**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT**



**KHOA CÔNG NGHỆ THÔNG TIN**

**-----o0o-----**

**BÀI TẬP LỚN**

**TRÍ TUỆ NHÂN TẠO**

**Tên đề tài:**

|  |  |
| --- | --- |
| **GIÁO VIÊN:** | **ĐỖ BẢO SƠN** |
| **SINH VIÊN THỰC HIỆN:**  **LỚP:**  **NHÓM:** | **TRƯƠNG THÀNH ĐẠT**  **NGUYỄN TUẤN DŨNG**  **ĐỖ ĐÌNH ANH ĐỨC**  **73DCTT23**  **3** |

*Hà Nội, tháng 1 năm 2025*

# LỜI NÓI ĐẦU

Trí tuệ nhân tạo, một khái niệm từng chỉ xuất hiện trong các tác phẩm khoa học viễn tưởng, giờ đây đã trở thành hiện thực và đang định hình lại thế giới chúng ta đang sống. Từ những thuật toán đơn giản cho phép máy tính thực hiện các phép tính phức tạp, AI đã phát triển thành những hệ thống thông minh có khả năng học hỏi, suy luận và sáng tạo.

Lịch sử của AI bắt nguồn từ những năm 1950, khi các nhà khoa học bắt đầu nghiên cứu khả năng mô phỏng trí tuệ con người trên máy tính. Qua nhiều thập kỷ, AI đã trải qua nhiều giai đoạn thăng trầm, với những đột phá đáng kể trong lĩnh vực học máy (machine learning) và học sâu (deep learning). Ngày nay, AI đã được ứng dụng rộng rãi trong nhiều lĩnh vực, từ y tế, tài chính, đến sản xuất và dịch vụ.

Tuy nhiên, cùng với những lợi ích to lớn, AI cũng đặt ra nhiều thách thức. Các vấn đề như an ninh mạng, mất việc làm, và những tác động không lường trước đến xã hội đang là những mối quan tâm hàng đầu. Bên cạnh đó, việc phát triển AI một cách có trách nhiệm và đạo đức cũng là một vấn đề cấp bách.

Trong tương lai, AI được dự đoán sẽ tiếp tục phát triển mạnh mẽ và tạo ra những đột phá chưa từng có. Các hệ thống AI sẽ trở nên thông minh hơn, linh hoạt hơn và có khả năng tự học hỏi cao hơn. Điều này sẽ mở ra những cơ hội mới cho con người, nhưng cũng đặt ra những câu hỏi về bản chất của chúng ta và vai trò của chúng ta trong một thế giới ngày càng được điều khiển bởi máy móc.

**NHẬN XÉT CỦA GIÁO VIÊN**

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

.................................................................................................................................

***Chữ ký của giáo viên***

MỤC LỤC

[LỜI NÓI ĐẦU 2](#_Toc184852356)

[CHƯƠNG 1. Thuật toán Simulated Annealing 6](#_Toc184852357)

[1.1.Tổng quan về thuật toán tìm kiếm cục bộ - Local Search 6](#_Toc184852358)

[1.2. Giới thiệu về thuật toán Simulated Annealing 6](#_Toc184852359)

[1.2.1 Nguồn gốc của thuật toán: 6](#_Toc184852360)

[1.2.2. Khái niệm và mục đích của thuật toán: 7](#_Toc184852361)

[1.2.3. Tính chất của thuật toán: 7](#_Toc184852362)

[1.3. Ứng dụng thực tiễn của Simulated Annealing 8](#_Toc184852363)

[1.4. Cách thực hiện thuật toán Simulated Annealing 9](#_Toc184852364)

[1.5. Mô hình hóa cài đặt thuật toán Simulated Annealing – Code mẫu 9](#_Toc184852365)

[Mã giả của thuật toán: 9](#_Toc184852366)

[Giải thích các thành phần chính của thuật toán: 10](#_Toc184852367)

[CHƯƠNG 2. Thuật toán Gradient Descent 11](#_Toc184852368)

[2.1. Lý thuyết và công thức 11](#_Toc184852369)

[Ưu điểm của thuật toán Gradient Descent: 14](#_Toc184852370)

[Nhược điểm của thuật toán Gradient Descent: 14](#_Toc184852371)

[Cách khắc phục các hạn chế của Gradient Descent: 14](#_Toc184852372)

[2.2 Phương pháp 15](#_Toc184852373)

[Các thư viện cần thiết: 15](#_Toc184852374)

[Đọc và định dạng dữ liệu: 15](#_Toc184852375)

[Khởi tạo các tham số ban đầu: 15](#_Toc184852376)

[Định nghĩa đạo hàm tính toán: 16](#_Toc184852377)

[Thực hiện Gradient Descent: 16](#_Toc184852378)

[Cập nhật trọng số: 17](#_Toc184852379)

[Kết quả sau khi chạy epochs: 17](#_Toc184852380)

[Vẽ biểu đồ phân tán:Hình 2 .1: Kết quả sau khi chạy epochs 17](#_Toc184852381)

[Biểu đồ phân tán: 18](#_Toc184852382)

[Kết quả dự đoán: 19](#_Toc184852383)

[2.3. Sử dụng thuật toán Stochastic Gradient Descent, Mini-batch Gradient Descent và RMSprop để tối ưu hoá bài toán dự báo thức ăn nhanh. 19](#_Toc184852384)

[Tổng quan bài toán 19](#_Toc184852385)

[Lý thuyết các thuật toán 19](#_Toc184852386)

[Các nghiên cứu liên quan 20](#_Toc184852387)

[2.4. Thảo luận 30](#_Toc184852388)

[2.5. Kết luận 30](#_Toc184852389)

# CHƯƠNG 1. Thuật toán Simulated Annealing

* 1. **Tổng** **quan** **về** **thuật** **toán** **tìm** **kiếm** **cục** **bộ** **-** **Local** **Search**

Thuật toán tìm kiếm cục bộ (local search) là một thuật toán tìm kiếm được sử dụng để giải quyết các bài toán tối ưu hóa. Nó bắt đầu từ một giải pháp khởi tạo và cố gắng tìm một giải pháp tốt hơn bằng cách tìm kiếm qua các giải pháp kề của nó.

Thuật toán tìm kiếm cục bộ không đưa ra được một giải pháp tối ưu toàn cục cho một bài toán, nhưng nó thường đưa ra được một giải pháp tốt đối với bài toán đó trong một thời gian ngắn. Vì vậy, nó được sử dụng rộng rãi trong các bài toán tối ưu với số lượng biến lớn.

Trên thực tế, đối với một số bài toán, đường đi đến lời giải không thực sự quan trọng, mà quan trọng là trạng thái đích (lời giải của bài toán)

Với mục tiêu tìm một thuật giải thỏa mãn các ràng buộc, có thể ví như bài toán n quân chốt trên 1 bàn cờ kích thước mxn, sao cho các quân chốt không ăn nhau.

Tại mỗi thời điểm, thuật toán tìm kiếm cục bộ dựa vào hai dữ kiện:

Trạng thái hiện thời (Current State): Từ trạng thái này, tìm kiếm các trạng thái kề cận. Mục tiêu (Objective): nhằm “cải thiện” trạng thái hiện thời này đối với một tiêu chí nào đó.

