

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



**NGUYỄN VŨ GIA PHƯƠNG – 52200205
MAI NGUYỄN PHƯƠNG TRANG – 52200051
LÊ CÔNG TUẤN – 52200033**

BÁO CÁO CUỐI KỲ NHẬP MÔN HỌC MÁY

Người hướng dẫn
TS. Trần Lương Quốc Đại

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**



**NGUYỄN VŨ GIA PHƯƠNG – 52200205
MAI NGUYỄN PHƯƠNG TRANG – 52200051
LÊ CÔNG TUẤN – 52200033**

BÁO CÁO CUỐI KỲ NHẬP MÔN HỌC MÁY

Người hướng dẫn
TS. Trần Lương Quốc Đại

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại Học Tôn Đức Thắng và khoa Công Nghệ Thông Tin đã đưa môn học máy vào chương trình giảng dạy. Đặc biệt là giảng viên bộ môn, thầy Trần Lương Quốc Đại, người đã truyền cảm hứng và những kiến thức bổ ích cho chúng em, người đã giải đáp những thắc mắc còn tồn đọng trong chúng em.

Môn nhập môn học máy là môn học thú vị, trường và khoa đã sắp xếp vô cùng hợp lý giữa các tiết lý thuyết và thực hành, từ những kiến thức ở lớp lý thuyết, thầy Trần Lương Quốc Đại đã giúp chúng em hiểu rõ trình tự thực thi của từng bài toán. Từ đó giúp nâng cao tư duy lập trình của chúng em, củng cố kiến thức toán học mà chúng em còn mơ hồ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài làm khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy xem xét và góp ý để bài của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 15 tháng 12 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Nguyễn Vũ Gia Phương

Mai Nguyễn Phương Trang

Lê Công Tuấn

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học 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 Dự 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 Dự 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 15 tháng 12 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Nguyễn Vũ Gia Phương

Mai Nguyễn Phương Trang

Lê Công Tuấn

TÓM TẮT

Bài làm gồm 3 chương chính:

- Chương 1: Trình bày cơ sở lý thuyết và đánh giá thực nghiệm 5 phương pháp tối ưu cost (chi phí) trong mô hình học sâu.
- Chương 2: Dự đoán giá mở cửa cổ phiếu
- Chương 3: Trình bày cơ sở lý thuyết và đánh giá thực nghiệm mô hình học sâu CNN.

MỤC LỤC

DANH MỤC BẢNG BIỂU	6
DANH MỤC HÌNH VẼ	7
DANH MỤC CÁC CHỮ VIẾT TẮT.....	8
CHƯƠNG 1. CÁC THUẬT TOÁN TỐI ƯU HÓA	9
1.1 Giới thiệu.....	9
1.2 Thuật toán Gradient Descent.....	9
1.2.1 Ý tưởng chính	9
1.2.2 Công thức tổng quát.....	9
1.2.3 Cơ chế hoạt động	11
1.2.4 Các biến thể	13
1.2.5 Ứng dụng.....	19
1.3 Thuật toán Momentum	19
1.3.1 Ý tưởng chính	19
1.3.2 Công thức toán học	20
1.3.3 Ưu điểm.....	21
1.3.4 Nhược điểm	21
1.3.5 Ứng dụng.....	21
1.4 Thuật toán Adagrad (Adaptive Gradient Algorithm).....	22
1.4.1 Ý tưởng chính	22
1.4.2 Công thức toán học	22
1.4.3 Ưu điểm.....	24
1.4.4 Nhược điểm	24

1.4.5 Ứng dụng.....	24
1.5 Thuật toán RMSProp (Root Mean Square Propagation)	25
1.5.1 Ý tưởng chính.....	25
1.5.2 Công thức toán học.....	25
1.5.3 Ưu điểm.....	27
1.5.4 Nhược điểm.....	27
1.5.5 Ứng dụng.....	27
1.6 Thuật toán Adam (Adaptive Moment Estimation)	27
1.6.1 Ý tưởng chính.....	27
1.6.2 Công thức toán học.....	28
1.6.3 Ưu điểm.....	29
1.6.4 Nhược điểm.....	29
1.6.5 Ứng dụng.....	30
CHƯƠNG 2. BÀI TOÁN: DỰ ĐOÁN GIÁ CỔ PHIẾU (GIÁ MỞ CỬA)	31
2.1 Cơ sở lý thuyết	31
2.1.1 Feedforward Neural Network (FNN)	31
2.1.2 Recurrent Neural Network (RNN)	33
2.1.3 Các thuật toán học máy khác.....	34
2.1.4 Các kỹ thuật tránh overfitting.....	36
2.2 Thực nghiệm	37
2.2.1 Dataset	37
2.2.2 Kết quả dự đoán giá Mở cửa cổ phiếu và biểu đồ huấn luyện – Đánh giá ...	39
2.2.3 So sánh các mô hình với nhau thông qua các thang đo	45

CHƯƠNG 3. CNN - CONVOLUTIONAL NEURAL NETWORK)	47
3.1 Tổng quan về CNN:	47
3.1.1 Định nghĩa:	47
3.1.2 Tham số đầu vào cho CNN:	48
3.2 Cấu trúc của CNN	49
3.2.1 Kiến trúc cơ bản:	49
3.2.2 Các lớp chính trong CNN	50
3.3 Nguyên lý hoạt động	52
3.3.1 Phép tích chập:	53
3.3.2 Hàm kích hoạt	53
3.3.3 Lớp Pooling:	53
3.3.4 Lớp kết nối đầy đủ:	53
3.3.5 Backpropagation trong CNN	53
3.4 Ưu điểm và nhược điểm	54
3.4.1 Ưu điểm:	54
3.4.2 Nhược điểm	54
3.5 Ứng dụng	55
3.5.1 Nhận Dạng Hình Ảnh:	55
3.5.2 Phân Tích Cảm Xúc từ Biểu Cảm Khuôn Mặt:	55
3.5.3 Phát Hiện và Phân Loại Vật Thể:	55
3.5.4 Phân Tích Y Học	55
3.5.5 Hệ Thống Tự Động Lái	55
3.5.6 Dịch Máy và Xử Lý Ngôn Ngữ Tự Nhiên:	56

3.5.7 Phát Hiện Gian Lận	56
3.5.8 Giáo Dục và Nghiên Cứu.....	56
3.6 Triển khai thuật toán	56
3.6.1 Dataset: MNIST	56
3.6.2 Kết quả thực hiện thuật toán:	57
TÀI LIỆU THAM KHẢO	63

DANH MỤC BẢNG BIỂU

Bảng 1.1: So sánh các biến thể của Gradient Descent	19
Bảng 2.1: Chia tập train và test	39
Bảng 2.2: Bảng đánh giá tổng quan các model.....	46

DANH MỤC HÌNH VẼ

Hình 2.1: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình FNN	39
Hình 2.2: Biểu đồ boss qua các epochs train và validation FNN model	39
Hình 2.3: Đánh giá mô hình FNN cho mã cổ phiếu META qua các thang đo	40
Hình 2.4: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình RNN	41
Hình 2.5: Biểu đồ boss qua các epochs train và validation RNN model	41
Hình 2.6: Đánh giá mô hình RNN cho mã cổ phiếu META qua các thang đo.....	42
Hình 2.7: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình SVM	43
Hình 2.8: Đánh giá mô hình SVM cho mã cổ phiếu META qua các thang đo	43
Hình 2.9: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình Random Forest	44
Hình 2.10: Đánh giá mô hình RF cho mã cổ phiếu META qua các thang đo	45
Hình 2.11: Biểu đồ các thang đo so sánh 4 model	45
Hình 3.1: Ví dụ về hoạt động của CNN	47
Hình 3.2: Ví dụ về hoạt động của Convolutional [53]	48
Hình 3.3: Ví dụ cho hoạt động theo lớp của CNN [1]	49
Hình 3.4: Ví dụ cho convolutional layer [2]	51
Hình 3.5: Ví dụ cho max pooling [55]	52
Hình 3.6: Các trường hợp dự đoán trong tập test của MNITS.....	59
Hình 3.7: Các chữ viết tay từ 0 đến 9 để test độ chính xác thực tế của CNN.....	62

DANH MỤC CÁC CHỮ VIẾT TẮT

RNN	Recurrent Neural Network
FNN	Feedforward Neural Network
SVM	Support Vector Machine
RF	Random Forest
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
R^2	R-squared
CNN	Convolutional Neural Network

CHƯƠNG 1. CÁC THUẬT TOÁN TỐI ƯU HÓA

1.1 Giới thiệu

[3] [4] [5] Trong lĩnh vực học sâu (Deep Learning), mục tiêu chính của quá trình tối ưu hóa (optimization) là giảm thiểu hàm mất mát (loss function) nhằm giúp mô hình có thể dự đoán chính xác hơn. Các thuật toán tối ưu hóa được thiết kế nhằm điều chỉnh các tham số như trọng số và bias của mô hình để tìm ra giá trị tối ưu của tham số, giúp mô hình đạt hiệu suất tốt nhất.

Các thuật toán tối ưu hóa hoạt động dựa trên cách tính gradient của hàm mất mát để xác định hướng điều chỉnh tham số. Tuy nhiên, cách tính toán và sử dụng gradient sẽ thay đổi vào theo từng thuật toán, dẫn đến tạo sự khác biệt về hiệu quả và tốc độ hội tụ.

Các thuật toán tối ưu hóa phổ biến và quan trọng trong lĩnh vực học sâu như Gradient Descent (GD), Momentum, Adagrad (Adaptive Gradient Algorithm), RMSProp (Root Mean Square Propagation), Adam (Adaptive Moment Estimation).

1.2 Thuật toán Gradient Descent

1.2.1 Ý tưởng chính

[6] Gradient Descent là một thuật toán tối ưu hóa dựa trên ý tưởng cốt lõi: đi xuống theo hướng dốc nhất của hàm mất mát để tìm cực tiểu. Trong học sâu, các tham số θ (trọng và bias) được cập nhật liên tục để giảm giá trị hàm mất mát $L(\theta)$, qua đó làm tăng khả năng dự đoán của mô hình.

1.2.2 Công thức tổng quát

- Công thức tổng quát

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

- [7] θ_t : Tham số tại thời điểm t
 - Đây là giá trị hiện tại của các tham số mô hình (như trọng số và bias).

- Mục tiêu của Gradient Descent là cập nhật θ_t để giảm giá trị của hàm mất mát $L(\theta)$.
- η : Learning rate (tốc độ học)
 - η quyết định kích thước của mỗi bước nhảy trong quá trình cập nhật tham số.
 - Nếu η quá nhỏ: Quá trình hội tụ sẽ chậm.
 - Nếu η quá lớn: Có thể vượt qua cực tiểu toàn cục hoặc dao động xung quanh cực tiểu.
 - Learning rate là một tham số quan trọng cần thiết cần được điều chỉnh cẩn thận. Các kỹ thuật như learning rate scheduler hoặc adaptive learning rate (ví dụ: trong Adagrad hoặc Adam) thường được sử dụng để tối ưu hóa η .
- $\eta \nabla L(\theta_t)$: Gradient của hàm mất mát tại θ_t
 - Gradient $\nabla L(\theta_t)$ là đạo hàm riêng của hàm mất mát $L(\theta)$.
 - Gradient biểu diễn độ dốc và hướng thay đổi của hàm mất mát tại thời điểm hiện tại.
 - Hướng của gradient âm ($-\nabla L(\theta_t)$) cho biết hướng mà hàm mất mát giảm nhanh nhất.
- **Công thức hàm mất mát tổng quát:**

[7] Hàm mất mát $L(\theta)$ đo lường mức độ sai lệch giữa giá trị dự đoán \hat{y}_i và giá trị thực tế y_i trên toàn bộ tập dữ liệu. Công thức:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L_i(\theta)$$

- N: số lượng mẫu trong tập dữ liệu
- $L_i(\theta)$: Hàm mất mát của mẫu thứ i.
- **Gradient của hàm mất mát:**
- [7] Gradient của hàm mất mát $L(\theta)$ theo tham số θ được tính bằng đạo hàm riêng (partial derivative).

Công thức:

$$\nabla L(\theta) = \frac{\partial L(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^N \frac{\partial L_i(\theta)}{\partial \theta}$$

- $\nabla L(\theta)$: Gradient tổng thể, là trung bình gradient của tất cả các mẫu trong tập dữ liệu.

- **Hàm mất mát Mean Squared Error (MSE):**

[8] [9] Hàm mất mát:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Gradient của MSE:

$$\nabla L(\theta) = -\frac{2}{N} \sum_{i=1}^N x_i (y_i - \hat{y}_i)$$

- x_i : Đầu vào của mẫu thứ i .
- y_i : Nhãn thực tế
- $\hat{y}_i = f(x_i; \theta)$: Giá trị dự đoán của mô hình.

- **Hàm mất mát Cross-Entropy:**

[10] [11] Hàm mất mát:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Gradient của MSE:

$$\nabla L(\theta) = -\frac{1}{N} \sum_{i=1}^N x_i (y_i - \hat{y}_i)$$

- x_i : Đầu vào của mẫu thứ i .
- y_i : Nhãn thực tế
- $\hat{y}_i = f(x_i; \theta)$: Giá trị dự đoán của mô hình.

1.2.3 Cơ chế hoạt động

[6] [7]

Bước 1: Khởi tạo tham số

- Giá trị khởi tạo của tham số θ (các trọng số và bias) thường được chọn ngẫu nhiên hoặc bằng một giá trị cố định.
- Ví dụ, nếu mô hình có n tham số, vector tham số ban đầu là:

$$\theta = [\theta_1, \theta_2, \dots, \theta_n]$$

Bước 2: Tính giá trị hàm mất mát

- Hàm mất mát $L(\theta)$ đo lường sự sai lệch giữa giá trị thực tế y và giá trị dự đoán \hat{y} . Một số hàm mất mát phổ biến:
 - Mean Squared Error (MSE):

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

- Cross-Entropy Loss:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Bước 3: Tính gradient của hàm mất mát

- Gradient $\nabla L(\theta)$ là vector đạo hàm riêng của hàm mất mát $L(\theta)$ theo từng tham số θ_j :

$$\nabla L(\theta) = \left[\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \dots, \frac{\partial L}{\partial \theta_n} \right]$$

- Gradient cho biết:
 - Độ dốc: Mức độ thay đổi của hàm mất mát.
 - Hướng: Gradient dương cho biết hàm mất mát tăng, gradient âm cho biết hàm mất mát giảm.

Bước 4: Cập nhật tham số

- Sử dụng công thức Gradient Descent để cập nhật tham số:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

- η : Learning rate, xác định kích thước bước nhảy trong không gian tham số.

- θ_{t+1} : Tham số mới được cập nhật.

Bước 5: Kiểm tra điều kiện hội tụ

- Sau mỗi bước cập nhật, kiểm tra xem thuật toán có hội tụ hay không:
 - Nếu sự thay đổi trong giá trị hàm mất mát $|L(\theta_{t+1}) - L(\theta_t)|$ nhỏ hơn một ngưỡng ϵ , thuật toán dừng lại.
 - Nếu đạt số lần lặp tối đa, thuật toán cũng dừng.

Bước 6: Lặp lại các bước 2 đến bước 5

- Quá trình này được lặp lại cho đến khi đạt điều kiện hội tụ hoặc đạt cực tiểu của hàm mất mát.

1.2.4 Các biến thể

Gradient Descent có ba biến thể chính, được thiết kế để cải thiện hiệu quả tính toán và tốc độ hội tụ trên các loại dữ liệu khác nhau. Chúng bao gồm Batch Gradient Descent, Stochastic Gradient Descent (SGD) và Mini-Batch Gradient Descent.

1.2.4.1 Batch Gradient Descent (BGD)

[3] [12]

Nguyên lý hoạt động:

- Batch Gradient Descent tính gradient dựa trên toàn bộ tập dữ liệu trong mỗi bước cập nhật.
- Hàm mất mát trung bình trên tất cả các mẫu trong tập dữ liệu được sử dụng để tính gradient.

