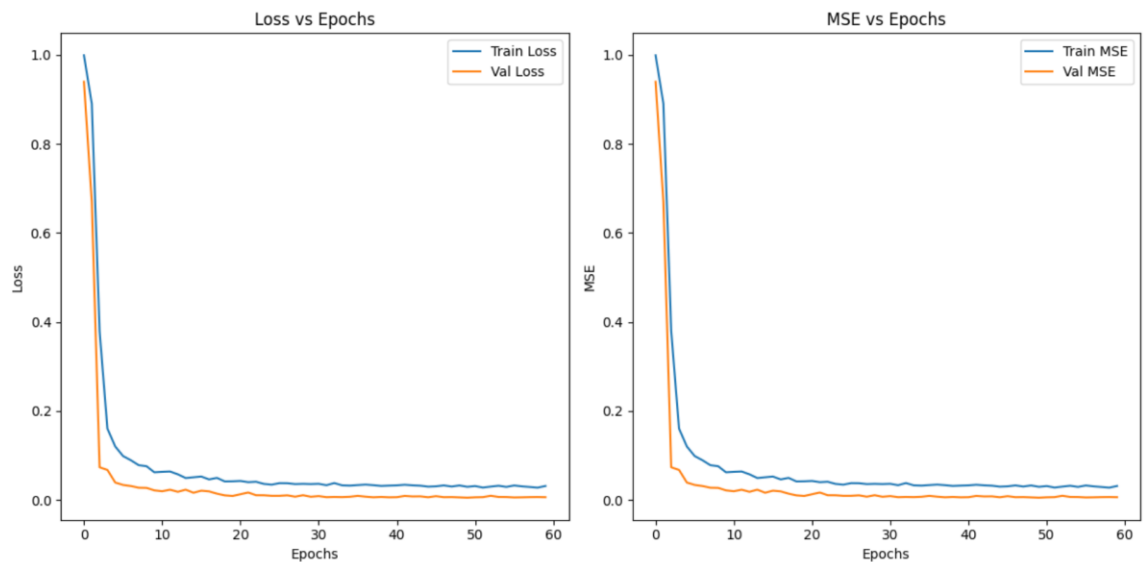
Cross-validation: Kỹ thuật đánh giá mô hình giúp kiểm tra hiệu suất của mô hình trên nhiều tập con dữ liệu khác nhau. Một phương pháp phổ biến là K-fold cross-validation, trong đó dữ liệu huấn luyện được chia thành K phần và mô hình được huấn luyện và kiểm tra K lần, mỗi lần sử dụng một phần dữ liệu làm tập kiểm tra và phần còn lại làm tập huấn luyện. Kết quả từ các lần kiểm tra này sau đó được tổng hợp lại để đưa ra đánh giá tổng thể về khả năng tổng quát hóa của mô hình. Việc sử dụng cross-validation giúp đảm bảo rằng mô hình không chỉ học tốt trên một tập huấn luyện cụ thể mà còn có thể hoạt động hiệu quả trên các dữ liệu chưa thấy.

### ***2.2.2 Đồ thị đào tạo***

Trong bài này, ta áp dụng hai kỹ thuật là **Early stopping** và **Dopout** để giảm thiểu overfitting và cải thiện khả năng tổng quát hóa của mô hình. Early stopping giúp ngừng huấn luyện khi mô hình không còn cải thiện trên tập kiểm tra, tránh việc học quá mức vào dữ liệu huấn luyện. Trong khi đó, Dropout ngẫu nhiên loại bỏ các nơ-ron trong quá trình huấn luyện, giúp giảm sự phụ thuộc vào một số đặc trưng và làm cho mô hình tổng quát hơn khi áp dụng vào dữ liệu mới.



Hình 2. 9 Giảm overfitting cho FFNN



Hình 2. 10 Giảm overfitting cho mô hình LSTM của RNN

### 2.3 Đánh giá các mô hình, so sánh và vẽ đồ thị kết quả

Hiệu suất của các mô hình sẽ được đánh giá thông qua các chỉ số sau:

1. **Mean Absolute Error (MAE):** Đo độ lệch tuyệt đối trung bình giữa giá trị dự đoán và thực tế, giúp dễ hiểu và không bị ảnh hưởng bởi giá trị ngoại lai.