Các thuật toán tìm kiếm cục bộ phổ biến bao gồm:

+ Hill Climbing: Tìm kiếm các giải pháp tốt hơn bằng cách tìm kiếm các giải pháp kề có giá trị hàm mục tiêu tốt nhất.

+ Simulated Annealing: Một thuật toán tìm kiếm cục bộ ngẫu nhiên, nó tìm kiếm các giải pháp kề của nó và chấp nhận một số giải pháp xấu hơn với xác suất nhất định.

+ Tabu Search: Tìm kiếm các giải pháp tốt hơn bằng cách tìm kiếm các giải pháp kề, nhưng tránh lặp lại các giải pháp đã thăm qua.

+ Genetic Algorithm: Một thuật toán tìm kiếm cục bộ dựa trên nguyên lý lọc tự nhiên, nó sử dụng các phép lai và đột biến để tạo ra các giải pháp mới.

## 1.2. Giới thiệu về thuật toán Simulated Annealing

### 1.2.1 Nguồn gốc của thuật toán:

Thuật toán Simulated Annealing được phát triển vào những năm 1980 bởi nhà toán học người Mỹ S. Kirkpatrick, C. D. Gelatt và M. P. Vecchi. Thuật toán được phát triển để giải quyết các bài toán tối ưu bằng cách mô phỏng quá trình nung chảy kim loại (Annealing) trong công nghệ chế tạo kim loại.

### 1.2.2. Khái niệm và mục đích của thuật toán:

Ý tưởng của thuật toán Simulated Annealing là khởi tạo một trạng thái ban đầu của hệ thống và thực hiện các bước tìm kiếm cục bộ (local search) để tìm kiếm các giải pháp tốt hơn. Tuy nhiên, khác với các thuật toán tìm kiếm cục bộ thông thường, thuật toán Simulated Annealing cho phép chấp nhận các giải pháp xấu hơn với một xác suất nhất định. Xác suất này được tính dựa trên một hàm mục tiêu (objective function) và nhiệt độ của hệ thống. Khi nhiệt độ càng cao, xác suất chấp nhận các giải pháp xấu càng cao. Tuy nhiên, khi nhiệt độ giảm dần, xác suất chấp nhận các giải pháp xấu cũng giảm dần.

Quá trình giảm nhiệt độ được gọi là quá trình làm mát (cooling schedule). Quá trình này thường được thiết lập sao cho nhiệt độ giảm dần theo thời gian và đạt đến một mức độ thấp nhất trước khi thuật toán kết thúc.

Mục đích chính của thuật toán Simulated Annealing là tìm kiếm giải pháp tối ưu cho một bài toán tối ưu hóa bằng cách sử dụng phương pháp tìm kiếm cục bộ và xác suất để tránh rơi vào các cực tiểu cục bộ. Thuật toán được áp dụng để giải quyết các bài toán tối ưu có thể có rất nhiều giải pháp khác nhau và không thể sử dụng các phương pháp tìm kiếm thông thường để đạt được kết quả tối ưu toàn cục.

### 1.2.3. Tính chất của thuật toán:

Thuật toán Simulated Annealing có các tính chất sau:

* Khả năng tìm kiếm toàn cục: Thuật toán Simulated Annealing có khả năng tìm kiếm giải pháp tối ưu toàn cục, tức là giải pháp tốt nhất trong tất cả các giải pháp có thể.

Điều này là do thuật toán cho phép chấp nhận các giải pháp xấu hơn với xác suất nhất định, giúp tránh rơi vào các cực tiểu cục bộ.

* Có tính ngẫu nhiên: Thuật toán Simulated Annealing sử dụng xác suất để quyết định việc chấp nhận hay từ chối các giải pháp, do đó có tính ngẫu nhiên. Tuy nhiên, việc sử dụng xác suất được điều chỉnh sao cho thuật toán có thể hội tụ đến một giải pháp tối ưu với xác suất cao.
* Thích nghi với bài toán: Thuật toán Simulated Annealing có thể được điều chỉnh để phù hợp với các loại bài toán khác nhau, thông qua việc thay đổi các tham số của thuật toán như hàm năng lượng, hệ số giảm nhiệt, và nhiều tham số khác.
* Có hiệu suất cao: Thuật toán Simulated Annealing có hiệu suất cao trong việc tìm kiếm giải pháp tối ưu cho các bài toán tối ưu hóa phức tạp, nơi mà các phương pháp tìm kiếm thông thường có thể không thể đạt được kết quả tối ưu toàn cục.
* Không đòi hỏi đạo đức đặc biệt: Thuật toán Simulated Annealing không đòi hỏi đạo đức đặc biệt như các thuật toán di truyền hoặc thuật toán tiến hóa, do không sử dụng các phép lai ghép hay đột biến.

## Ứng dụng thực tiễn của Simulated Annealing

Thuật toán Simulated Annealing có nhiều ứng dụng trong thực tiễn, bao gồm:

* Tối ưu hóa kết cấu phân tử: trong việc tìm kiếm cấu trúc phân tử tối ưu với năng lượng thấp nhất.
* Phân tích đồ thị: trong việc tìm kiếm đường đi tối ưu trên một đồ thị, bao gồm cả các ứng dụng trong vận tải, lộ trình đi lại, v.v.
* Lập lịch sản xuất: trong việc tìm kiếm lịch sản xuất tối ưu để giảm thiểu thời gian chờ đợi và tăng năng suất sản xuất.
* Tối ưu hóa mô hình học máy: trong việc tìm kiếm các tham số tối ưu của mô hình học máy để đạt được độ chính xác cao nhất.
* Tối ưu hóa thiết kế kỹ thuật: trong việc tìm kiếm thiết kế tối ưu cho các hệ thống kỹ thuật như máy móc, công trình xây dựng, v.v.

## Cách thực hiện thuật toán Simulated Annealing

Khởi tạo trạng thái ban đầu: chọn một trạng thái ngẫu nhiên trong không gian tìm kiếm làm trạng thái ban đầu.

* Thiết lập hàm mục tiêu và hàm đánh giá: xác định hàm mục tiêu cần tối ưu và hàm đánh giá để đánh giá chất lượng của các trạng thái tìm kiếm.
* Thiết lập lịch trình và nhiệt độ ban đầu: xác định lịch trình ( kế hoạch giảm nhiệt độ ) và nhiệt độ ban đầu.
* Lặp lại quá trình tìm kiếm để tối ưu hoá hàm mục tiêu:

1. Lựa chọn một trạng thái mới dựa trên trạng thái hiện tại và quy tắc chuyển tiếp: chọn một trạng thái mới bằng cách áp dụng quy tắc chuyển tiếp từ trạng thái hiện tại.
2. Đánh giá trạng thái mới: sử dụng hàm đánh giá để đánh giá chất lượng của trạng thái mới.
3. Xác định xác suất chấp nhận trạng thái mới: xác định xác suất chấp nhận trạng thái mới dựa trên nhiệt độ và chênh lệch giữa giá trị mục tiêu của trạng thái hiện tại và trạng thái mới.
4. Cập nhật trạng thái hiện tại nếu trạng thái mới được chấp nhận: nếu trạng thái mới được chấp nhận, cập nhật trạng thái hiện tại bằng trạng thái mới.
5. Giảm nhiệt độ: cập nhật nhiệt độ theo lịch trình giảm nhiệt độ đã thiết lập.