Công thức toán học:

- Công thức hàm mất mát tổng quát:

Hàm mất mát $L(\theta)$ đo lường mức độ sai lệch giữa giá trị dự đoán \hat{y}_i và giá trị thực tế y_i trên toàn bộ tập dữ liệu. Công thức:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L_i(\theta)$$

- N: Số lượng mẫu trong tập dữ liệu.

- $L_i(\theta)$: Hàm mất mát của mẫu thứ i .

- *Gradient của hàm mất mát:*

Gradient của hàm mất mát $L(\theta)$ theo tham số θ được tính bằng đạo hàm riêng (partial derivative).

Công thức:

$$\nabla L(\theta) = \frac{\partial L(\theta)}{\partial \theta} = \frac{1}{N} \sum_{i=1}^N \frac{\partial L_i(\theta)}{\partial \theta}$$

- $\nabla L(\theta)$: Gradient tổng thể, là trung bình gradient của tất cả các mẫu trong tập dữ liệu

- *Hàm mất mát Mean Squared Error (MSE):*

[8] [9] Hàm mất mát:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Gradient của MSE:

$$\nabla L(\theta) = -\frac{2}{N} \sum_{i=1}^N x_i (y_i - \hat{y}_i)$$

- x_i : Đầu vào của mẫu thứ i .
- y_i : Nhãn thực tế
- $\hat{y}_i = f(x_i; \theta)$: Giá trị dự đoán của mô hình.

- *Hàm mất mát Cross-Entropy:*

[10] [11] Hàm mất mát:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Gradient của MSE:

$$\nabla L(\theta) = -\frac{1}{N} \sum_{i=1}^N x_i (y_i - \hat{y}_i)$$

- x_i : Đầu vào của mẫu thứ i .
- y_i : Nhãn thực tế

- $\hat{y}_i = f(x_i; \theta)$: Giá trị dự đoán của mô hình.
- Công thức cập nhật tham số:

Công thức cập nhật tham số của Batch Gradient Descent:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

- θ_t : Tham số tại bước t.
- η : Learning rate (tốc độ học).
- $\nabla L(\theta_t)$: Gradient của hàm mất mát tại θ_t .

Ưu điểm:

- Ổn định: Gradient được tính trên toàn bộ tập dữ liệu, đảm bảo hướng di chuyển đúng đắn.
- Dễ dàng hội tụ khi làm mất mát lồi (convex).

Nhược điểm:

- Chậm với tập dữ liệu lớn: Tính toán bộ gradient trên tập dữ liệu lớn cần nhiều tài nguyên.
- Không phù hợp với học trực tuyến (online learning).

1.2.4.2 Stochastic Gradient Descent (SGD)

[7] [13] [14]

Nguyên lý hoạt động:

- Stochastic Gradient Descent cập nhật tham số dựa trên gradient của một mẫu dữ liệu duy nhất tại mỗi bước.
- Gradient không còn là trung bình của tất cả các mẫu mà chỉ trên một điểm duy nhất.

Công thức toán học:

- Công thức hàm mất mát tổng quát:

Hàm mất mát tổng quát (tương tự như Batch Gradient Descent):

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L_i(\theta)$$

- N: Số lượng mẫu trong tập dữ liệu.

- $L_i(\theta)$: Hàm mất mát của mẫu thứ i .
- Tuy nhiên, trong SGD, chúng ta không tính gradient trên toàn bộ tập dữ liệu mà chỉ dựa trên một mẫu duy nhất tại mỗi bước.
- *Gradient của Stochastic Gradient Descent:*
Gradient của hàm mất mát trên một mẫu dữ liệu i được tính như sau:
$$\nabla L(\theta) = \nabla L_i(\theta)$$
 - $\nabla L(\theta)$: Hàm mất mát của mẫu dữ liệu thứ i .
 - $\nabla L_i(\theta)$: Gradient của hàm mất mát dựa trên mẫu i .
 - Gradient này là một xấp xỉ của gradient tổng thể (dựa trên toàn bộ dữ liệu), vì chỉ sử dụng thông tin từ một mẫu duy nhất.
- *Công thức cập nhật tham số:*
Công thức cập nhật tham số:
$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$
 - θ_t : Tham số tại bước t .
 - η : Learning rate (tốc độ học).
 - $\nabla L(\theta_t)$: Gradient của hàm mất mát tại θ_t .

Ưu điểm:

- Hiệu quả: Mỗi bước chỉ tính gradient trên một mẫu, giảm yêu cầu bộ nhớ.
- Phù hợp với dữ liệu lớn: SGD xử lý từng mẫu một, tránh tính toán trên toàn bộ tập dữ liệu.

Nhược điểm:

- Dao động mạnh: Gradient tính trên một mẫu không ổn định, dẫn đến việc mô hình dao động quanh cực tiểu.
- Cần giảm tốc độ học (η) theo thời gian để tăng độ ổn định.

1.2.4.3 Mini-Batch Gradient Descent

[15] [16]

Nguyên lý hoạt động:

- Mini-Batch Gradient Descent tính gradient trên một nhóm nhỏ dữ liệu (mini-batch) thay vì toàn bộ dữ liệu hoặc một mẫu duy nhất.
- Mini-Batch Gradient Descent là sự kết hợp giữa Batch Gradient Descent và Stochastic Gradient Descent, tận dụng ưu điểm của cả hai.

Công thức toán học:

- *Công thức hàm mất mát trung bình trên một mini-batch:*

Hàm mất mát trên mini-batch B có kích thước $|B|$:

$$L_B(\theta) = \frac{1}{|B|} \sum_{i \in B} L_i(\theta)$$

- B: Mini-batch (tập hợp các chỉ số của các mẫu dữ liệu trong nhóm).
- $L_i(\theta)$: Hàm mất mát của mẫu thứ i.
- $|B|$: Kích thước của mini-batch.
- *Gradient trên mini-batch:*

Gradient của hàm mất mát trên mini-batch B được tính như sau:

$$\nabla L_B(\theta) = \frac{\partial L_B(\theta)}{\partial \theta} = \frac{1}{|B|} \sum_{i \in B} \nabla L_i(\theta)$$

- $\nabla L_B(\theta)$: Trung bình gradient của tất cả các mẫu trong mini-batch B.
 - Gradient này là một xấp xỉ của gradient toàn bộ dữ liệu, nhưng ổn định hơn so với SGD.
 - *Công thức cập nhật tham số:*
- Công thức cập nhật tham số:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

- θ_t : Tham số tại bước t.
- η : Learning rate (tốc độ học).
- $\nabla L(\theta_t)$: Gradient của hàm mất mát tại θ_t .

Ưu điểm:

- Cân bằng giữa tốc độ và độ ổn định:
 - Nhanh hơn Batch Gradient Descent nhờ tính toán trên mini-batch.
 - Ổn định hơn Stochastic Gradient Descent do sử dụng trung bình gradient của nhiều mẫu.
- Tận dụng tốt GPU: Mini-Batch phù hợp với tính toán song song trên GPU.

Nhược điểm:

- Cần chọn kích thước mini-batch hợp lý:
 - Mini-Batch nhỏ: Gần giống Stochastic Gradient Descent, dao động nhiều.
 - Mini-Batch lớn: Gần giống Batch Gradient Descent, tốn tài nguyên.

1.2.4.4 Áp dụng các biến thể

[16] [15] [7] [14] [13]

Batch Gradient Descent:

- Phù hợp khi tập dữ liệu nhỏ và có thể xử lý toàn bộ trong một lần.
- Dùng để kiểm tra độ ổn định của mô hình trước khi áp dụng Stochastic Gradient Descent hoặc Mini-Batch Gradient Descent.

Stochastic Gradient Descent:

- Phù hợp khi làm việc với dữ liệu rất lớn hoặc dữ liệu đến theo dòng (streaming data).
- Sử dụng khi bạn có tài nguyên hạn chế hoặc muốn huấn luyện nhanh.

Mini-Batch Gradient Descent:

- Là lựa chọn phổ biến nhất trong học sâu.
- Phù hợp khi cần tối ưu hiệu năng tính toán và tốc độ hội tụ.
- Mini-Batch với kích thước 32, 64 hoặc 128 là các lựa chọn phổ biến.

1.2.4.5 So sánh các biến thể

[16] [15] [7] [14] [13]

Biến thể	Gradient được tính trên	Ưu điểm	Nhược điểm
Batch Gradient Descent	Toàn bộ tập dữ liệu	Ổn định, hội tụ chính xác	Chậm, tốn nhiều bộ nhớ với dữ liệu lớn
Stochastic Gradient Descent (SGD)	Một mẫu dữ liệu	Nhanh hơn Batch Gradient Descent, hiệu quả với dữ liệu lớn	Dao động mạnh, cần giảm learning rate để hội tụ ổn định
Mini-Batch Gradient Descent (SGD)	Một nhóm nhỏ (mini-batch)	Cân bằng giữa tốc độ và độ ổn định, tận dụng tốt GPU	Cần chọn kích thước mini-batch phù hợp

Bảng 1.1: So sánh các biến thể của Gradient Descent

1.2.5 Ứng dụng

[17] Gradient Descent được sử dụng trong:

- Học máy: Tối ưu trọng số trong hồi quy logistic, hồi quy tuyến tính.
- Học sâu: Tối ưu tham số trong mạng CNN, RNN, Transformer.
- Kỹ thuật và tài chính: Tối ưu hóa các hàm chi phí phi tuyến.

1.3 Thuật toán Momentum

1.3.1 Ý tưởng chính

[18] [19] [20] **Momentum** là một kỹ thuật tối ưu hóa được thêm vào Gradient Descent để tăng tốc quá trình hội tụ và giảm dao động trong quá trình cập nhật tham số. Momentum sử dụng ý tưởng tương tự trong vật lý học: duy trì quán tính để tiếp tục di chuyển theo hướng đã có, ngay cả khi gradient nhỏ hoặc dao động.

Trong Gradient Descent thông thường, mỗi bước di chuyển phụ thuộc hoàn toàn vào gradient tại điểm hiện tại:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

Điều này dẫn đến dao động khi gradient thay đổi nhanh giữa các bước lặp.

Momentum bổ sung một thành phần quán tính để giúp thuật toán di chuyển mượt mà hơn và nhanh hơn:

- **Thành phần quán tính:** Tích lũy động lượng từ các gradient trước đó.
- **Hướng dẫn:** Tiếp tục di chuyển theo hướng tích lũy, ngay cả khi gradient hiện tại nhỏ

1.3.2 Công thức toán học

[21] [18] Momentum thêm một biến v_t , đại diện cho vận tốc, để lưu trữ động lượng tích lũy:

1.3.2.1 Vận tốc v_t

$$v_t = \gamma v_{t-1} - \eta \nabla L(\theta_t)$$

- v_t : Vận tốc tại bước t .
- γ : Hệ số momentum (quán tính), thường nằm trong khoảng $0.9 \leq \gamma < 1$.
- η : Learning rate (tốc độ học).
- $\nabla L(\theta_t)$: Gradient của hàm mất mát tại bước t .
- $\eta \nabla L(\theta_t)$: Đây là thành phần từ Gradient Descent thông thường, biểu diễn hướng giảm giá trị hàm mất mát tại bước hiện tại.
- γv_{t-1} : Thành phần này tích lũy gradient từ các bước trước, tạo ra quán tính để tiếp tục di chuyển theo hướng tích lũy.

1.3.2.2 Cập nhật tham số θ

$$\theta_{t+1} = \theta_t + v_t$$

- v_t : Là sự kết hợp giữa quán tính và gradient hiện tại, giúp giảm dao động và tăng tốc hội tụ.

- Thay vì di chuyển trực tiếp theo gradient, tham số θ được cập nhật dựa trên vận tốc v_t .

1.3.3 Ưu điểm

[18] [22] Giảm dao động:

- Trong Gradient Descent thông thường, các bước nhảy có thể dao động quanh cực tiểu (đặc biệt trong các hố dốc hẹp).
- Momentum giảm dao động bằng cách duy trì quán tính theo hướng di chuyển.

Tăng tốc hội tụ:

- Momentum giúp tăng tốc trong các vùng mà gradient nhỏ nhưng hướng di chuyển vẫn còn phù hợp (ví dụ: khi tiến đến đáy của hàm mất mát).

Ổn định hơn trên các hàm mất mát phức tạp:

- Với các hàm mất mát có nhiều hố và cực tiểu cục bộ, Momentum giúp mô hình tránh bị kẹt ở các điểm đó.

1.3.4 Nhược điểm

[18] [22] Phụ thuộc vào hệ số γ :

- Nếu γ quá nhỏ, Momentum không tận dụng được quán tính.
- Nếu γ quá lớn, mô hình có thể bị dao động.

Không tự động điều chỉnh learning rate:

- Momentum chỉ giúp giảm dao động, nhưng không giải quyết được vấn đề learning rate không phù hợp.

Không tối ưu với tất cả các bài toán:

- Một số bài toán phức tạp yêu cầu các kỹ thuật như RMSProp, Adam (kết hợp Momentum với các yếu tố khác).

1.3.5 Ứng dụng

[20] [23] Học sâu:

- Momentum là một thành phần quan trọng trong nhiều thuật toán tối ưu như RMSProp và Adam.
- Thường được sử dụng trong các bài toán lớn như huấn luyện mạng CNN hoặc RNN.

Giảm hiệu ứng gradient vanish:

- Momentum giúp giảm tốc độ "mất dần gradient" trong các mạng sâu (Deep Neural Networks).

Giúp vượt qua các cực tiểu cục bộ:

- Với các hàm mất mát không lồi, Momentum giúp thuật toán thoát khỏi các điểm dừng cục bộ.

1.4 Thuật toán Adagrad (Adaptive Gradient Algorithm)

1.4.1 Ý tưởng chính

[24] [25] [26] Trong Gradient Descent thông thường, tất cả các tham số θ đều được cập nhật với cùng một giá trị learning rate η . Điều này không tối ưu nếu gradient của các tham số thay đổi rất khác nhau.

Adagrad khắc phục điều này bằng cách:

- Tự động điều chỉnh learning rate cho mỗi tham số θ_i , dựa trên lịch sử của các gradient trước đó.
- Điều chỉnh learning rate dựa trên tổng bình phương gradient tích lũy, làm giảm learning rate khi gradient lớn và giữ learning rate cao hơn khi gradient nhỏ.

1.4.2 Công thức toán học

1.4.2.1 Gradient thông thường

Gradient Descent cập nhật tham số θ :

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

1.4.2.2 Công thức của Adagrad

[24] [25] [26]

Tích lũy bình phương gradient

Adagrad sử dụng một vector G_t , tích lũy bình phương của tất cả gradient trước đó:

$$G_t = G_{t-1} + \nabla L(\theta_t)^2$$

- G_t : Vector tổng bình phương gradient tại bước t .
- $\nabla L(\theta_t)$: Gradient tại bước t .
- G_t có kích thước bằng số tham số θ .

Cập nhật tham số với learning rate được điều chỉnh:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla L(\theta_t)$$

- $\frac{\eta}{\sqrt{G_t + \epsilon}}$: Learning rate được điều chỉnh dựa trên G_t .
- ϵ : Một số nhỏ (thường là 10^{-8}) để tránh chia cho 0.

Learning rate được điều chỉnh:

Learning rate cho mỗi tham số θ_i :

$$\eta_i = - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}}$$

- $G_{t,i}$: Phần tử thứ i của G_t , đại diện cho tổng bình phương gradient của θ_i .

Phân tích cơ chế

- Tích lũy gradient:
 - Các tham số có gradient lớn sẽ làm tăng nhanh G_t , dẫn đến giảm learning rate cho tham số đó.
 - Các tham số có gradient nhỏ sẽ duy trì learning rate cao hơn, giúp tối ưu hóa tốt hơn.
- Điều chỉnh learning rate theo từng tham số:
 - Mỗi tham số θ_i có một learning rate riêng, thay đổi theo gradient của chính nó.

- Hiệu ứng tự động giảm learning rate:
 - Khi số bước lặp tăng, G_t tăng, làm giảm η_i . Điều này phù hợp với các bài toán có gradient giảm dần theo thời gian.