2. **Mean Squared Error (MSE)**: Đo sai số bình phương trung bình, nhấn mạnh các sai số lớn, nhưng nhạy cảm với giá trị ngoại lai.
3. **Root Mean Squared Error (RMSE)**: Là căn bậc hai của MSE, giúp đưa chỉ số về cùng đơn vị với dữ liệu thực tế, dễ hiểu và trực quan hơn.
4. **R<sup>2</sup> Score**: Đo lường mức độ mà mô hình giải thích được phương sai trong dữ liệu, với giá trị từ 0 đến 1, cho thấy độ chính xác của mô hình trong việc mô phỏng mối quan hệ giữa các biến.

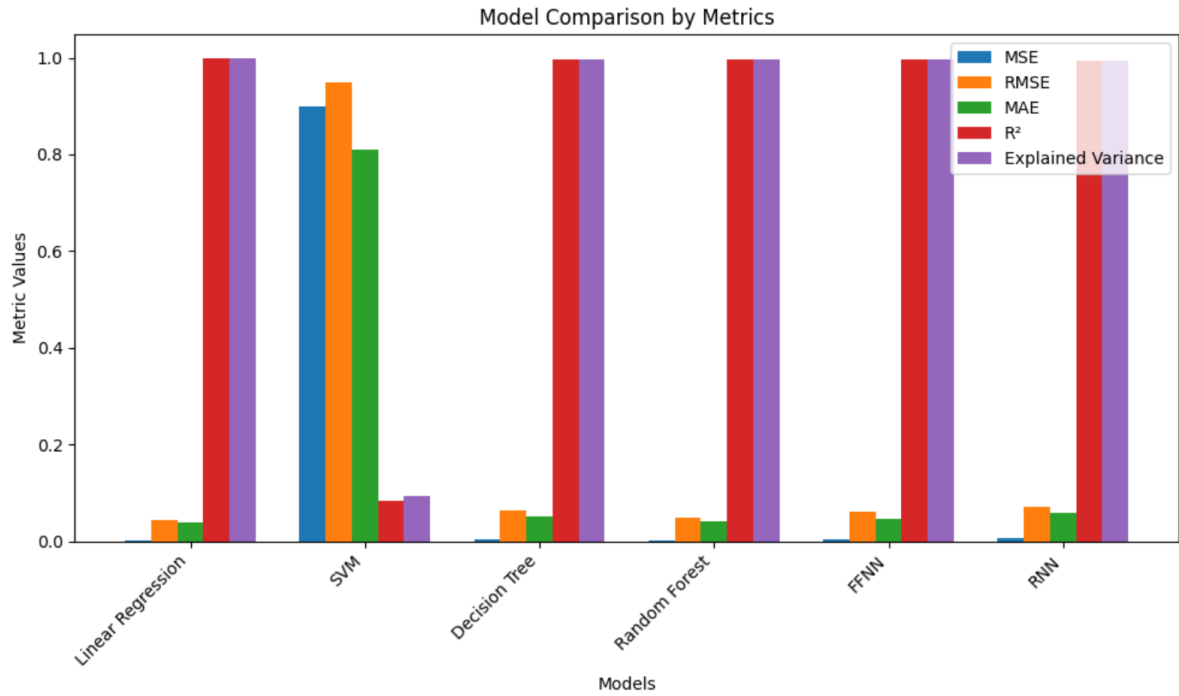
Nhận xét:

Dựa trên các chỉ số hiệu suất, mô hình RNN thể hiện sự vượt trội rõ rệt với MSE, RMSE và MAE rất thấp, cùng với R<sup>2</sup> và Explained Variance gần 1, cho thấy khả năng dự đoán chính xác và giải thích phương sai dữ liệu một cách xuất sắc.