## Mô hình hóa cài đặt thuật toán Simulated Annealing – Code mẫu

### Mã giả của thuật toán:

A white text with black text

Description automatically generated

*Chapter* *4* *-* *Artificial* *Intelligence* *A* *Modern* *Approach* *(3rd* *Edition)*

### Giải thích các thành phần chính của thuật toán:

* Phương pháp SA là dựa trên ý tưởng về thuật “luyện kim”, trong đó kim loại được nung nóng vượt quá nhiệt độ tới hạn và được làm mát theo một lịch trình cụ thể cho đến khi đạt đến trạng thái năng lượng tối thiểu.
* Về cốt lõi thì phương pháp này dựa trên phương trình [1] đại diện cho xác suất nhảy lên mức năng lượng tiếp theo. Trong bối cảnh của thuật toán, mức năng lượng chỉ đỡn giản là giá trị hiện tại của bất kỳ chức năng nào đang được tối ưu hoá.

P(Ec,En,T)=e-deltaE/T (1)

Trong đó:

* Ec : năng lượng ở trạng thái hiện tại.
* En : năng lượng tại địa điểm lân cận được đề xuất.
* deltaE : sự thay đổi năng lượng giữa trạng thái hiện tại và trạng thái láng giềng được đề xuất.
* T : nhiệt độ hệ thống hiện tại.

A group of math equations

Description automatically generated

*Ví* *dụ*

* Hàm trạng thái ban đầu (initial\_state): khởi tạo trạng thái ban đầu cho phương pháp, chọn một trạng thái ngẫu nhiên trong mục tiêu không gian tìm kiếm.
* Hàm chức năng lân cận (neighbor\_function): tạo ra các mục tiêu lân cận để đánh giá trạng thái giữa hiện tại và lân cận.
* Hàm chức năng giá trị (cost\_function): tính toán giá trị của trạng thái mục tiêu nhằm tạo ra giá trị delta để bắt đầu quá trình thực hiện thuật toán.
* Hàm chức năng nhiệt độ (temperature\_function): được sử dụng xem như tham số trong điều kiện dừng thuật toán, xem xét và xác định trạng thái mới dựa trên nhiệt độ.
* Hàm điều kiện dừng (stopping\_condition): chấp nhận trạng thái mới, nhằm tìm ra trạng thái tối ưu của thuật toán.

**CHƯƠNG 2. THUẬT TOÁN Gradient Descent**

## 2.1. Lý thuyết và công thức

Thuật toán Gradient Descent là một trong những phương pháp phổ biến nhất được sử dụng để tối giúp tối ưu hóa các tham số trong mô hình hồi quy tuyến tính. Thuật toán Gradient Descent sẽ cập nhật giá trị bằng cách tính đạo hàm riêng của hàm mất mát (loss function) theo mỗi tham số. Mục tiêu là giảm thiểu giá trị của hàm mất mát để đạt được mô hình dự báo tốt nhất.

Để hiểu hơn về cách hoạt động của thuật toán chúng ta đến với phần ví dụ sau:

Bài toán dự báo giá nhà, với số lượng mẫu là 30 căn nhà bài toán sử dụng mô hình Hồi quy tuyến tính đơn biến gồm 1 biến đầu vào là diện tích (đơn vị là 𝑚2) và 1 biến đầu ra là giá bán (đơn vị là triệu đồng).

**Bảng 2.1: Dataset**

|  |  |
| --- | --- |
| Diện tích | Giá nhà |
| 30 | 448.524 |
| 32.4138 | 509.248 |
| 34.8276 | 535.104 |
| 37.2414 | 551.432 |
| … | … |
| 90.3448 | 1374.92 |
| 92.7586 | 1410.16 |
| 95.1724 | 1469.69 |
| 97.5862 | 1478.54 |
| 100 | 1515.28 |

Công thức mô hình hồi quy tuyến tính: 𝑦̂ = 𝑤1𝑥𝑖 + 𝑤0

Các bước của thuật toán Gradient Descent trong bài toán này như sau:

* Khởi tạo giá trị ban đầu cho 𝑤0 và 𝑤1.
* Xác định hàm mất mát (loss function) để đo lường sai số giữa giá trị dự đoán và giá trị thực tế.

Công thức loss function: 𝐿 = 1 ⋅ ∑𝑁 (𝑦̂

− 𝑦)2

𝑁 𝑖=1 𝑖

Trong đó N là số lượng điểm dữ liệu, y là giá bán thực tế, và 𝑦̂𝑖 là giá trị dự đoán.

* Tính đạo hàm riêng của hàm mất mát.

1. Đạo hàm theo 𝑤0:

30

𝜕𝐿 1

1. Đạo hàm theo 𝑤1:

𝜕𝑤0

= 𝑁 ∑ 2((𝑤1𝑥𝑖 + 𝑤0) − 𝑦𝑖)

𝑖=1

30

𝜕𝐿 1

𝜕𝑤1

= 𝑁 ∑ 2((𝑤1𝑥𝑖 + 𝑤0) − 𝑦𝑖) ⋅ 𝑥𝑖

𝑖=1

1. Cập nhật giá trị của 𝑤0 và 𝑤1. Công thức cập nhật:

𝑤0

= 𝑤0

− 𝑙𝑒𝑎𝑟𝑛𝑖𝑛𝑔 𝑟𝑎𝑡𝑒 ⅆ𝐿

ⅆ𝑤0

(𝑤1

, 𝑤0

) và 𝑤1

= 𝑤1

− 𝑙𝑒𝑎𝑟𝑛𝑖𝑛𝑔 𝑟𝑎𝑡𝑒 ⅆ𝐿

ⅆ𝑤1

(𝑤1

, 𝑤0)

Trong đó learning\_rate là một hằng số được gọi là tốc độ học (learning rate) và ⅆ𝐿 , ⅆ𝐿 là

đạo hàm riêng của hàm mất mát theo 𝑤0 và 𝑤1.ⅆ𝑤0

ⅆ𝑤1

1. Thuật toán Gradient Descent sẽ tiếp tục cập nhật giá trị của 𝑤1 và 𝑤0 trong mỗi vòng lặp, để tìm ra giá trị của các tham số mô hình sao cho hàm mất mát (loss function) đạt giá trị nhỏ nhất. Khi thuật toán kết thúc, ta sẽ thu được giá trị tối ưu của 𝑤1 và 𝑤0 để xây dự báo giá bán nhà dựa trên diện tích.

### Ưu điểm của thuật toán Gradient Descent:

* Đơn giản và dễ hiểu: Gradient Descent là thuật toán tối ưu hóa phổ biến với cách thức hoạt động đơn giản, dễ hiểu. Thuật toán này dựa trên việc lặp đi lặp lại các bước cập nhật tham số theo hướng dốc âm của hàm mất mát, giúp tìm kiếm giá trị tối ưu cho các tham số của mô hình dự báo.
* Hiệu quả: Gradient Descent có hiệu quả cao trong việc giải quyết các bài toán tối ưu hóa, đặc biệt là khi số lượng tham số của mô hình không quá lớn.
* Linh hoạt: Gradient Descent có thể được áp dụng cho nhiều loại mô hình dự báo khác nhau, bao gồm cả mô hình tuyến tính và phi tuyến tính.