1.4.3 Ưu điểm

[24] [27] [28] Learning rate thích nghi:

- Tự động giảm learning rate cho các tham số có gradient lớn.
- Giữ learning rate cao hơn cho các tham số có gradient nhỏ, tăng hiệu quả tối ưu hóa.

Không cần giảm learning rate thủ công:

- Với Gradient Descent, chúng ta cần giảm learning rate thủ công theo thời gian. Adagrad tự động xử lý điều này.

Hiệu quả cho dữ liệu thưa (Sparse Data):

- Trong các bài toán xử lý dữ liệu thưa, như xử lý ngôn ngữ tự nhiên (NLP) hoặc hệ thống gợi ý, Adagrad hiệu quả nhờ tối ưu hóa tốt các tham số hiếm khi được cập nhật.

1.4.4 Nhược điểm

[24] [27] [28] Tích lũy gradient quá mức:

- Khi G_t tăng lớn, learning rate giảm dần về gần 0. Điều này có thể làm thuật toán dừng trước khi đạt cực tiểu.

Không thích hợp với bài toán dài hạn:

- Trong các bài toán yêu cầu tối ưu hóa lâu dài (như mạng sâu), Adagrad có thể gặp khó khăn vì learning rate giảm quá nhanh.

1.4.5 Ứng dụng

[24] [28] [29] Adagrad thường được sử dụng trong các bài toán:

- Xử lý ngôn ngữ tự nhiên (NLP): Phân loại văn bản, Word2Vec, GloVe.
- Hệ thống gợi ý: Học tham số của hệ thống gợi ý với dữ liệu người dùng lớn và thưa.

- Xử lý dữ liệu lớn và phân tán: Khi làm việc với các tập dữ liệu lớn và cần hiệu quả tính toán.

1.5 Thuật toán RMSProp (Root Mean Square Propagation)

1.5.1 Ý tưởng chính

[30] [31] [32] RMSProp điều chỉnh learning rate cho từng tham số bằng cách:

- Theo dõi gradient của từng tham số: Sử dụng trung bình lũy thừa mũ của bình phương gradient để làm mịn (smooth) tác động của các gradient lớn.
- Điều chỉnh learning rate: Dựa trên giá trị trung bình gradient, RMSProp duy trì một learning rate ổn định bằng cách giảm ảnh hưởng của các gradient lớn.

1.5.2 Công thức toán học

[30] [31] [32] [33]

1.5.2.1 Gradient thông thường trong Gradient Descent

Công thức cập nhật tham số trong Gradient Descent:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

1.5.2.2 Cập nhật trong RMSProp

Tích lũy trung bình lũy thừa mũ của bình phương gradient:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) \nabla L(\theta_t)^2$$

- $E[g^2]_t$: Trung bình lũy thừa mũ của bình phương gradient tại bước t.
- β : Hệ số làm mịn (decay rate), thường được đặt khoảng 0.9.
- $\nabla L(\theta_t)$: Gradient của hàm mất mát tại bước t.

Điều chỉnh learning rate:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla L(\theta_t)$$

- η : Learning rate.

- ϵ : Một số nhỏ (thường là 10^{-8}) để tránh chia cho 0.

[33]

Ý nghĩa thành phần:

- Trung bình lũy thừa mũ của bình phương gradient $E[g^2]_t$:
 - Theo dõi lịch sử của gradient, nhấn mạnh các gradient gần đây hơn và giảm ảnh hưởng của các gradient cũ hơn.
 - Gradient lớn được làm mịn, giúp giảm dao động và ổn định cập nhật tham số.
- Điều chỉnh learning rate ($\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}$):
 - Khi $E[g^2]_t$ lớn (gradient lớn), learning rate giảm, tránh bước nhảy quá lớn.
 - Khi $E[g^2]_t$ nhỏ, learning rate tăng, giúp tham số thay đổi hiệu quả hơn.

[24] [28]

Phân tích cơ chế:

- Hạn chế của Adagrad:
 - Adagrad tích lũy toàn bộ bình phương gradient từ đầu đến bước hiện tại:

$$G_t = G_{t-1} + \nabla L(\theta_t)^2$$

- Điều này khiến G_t tăng không giới hạn, làm learning rate giảm quá nhanh và hội tụ chậm.
- Giải pháp của RMSProp:
 - RMSProp thay thế tích lũy tổng G_t bằng trung bình lũy thừa mũ $E[g^2]_t$, giúp:
 - Làm mịn tác động của gradient lớn.
 - Duy trì learning rate ổn định theo thời gian.

1.5.3 Ưu điểm

[34] [35] Duy trì learning rate ổn định: RMSProp ngăn chặn learning rate giảm quá nhanh như trong Adagrad.

Hiệu quả với gradient thay đổi mạnh: Phù hợp với các bài toán mà gradient biến đổi lớn hoặc không đồng đều giữa các tham số.

Phù hợp với học sâu: RMSProp hoạt động hiệu quả trong các mạng học sâu như RNN và CNN.

1.5.4 Nhược điểm

[34] [35] Phụ thuộc vào tham số β : Giá trị β cần được chọn cẩn thận để đảm bảo sự cân bằng giữa gradient hiện tại và gradient lịch sử.

Không tích hợp quán tính: RMSProp không tích hợp quán tính (momentum) để tăng tốc hội tụ.

1.5.5 Ứng dụng

[31] [36] Học sâu (Deep Learning): RMSProp được sử dụng phổ biến trong RNN, LSTM và CNN.

1.6 Thuật toán Adam (Adaptive Moment Estimation)

1.6.1 Ý tưởng chính

[37] [3] [38] **Adam** là một thuật toán tối ưu hóa phổ biến trong học sâu (Deep Learning), kết hợp các ưu điểm của **Momentum** và **RMSProp**. Adam tự động điều chỉnh learning rate cho từng tham số, đồng thời sử dụng quán tính để tăng tốc hội tụ và giảm dao động.

- Momentum: Sử dụng trung bình lũy thừa mũ (Exponential Moving Average) để tích lũy gradient, giúp giảm dao động và tăng tốc hội tụ.
- RMSProp: Tính trung bình lũy thừa mũ của bình phương gradient, giúp điều chỉnh learning rate cho từng tham số.

Điểm mạnh của Adam:

- Tự động điều chỉnh learning rate theo từng tham số.

- Giảm dao động và tăng tốc hội tụ trong các hàm mất mát không lồi (non-convex).

1.6.2 Công thức toán học

1.6.2.1 Các bước chính trong Adam

[39] [40]

Trung bình lũy thừa mũ của gradient (Momentum):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L(\theta_t)$$

- m_t : Giá trị trung bình lũy thừa mũ (ước tính momentum) tại bước t .
- β_1 : Hệ số quán tính (thường là 0.9).
- $\nabla L(\theta_t)$: Gradient tại bước t .

Trung bình lũy thừa mũ của bình phương gradient (RMSProp):

$$v_t = \beta_2 v_t + (1 - \beta_2) \nabla L(\theta_t)^2$$

- v_t : Trung bình lũy thừa mũ của bình phương gradient tại bước t .
- β_2 : Hệ số làm mịn (thường là 0.999).

Hiệu chỉnh xu hướng lệch (Bias Correction):

m_t và v_t bị lệch do khởi tạo ban đầu. Để điều chỉnh:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- \hat{m}_t : Giá trị trung bình gradient đã hiệu chỉnh.
- \hat{v}_t : Giá trị trung bình bình phương gradient đã hiệu chỉnh.

Cập nhật tham số:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- η : Learning rate.
- ϵ : Số nhỏ (thường là 10^{-8}) để tránh chia cho 0.

1.6.2.2 Phân tích từng thành phần

[39] [40]

Momentum (trung bình gradient):

- m_t lưu giữ thông tin về hướng gradient trong các bước trước đó.
- Giúp giảm dao động khi gradient thay đổi nhanh.

RMSProp (trung bình bình phương gradient):

- v_t làm mịn gradient lớn, giúp điều chỉnh learning rate cho từng tham số.
- Ngăn các gradient lớn gây ra bước nhảy quá lớn.

Hiệu chỉnh xu hướng lệch (Bias Correction):

- m_t và v_t ban đầu bị lệch về 0 do khởi tạo. Hiệu chỉnh này giúp Adam khởi động ổn định hơn.

Cập nhật tham số:

- Cân bằng giữa tốc độ hội tụ (nhờ Momentum) và độ ổn định (nhờ RMSProp).

1.6.3 Ưu điểm

[38] [41] Hiệu quả với gradient không đồng nhất: Adam điều chỉnh learning rate theo từng tham số, phù hợp với các bài toán mà gradient có độ lớn khác nhau.

Hội tụ nhanh: Kết hợp Momentum để tăng tốc hội tụ, đặc biệt trong các bài toán phức tạp.

Tự động điều chỉnh learning rate: Không cần giảm learning rate thủ công, Adam tự điều chỉnh dựa trên gradient.

Phù hợp với học sâu: Adam hoạt động tốt với các mô hình học sâu như CNN, RNN, và Transformer.

1.6.4 Nhược điểm

[38] [41] Phụ thuộc vào hyperparameters: Các giá trị β_1 , β_2 , η cần được chọn phù hợp với từng bài toán.

Có thể không đạt tối ưu toàn cục: Adam đôi khi hội tụ đến cực tiểu cục bộ thay vì cực tiểu toàn cục, đặc biệt với các bài toán không lồi.

1.6.5 Ứng dụng

[37] Học sâu:

- Adam là thuật toán mặc định trong nhiều framework học sâu như TensorFlow, PyTorch.
- Phù hợp với mạng CNN, RNN, và Transformer.

Xử lý dữ liệu lớn: Tối ưu hoá hiệu quả trên dữ liệu lớn và phức tạp.

Bài toán không lồi: Hoạt động tốt trên các hàm mất mát không lồi.

CHƯƠNG 2. BÀI TOÁN: DỰ ĐOÁN GIÁ CỔ PHIẾU (GIÁ MỞ CỬA)

2.1 Cơ sở lý thuyết

2.1.1 Feedforward Neural Network (FNN)

Mạng nơ-ron truyền thẳng (FNN) là một loại mạng nơ-ron nhân tạo, trong đó dữ liệu di chuyển theo một hướng duy nhất từ lớp đầu vào, qua các lớp ẩn, đến lớp đầu ra mà không có vòng lặp hoặc kết nối ngược [42]. Cấu trúc cơ bản của nó bao gồm 3 lớp:

- Lớp đầu vào (Input Layer): Nhận dữ liệu đầu vào dạng vector $X = [x_1, x_2, \dots, x_n]$, trong đó, mỗi phần tử x_i là một đặc trưng dữ liệu
- Lớp ẩn (Hidden Layers): Mỗi nơ-ron trong lớp ẩn thực hiện hai phép toán:

- Tổng trọng số (Weighted Sum):

$$z_j^{(l)} = \sum_{i=1}^n w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)} \quad [43]$$

- $z_j^{(l)}$: Tổng trọng số của nơ-ron j tại lớp l
- $w_{ij}^{(l)}$: Trọng số kết nối giữa nơ-ron i tại lớp $l - 1$ và nơ-ron j tại lớp l
- $a_i^{(l-1)}$: Đầu ra của nơ-ron i từ lớp trước đó.
- $b_j^{(l)}$: Bias của nơ-ron j tại lớp l

- Hàm kích hoạt (Activation Function): $a_i^{(l)} = f(z_j^{(l)})$ [43]

- $f(z)$: Hàm kích hoạt phi tuyến như sigmoid, tanh, hoặc ReLU.

- Lớp đầu ra (Output Layer): Cung cấp kết quả cuối cùng \hat{y} :

$$\hat{y} = f_{\text{output}}(W^{(L)} a^{(L-1)} + b^{(L)}) \quad [41]$$

- f_{output} : Hàm kích hoạt tại lớp đầu ra, thường là: Softmax (phân loại đa lớp) hoặc Linear (hồi quy).

Mỗi nơ-ron trong mạng tính toán tổng có trọng số của các đầu vào, áp dụng một hàm kích hoạt (activation function) để tạo ra đầu ra. Các hàm kích hoạt phổ biến bao gồm:

- Hàm sigmoid: Đưa đầu ra vào khoảng $(0, 1)$, thường được sử dụng trong các bài toán phân loại nhị phân.
- Hàm tanh (hyperbolic tangent): Đưa đầu ra vào khoảng $(-1, 1)$, giúp trung hòa dữ liệu.
- Hàm ReLU (Rectified Linear Unit): Đưa đầu ra là 0 nếu đầu vào âm, và giữ nguyên giá trị nếu đầu vào dương; thường được sử dụng trong các mạng sâu do khả năng giảm thiểu vấn đề biến mất gradient.

Quá trình học của FNN thường sử dụng thuật toán lan truyền ngược (backpropagation) kết hợp với phương pháp tối ưu hóa như gradient descent. Các bước chính bao gồm:

- Lan truyền tiến (Feedforward): Dữ liệu đầu vào được truyền qua mạng để tạo ra đầu ra dự đoán.
- Tính toán lỗi: So sánh đầu ra dự đoán với giá trị thực tế để xác định lỗi (thường sử dụng hàm mất mát như mean squared error hoặc cross-entropy).
- Lan truyền ngược (Backpropagation): Tính toán gradient của hàm mất mát đối với các trọng số trong mạng, bắt đầu từ lớp đầu ra ngược về lớp đầu vào.
- Cập nhật trọng số: Điều chỉnh các trọng số bằng cách sử dụng gradient descent hoặc các biến thể của nó để giảm thiểu lỗi.

Lý thuyết xấp xỉ phổ quát: Một kết quả quan trọng trong lý thuyết mạng nơ-ron là định lý xấp xỉ phổ quát, khẳng định rằng một mạng nơ-ron truyền thẳng với ít nhất một lớp ẩn và sử dụng các hàm kích hoạt phi tuyến có thể xấp xỉ bất kỳ hàm liên tục nào với độ chính xác tùy ý, miễn là mạng có đủ số lượng nơ-ron trong lớp ẩn. [44]

Ứng dụng của FNN: Phân loại hình ảnh, văn bản, âm thanh; Dự báo giá trị liên tục như dự báo thời tiết, giá cổ phiếu; Nhận dạng chữ viết tay, khuôn mặt, giọng nói.

2.1.2 Recurrent Neural Network (RNN)

Mạng Nơ-ron Hồi quy (RNN) là một loại mạng nơ-ron nhân tạo được thiết kế để xử lý dữ liệu tuần tự, nơi các kết nối giữa các nơ-ron tạo thành đồ thị có hướng theo trình tự thời gian. Điều này cho phép RNN lưu giữ thông tin từ các bước thời gian trước đó, giúp mô hình hóa các phụ thuộc tạm thời trong dữ liệu. [45]

Cấu trúc và Hoạt động của RNN:

- **Trạng thái ẩn (Hidden State):** Tại mỗi bước thời gian t , RNN duy trì một trạng thái ẩn là h_t , được coi như bộ nhớ của mạng, lưu trữ thông tin về các đầu vào trước đó. [46]
- **Cập nhật trạng thái ẩn:** Trạng thái ẩn h_t được tính dựa trên đầu vào hiện tại x_t và trạng thái ẩn trước đó h_{t-1} : $h_t = \Phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$.

Trong đó:

- W_{xh} : Trọng số kết nối từ đầu vào đến trạng thái ẩn.
- W_{hh} : Trọng số kết nối từ trạng thái ẩn trước đó đến trạng thái ẩn hiện tại.
- b_h : Hệ số điều chỉnh (bias) của trạng thái ẩn.
- Φ : Hàm kích hoạt, thường là hàm tanh hoặc ReLU.
- **Đầu ra:** Dựa trên trạng thái ẩn hiện tại h_t , RNN tạo output \hat{y} :
 $\hat{y} = \psi(W_{hy}h_t + b_y)$. Trong đó:
 - W_{hy} : Trọng số kết nối từ trạng thái ẩn đến đầu ra.
 - b_y : Hệ số điều chỉnh (bias) của đầu ra.
 - ψ : Hàm kích hoạt phù hợp với nhiệm vụ, chẳng hạn như hàm softmax cho phân loại.

Lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT): Để huấn luyện RNN, thuật toán lan truyền ngược qua thời gian được sử dụng để tính toán gradient của hàm mất mát đối với các trọng số, bằng cách chia các bước thời gian và áp dụng lan truyền ngược tiêu chuẩn. [47]

Ứng dụng: RNN được ứng dụng khá rộng trong các lĩnh vực như xử lý ngôn ngữ tự nhiên (NLP), dự đoán chuỗi thời gian (bao gồm cả giá cổ phiếu) nơi dữ liệu có tính tuần tự (time series).

Ngoài ra còn có các biến thể của RNN như LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Unit).

2.1.3 Các thuật toán học máy khác

2.1.3.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) là một thuật toán học máy có giám sát, được sử dụng cho cả phân loại và hồi quy. Mục tiêu chính của SVM là tìm ra một siêu phẳng (hyperplane) tối ưu để phân tách các lớp dữ liệu với khoảng cách biên lớn nhất giữa các điểm dữ liệu gần nhất của các lớp khác nhau. [48]

SVM gồm các thành phần như: Siêu phẳng hyperplane, khoảng cách biên margin, hàm mục tiêu, hàm mất mát và kernel.

Siêu phẳng phân tách (Separating Hyperplane): Trong không gian nhiều chiều, siêu phẳng là một không gian con có chiều thấp hơn một đơn vị so với không gian ban đầu, được sử dụng để phân tách các điểm dữ liệu thuộc các lớp khác nhau. Phương trình của siêu phẳng được biểu diễn dưới dạng: $w^T x + b = 0$ [48] trong đó w là vector trọng số, x là vector đầu vào và b là hệ số điều chỉnh (bias).

Khoảng cách biên (Margin): Khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất từ mỗi lớp được gọi là margin. SVM tìm cách tối đa hóa margin này để đảm bảo phân tách rõ ràng giữa các lớp, giúp tăng khả năng tổng quát hóa của mô hình.

Hàm mục tiêu: Bài toán tối ưu trong SVM là tìm w và b sao cho margin được tối đa hóa, đồng thời đảm bảo rằng tất cả các điểm dữ liệu được phân loại chính xác: $\min_{w,b} \frac{1}{2} \|w\|^2$ [48] với ràng buộc $y_i(w^T x_i + b) \geq 1 \forall i$ [48]. y_i là nhãn (output) của điểm dữ liệu x_i .

Hàm mất mát để xử lý các trường hợp dữ liệu không thể phân tách hoàn toàn, SVM sử dụng các biến trượt ξ_i và hàm mất mát hinge: $\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$ [48]

với ràng buộc $y_i(w^T + b) \geq 1 - \xi_i \quad \forall i$ [48]. C là tham số điều chỉnh giữa độ phức tạp của mô hình và số lượng lỗi cho phép.

Kernel để xử lý các bài toán phân loại phi tuyến, SVM sử dụng kỹ thuật kernel trick, cho phép ánh xạ dữ liệu từ không gian đầu vào sang không gian đặc trưng cao hơn, nơi dữ liệu có thể được phân tách tuyến tính. Một số hàm kernel phổ biến:

- Kernel tuyến tính (Linear Kernel):

$$K(x_i, y_i) = x_i^T x_j \text{ [48]}$$

- Kernel Gaussian (RBF Kernel):

$$K(x_i, y_i) = \exp\left(-\gamma \|x_i - y_j\|^2\right) \text{ [48]}$$

- Kernel đa thức (Polynomial Kernel):

$$K(x_i, y_i) = (\alpha x_i^T x_j + c)^d \text{ [48]}$$

Ứng dụng: SVM được áp dụng rộng rãi trong nhiều lĩnh vực, bao gồm: Phân loại văn bản và hình ảnh. Nhận dạng chữ viết tay; Phân tích chuỗi thời gian; Phân loại gen trong sinh học. Với khả năng tìm kiếm siêu phẳng phân tách tối ưu và sử dụng các hàm kernel để xử lý dữ liệu phi tuyến, SVM là một công cụ mạnh mẽ trong học máy, đặc biệt hiệu quả trong các bài toán phân loại với số lượng mẫu hạn chế [48]

2.1.3.2 Random Forest (RF)

Random Forest là một thuật toán học máy thuộc nhóm ensemble learning, được sử dụng cho cả phân loại và hồi quy. Thuật toán này kết hợp nhiều cây quyết định (decision trees) để cải thiện độ chính xác và khả năng tổng quát hóa của mô hình [49]

Tập hợp các cây quyết định (Ensemble of Decision Trees): Random Forest xây dựng nhiều cây quyết định độc lập, mỗi cây được huấn luyện trên một tập con dữ liệu khác nhau và một tập con các thuộc tính được chọn ngẫu nhiên. [49]

Kỹ thuật Bootstrap Aggregating (Bagging): Mỗi cây trong rừng được huấn luyện trên một tập dữ liệu được tạo ra bằng phương pháp bootstrap sampling, tức là chọn ngẫu nhiên với hoàn lại từ tập dữ liệu gốc. [49]

Chọn ngẫu nhiên các thuộc tính (Random Feature Selection): Tại mỗi nút phân chia trong cây, một tập con ngẫu nhiên của các thuộc tính được chọn để tìm thuộc tính tốt nhất cho việc phân chia, nhằm giảm tương quan giữa các cây và tăng tính đa dạng của mô hình. [49]

Quyết định dựa trên số đông (Majority Voting) hoặc trung bình (Averaging):

- Đối với bài toán phân loại, Random Forest sử dụng phương pháp bỏ phiếu đa số từ các cây để quyết định nhãn cuối cùng.
- Đối với bài toán hồi quy, kết quả dự đoán được tính bằng trung bình các dự đoán từ các cây. [49]

Ứng dụng của Random Forest: Phân loại văn bản và hình ảnh; Dự báo tài chính và phân tích rủi ro; Phân tích gen và sinh học phân tử; Hệ thống gợi ý và cá nhân hóa.

2.1.4 Các kỹ thuật tránh overfitting

Các kỹ thuật tránh overfitting mà nhóm đã sử dụng: bao gồm Early Stopping và Dropout.

Early Stopping dừng quá trình huấn luyện khi lỗi trên tập kiểm tra (Validation Loss) ngừng cải thiện trong một số epoch liên tiếp, thường được gọi là patience. Quá trình huấn luyện dừng lại nếu:

$$L_{val}^{(t+k)} > L_{val}^{(t)} \quad \forall k \in [1, \text{patience}] \quad [50]$$

với:

- L_{val} : Hàm mất mát trên tập kiểm tra.
- t : Epoch hiện tại.
- $t + k$: Các epoch sau đó trong phạm vi patience.

Dropout hoạt động bằng cách bỏ qua một tập con các nơ-ron trong mỗi bước huấn luyện. Với tỷ lệ dropout p , mỗi nơ-ron có xác suất $1 - p$ để được giữ lại. Công thức tính đầu ra của một nơ-ron h_i trong quá trình huấn luyện:

$$\tilde{h} = h_i \times z_i, \quad z_i \sim \text{Bernoulli}(1 - p) \quad [51]$$

trong đó:

- h_i : Đầu ra ban đầu của nơ-ron i
- z_i : Biến Bernoulli với xác suất $1 - p$ (giá trị là 1 nếu nơ-ron được giữ lại, và 0 nếu bị bỏ qua).

Trong quá trình dự đoán, tất cả các nơ-ron đều hoạt động, nhưng trọng số được nhân với $(1 - p)$ để duy trì giá trị kỳ vọng của đầu ra:

$$h_i^{test} = h_i^{train} \times (1 - p) \quad [51]$$

Hàm mất mát với Dropout: Nếu L là hàm mất mát gốc, Dropout áp dụng thêm kỳ vọng trên các cấu hình ngẫu nhiên của mạng:

$$\mathbb{E}|L(W, z)| \quad [51]$$

- z : Vector nhị phân biểu thị các nơ-ron được giữ lại hoặc bỏ qua.

2.2 Thực nghiệm

2.2.1 Dataset

Nhóm lấy dữ liệu cổ phiếu của **Yahoo Finance** mà Yahoo Finance không có danh sách chính thức và cố định về loại ngành của cổ phiếu (Industry type of the stock) mà họ sử dụng hệ thống phân loại ngành dựa trên Global Industry Classification Standard (GICS), một tiêu chuẩn phân loại ngành toàn cầu được phát triển bởi MSCI và Standard & Poor's.

Trong các lĩnh vực theo GICS thì nhóm có chọn ra 02 lĩnh vực và các công ty/tập đoàn thuộc lĩnh vực tương ứng là:

- Tiêu dùng thường xuyên (Consumer Staples):
 - Coca-Cola (KO)
 - Nestle (NESN.SW)
 - PepsiCo (PEP)

- Viễn thông (Communication Services):
 - Walt Disney (DIS)
 - Netflix (NFLX)
 - Meta Platforms (META)

Đồng thời nhóm đã chọn các chỉ số kinh tế vĩ mô (macroeconomic indicators) cho các doanh nghiệp trên và các chỉ số này đóng góp vào việc train & test model. Các chỉ số nhóm đã chọn là:

- GDP
- CPI (Lạm phát)
- Tỷ lệ thất nghiệp
- Lãi suất

Tất cả các dữ liệu về chỉ số kinh tế nhóm đều lấy nguồn từ **FRED** (fredapi) - Federal Reserve Economic Data: <https://fred.stlouisfed.org>

Dữ liệu cổ phiếu lấy từ Yahoo Finance từ 1/1/2014 đến ngày 31/10/2024, gồm 2665 mẫu trong đó có các feature là:

- Date: Ngày giao dịch.
- Open: Giá mở cửa.
- High: Giá cao nhất.
- Low: Giá thấp nhất.
- Close: Giá đóng cửa.
- Adj_Close: Giá điều chỉnh (sau chia cổ tức hoặc tách cổ phiếu).
- Volume: Khối lượng giao dịch.

Nhóm chọn mốc 1/1/2022 để phân tách dữ liệu, nghĩa là số lượng tập train và test sẽ bao gồm như sau:

Tên doanh nghiệp	Số lượng mẫu tập train	Số lượng mẫu tập test
Coca-Cola (KO)	1986	721
Nestle (NESN.SW)	1990	731
PepsiCo (PEP)	1986	721

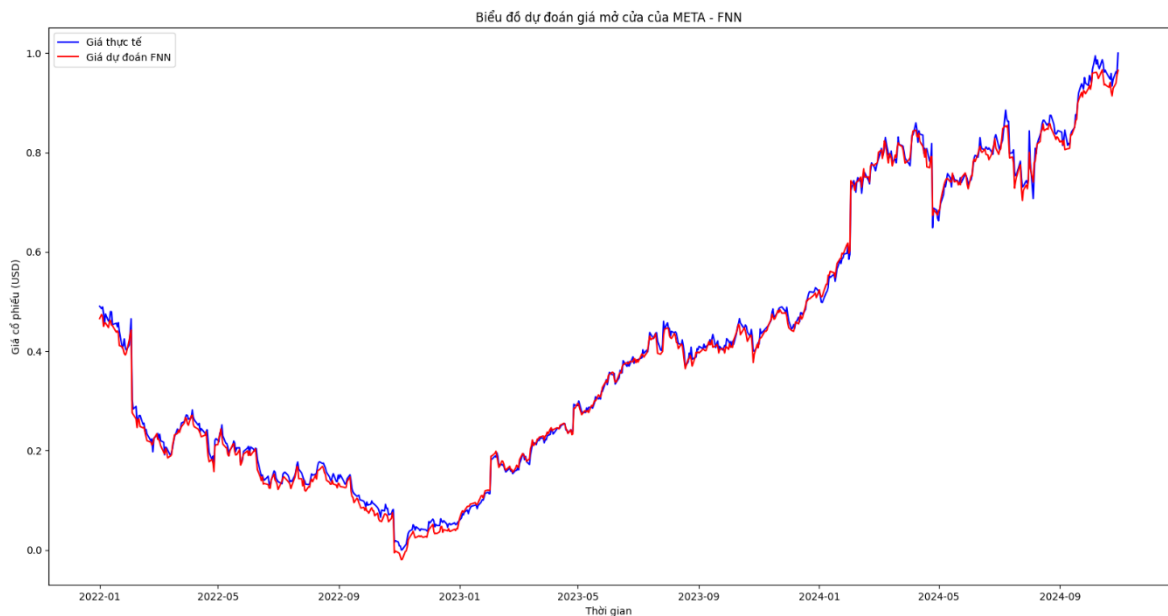
Walt Disney (DIS)	1986	721
Netflix (NFLX)	1986	721
Meta Platforms (META)	1986	721

Bảng 2.1: Chia tập train và test

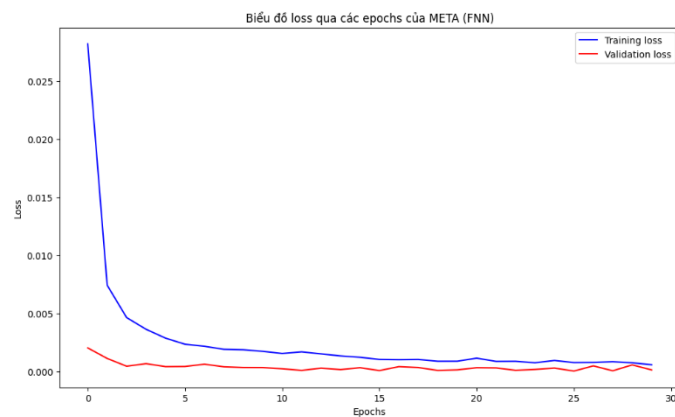
2.2.2 Kết quả dự đoán giá Mở cửa cổ phiếu và biểu đồ huấn luyện – Đánh giá

Vì có tận 6 doanh nghiệp nên nhóm sẽ chọn 1 doanh nghiệp là META để trình bày vào báo cáo.

Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình FNN:



Hình 2.1: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình FNN



Hình 2.2: Biểu đồ loss qua các epochs train và validation FNN model

Nhận xét:

- Ưu điểm: Mô hình FNN dự đoán xu hướng giá cổ phiếu META khá tốt, bám sát giá thực tế, Training và Validation loss thấp, ổn định, không có dấu hiệu overfitting.
- Hạn chế: Mô hình phản ứng chưa tốt với các biến động mạnh, có xu hướng dự đoán trễ. Validation loss thấp hơn Training loss, do dữ liệu validation chưa đủ đa dạng.