Mô hình Linear Regression cũng cho kết quả ấn tượng, đặc biệt với chỉ số R<sup>2</sup> và Explained Variance gần như đạt mức tối đa, tuy nhiên vẫn thấp hơn so với RNN về mặt tổng quát.

Các mô hình Random Forest và Decision Tree đều cho kết quả khá tốt với MSE, RMSE và MAE thấp, đồng thời khả năng giải thích phương sai cũng rất cao, cho thấy độ chính xác cao khi làm việc với dữ liệu. Trong khi đó, SVM lại có hiệu suất kém hơn nhiều, với các chỉ số lỗi cao và khả năng giải thích phương sai thấp, cho thấy mô hình này không phù hợp với bài toán này.

Tổng thể, RNN là mô hình tối ưu nhất, tiếp theo là Linear Regression và Random Forest, còn SVM là lựa chọn kém hiệu quả nhất trong trường hợp này.



Hình 2. 11 Biểu đồ so sánh hiệu suất các mô hình

## CHƯƠNG 3 – VẤN ĐỀ THỨ BA

### 3.1 Deep Learning

#### 3.1.1 Giới thiệu chung

Deep learning (Học sâu) là một dạng của Machine learning (Học máy), sử dụng mạng nơ-ron nhân tạo để học từ tập dữ liệu. Mạng nơ-ron nhân tạo này lấy cảm hứng từ bộ não con người, được lập trình để có thể thực hiện các hành vi tương tự não người như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói,...

Giải thuật Học sâu thường được đào tạo trên các tập dữ liệu lớn có gắn nhãn. Các thuật toán sẽ học cách liên kết các thuộc tính (features) trong dữ liệu với nhãn chính xác. Sau khi được đào tạo, giải thuật Học sâu có thể đưa ra dự đoán về dữ liệu mới.

#### 3.1.2 Thuận lợi và Thách thức

Học sâu mang đến những thuận lợi rất đáng chú ý như:

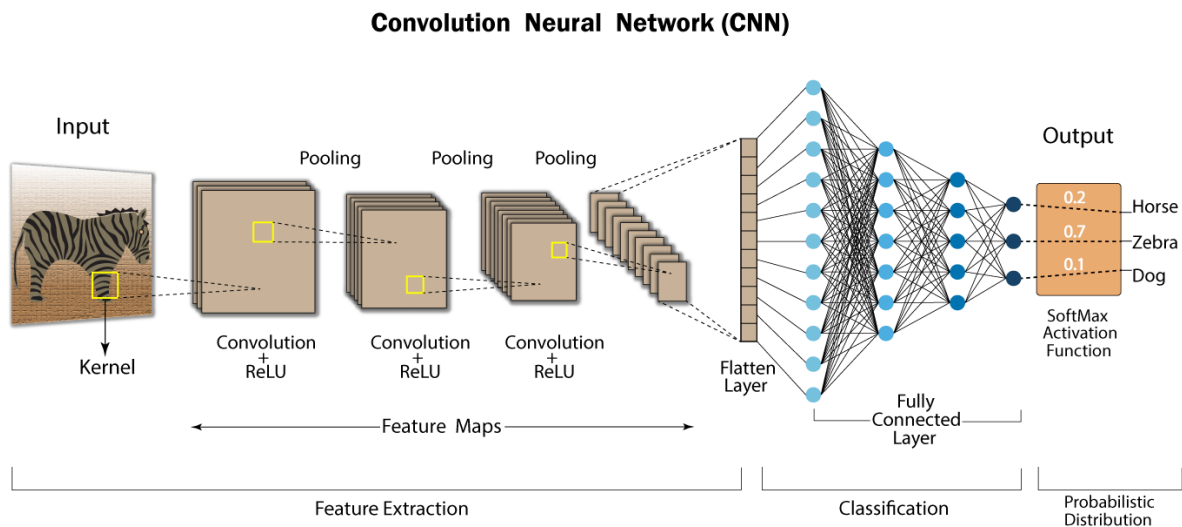
1. Có thể học các mối quan hệ phức tạp giữa các tính năng trong dữ liệu, điều này khiến nó trở nên cạnh tranh hơn so với các phương pháp Học máy truyền thống.
2. Đào tạo được tập dữ liệu lớn, ưu điểm này khiến Học sâu có khả năng mở rộng rất cao và có thể học từ nhiều trải nghiệm hơn, đưa ra các dự đoán chính xác hơn.
3. Học tập dựa trên dữ liệu: Các mô hình Học sâu có thể học theo cách dựa trên dữ liệu, đòi hỏi ít sự can thiệp của con người hơn trong quá trình đào tạo mô hình, điều này giúp tăng hiệu quả và khả năng mở rộng về sau.