### Nhược điểm của thuật toán Gradient Descent:

* Tốc độ hội tụ: Tốc độ hội tụ của Gradient Descent có thể chậm, đặc biệt là khi mô hình có nhiều tham số hoặc dữ liệu nhiễu.
* Cực tiểu địa phương: Gradient Descent có thể bị mắc kẹt tại các cực tiểu địa phương, dẫn đến kết quả dự báo không chính xác.
* Lựa chọn learning rate: Việc lựa chọn learning rate phù hợp là rất quan trọng để đảm bảo hiệu quả của Gradient Descent. Nếu learning rate quá nhỏ, quá trình hội tụ sẽ diễn ra chậm chạp. Ngược lại, nếu learning rate quá lớn, thuật toán có thể không hội tụ được.

### Cách khắc phục các hạn chế của Gradient Descent:

* Sử dụng các biến thể của Gradient Descent: Có nhiều biến thể của Gradient Descent được phát triển để cải thiện tốc độ hội tụ và khả năng tránh các cực tiểu địa phương, ví dụ như Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent,….
* Chuẩn hóa dữ liệu: Việc chuẩn hóa dữ liệu giúp thu hẹp khoảng giá trị của các thuộc tính, từ đó giúp Gradient Descent hoạt động hiệu quả hơn.
* Sử dụng kỹ thuật cross-validation: Kỹ thuật cross-validation giúp đánh giá hiệu quả của mô hình dự báo và lựa chọn learning rate phù hợp.
  1. **Phương pháp**

### Các thư viện cần thiết:

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

* Numpy: Được sử dụng để thực hiện các phép toán số học trong ma trận.
* Pandas: Được sử dụng để đọc dữ liệu từ tệp CSV.
* Matplotlib.pyplot: Được sử dụng để vẽ đồ thị và biểu đồ.

### Đọc và định dạng dữ liệu:

data = pd.read\_csv('data\_linear.csv').values

#shape[0] đại diện cho sổ hàng, shape[1] đại diện cho cột N = data.shape[0]

# x là biến độc lập (diện tích), lấy tất cả các dòng ở cột 1

x = data[:, 0].reshape(-1, 1)#reshape là chuyển ma trận sang ma trận

1 cột

# y là biến phụ thuộc (giá nhà), lấy tất cả các dòng ở cột 2 y = data[:, 1].reshape(-1, 1)

* pd.read\_csv('data\_linear.csv'): Đọc dữ liệu từ tệp CSV có tên "data\_linear.csv" vào một DataFrame của Pandas.
* .values: Chuyển đổi DataFrame thành mảng NumPy
* N = data.shape[0]: Tính số lượng điểm dữ liệu (N) bằng cách lấy kích thước hàng

(0) của mảng dữ liệu.

* x = data[:, 0].reshape(-1, 1): Trích xuất cột đầu tiên (diện tích) vào mảng NumPy x và định dạng lại thành mảng hai chiều với một cột.
* y = data[:, 1].reshape(-1, 1): Tương tự, trích xuất cột thứ hai (giá) vào mảng NumPy y và định dạng lại thành mảng hai chiều với một cột.

### Khởi tạo các tham số ban đầu:

# Giá trị đầu của trọng số

w1 = 0

w0 = 0

# The learning Rate alpha = 0.0001

# The number of iterations to perform gradient descent epochs = 50

* w1 = 0, w0 = 0: Khởi tạo giá trị ban đầu cho các trọng số w1 (hệ số cho diện tích) và w0 (hệ số chặn) đều bằng 0.
* alpha = 0.0001: Thiết lập tốc độ học tập (alpha) là 0.0001, kiểm soát mức độ điều chỉnh trọng số trong quá trình tối ưu hóa.
* epochs = 50: Thiết lập số lần lặp (epochs) của thuật toán gradient descent là 50, tượng trưng cho số lần cập nhật trọng số.

### Định nghĩa đạo hàm tính toán:

n = len(x) # Number of elements in X def dL\_w1(w1,w0):

return (np.sum(2 \*(w1 \* x + w0 -y) \* x))/N def dL\_w0(w1,w0):

return (np.sum(2 \*(w1 \* x + w0 - y)))/N # Performing Gradient Descent

for i in range(epochs):

print('i =',i,'w1 =',w1,'w0 =',w0) w1 = w1 - alpha\*dL\_w1(w1, w0)

w0 = w0 - alpha\*dL\_w0(w1, w0)

* n = len(x): Tính lại số lượng điểm dữ liệu (n) bằng độ dài của mảng x.
* dL\_dw1(w1, w0): Định nghĩa hàm tính đạo hàm của hàm mất mát theo w1.
* dL\_dw0(w1, w0): Định nghĩa hàm tính đạo hàm của hàm mất mát theo w0.
* Sử dụng công thức đạo hàm của hàm mất mát MSE (Mean Squared Error) cho hồi quy tuyến tính.
* Tính toán tổng bình phương sai lệch giữa giá trị dự đoán (w1 \* x + w0) và giá trị thực tế (y) nhân với diện tích (x), chia cho tổng số điểm dữ liệu (N).

### Thực hiện Gradient Descent:

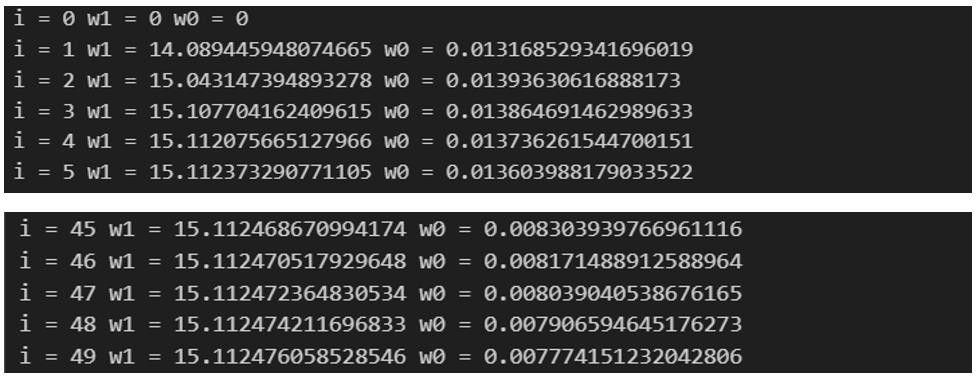
Vòng lặp for thực hiện chạy 50 epochs. Mỗi lần lặp thực hiện các bước sau: In thông tin:

* i: In ra số thứ tự của vòng lặp hiện tại.
* w1, w0: In ra giá trị của trọng số w1 và w0 tại thời điểm hiện tại.

### Cập nhật trọng số:

* w1 = w1 - alpha \* dL\_dw1(w1, w0):
* w0 = w0 - alpha \* dL\_dw0(w1, w0)
* Cập nhật giá trị của w1 bằng cách trừ đi tích của alpha (tốc độ học) và đạo hàm của hàm mất mát theo w1 (dL\_dw1) và cập nhật giá trị của w0 bằng cách trừ đi tích của alpha và đạo hàm của hàm mất mát theo w0 (dL\_dw0).
* alpha là giá trị nhỏ (0.0001) để đảm bảo bước cập nhật không quá lớn, giúp tránh vượt qua giá trị tối ưu.
* Dấu trừ đảm bảo di chuyển w1 theo hướng giảm thiểu hàm mất mát.