Đánh giá qua các thang đo:

```

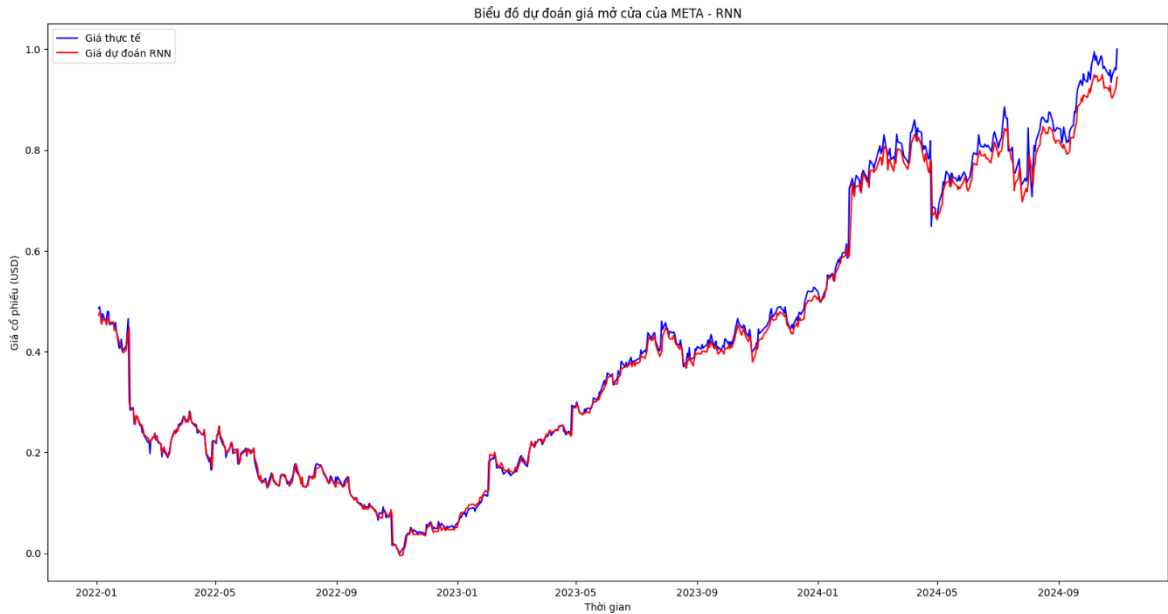
Thông số đánh giá mô hình FNN cho mã cổ phiếu META:
MAE: 0.00999406073242426
MSE: 0.0001537836651550606
RMSE: 0.012400954203409534
R^2: 0.9979917407035828

```

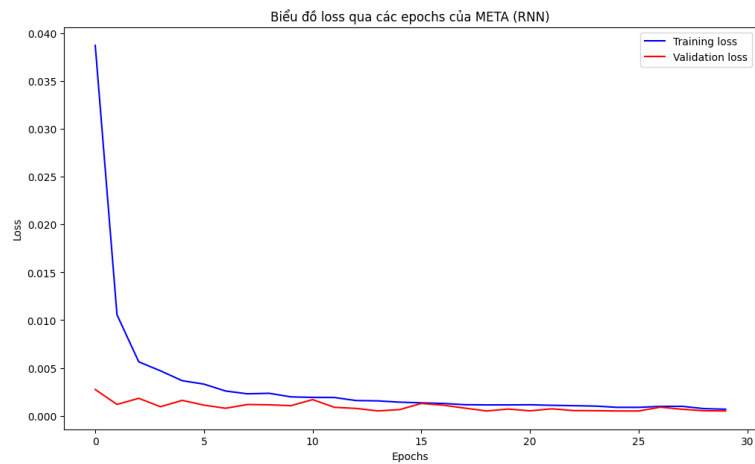
Hình 2.3: Đánh giá mô hình FNN cho mã cổ phiếu META qua các thang đo

- MAE (Mean Absolute Error): 0.00999: Sai số trung bình tuyệt đối rất nhỏ, cho thấy mô hình dự đoán giá mở cửa khá chính xác.
- MSE (Mean Squared Error): 0.00015: Sai số bình phương trung bình cực nhỏ, thể hiện mức độ sai lệch thấp.
- RMSE (Root Mean Squared Error): 0.0124: Sai số bình phương gốc nhỏ, khẳng định mô hình dự đoán ổn định và chính xác.
- R^2 (R-squared): 0.998: Mô hình giải thích đến 99.8% phương sai trong dữ liệu, cho thấy khả năng dự đoán mạnh mẽ.

Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình RNN:



Hình 2.4: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình RNN



Hình 2.5: Biểu đồ loss qua các epochs train và validation RNN model

Nhận xét:

- Ưu điểm: Mô hình RNN dự đoán chính xác xu hướng giá mở cửa của cổ phiếu META, bám sát giá thực tế kể cả trong giai đoạn biến động

mạnh. Loss (training và validation) giảm nhanh và ổn định ở mức thấp, không có dấu hiệu overfitting.

- Hạn chế: Có một số chênh lệch nhỏ trong giai đoạn giá biến động nhanh, dự đoán có xu hướng trễ.

Đánh giá qua các thang đo:

```
Thông số đánh giá mô hình RNN cho mã cổ phiếu META:
MAE: 0.01169975864889518
MSE: 0.0003315936422666812
RMSE: 0.01820971285514083
R^2: 0.995675265789032
```

Hình 2.6: Đánh giá mô hình RNN cho mã cổ phiếu META qua các thang đo

- MAE (Mean Absolute Error): 0.0117 : Sai số trung bình nhỏ, mô hình dự đoán khá chính xác.
- MSE (Mean Squared Error): 0.00033: Sai số bình phương trung bình thấp, thể hiện độ chính xác tốt.
- RMSE (Root Mean Squared Error): 0.0182: Sai số gốc nhỏ, phản ánh mức độ chênh lệch không đáng kể.
- R^2 (R-squared): 0.9957: Mô hình giải thích được 99.57% phương sai trong dữ liệu, hiệu suất rất cao.

Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình SVM:



Hình 2.7: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình SVM

Nhận xét:

- Ưu điểm: Mô hình dự đoán tốt xu hướng tổng thể (tăng/giảm) của giá cổ phiếu META, đặc biệt trong các giai đoạn ổn định.
- Hạn chế: Không xử lý tốt các biến động mạnh và ngắn hạn, dự đoán có xu hướng bị mượt hóa, dẫn đến sai lệch lớn tại các điểm đảo chiều.

Đánh giá qua các thang đo

```
Thông số đánh giá mô hình SVM cho mã cổ phiếu META:
MAE: 0.06971582910539942
MSE: 0.006264202390520276
RMSE: 0.07914671433812193
R^2: 0.9181963492904034
```

Hình 2.8: Đánh giá mô hình SVM cho mã cổ phiếu META qua các thang đo

- MAE (Mean Absolute Error): 0.0697: Sai số trung bình không quá lớn tuy vậy nhưng mô hình chưa dự đoán chính xác.
- MSE (Mean Squared Error): 0.0063: Sai số bình phương trung bình cao hơn đáng kể so với các mô hình khác (FNN, RNN).

- RMSE (Root Mean Squared Error): 0.0791: Sai số gốc lớn, phản ánh chênh lệch đáng kể giữa giá dự đoán và thực tế.
- R^2 (R-squared): 0.918: Mô hình giải thích được 91.8% phương sai, nhưng thấp hơn so với các mô hình học sâu như RNN hoặc FNN.

Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình Random

Forest (RF):



Hình 2.9: Kết quả dự đoán giá mở cửa của cổ phiếu META đối với mô hình Random Forest

Nhận xét:

- Ưu điểm: Mô hình Random Forest dự đoán khá tốt xu hướng giá mở cửa, bám sát giá thực tế trong phần lớn thời gian. Hoạt động tốt ở giai đoạn giá ổn định hoặc có sự tăng trưởng chậm.
- Hạn chế: Mô hình không xử lý tốt các biến động mạnh và bất ngờ trong ngắn hạn, thể hiện qua các giai đoạn giá thay đổi nhanh, như năm 2024. Một số dự đoán bị lệch đáng kể so với giá thực tế ở các điểm đảo chiều.

Đánh giá qua các thang đo:

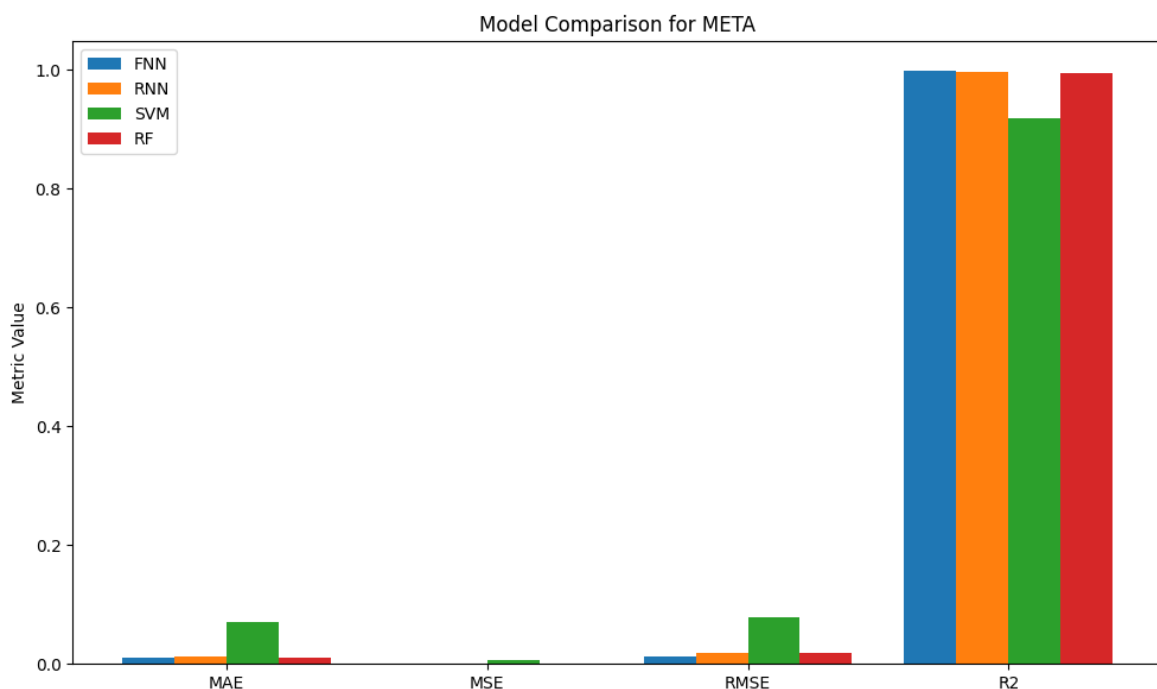
```

Thông số đánh giá mô hình RF cho mã cổ phiếu META:
MAE: 0.010851274778372772
MSE: 0.00035522007038261733
RMSE: 0.018847282838186978
R^2: 0.9953612133275591
  
```

Hình 2.10: Đánh giá mô hình RF cho mã cổ phiếu META qua các thang đo

- MAE (Mean Absolute Error): 0.0109: Sai số trung bình nhỏ, dự đoán khá chính xác.
- MSE (Mean Squared Error): 0.00036: Sai số bình phương trung bình thấp, nhưng kém hơn một chút so với RNN.
- RMSE (Root Mean Squared Error): 0.0188: Sai số gốc nhỏ, hiệu suất dự đoán ổn định.
- R^2 (R-squared): 0.9954: Mô hình giải thích được 99.54% phương sai, thể hiện hiệu suất cao.

2.2.3 So sánh các mô hình với nhau thông qua các thang đo



Hình 2.11: Biểu đồ các thang đo so sánh 4 model

Thông qua biểu đồ thang đo trên, nhóm trích xuất ra bảng sau:

Model	MAE	MSE	RMSE	R ²	Đánh giá tổng thể
FNN	0.0100	0.0002	0.0124	0.9980	Tốt nhất
RNN	0.0117	0.0003	0.0182	0.9957	Tốt, nhưng kém hơn FNN
SVM	0.0697	0.0063	0.0791	0.9182	Kém nhất
RF	0.0109	0.0004	0.0188	0.9954	Khá tốt

Bảng 2.2: Bảng đánh giá tổng quan các model

Kết luận:

- FNN là mô hình tốt nhất, vượt trội về mọi thang đo, phù hợp nhất cho dự đoán giá cổ phiếu META.
- RNN và RF cũng đạt hiệu suất tốt nhưng không bằng FNN.
- SVM là mô hình yếu nhất, không phù hợp với bài toán có biến động ngắn hạn.

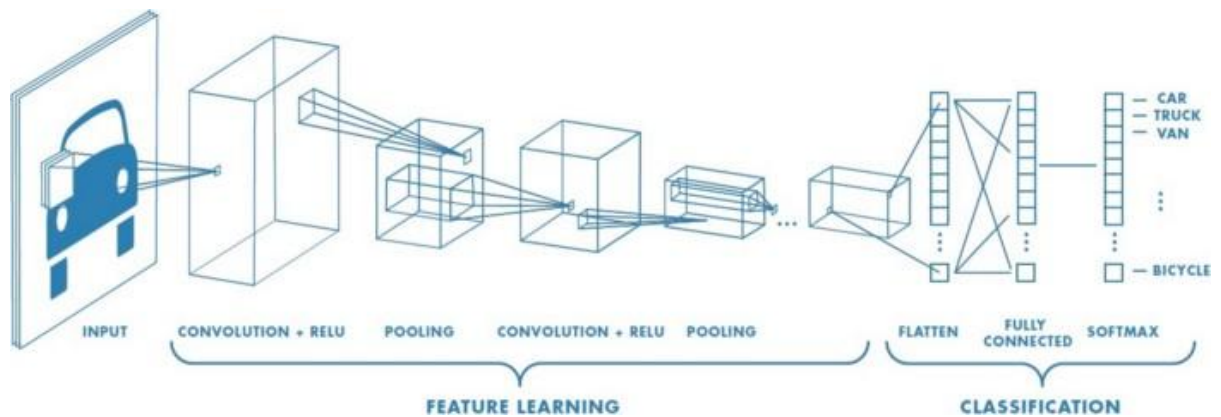
Tuy nhiên, theo lý thuyết RNN sẽ dự đoán được tốt hơn FNN nhưng vì dữ liệu khá ít (chỉ có 2665 mẫu) và không có độ biến động cao nên FNN có thể dự đoán tốt hơn. Suy ra bài toán thực tế thì RNN sẽ tối ưu hơn trong việc dự đoán dữ liệu có độ biến động lớn.

CHƯƠNG 3. CNN - CONVOLUTIONAL NEURAL NETWORK)

3.1 Tổng quan về CNN:

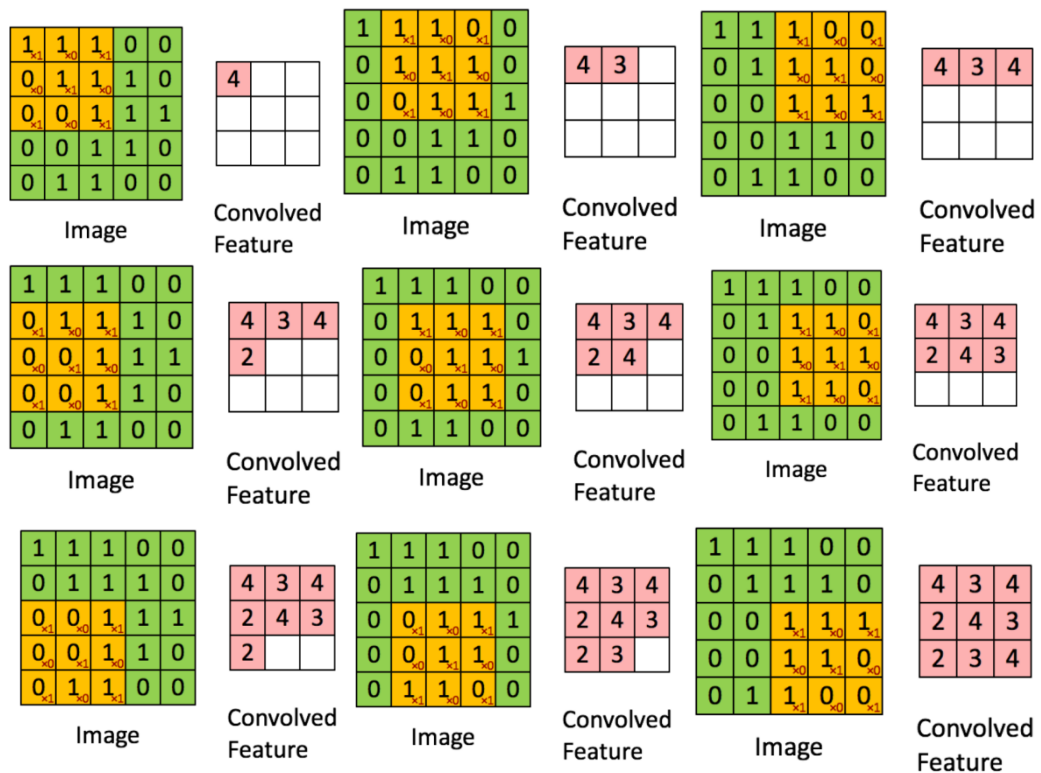
3.1.1 Định nghĩa:

- CNN (Convolutional Neural Network) là một loại mô hình học sâu thường được sử dụng trong xử lý ảnh, video, xử lý ngôn ngữ tự nhiên và nhiều ứng dụng khác [52]. CNN được thiết kế đặc biệt để nhận diện và xử lý dữ liệu có cấu trúc dạng lưới, chẳng hạn như hình ảnh (lưới của pixel).



Hình 3.1: Ví dụ về hoạt động của CNN

- Convolutional:
 - Một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới [1].
 - Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature [1].