Học sâu là một công nghệ tương đối phức tạp, do đó khi triển khai sẽ có một số thách thức:

1. Quy trình tiền xử lý dữ liệu đầu vào phức tạp: Các thuật toán Học sâu sẽ cho ra kết quả tốt hơn khi được đào tạo trên dữ liệu lớn có chất lượng cao, vì vậy, nếu không tiền xử lý tốt, các giá trị ngoại lai hoặc sai sót trong tập dữ liệu đầu vào có thể ảnh hưởng đáng kể đến quy trình học sâu.
2. Thời gian để xử lý kết quả: Các thuật toán học sâu thiên về điện toán và yêu cầu cơ sở hạ tầng đủ năng lực điện toán để hoạt động đúng cách, nếu không, các thuật toán này sẽ rất mất thời gian để xử lý kết quả.

### 3.2 Convolutional Neural Network

Convolutional Neural Network (CCN hoặc ConvNet) – Mạng neural tích chập là một trong những mô hình Học sâu tiên tiến, giúp xây dựng những hệ thống thông minh với độ chính xác cao. Mô hình này được ứng dụng cho các tác vụ phân loại và thị giác máy tính.



Hình 3. 1 CNN

Trước CNN, ta cũng có các phương pháp để xác định các đối tượng trong hình ảnh, tuy nhiên chúng khá thủ công và tốn thời gian. Trong khi đó, mạng neural tích chập



cung cấp một phương pháp tiếp cận có khả năng mở rộng hơn với các tác vụ phân loại hình ảnh và nhận dạng đối tượng, tận dụng các nguyên tắc từ đại số tuyến tính, như phép nhân ma trận, để xác định các mẫu trong hình ảnh.

### ***3.2.1 Cách hoạt động của mạng neural tích chập***

Mạng có 3 loại lớp chính:

Lớp tích chập (Convolutional layer)

Lớp gộp (Pooling layer)

Lớp kết nối đầy đủ (Fully – connected layer)

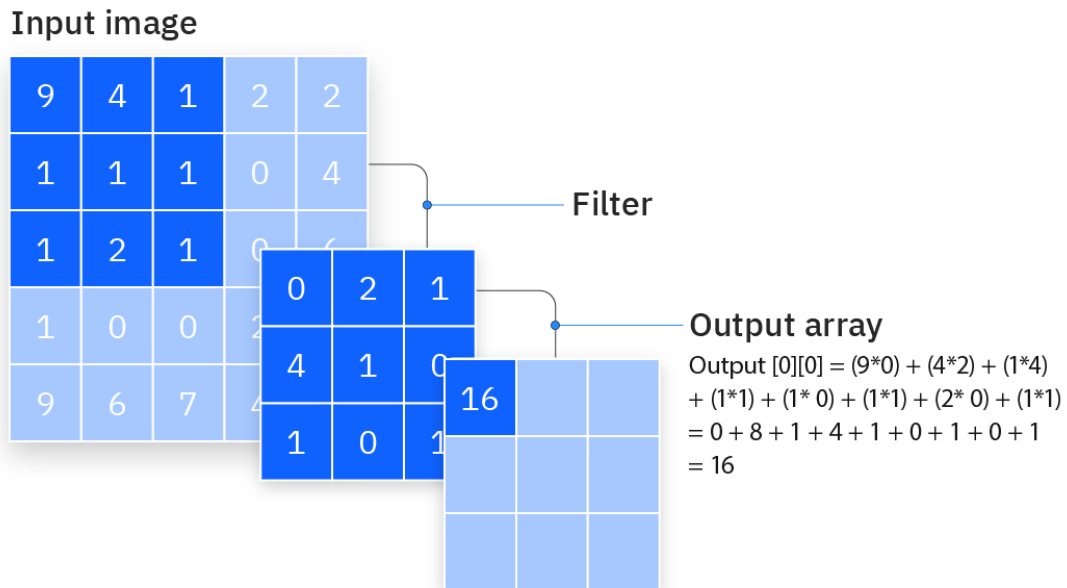
Lớp đầu tiên của mạng là Lớp tích chập, theo sau đó là các Lớp tích chập bổ sung hoặc các Lớp gộp, và Lớp được kết nối đầy đủ là lớp cuối cùng.