### Kết quả sau khi chạy epochs:



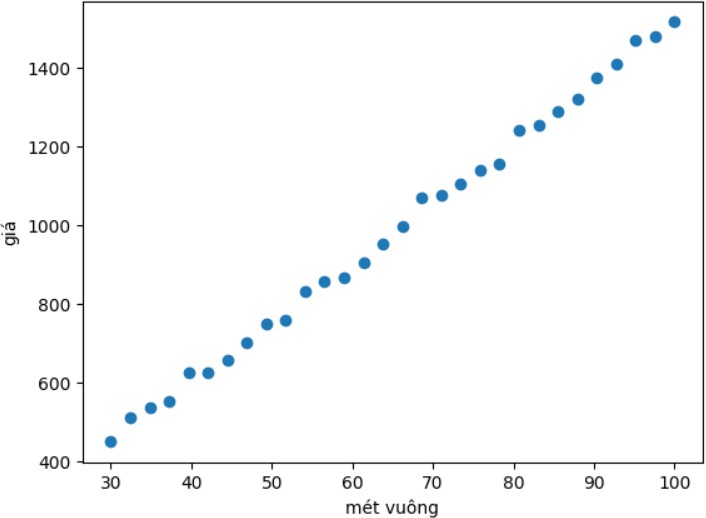
### Vẽ biểu đồ phân tán:Hình 2 .1: Kết quả sau khi chạy epochs

plt.scatter(x, y) plt.xlabel('mét vuông') plt.ylabel('giá')

plt.plot((x[0], x[n-1]),(x[0]\*w1 + w0, x[n-1]\*w1 + w0), 'r') plt.show()

* plt.scatter(x, y): Vẽ biểu đồ phân tán của dữ liệu, với x trục hoành và y trục tung.
* plt.xlabel('mét vuông'): Đặt nhãn cho trục x là "mét vuông".
* plt.ylabel('giá'): Đặt nhãn cho trục y là "giá".
* Hiển thị lại biểu đồ dạng điểm rải rác (scatter) với các điểm dữ liệu.
* Giữ nguyên nhãn trục x và y.
* Vẽ một đường thẳng màu đỏ ((x[0], x[n-1]),(x[0]\*w1 + w0, x[n-1]\*w1 + w0)) biểu thị đường hồi quy được tính toán dựa trên giá trị w1 và w0 cuối cùng.

### Biểu đồ phân tán:



**Hình 2. 2: Biểu đồ phân tán**

In ra giá dự đoán y1 cho một diện tích cụ thể (50 mét vuông) được tính toán bằng y1 = x1 \* w1 + w0.

x1 = 50

y1 = x1\*w1 + w0

print("Gia nha co dien tich 50m2 la:", y1)

### Kết quả dự đoán:

**Hình 2. 3: Kết quá dự đoán**

## 2.3. Sử dụng thuật toán Stochastic Gradient Descent, Mini-batch Gradient Descent và RMSprop để tối ưu hoá bài toán dự báo thức ăn nhanh.

### Tổng quan bài toán

Trong ngành dịch vụ ẩm thực, việc nắm bắt được những thay đổi trong sở thích và nhu cầu của khách hàng có thể quyết định sự phát triển và tồn tại của các doanh nghiệp đó. Để thực hiện điều này, việc áp dụng các phương pháp tối ưu hóa gradient trong quá trình huấn luyện mạng nơ-ron là một cách hiệu quả, cụ thể là áp dụng công nghệ này vào để theo dõi và phân loại đồ ăn được bán ra.

Trong đồ án này, chúng em tập trung nghiên cứu các chuỗi kinh doanh đồ ăn nhanh, điểm chung của những cửa hàng bán đồ ăn nhanh là thực đơn khá đơn giản, các món ăn lại làm rất nhanh, đáp ứng được nhu cầu nhanh chóng - tiện lợi nên số lượng người tiêu thụ một ngày cũng đông đảo. Như vậy, với tần suất bán ra liên tục như thế, việc sử dụng các phương pháp tối ưu hóa gradient trong quá trình huấn luyện mạng nơ-ron để theo dõi và phân loại lượng thức ăn được bán ra trở nên cần thiết. Bởi vì thông qua đó, chúng ta có thể nhận biết món ăn nào được tiêu thụ nhiều nhất, từ đó tiếp cận được với sở thích của khách hàng. Ngoài ra, việc theo dõi số lượng bán ra của mỗi món ăn cũng giúp cân chỉnh sản xuất một cách hợp lí, tối ưu chi phí và tăng cường lợi nhuận, đồng thời tránh lãng phí tài nguyên. Điều này không chỉ mang lại lợi ích kinh doanh mà còn giúp bảo vệ môi trường đáng kể.

Để đạt được mục tiêu đã đề ra, chúng em lựa chọn ba phương pháp cụ thể bao gồm: Mini- Batch Gradient Descent, Stochastic Gradient Descent và RMSprop để phân loại hình ảnh của ba loại đồ ăn phổ biến: Burger, Donut và Sandwich.

### Lý thuyết các thuật toán

* **Mini-batch Gradient Descent**

Mini-batch Gradient Descent là một biến thể của phương pháp tối ưu hóa Gradient Descent. Khác với phương pháp nguyên thuỷ của nó, trong Mini-batch Gradient Descent, dữ liệu huấn luyện được chia thành các mini-batches nhỏ hơn, và việc cập nhật trọng số được thực hiện sau mỗi mini-batch.

Việc chia nhỏ dữ liệu huấn luyện thành các mini-batches giúp giảm bớt thời gian tính toán và tiết kiệm bộ nhớ, đặc biệt là đối với các tập dữ liệu lớn. Đồng thời, việc sử dụng mini- batches cũng giúp tạo ra các bước cập nhật trọng số có tính ngẫu nhiên hơn, giúp mô hình tránh được các điểm cực tiểu cục bộ và có khả năng tìm được điểm cực tiểu toàn cục tốt hơn.

* **Stochastic Gradient Descent**

Stochastic Gradient Descent là một biến thể khác của Gradient Descent. Đặc điểm chính của Stochastic Gradient Descent là việc cập nhật trọng số được thực hiện dựa trên từng điểm dữ liệu huấn luyện một cách ngẫu nhiên. Khác với phương pháp Gradient Descent truyền thống, Stochastic Gradient Descent không đợi đến khi duyệt qua toàn bộ tập dữ liệu để cập

nhật trọng số một lần, mà thường cập nhật sau mỗi điểm dữ liệu. Điều này giúp giảm bớt thời gian tính toán và bộ nhớ cần thiết, đặc biệt là đối với các tập dữ liệu lớn.

Stochastic Gradient Descent có tính ngẫu nhiên trong việc chọn mẫu dữ liệu để cập nhật trọng số, điều này giúp tránh được các điểm cực tiểu cục bộ và tăng khả năng tìm ra điểm cực tiểu toàn cục. Tuy nhiên, tính ngẫu nhiên cũng có thể làm cho quá trình huấn luyện dao động nhiều hơn và có thể cần thêm thời gian để đạt được điểm hội tụ.