Hình 3.2: Ví dụ về hoạt động của Convolutional [53]

3.1.2 Tham số đầu vào cho CNN:

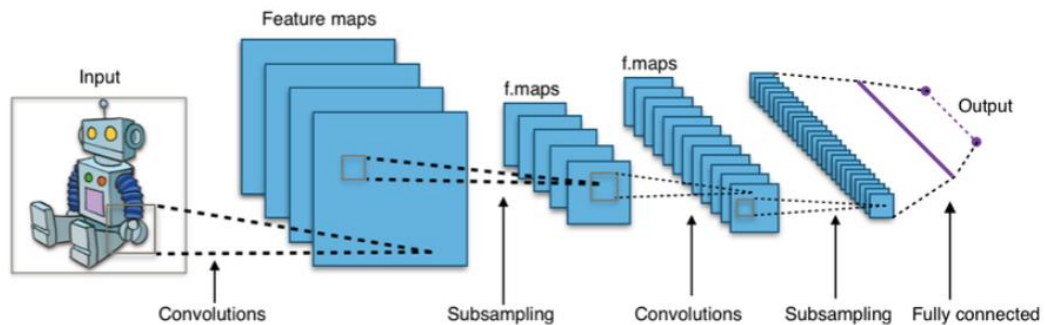
Tham số cho mạng Convolutional Neural Network cần những yếu tố như: filter size, số convolution, pooling size và việc train – test [2].

- **Lớp Convolution:** Số lượng lớp này càng nhiều thì sẽ giúp cải thiện được hoạt động của chương trình. Sử dụng những lớp với số lượng lớn thì khả năng hạn chế các tác động các tốt. Thông thường, chỉ sau khoảng 3 đến 4 lớp bạn sẽ đạt được kết quả như kỳ vọng [2].
- **Filter size:** Kích thước thường chọn là ma trận 3×3 hoặc ma trận 5×5 [2].
- **Pooling size:** Với những hình ảnh thông thường, bạn nên chọn ma trận pooling kích thước 2×2. Với những ảnh kích thước lớn thì nên chọn ma trận kích thước 3×3 [2].
- **Train – test:** Cần thực hiện train – test nhiều lần để có thể cho ra những parameter tốt nhất [2].

3.2 Cấu trúc của CNN

3.2.1 Kiến trúc cơ bản

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và Tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo [1].



Hình 3.3: Ví dụ cho hoạt động theo lớp của CNN [1]

Mạng này có tính kết hợp và tính bất biến. Một đối tượng sử dụng chiều theo các góc độ khác nhau thì sẽ có ảnh hưởng đến độ chính xác. Với dịch chuyển, co giãn hay quay ma trận ảnh thì lớp Pooling sẽ được dùng để hỗ trợ làm bất biến các tính chất này. Chính vì vậy mà mạng nơ-ron này sẽ đưa ra những kết quả có độ chính xác tương ứng với từng mô hình [2].

Trong đó, lớp Pooling sẽ có khả năng tạo tính bất biến với phép dịch chuyển, co giãn và quay. Còn tính kết hợp cục bộ sẽ cho thấy những cấp độ biểu diễn, dữ liệu từ thấp đến cao với mức trừu tượng thông qua Convolution từ filter. Mạng CNN có những lớp liên kết nhau dựa vào cơ chế Convolution [2].

Các lớp tiếp theo sẽ là kết quả từ những lớp trước đó, vì vậy mà bạn sẽ có những liên kết cục bộ phù hợp nhất. Trong quá trình huấn luyện mạng, mạng nơ-ron này sẽ tự học hỏi những giá trị thông qua filter layer dựa theo cách thức mà bạn thực hiện [2].

Cấu trúc cơ bản của một mô hình mạng CNN thường bao gồm 3 phần chính bao gồm:

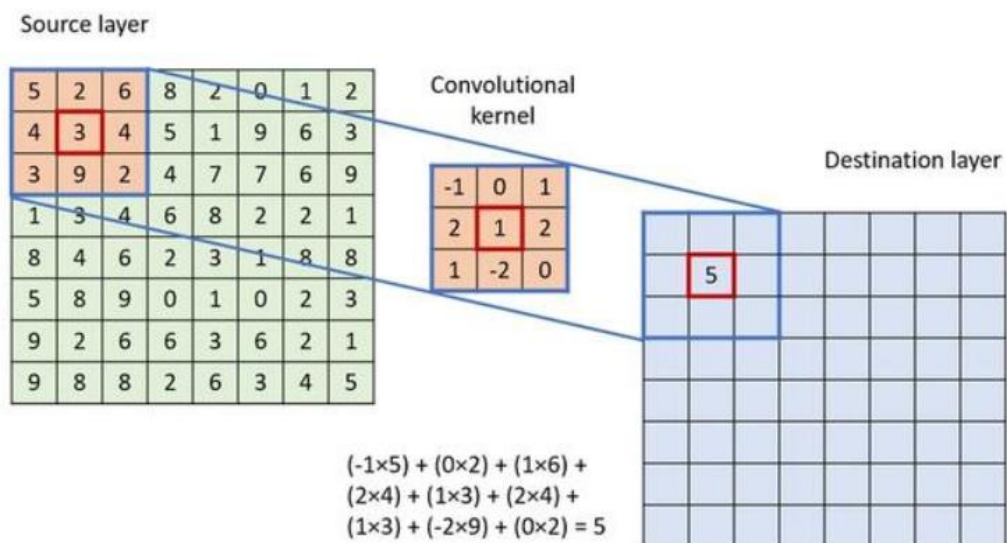
- **Trường cục bộ/ Local receptive field:** Lớp này sử dụng để tách lọc dữ liệu, thông tin hình ảnh để từ đó có thể lựa chọn các vùng có giá trị sử dụng hiệu quả cao nhất [2].
- **Trọng số chia sẻ/ Shared weights and bias:** Lớp này hỗ trợ làm giảm các tham số đến mức tối thiểu trong mạng CNN. Trong từng lớp convolution sẽ chứa các feature map riêng và từng feature thì sẽ có khả năng phát hiện một vài feature trong hình ảnh [2].
- **Lớp tổng hợp/ Pooling layer:** lớp cuối cùng và sử dụng để làm đơn giản các thông tin output. Tức là, sau khi tính toán xong và quét qua các layer trong mạng thì pooling layer sẽ được dùng để lược bỏ các thông tin không hữu ích. Từ đó cho ra kết quả theo kỳ vọng người dùng [2].

3.2.2 Các lớp chính trong CNN

Convolutional Layer (Lớp tích chập):

- Là thành phần cốt lõi của CNN, lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào [2]
- Thực hiện phép toán tích chập để duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách dùng một tập hợp các bộ lọc (filters/kernels) để trích xuất đặc trưng (features) từ ảnh đầu vào [54].
- Các yếu tố quan trọng trong lớp Convolutional là: padding, stride, feature map và filter map [2]
- Sử dụng filter để áp dụng vào các vùng của ma trận hình ảnh. Các filter map là các ma trận 3 chiều, bên trong đó là những tham số và chúng được gọi là parameters [2].
 - Stride tức là bạn dịch chuyển filter map theo từng pixel dựa vào các giá trị từ trái qua phải [2].

- Padding: Thường, giá trị viền xung quanh của ma trận hình ảnh sẽ được gán các giá trị 0 để có thể tiến hành nhân tích chập mà không làm giảm kích thước ma trận ảnh ban đầu [2].
- Feature map: Biểu diễn kết quả sau mỗi lần feature map quét qua ma trận ảnh đầu vào. Sau mỗi lần quét thì lớp Convolutional sẽ tiến hành tính toán [2].



Hình 3.4: Ví dụ cho convolutional layer [2]

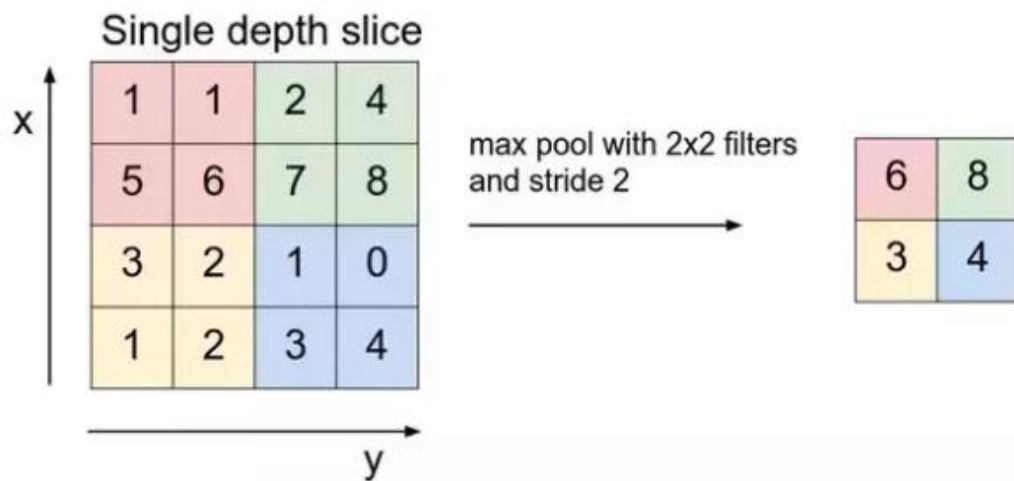
Relu Layer - Rectified Linear Unit (Hàm kích hoạt):

- Lớp ReLU này là hàm kích hoạt trong mạng CNN, được gọi là activation function. Nó có tác dụng mô phỏng những nơ ron có tỷ lệ truyền xung qua axon. Các hàm activation khác như Leaky, Sigmoid, Leaky, Maxout,.. tuy nhiên hiện nay, hàm ReLU được sử dụng phổ biến và thông dụng nhất [2].
- Hàm này được sử dụng cho những yêu cầu huấn luyện mạng nơ ron với những ưu điểm nổi bật điển hình là hỗ trợ tính toán nhanh hơn. Tăng khả năng học phi tuyến tính của hàm. Những lớp ReLU được dùng sau khi filter map được tính và áp dụng ReLU lên các giá trị của filter map [2].

Pooling Layer (Lớp gộp):

- Thường dùng để giảm kích thước không gian của đặc trưng và tăng hiệu quả tính toán [2].

- Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng [2].
- Các pooling có thể có nhiều loại khác nhau [55]:
 - Max Pooling
 - Average Pooling
 - Sum Pooling
- Phổ biến nhất là Max Pooling, lấy giá trị lớn nhất trong một vùng, và Average Pooling, lấy giá trị trung bình [53].



Hình 3.5: Ví dụ cho max pooling [55]

Fully Connected Layer (Lớp kết nối đầy đủ):

- Kết hợp các đặc trưng đã được trích xuất để dự đoán đầu ra (nhãn của ảnh) [2].
- Hoạt động tương tự như các lớp trong mạng nơ-ron truyền thống [2].

Dropout Layer (Lớp giảm overfitting):

- Ngẫu nhiên bỏ qua một số kết nối trong quá trình huấn luyện để giảm hiện tượng overfitting [2].

3.3 Nguyên lý hoạt động

CNN được thiết kế với giả định rằng đầu vào là hình ảnh, cho phép tận dụng cấu trúc 2D của hình ảnh. Điều này giúp giảm số lượng tham số cần phải học và tăng

hiệu suất xử lý. Mỗi hình ảnh được biểu diễn dưới dạng ma trận 3 chiều với chiều dài, chiều rộng và chiều sâu [54].

3.3.1 Phép tích chập

Tại lớp này, mạng thực hiện phép tích chập giữa đầu vào và một bộ lọc (hoặc kernel) để tạo ra map đặc trưng. Quá trình này giúp trích xuất các đặc điểm quan trọng từ hình ảnh như cạnh, góc, v.v [54].

3.3.2 Hàm kích hoạt

Sau mỗi lớp tích chập, một hàm kích hoạt như ReLU được áp dụng. Mục đích của hàm kích hoạt là giới thiệu tính phi tuyến vào mạng, giúp mạng có khả năng học được các mối quan hệ phức tạp hơn [54].

3.3.3 Lớp Pooling

Pooling (hay còn gọi là subsampling hay downsampling) giúp giảm kích thước không gian của map đặc trưng, làm giảm số lượng tham số và tính toán trong mạng, và giúp tránh overfitting. Các phương pháp pooling phổ biến bao gồm max pooling và average pooling [54].

3.3.4 Lớp kết nối đầy đủ

Sau một hoặc nhiều lớp tích chập và pooling, mạng sẽ có các lớp kết nối đầy đủ, nơi tất cả neuron trong một lớp được kết nối với mọi neuron trong lớp trước đó. Đây là nơi mạng tổng hợp tất cả thông tin đã học và thực hiện phân loại hoặc dự đoán [54].

3.3.5 Backpropagation trong CNN

Cuối cùng, một hàm mất mát được tính toán để đo lường sai lệch giữa dự đoán và nhãn thực tế. CNN sử dụng quá trình backpropagation để cập nhật trọng số và giảm thiểu hàm mất mát, thông qua đó cải thiện khả năng dự đoán của mô hình [54].

Tóm lại, CNN hoạt động dựa trên cấu trúc phức tạp của mình, bao gồm lớp tích chập, hàm kích hoạt, pooling, và lớp kết nối đầy đủ, để tự động học các đặc trưng từ hình ảnh và thực hiện các nhiệm vụ như phân loại hình ảnh với độ chính xác [54].

3.4 Ưu điểm và nhược điểm

3.4.1 Ưu điểm

- Hiệu suất cao: CNN cho phép xử lý ảnh với độ phân giải cao và độ phức tạp cao với độ chính xác cao hơn so với các phương pháp truyền thống.
- Tự động học các đặc trưng: CNN có khả năng tự động học các đặc trưng của ảnh một cách tự nhiên thông qua việc lặp lại các lớp tích chập, giúp cho mô hình có khả năng phân loại và nhận diện đối tượng tốt hơn.
- Tính đa dụng: CNN có thể được áp dụng cho nhiều bài toán trong lĩnh vực thị giác máy tính, chẳng hạn như phân loại ảnh, nhận diện đối tượng, nhận dạng chữ viết tay, nhận diện khuôn mặt và dấu vân tay, v.v.
- Thích ứng với các bài toán mới: CNN có khả năng học và thích ứng với các bài toán mới bằng cách sử dụng các trọng số của mạng đã được huấn luyện trước đó.

3.4.2 Nhược điểm

- Đòi hỏi nhiều dữ liệu huấn luyện: Để đạt được hiệu suất tốt, CNN đòi hỏi một lượng lớn dữ liệu huấn luyện để có thể học các đặc trưng của ảnh một cách tự nhiên.
- Chi phí tính toán cao: CNN có nhiều lớp tích chập và lớp kết nối đầy đủ, do đó đòi hỏi nhiều tài nguyên tính toán và thời gian để huấn luyện.
- Không hiệu quả đối với các bài toán đơn giản: Khi sử dụng CNN cho các bài toán đơn giản, nó có thể dẫn đến sự quá khớp dữ liệu, khiến cho hiệu suất của mô hình không được tốt.
- Khó hiểu: Do CNN có nhiều lớp và nhiều tham số, nên việc hiểu và giải thích hoạt động của nó là khá khó.

3.5 Ứng dụng

3.5.1 Nhận Dạng Hình Ảnh

CNNs là công cụ hàng đầu trong việc phân loại hình ảnh và nhận dạng đối tượng. Chúng được sử dụng rộng rãi trong các ứng dụng từ nhận dạng khuôn mặt cho đến phân loại sản phẩm trong thương mại điện tử [2].

3.5.2 Phân Tích Cảm Xúc từ Biểu Cảm Khuôn Mặt

CNNs có khả năng phân tích biểu cảm khuôn mặt để xác định cảm xúc, hỗ trợ trong lĩnh vực tương tác máy người và cải thiện trải nghiệm người dùng [2].

3.5.3 Phát Hiện và Phân Loại Vật Thể

Trong an ninh và giám sát, CNNs được sử dụng để phát hiện và phân loại vật thể trong video hay hình ảnh, từ việc nhận diện phương tiện giao thông đến phát hiện hành vi bất thường [2].