Với mỗi lớp, CNN tăng dần độ phức tạp, xác định các phần lớn hơn của hình ảnh. Các lớp trước đó tập trung vào các tính năng đơn giản, chẳng hạn như màu sắc và các cạnh. Khi dữ liệu hình ảnh qua các lớp của CNN, nó bắt đầu nhận dạng các phần tử hoặc hình dạng lớn hơn của đối tượng cho đến khi cuối cùng xác định được đối tượng mong muốn.

#### **3.2.1.1 Convolutional layer**

Lớp tích chập là khối xây dựng cốt lõi của CNN và là nơi diễn ra phần lớn quá trình tính toán. Nó yêu cầu các thành phần là dữ liệu đầu vào, bộ lọc (filter) và bản đồ đặc điểm (feature map).

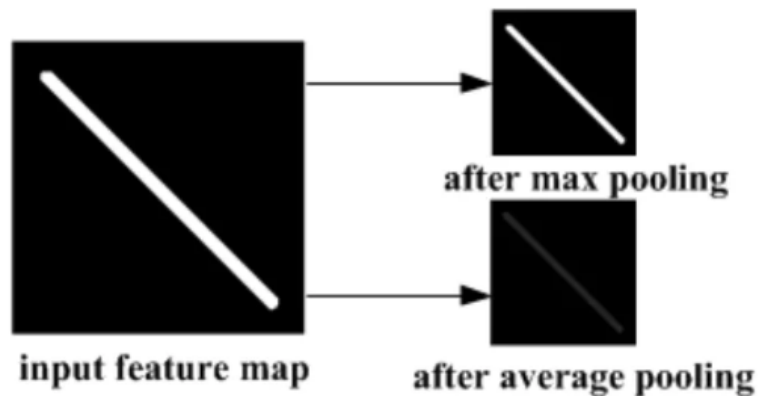
Giả sử đầu vào sẽ là một hình ảnh màu, được tạo thành từ ma trận các điểm ảnh trong 3D. Bộ dò đặc điểm (feature detector / kernel / filter) sẽ di chuyển qua các trường tiếp nhận của hình ảnh, kiểm tra xem đặc điểm có hiện diện hay không. Quá trình này được gọi là tích chập.



Hình 3. 2 Convolution

### 3.2.1.2 Pooling layer

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model. Tương tự như lớp tích chập, hoạt động gộp quét một bộ lọc trên toàn bộ đầu vào, nhưng bộ lọc này không có bất kỳ trọng số nào. Thay vào

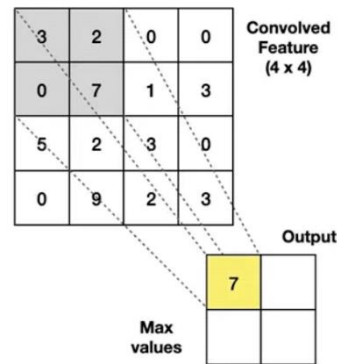


Hình 3. 3 Pooling

đó, hạt nhân áp dụng một hàm tổng hợp cho các giá trị trong trường tiếp nhận, điền vào mảng đầu ra.