* **RMSprop**

RMSprop (Root Mean Square Propagation) là một phương pháp tối ưu hóa trong máy học và deep learning, dùng để cập nhật trọng số của mạng nơ-ron trong quá trình huấn luyện. So với hai phương pháp trên, RMSprop thường cho hiệu suất tốt và hội tụ nhanh hơn so với Stochastic Gradient Descent và Mini-batch Gradient Descent trong nhiều trường hợp, đặc biệt là trong các mô hình sâu và phức tạp.

Phương pháp này được thiết kế để giảm tác động của gradient bất ổn trong quá trình huấn luyện. RMSprop sử dụng learning-rate khác nhau cho từng tham số của mạng nơ-ron dựa trên tỷ lệ của gradient trung bình bình phương của các gradient trước đó. Bằng cách này, RMSprop có khả năng thích ứng với tốc độ học của từng tham số, giúp tăng tính ổn định của quá trình huấn luyện.

### Các nghiên cứu liên quan

Trong lĩnh vực xử lý dữ liệu, việc phân loại thức ăn nhanh đã thu hút sự quan tâm lớn từ cộng đồng nghiên cứu. Một số nghiên cứu trước đây tập trung nghiên cứu và sử dụng các thuật toán phân loại tối ưu hoá để cải thiện hiệu suất của các mô hình. Ví dụ, nghiên cứu của Zhang et al.(2019) đã đi sâu vào ứng dụng của hai thuật toán quan trọng là SGD và Mini-batch Gradient Descent trong bài toán phân loại hình ảnh. Từ những kết quả này, chúng ta có thể hiểu được cách các phương pháp tối ưu này hoạt động và cải thiện khả năng phân loại và nhận dạng thức ăn từ hình ảnh. Ngoài ra, Akhi, Akter và Uddin (2018) đã tập trung vào thuật toán SGD trong nghiên cứu của họ về việc phân loại thức ăn nhanh. Các nghiên cứu này cung cấp một cơ sở quan trọng cho việc áp dụng các phương pháp tối ưu hoá vào bài toán phân loại thức ăn nhanh từ hình ảnh, giúp tối ưu hoá quá trình hoạt động và tăng cường hiệu suất của mô hình.

#### Ưu điểm và khuyết điểm của các nghiên cứu liên quan.

Các nghiên cứu trước đây đã tận dụng cơ sở lý luận mạnh mẽ cùng với việc áp dụng các phương pháp và kỹ thuật nghiên cứu chi tiết và tối ưu. Tuy nhiên, một số nghiên cứu vẫn còn đối mặt với thách thức trong quá trình học, chẳng hạn như sự không cân bằng trong dữ liệu và sai lệnh có thể xuất hiện do sử dụng phương pháp Gradient Descent truyền thống, dẫn đến kết quả tối cưu hoá kém (Yingjie Tian, Yuqi Zhang and Haibin Zhang,2023). Để khắc phục những điểm yếu này, chúng em sẽ cố gắng áp dụng những phương pháp tiên tiến và tối ưu hơn nhưStochastic Gradient Descent, Mini-batch Gradient Descent và RMSprop và so sánh cả ba thuật toán để tìm ra thuật toán tối ưu tố nhất. Đồng thời, chúng em sẽ thực hiện các thử nghiệm và phân tích chi tiết để đảm bảo tính tin cậy và hiệu suất của kết quả nghiên cứu.

#### Mô tả dữ liệu

Dữ liệu trong nghiên cứu bao gồm 4500 hình ảnh, có ba loại thức ăn chính là Burger, Donut và Sandwich. Mỗi loại thức ăn có 1500 hình ảnh, tạo thành một mẫu dữ liệu cân bằng. Các hình ảnh được thu thập từ nhiều nguồn khác nhau, bao gồm sự khác biệt về màu sắc, hình dáng và kích thước của từng loại thức ăn. Dữ liệu đã được chuẩn bị và tiền xử lý đảm bảo tính đa dạng và thực tế. Mục tiêu là để huấn luyện và đánh giá hiệu suất của mô hình trong phân loại thức ăn nhanh.

#### Phương pháp thực hiện

* Công cụ và môi trường

Sử dụng ngôn ngữ lập trình Python và thư viện TensorFlow/Keras để triển khai mô hình và huấn luyện. Môi trường thực thi là máy tính cá nhân với các tài nguyên phần cứng như CPU và GPU. Các công cụ để chạy mô hình như Visual Studio Code, Jupyter Notebook hoặc Google Colab.

* Tiền xử lý dữ liệu

Các bước tiền xử lý bao gồm chuẩn hoá giá trị pixel của các hình ảnh về khoảng [0,1] bằng cách chia tỷ lệ giá trị của mỗi pixel cho 255. Điều này giúp làm giảm độ lớn của dữ liệu đầu vào và làm cho quá trình huấn luyện mô hình hiệu quả hơn. Tiếp theo, đặt kích thước cho batch (tuỳ theo thuật toán) và định dạng các nhãn dưới dạng số nguyên. Cuối cùng, xáo trộn đữ liệu trước khi tạo batch, giúp đảm bảo rằng mô hình không học theo thứ tự của các hình ảnh trong thư mục.

train\_datagen = ImageDataGenerator(rescale=1./255) train\_generator = train\_datagen.flow\_from\_directory(

'Food', target\_size=(224, 224),

batch\_size=…,#Tuỳ theo thuật toán class\_mode='sparse', shuffle=True)

* Mô hình mạng nơ-ron

Sử dụng mạng nơ-ron tích chập (CNN) để nhận dạng hình ảnh. Mô hình bao gồm các lớp Conv2D, MaxPooling2D, Flatten, Dense và Dropout để tránh overfitting.

Mạng nơ-ron có cấu trúc như sau:

* Lớp Conv2D: Sử dụng 32 bộ lọc với kích thức kernel là (3,3). hàm kích hoạt là ReLU,và kích thước đầu vào là (224, 224, 3) (chiều cao, chiều rộng, số kênh màu).
* Lớp MaxPooling2D: Áp dụng phép gộp cực đại với kích thước pool là (2, 2) để giảm kích thước của feature map.
* Lớp Flatten: Chuyển đổi các feature map thành một vector 1 chiều trước khi đưa vào các lớp kết nối đầy đủ.
* Lớp Dense: Là lớp kết nối đầy đủ với 512 đơn vị và hàm kích hoạt là ReLU.
* Lớp Dropout: Sử dụng để ngăn chặn overfitting bằng cách loại bỏ ngẫu nhiên một phần các đơn vị trong quá trình huấn luyện.
* Lớp Dense cuối cùng: Là lớp kết nối đầy đủ với 3 đơn vị (tương ứng với số lớp đầu ra), không có hàm kích hoạt được áp dụng ở đây vì đầu ra sẽ được sử dụng với hàm softmax trong quá trình tính toán loss function trong quá trình huấn luyện

model = Sequential([ layers.Conv2D(32,(3,3),padding='same',activation='relu',

input\_shape=(224, 224, 3)),

layers.MaxPooling2D((2, 2)), layers.Conv2D(64,(3,3),padding='same',activation='relu'),

layers.MaxPooling2D((2, 2)), layers.Conv2D(128,(3,3),padding='same',activation='relu'), layers.MaxPooling2D((2, 2)), layers.Conv2D(256,(3,3),padding='same',activation='relu'), layers.MaxPooling2D((2, 2)),

layers.Flatten(),

layers.Dense(512, activation='relu'), layers.Dropout(0.5),

layers.Dense(3)

])

* Biên dịch mô hình

Sử dụng hàm compile để biên dịch mô hình với các tham só như optimizer, loss, learning rate. Tuỳ vào thuật toán tối ưu mà chúng ta sử dụng thì sẽ điều chỉnh các tham số khác nhau.