3.5.4 Phân Tích Y Học

CNNs đóng một vai trò quan trọng trong việc phân tích hình ảnh y tế, giúp chẩn đoán bệnh qua hình ảnh X-quang, MRI, và các loại hình ảnh chẩn đoán khác, cải thiện độ chính xác và giảm thời gian chẩn đoán [2].

3.5.5 Hệ Thống Tự Động Lái

CNNs là trái tim của hệ thống tự động lái xe, nơi chúng phân tích dữ liệu từ camera và cảm biến để nhận diện đường đi, vật cản và chỉ dẫn giao thông, đảm bảo an toàn cho hệ thống lái tự động [2].

3.5.6 Dịch Máy và Xử Lý Ngôn Ngữ Tự Nhiên

Mặc dù không phải là ứng dụng trực tiếp, nhưng CNNs cũng đã được áp dụng trong xử lý ngôn ngữ tự nhiên và dịch máy, nơi chúng giúp cải thiện khả năng hiểu và dịch văn bản [2].

3.5.7 Phát Hiện Gian Lận

Trong ngành tài chính và thương mại điện tử, CNNs được sử dụng để phát hiện gian lận bằng cách phân tích hành vi giao dịch và so sánh chữ ký, giúp giảm thiểu rủi ro tài chính [2].

3.5.8 Giáo Dục và Nghiên Cứu

CNNs còn được sử dụng trong giáo dục để phát triển công cụ hỗ trợ học tập tương tác, và trong nghiên cứu để phân tích và xử lý dữ liệu phức tạp.

Khả năng tự học và trích xuất đặc trưng mạnh mẽ của CNNs đã giúp chúng trở thành công cụ không thể thiếu trong cuộc cách mạng AI hiện đại, mở ra nhiều khả năng mới và giải quyết nhiều thách thức từ trước đến nay [2].

3.6 Triển khai thuật toán

3.6.1 Dataset: MNIST

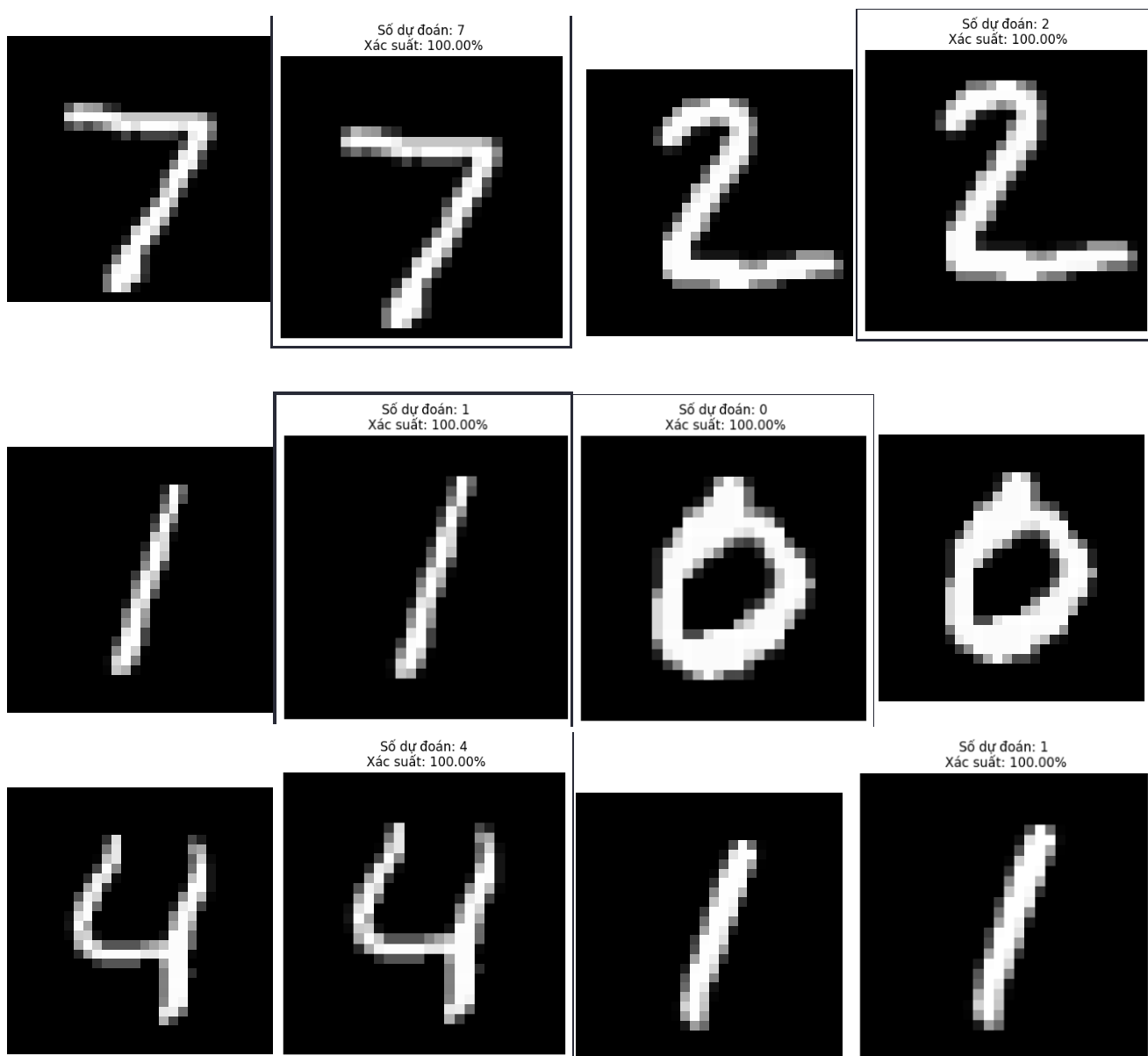
Modified National Institute of Standards and Technology là tập dữ liệu thường được dùng trong học máy và thị giác máy tính. Đây là tập dữ liệu thường được sử dụng để kiểm tra các thuật toán nhận dạng ký tự viết tay.

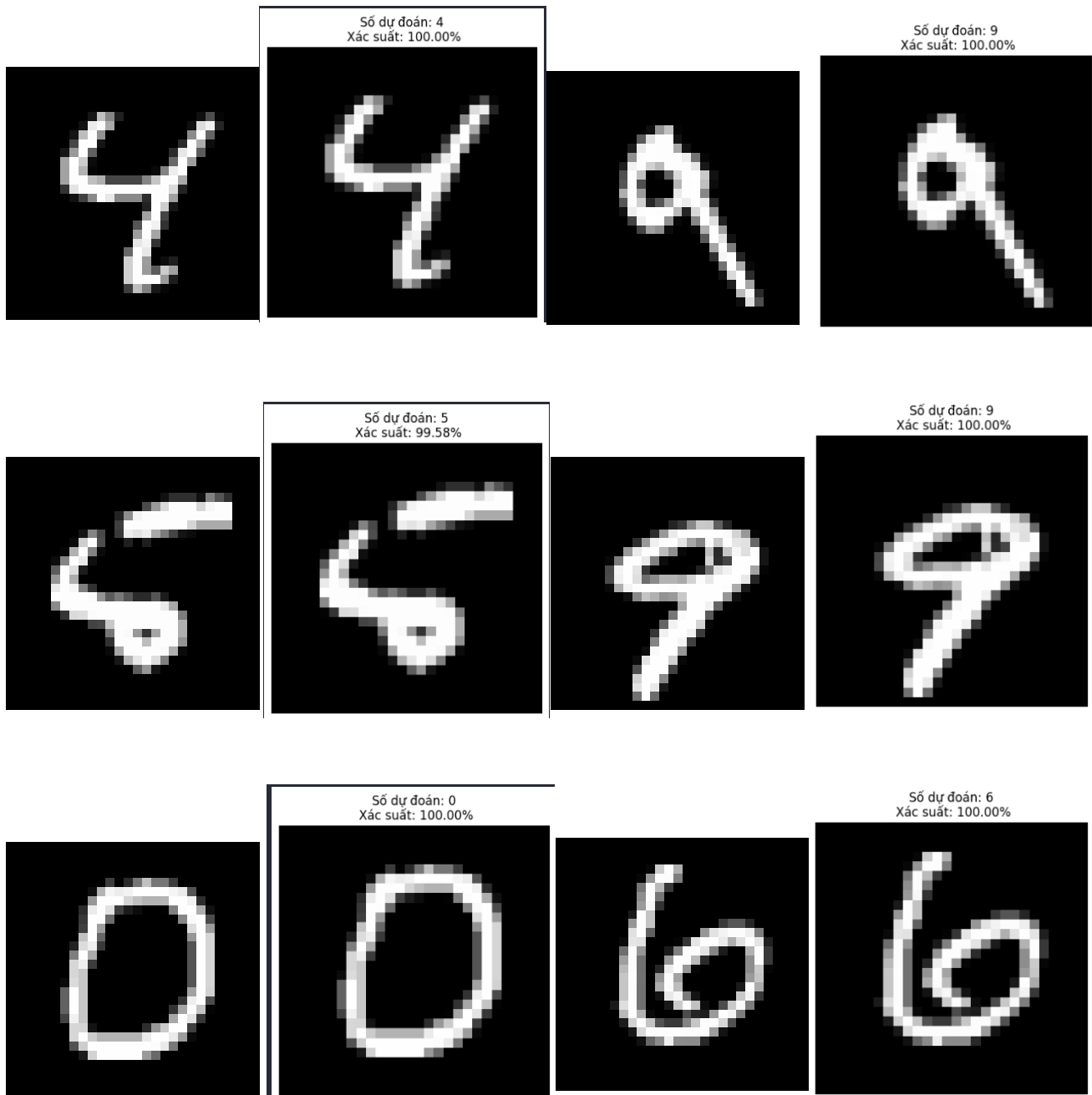
- Tập dữ liệu MNIST bao gồm các hình ảnh grayscale (màu xám) của các chữ số viết tay từ 0 đến 9.
- Mỗi hình ảnh có kích thước 28x28 pixel, tổng cộng 784 pixel.
- Cấu trúc dữ liệu:
 - Training set: 60,000 hình ảnh
 - Test set: 10,000 hình ảnh
- Thư viện hỗ trợ:

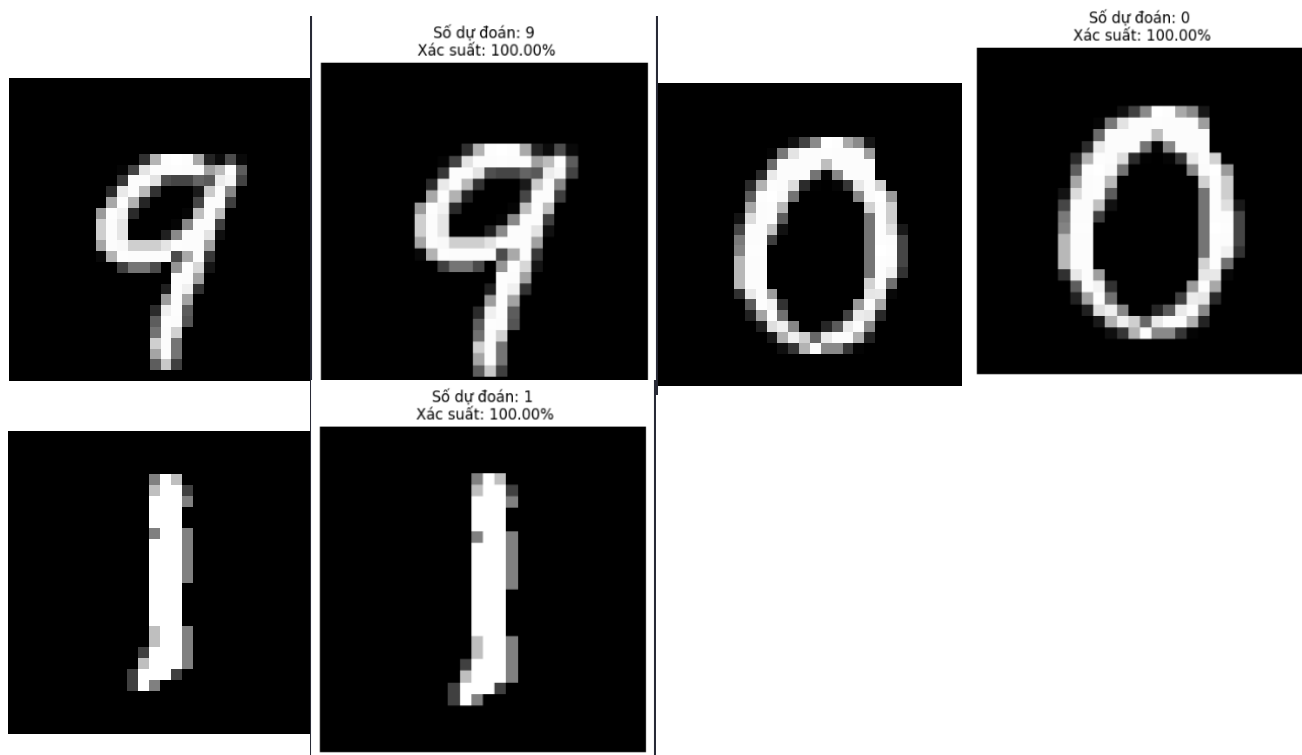
- TensorFlow/ Keras: tích hợp sẵn
 - PyTorch: torchvision cung cấp tool tải và tiền xử lý dữ liệu
- Các dữ liệu trên đều được lấy trên trang web của [TensorFlow](https://www.tensorflow.org/datasets/catalog/mnist)

3.6.2 Kết quả thực hiện thuật toán:

Dưới đây là 14 ảnh dùng để kiểm tra mô hình dự đoán ảnh bị làm mờ thông qua mô hình CNN:

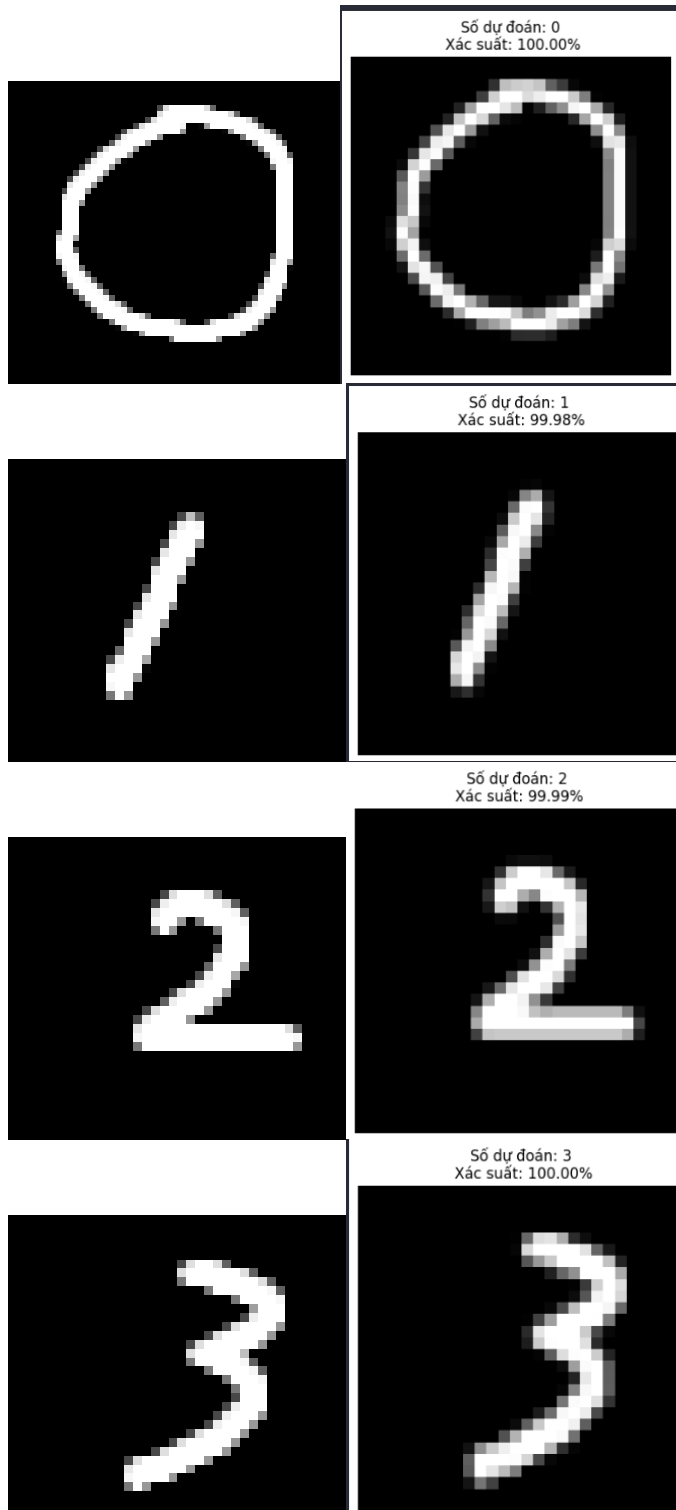


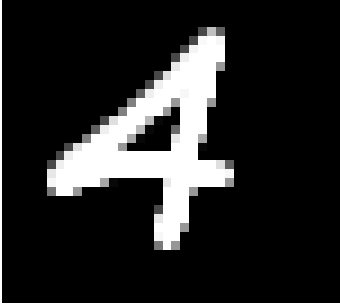
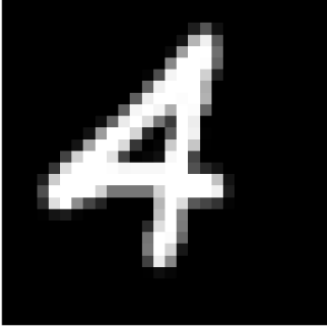
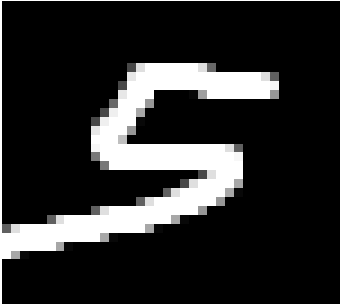