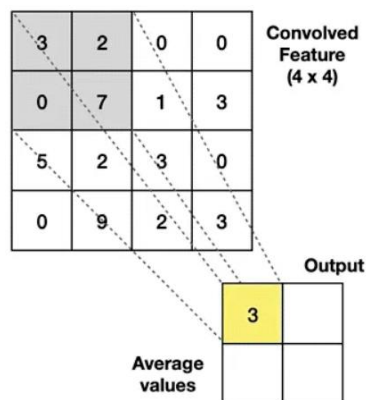
Có hai loại gộp chính:

1. Max pooling: Chọn pixel có giá trị lớn nhất để gửi đến mảng đầu ra. Cách tiếp cận này thường được sử dụng thường xuyên hơn.



Hình 3. 4 Max pooling

2. Average pooling: Tính toán giá trị trung bình trong trường tiếp nhận để gửi đến mảng đầu ra.



Hình 3. 5 Average pooling

### 3.2.1.3 Fully-connected (FC) layer

Trong lớp này, mỗi nút trong lớp đầu ra kết nối trực tiếp với một nút trong lớp trước đó. Lớp thực hiện nhiệm vụ phân loại dựa trên các đặc điểm được trích xuất thông qua các lớp trước đó và các bộ lọc khác nhau của chúng bằng cách tận dụng hàm kích hoạt softmax để phân loại đầu vào một cách thích hợp, tạo ra xác suất từ 0 đến 1.

### 3.3 Chương trình thực thi

#### 3.3.1 Tập dữ liệu và xử lý dữ liệu

Tập dữ liệu được sử dụng là tập dữ liệu *CIFAR-10 - Object Recognition in Images* (<https://www.kaggle.com/c/cifar-10/data>).

Tập dữ liệu này bao gồm 60.000 hình ảnh màu 32x32 trong 10 lớp, với 6000 hình ảnh cho mỗi lớp. Có 50.000 hình ảnh đào tạo và 10.000 hình ảnh thử nghiệm trong dữ liệu chính thức.

The label classes in the dataset are:

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck

Hình 3. 6 Các labels trong tập dữ liệu Cifar – 10

Trong Python, ta có thể sử dụng hàm `datasets.cifar10.load_data()` của thư viện TensorFlow để tải bộ dữ liệu CIFAR-10.

```

import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

# Tải bộ dữ liệu CIFAR-10 từ TensorFlow
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Hiển thị thông tin bộ dữ liệu
print(f"Training data shape: {train_images.shape}")
print(f"Testing data shape: {test_images.shape}")

# Chuẩn hóa dữ liệu (giảm giá trị pixel từ 0-255 về 0-1)
train_images, test_images = train_images / 255.0, test_images / 255.0

# Các nhãn cần được chuyển thành dạng one-hot encoding
train_labels = tf.keras.utils.to_categorical(train_labels, num_classes=10)
test_labels = tf.keras.utils.to_categorical(test_labels, num_classes=10)

```

Dữ liệu hình ảnh trong CIFAR-10 có giá trị pixel từ 0 đến 255. Để giúp mô hình học tốt hơn và nhanh hơn, giá trị pixel được chuẩn hóa về phạm vi từ 0 đến 1.

```

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ————— 4s 0us/step
Training data shape: (50000, 32, 32, 3)
Testing data shape: (10000, 32, 32, 3)

```

Hình 3. 7 Cấu trúc tập dữ liệu

### 3.3.2 Huấn luyện mô hình và kiểm tra

Lớp Conv2D (Convolutional Layer) tạo ra 64 bộ lọc (kernels) có kích thước 3x3 để xử lý hình ảnh đầu vào với kích thước 32x32 pixel và 3 kênh màu RGB và sử dụng hàm kích hoạt ReLU (Rectified Linear Unit) giúp mô hình học các đặc trưng phức tạp từ dữ liệu hình ảnh. `padding='same'` sẽ giữ cho kích thước của hình ảnh không thay đổi sau khi chập.

```
[3] from tensorflow.keras import layers, models

# Mô hình CNN
model = models.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3), padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.Flatten(),
    layers.Dense(10, activation='softmax')
])

# Biên dịch mô hình
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Huấn luyện mô hình
history = model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))

# Đánh giá mô hình
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
```