* Huấn luyện mô hình

Sau khi biên dịch mô hình, chúng ra có thể bắt đầu quá trình huấn luyện bằng cách sử dụng phương thức fit của đối tượng mô hình. Trong phương thức này, chúng ta cần chỉ định dữ liệu huấn luyện và các tham số như số lượng epoch, kích thước batch, và các thông số khác.

Sau khi kết thúc quá trình huấn luyện đầu tiên, chúng ta sẽ tiến hành dự đoán cho hình ảnh mới bằng các sử dụng các trọng số đã được huấn luyện trước đó. Dưới đây là quá trình thực hiện:

# Định nghĩa lớp cho dự đoán

classes = ['burger', 'donut', 'sandwich']

# Load trọng số của mô hình

model.load\_weights('RMS.weights.h5') # điều chỉnh cái này thành mô hình trọng số muốn dùng

# Đường dẫn đến hình ảnh mới bạn muốn đưa ra dự đoán new\_image\_path = 'Sandwich-Train (2).jpeg'

# Load và tiền xử lý hình ảnh mới

img = image.load\_img(new\_image\_path, target\_size=(224, 224)) img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0) # Thêm một chiều để phù hợp với shape của batch

img\_array /= 255. # Chuẩn hóa hình ảnh

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

predicted\_class\_index = np.argmax(predictions) predicted\_class = classes[predicted\_class\_index] probability = predictions[0][predicted\_class\_index]

# In kết quả

print("Predict the dish in the photo is", predicted\_class)

* Đầu tiên, chúng tôi đã định nghĩa danh sách các lớp mà mô hình có thể dự đoán.
* Xác định đường dẫn đến hình ảnh mới mà chúng tôi muốn mô hình dự đoán.
* Hình ảnh đã được tải, tiền xử lý và chuẩn bị để đưa vào mô hình để dự đoán.
* Cuối cùng, sử dụng mô hình để dự đoán lớp của hình ảnh và in ra kết quả dự đoán.

#### Điều chỉnh tham số phù hợp

**Mini-batch Gradient Descent**

Đối với thuật toán này chúng ta điều chỉnh các tham số như sau:

Trong phần tiền xử lý dữ liệu đặt batch = 20

Trong phần biên dịch mô hình đặt learning rate = 0.01

Trong phần huấn luyện mô hình đặt epochs = 35

**Stochastic Gradient Descent**

Đối với thuật toán này chúng ta điều chỉnh các tham số như sau:

Trong phần tiền xử lý dữ liệu đặt batch = 1

Trong phần biên dịch mô hình đặt learning rate = 0.001

Trong phần huấn luyện mô hình đặt epochs = 25

**RMSprop**

Đối với thuật toán này chúng ta điều chỉnh các tham số như sau:

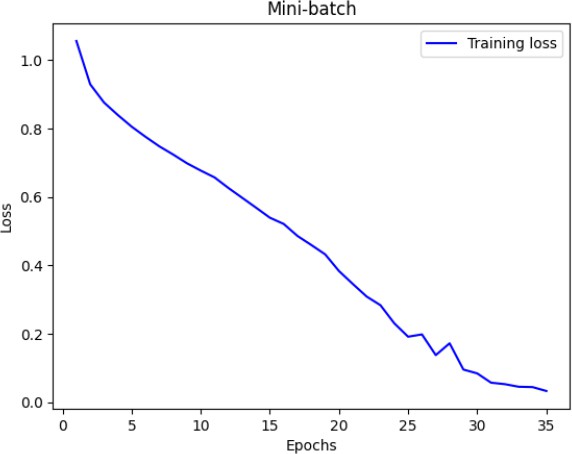
Trong phần tiền xử lý dữ liệu đặt batch = 20

Trong phần biên dịch mô hình đặt learning rate = 0.00001

Trong phần huấn luyện mô hình đặt epochs = 30

#### Biểu đồ quá trình tìm điểm tối ưu

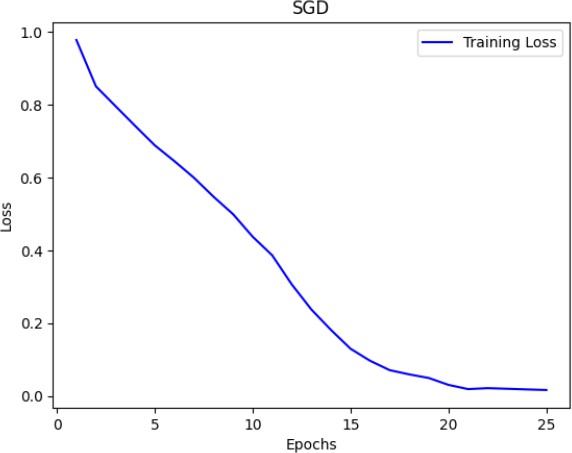
* Mini-batch Gradient Descent



**Hình 3. 1: Biểu đồ thuật toán Mini-batch Gradient Descent**

Nhìn vào biểu đồ, ta thấy giá trị độ lỗi giảm mạnh mẽ từ vòng lặp đầu tiên đến vòng lặp 25, từ vòng lặp 25 đến 28 có sự dao động nhẹ, có lẽ là mô hình đang gặp phải trường hợp overfitting. Nhưng sau đó dần ổn định và mức độ chính xác dần cải thiện đến vòng lặp cuối cùng. Dựa vào kết quả của mô hình, ta có thể kết luận số lượng epochs 35 là khá hợp lí để mô hình vượt qua một số dao động và tìm được điểm tối ưu.

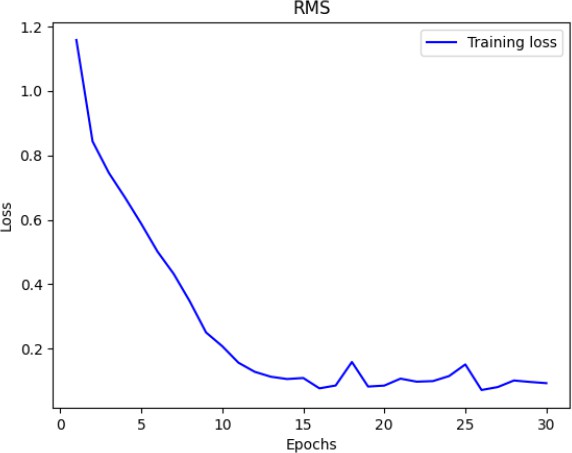
* Stochastic Gradient Descent



**Hình 3. 2: Biểu đồ thuật toán Stochastic Gradient Descent**

Nhìn vào biểu đồ, ta thấy độ lỗi giảm mạnh mẽ ở những vòng lặp đầu tiên, sau đó dần dần ổn định từ vòng lặp 17 đến 25 để tiến gần đến 0. Khoảng từ vòng lặp 23 đến 25 độ lỗi dao động từ 0 đến 0.1. Dựa trên điều này, có thể kết luận rằng mô hình đã hội tụ và đạt được một mức độ chính xác mong muốn sau khoảng 23 vòng lặp, và sau đó mức độ chính xác tiếp tục cải thiện nhưng ở mức độ nhỏ. Điều này có thể cho thấy mô hình đã biểu diễn tốt và không cần phải huấn luyện thêm đếm vòng lặp thứ 25. Dựa vào kết quả cùa mô hình trên ta có thể cân nhắc điều chỉnh số epochs khoảng 23 để tiết kiệm thời gian hơn.