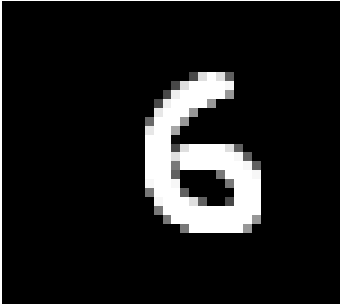
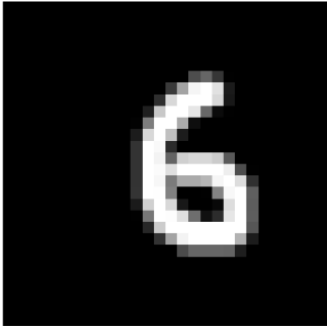
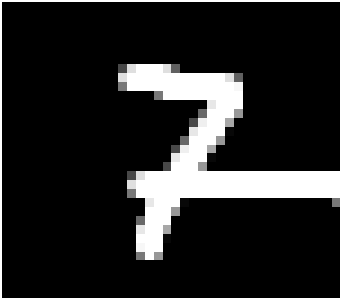
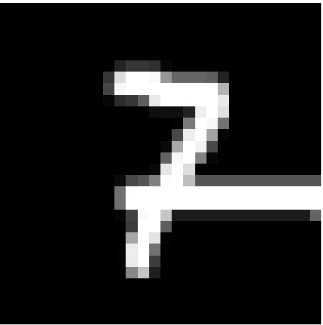


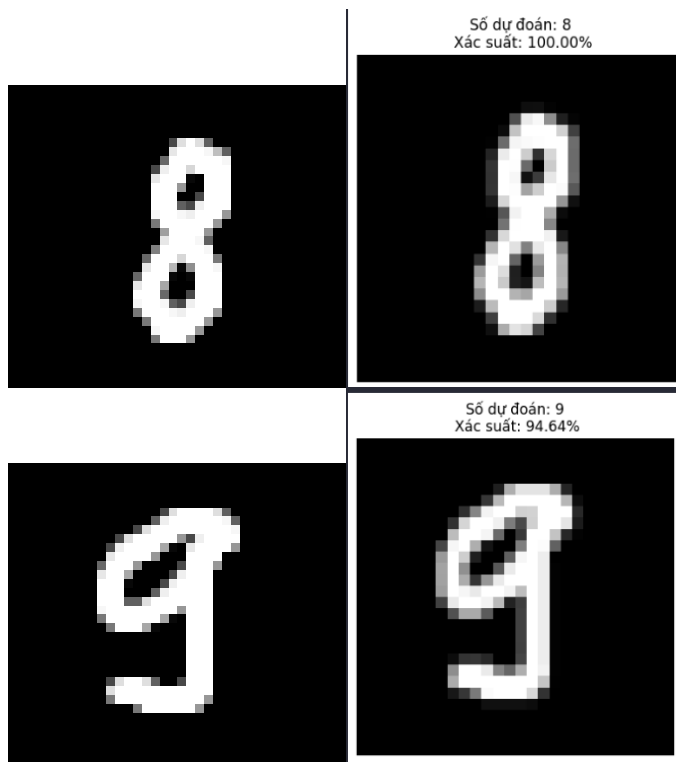


Hình 3.6: Các trường hợp dự đoán trong tập test của MNITS

Dưới đây là 10 ảnh dùng để kiểm tra mô hình dự đoán ảnh chữ viết tay mà nhóm đã thêm vào thông qua mô hình CNN:



	<div>Số dự đoán: 4 Xác suất: 100.00%</div> 
	<div>Số dự đoán: 5 Xác suất: 99.94%</div> 
	<div>Số dự đoán: 6 Xác suất: 99.22%</div> 
	<div>Số dự đoán: 7 Xác suất: 100.00%</div> 



Hình 3.7: Các chữ viết tay từ 0 đến 9 để test độ chính xác thực tế của CNN

Ngoài ra, nhóm còn thêm các trường hợp xấu bao gồm các ảnh bad_case từ 0 đến 9. Thầy có thể xem thêm trong file source code.

TÀI LIỆU THAM KHẢO

- [1] Thuật toán CNN, cấu trúc mạng Convolutional Neural Network, “TOPDev,”
[Trực tuyến]. Available: <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>.
- [2] Mạng CNN là gì và những kiến thức cơ bản cần biết về mạng CNN, "FPT,"
[Online]. Available: <https://aptech.fpt.edu.vn/cnn-la-gi.html>.
- [3] ayush, "Optimizers in Deep Learning: A Detailed Guide," 2024 December 10.
[Online]. Available: <https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/>.
- [4] D. Shulman, "Optimization Methods in Deep Learning: A Comprehensive," 19
February 2023. [Online]. Available: <https://arxiv.org/pdf/2302.09566v1>.
- [5] R. Sun, "Optimization for deep learning: an overview," 28 April 2020. [Online].
Available: <https://ise.ncsu.edu/wp-content/uploads/sites/9/2020/08/Optimization-for-deep-learning.pdf>.
- [6] T. V. Huu, “Machine Learning cơ bản,” 12 January 2017. [Trực tuyến].
Available:
<https://machinelearningcoban.com/2017/01/12/gradientdescent/>.
- [7] G. Farina, "Massachusetts Institute of Technology," 5 3 2024. [Online].
Available:
https://www.mit.edu/~gfarina/2024/67220s24_L07_gradient_descent/L07.pdf.

- [8] Wikipedia, "Mean squared error," 11 June 2024. [Online]. Available:
https://en.wikipedia.org/wiki/Mean_squared_error.
- [9] J. Frost, "Mean Squared Error (MSE)," [Online]. Available:
<https://statisticsbyjim.com/regression/mean-squared-error-mse/>.
- [10] Wikipedia, "Cross-entropy," 14 November 2024. [Online]. Available:
<https://en.wikipedia.org/wiki/Cross-entropy>.
- [11] K. Pykes, "Cross-Entropy Loss Function in Machine Learning: Enhancing Model Accuracy," 10 August 2024. [Online]. Available:
<https://www.datacamp.com/tutorial/the-cross-entropy-loss-function-in-machine-learning>.
- [12] Geeksforgeeks, "Gradient Descent Algorithm in Machine Learning," 12 September 2024. [Online]. Available:
<https://www.geeksforgeeks.org/gradient-descent-algorithm-and-its-variants/>.
- [13] Wikipedia, "Stochastic gradient descent," 16 December 2024. [Online].
Available: https://en.wikipedia.org/wiki/Stochastic_gradient_descent.
- [14] A. W. T. Y. J. M. T. O. Jonathon Price, "Stochastic gradient descent," 21 December 2020. [Online]. Available:
https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent.

- [15] baeldung, "Differences Between Gradient, Stochastic and Mini Batch Gradient Descent," 16 March 2023. [Online]. Available:
<https://www.baeldung.com/cs/gradient-stochastic-and-mini-batch>.
- [16] J. Brownlee, "A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size," 19 August 2019. [Online]. Available:
<https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/>.
- [17] Crypto1, "Gradient Descent Algorithm: How Does it Work in Machine Learning?," 9 October 2024. [Online]. Available:
<https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/>.
- [18] G. G. a. E. H. Thomas Lee, "Momentum," 15 December 2021. [Online]. Available:
<https://optimization.cbe.cornell.edu/index.php?title=Momentum>.
- [19] C. Jeeva, "Momentum-Based Gradient Descent," 9 May 2023. [Online]. Available: <https://www.scaler.com/topics/momentum-based-gradient-descent/>.
- [20] A. Kathuria, "Intro to optimization in deep learning: Momentum, RMSProp and Adam," 23 September 2024. [Online]. Available:
<https://www.digitalocean.com/community/tutorials/intro-to-optimization-momentum-rmsprop-adam>.

- [21] W. RECHT, "Chapter 4 Gradient Methods Using Momentum," [Online].
Available:
https://people.eecs.berkeley.edu/~brecht/opt4ml_book/O4MD_04_Momentum.pdf.

- [22] E. A. D. Learning, "What are the advantages of using momentum methods in optimization for machine learning, and how do they help in accelerating the convergence of gradient descent algorithms?," 22 May 2024. [Online].
Available: <https://eitca.org/artificial-intelligence/eitc-ai-adl-advanced-deep-learning/optimization/optimization-for-machine-learning/examination-review-optimization-for-machine-learning/what-are-the-advantages-of-using-momentum-methods-in-optimization-for-machine->.

- [23] D. I. D. LEARNING, "12.6. Momentum," 2024. [Online]. Available:
https://d2l.ai/chapter_optimization/momentum.html#summary.

- [24] S. Tripathi, "Adagrad Optimizer Explained: How It Works, Implementation, & Comparisons," 26 September 2024. [Online]. Available:
<https://www.datacamp.com/tutorial/adagrad-optimizer-explained>.

- [25] G. Farina, "Adaptive preconditioning: AdaGrad and ADAM," 11 April 2024. [Online]. Available:
https://www.mit.edu/~gfarina/2024/67220s24_L13_adagrad/L13.pdf.

- [26] D. Villarraga, "AdaGrad," 14 December 2021. [Online]. Available:
<https://optimization.cbe.cornell.edu/index.php?title=AdaGrad>.
- [27] DeepAI, "Adaptive Subgradient Methods (AdaGrad)," [Online]. Available:
<https://deepai.org/machine-learning-glossary-and-terms/adaptive-subgradient-methods>.
- [28] Y. Alok, "What is Adaptive Gradient(Adagrad) Optimizer?," 21 November 2024.
[Online]. Available:
<https://www.analyticsvidhya.com/blog/2024/11/adaptive-gradient-optimizer/>.
- [29] Adagrad, "Natural Language," Adagrad, 2024. [Online]. Available:
<https://www.adagrad.ai/nlp>.
- [30] J. Huang, "RMSProp," 21 December 2020. [Online]. Available:
<https://optimization.cbe.cornell.edu/index.php?title=RMSProp>.
- [31] B. B. Tuychiev, "RMSprop Optimizer Tutorial: Intuition and Implementation in Python," 23 October 2024. [Online]. Available:
<https://www.datacamp.com/tutorial/rmsprop-optimizer-tutorial>.
- [32] J. Brownlee, "Gradient Descent With RMSProp from Scratch," 12 October 2021.
[Online]. Available: <https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/>.

- [33] CampusX, "RMSProp Explained in Detail with Animations | Optimizers in Deep Learning Part 5," 2022. [Online]. Available:
<https://www.youtube.com/watch?v=p0wSmKslWi0>.
- [34] DeepAI, "RMSProp," [Online]. Available: <https://deepai.org/machine-learning-glossary-and-terms/rmsprop>.
- [35] deepchecks, "DEEPCHECKS GLOSSARY RMSProp," [Online]. Available:
<https://www.deepchecks.com/glossary/rmsprop/>.
- [36] A. C. Agniva Maiti, "RMSprop (Root Mean Square Propagation)," 2024.
[Online]. Available: <https://iq.opengenus.org/rmsprop/>.
- [37] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," 13 January 2021. [Online]. Available:
<https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.
- [38] N. Vishwakarma, "What is Adam Optimizer?," 15 October 2024. [Online].
Available: <https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/>.
- [39] A. Ajagekar, "Adam," 16 December 2021. [Online]. Available:
<https://optimization.cbe.cornell.edu/index.php?title=Adam>.
- [40] Manika, "Adam Optimizer Simplified for Beginners in ML," 28 October 2024.
[Online]. Available: <https://www.projectpro.io/article/adam-optimizer/986>.

- [41] B. Q. Agniva Maiti, "Adam Optimizer," 2024. [Online]. Available:
<https://iq.opengenus.org/adam-optimizer/>.
- [42] GeeksforGeeks, "Feedforward neural network," [Online]. Available:
<https://www.geeksforgeeks.org/feedforward-neural-network>.
- [43] Ian Goodfellow and Yoshua Bengio and Aaron Courvil, Deep Learning, MIT Press, 2016.
- [44] Yagawa, Genki and Oishi, Atsuya, Feedforward Neural Networks, Springer International Publishing, 2021, pp. 11--23.
- [45] Wikipedia, "Mạng thần kinh hồi quy," 6 6 2023. [Online]. Available:
https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_th%E1%BA%A7n_kinh_h%E1%BB%93i_quy.
- [46] Đoàn Võ Duy Thanh, Nguyễn Văn Cường, Nguyễn Lê Quang Nhật, Nguyễn Duy Du, Lê Khắc Hồng Phúc, Phạm Minh Đức, Trần Yến Thy, Phạm Hồng Vinh and Nguyễn Cảnh Thương, "8.4. Mạng nơ-ron Hồi tiếp," [Online]. Available: https://d2l.aivivn.com/chapter_recurrent-neural-networks/rnn_vn.html.
- [47] Afshine Amidi and Shervine Amidi, "Recurrent Neural Networks cheatsheet," [Online]. Available: <https://web.stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>.

- [48] © 2024 trituenhantao.io, "SVM quá khó hiểu! Hãy đọc bài này," [Online].
Available: <https://trituenhantao.io/kien-thuc/svm-qua-kho-hieu-hay-doc-bai-nay>.
- [49] Tiep Vu, "Random Forest algorithm," 2021. [Online]. Available:
https://machinelearningcoban.com/tabml_book/ch_model/random_forest.html.
- [50] Lutz Prechelt, "Early Stopping – But When?," in *Neural Networks: Tricks of the Trade*, Karlsruhe, Karlsruhe University, 2022, p. 55–69.
- [51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research* 15, 2014.
- [52] Convolution neural network in Machine learning, "geeksforgeeks," 13 03 2024.
[Online]. Available: <https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/>.
- [53] Nttuan8; Deep learning cơ bản; chap 6: CNN, "nttuan8," 30 03 2019. [Online].
Available: <https://nttuan8.com/bai-6-convolutional-neural-network/>.
- [54] JSTOR; Convolutional neural network, "Wikipedia," 2019. [Online]. Available:
https://en.wikipedia.org/wiki/Convolutional_neural_network.
- [55] Pham Van Chung; DeepLearning , "Viblo," 10 12 2020. [Online]. Available:
<https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2>.