Hình 3. 8 Huấn luyện CNN

Lớp `MaxPooling2D` thực hiện phép giảm mẫu (pooling) 2x2, giúp giảm kích thước hình ảnh sau mỗi lớp chập và tăng tính trừu tượng của đặc trưng. Việc giảm kích thước giúp mô hình học nhanh hơn và giảm độ phức tạp tính toán.

Lớp `Flatten` chuyển đổi dữ liệu từ dạng ma trận 2D thành vector 1D để có thể kết nối với các lớp `Dense` (mạng fully connected).

Lớp `Dense` với 10 nút (tương ứng với 10 lớp trong `Cifar-10`) dùng để phân loại đầu ra, `activation='softmax'` hỗ trợ chuyển các giá trị đầu ra thành một phân phối xác suất.

Sau khi huấn luyện xong, ta chỉ định một chỉ số index để chọn một hình ảnh từ tập kiểm tra. Trong trường hợp này, hình ảnh được chọn là hình thứ 1000.

`test_images[index:index + 1]` dùng để lấy ra hình ảnh thứ 1000 từ tập kiểm tra, tạo ra một mảng chứa một hình ảnh duy nhất có kích thước (1, 32, 32, 3) và lưu vào `img`. Lúc này, `img` là một mảng 4 chiều với kích thước (1, 32, 32, 3).

Sau khi dự đoán hình ảnh xong, `np.argmax(predictions)` sẽ giá trị lớn nhất trong vector xác suất và trả về chỉ số của lớp đó, chỉ số này tương ứng với lớp mà mô hình dự đoán hình ảnh thuộc về.

```
[12] import numpy as np
import matplotlib.pyplot as plt

# Lấy một hình ảnh ngẫu nhiên từ bộ dữ liệu CIFAR-10
index = 1000
img = test_images[index:index + 1]

# Dự đoán lớp của hình ảnh
predictions = model.predict(img)

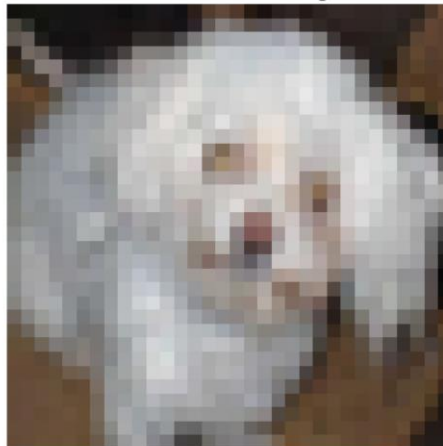
# Lấy lớp có xác suất cao nhất
predicted_class = np.argmax(predictions)
print(predicted_class)

# Danh sách tên các lớp
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Hiển thị ảnh với chất lượng cao
plt.imshow(test_images[index])
plt.title(f"Predicted class: {class_names[predicted_class]}")
plt.axis('off')
plt.show()
```