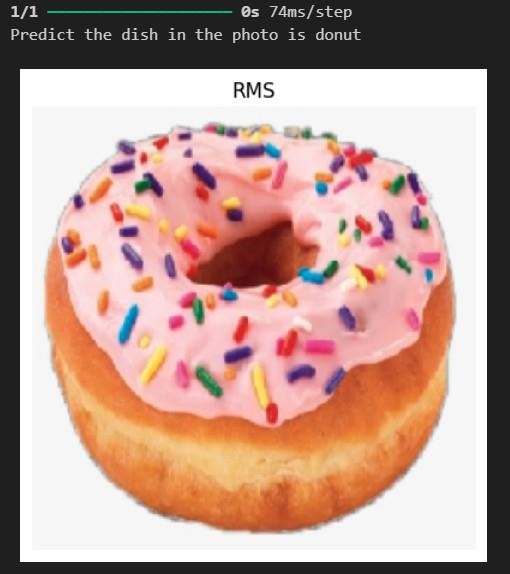
* RMSprop



**Hình 3. 3: Biểu đồ thuật toán RMSprop**

Nhìn vào biểu đồ, giá trị của độ lỗi ở vòng lặp đầu tiên khá cao (gần 1.2), quá trình học cho thấy độ lỗi giảm mạnh từ vòng lặp đầu đến vòng lặp 10. Từ vòng lặp 10 trở đi, tốc độ giảm của độ lỗi chậm lại, cho thấy mô hình đã hội tụ và cần một số vòng lặp ít hơn để tiếp tục cải thiện. Có sự dao động nhẹ của độ lỗi ở vòng lặp 18 và 27. Ở vòng lặp cuối cùng độ lỗi đạt giá trị thấp nhất là 0.725, tuy nhiên ở vòng lặp 27 độ lỗi cũng đạt giá trị gần như thấp nhất là 0.749. Dựa vào kết quả cùa mô hình trên ta có thể cân nhắc điều chỉnh số epochs khoảng 27 để tiết kiệm thời gian hơn.

#### Kết quả dự đoán

**Hình 3. 5: Mini-batch Gradient Descent** **Hình 3. 6: RMSprop**



**Hình 3. 4: Stochastic Grandient Descent**

##### So sánh

* Thời gian huấn luyện:

SGD: Mất 95 phút cho 25 epoch.

* Điểm mạnh: Đạt được độ chính xác cao.
* Điểm yếu: Thời gian huấn luyện khá lâu. RMSprop: Mất 49 phút cho 30 epoch.
* Điểm mạnh: Thời gian huấn luyện trung bình, đạt được độ chính xác tốt sau số lượng epoch vừa phải.
* Điểm yếu: Cần điều chỉnh lại tham số để cải thiện hiệu suất mô hình và tránh được những dao động trong quá trình dự đoán.

Mini-batch Gradient Descent: Mất 40 phút cho 35 epoch.

* Điểm mạnh: Huấn luyện nhanh nhất.
* Điểm yếu: Gặp vấn đề khi dự đoán, cần cải thiện tính ổn định và độ chính xác của mô hình.
* Độ chính xác và loss:

SGD: Độ chính xác là 0.9983 sau 25 epoch, với loss là 0.0103 sau 25 epoch.

RMSprop: Đạt được độ chính xác tốt nhất ở khoảng 0.9867 với loss là 0.0725 sau 30 epoch. Mini-batch Gradient Descent: Đạt được độ chính xác cao nhất là 0.9928 với loss là 0.0317 sau 35 epoch.

## 2.4. Thảo luận

Mặc dù ba phương pháp tối ưu hóa này đều mang lại hiệu suất huấn luyện tốt, nhưng vẫn còn cơ hội để cải thiện và tối ưu hóa quá trình huấn luyện mô hình. Cải thiện có thể đạt được thông qua việc tinh chỉnh siêu tham số, xử lý dữ liệu nhiễu và chọn kích thước mini-batch phù hợp. Ngoài ra, cũng có thể giải quyết các vấn đề như xử lý đa dạng định dạng ảnh, tăng cường dữ liệu và cải thiện hiệu suất dự đoán để đảm bảo tính nhất quán và độ chính xác của mô hình. Bằng cách này, chúng ta có thể tối ưu hóa hiệu suất và độ chính xác của mô hình, đồng thời tăng cường khả năng áp dụng trong các tình huống thực tế.

## 2.5. Kết luận

Mục đích của nghiên cứu này là việc sử dụng các phương pháp tối ưu hóa gradient trong quá trình huấn luyện mạng nơ-ron để theo dõi và phân loại lượng thức ăn được bán ra trở nên cần thiết. Ngoài ra, việc theo dõi số lượng bán ra của mỗi món ăn cũng giúp cân chỉnh sản xuất một cách hợp lí, tối ưu chi phí và tăng cường lợi nhuận, đồng thời tránh lãng phí tài nguyên. Điều này không chỉ mang lại lợi ích kinh doanh mà còn giúp bảo vệ môi trường đáng kể.

Chúng em đã huấn luyện thành công và so sánh ba thuật toán tối ưu cụ thể bao gồm: Mini-Batch Gradient Descent, Stochastic Gradient Descent và RMSprop để phân loại hình ảnh của ba loại đồ ăn phổ biến: Burger, Donut và Sandwich. Kết quả cho thấy thuật toán

Stochastic Gradient Descent có độ chính xác cao nhất là 0.9983 và với loss là 0.0103 sau 25 epochs. Thứ hai, là thuật toán Mini-batch Gradient Descent có độ chính xác 0.9928 với loss là 0.0317 sau 35 epochs. Cuối cùng, là thuật toán RMSprop là 0.9867 với loss là 0.0725 sau 30 epoch. Tuy nhiên, kết quả so sánh độ chính xác của 3 thuật toán này cũng không chính xác hoàn toàn vì các tham số của mỗi thuật toán là khác nhau.

Stochastic Gradient Descent có độ chính xác cao nhất là 0.9983 và với loss là 0.0103 sau 25 epochs. Thứ hai, là thuật toán Mini-batch Gradient Descent có độ chính xác 0.9928 với loss là 0.0317 sau 35 epochs. Cuối cùng, là thuật toán RMSprop là 0.9867 với loss là 0.0725 sau 30 epoch. Tuy nhiên, kết quả so sánh độ chính xác của 3 thuật toán này cũng không chính xác hoàn toàn vì các tham số của mỗi thuật toán là khác nhaU.