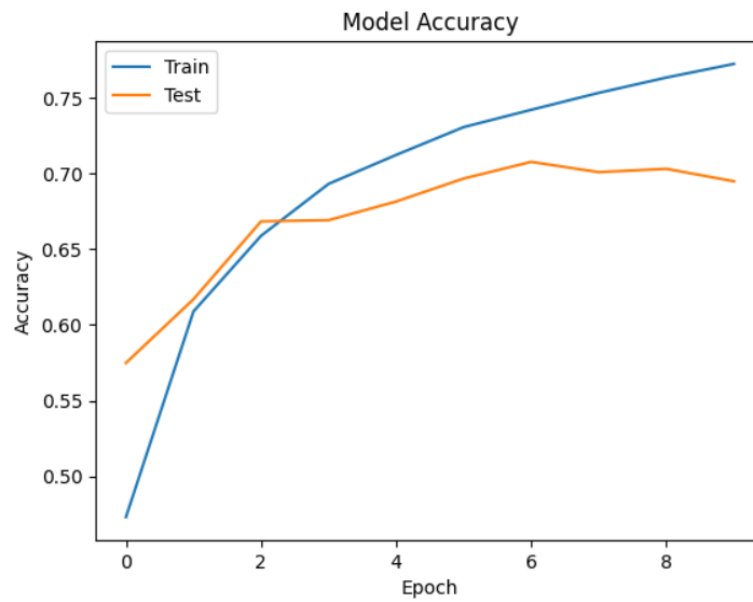
1/1 — 0s 38ms/step  
5

Predicted class: dog

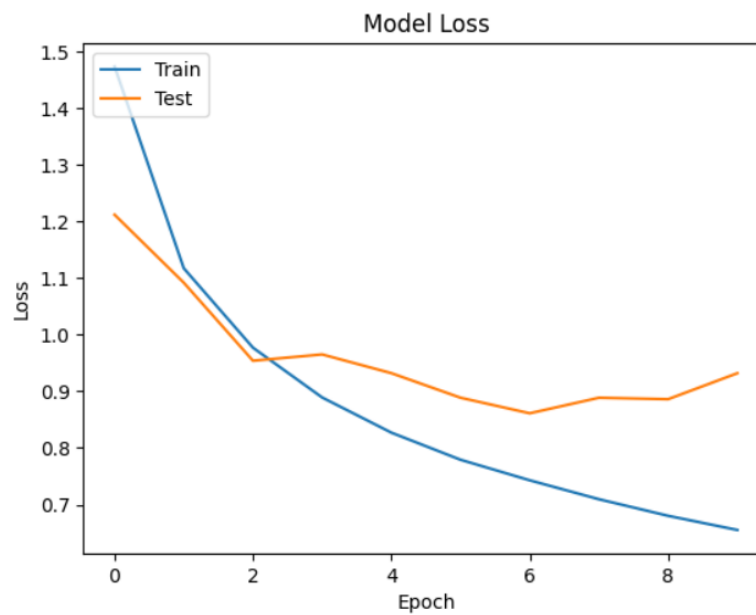


Hình 3. 9 Kết quả CNN

### 3.3.2 Nhận xét



Hình 3. 10 Accuracy khi huấn luyện bằng CNN



Hình 3. 11 Loss khi huấn luyện bằng CNN



Trong đồ thị Accuracy, độ chính xác trên tập huấn luyện (đường màu xanh) tăng liên tục và đạt giá trị cao sau mỗi epoch và độ chính xác trên tập kiểm tra (đường màu cam) tăng lên ở những epoch đầu, nhưng sau đó ổn định và dao động nhẹ (khoảng 0.7).

Trong đồ thị Loss, giá trị hàm mất mát trên tập huấn luyện (đường màu xanh) giảm đều và liên tục cho thấy mô hình học tốt trên tập này; giá trị hàm mất mát trên tập kiểm tra (đường màu cam) giảm ở những epoch đầu nhưng sau đó dao động nhẹ hoặc tăng lên một chút, phản ánh rằng mô hình bắt đầu không tổng quát hóa tốt cho dữ liệu kiểm tra.

Kết luận: Mô hình hoạt động khá tốt trên tập huấn luyện, tuy nhiên từ khoảng cách giữa Train Accuracy – Test Accuracy và Train Loss – Test Loss sau vài epoch, ta có thể nhận ra mô hình có dấu hiệu overfitting.

## TÀI LIỆU THAM KHẢO

### Tiếng Việt

1. Machine Learning cơ bản (2017), *Bài 7: Gradient Descent (phần 1/2)*

### Tiếng Anh

28. IBM, Artificial Intelligence, *What are convolutional neural networks.*
29. Kh. Nafizul Haque (2023), *What is Convolutional Neural Network – CNN (Deep Learning).*
30. cloudfactory, Key principles of Computer Vision, *Pooling.*

## **PHỤ LỤC**