|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG CNTT & TRUYỀN THÔNG**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  logo_128  **BÁO CÁO BÀI TẬP LỚN**  **TÍNH TOÁN HIỆU NĂNG CAO**  **(*Mã học phần: IT5408*)**  Đề tài : *Học sâu – Deep Learning (MPI)*   |  |  | | --- | --- | | ĐIỂM | NHẬN XÉT CỦA GIẢNG VIÊN | |  |  |  |  |  | | --- | --- | | Giảng viên hướng dẫn : | TS . Vũ Văn Thiệu | | Học viên : | Nguyễn Trần Hải Phong – 20232019M  Đào Quốc Phong - 20242024M |   Hà Nội, 12 - 2024 |

**Mục lục**

[**I.** **Giới thiệu** 3](#_Toc185625418)

[**II.** **Bài toán** 3](#_Toc185625419)

[**III.** **Thuật toán** 4](#_Toc185625420)

[**IV.** **Thuật toán song song** 5](#_Toc185625421)

[**V.** **Kết quả** 5](#_Toc185625422)

[**VI.** **Kết luận** 5](#_Toc185625423)

# **Giới thiệu**

Ngày nay, học sâu đã và đang trở thành một trong những phương pháp chủ lực trong công cuộc phát triển trí tuệ nhân tạo. Ngoài ra học sâu còn góp phần vào việc phát triển các công nghệ tiên tiến khác như phân tích giọng nói, xử lý ngôn ngữ tự nhiên, thị giác máy tính… Tuy nhiên, với sự phức tạp của các mô hình học sâu hiện đại nếu chỉ phụ thuộc vào việc xử lý tuần tự của CPU thì hiệu quả công việc sẽ trở nên kém đi đáng kể khiên cho thời gian huấn luyện trở nên dài hơn. Do vậy, việc áp dụng xử lý song song của hệ thống xử lý GPU sẽ giúp cho việc học máy trở nên hiệu quả cũng như giảm chi phí hệ thống, đặc biệt khi kết hợp xử lý cùng với CPU.

Dựa trên bài báo **“Parallel and Distributed Deep Learning”** của tác giả Vishakh Hegde và Sheema Usmani thuộc đại học Stanford. Nhóm sẽ thực hiện lại mô phỏng việc học sâu sự dụng tính toán song song dựa trên MPI để xác minh tính hiệu quả so với mô hình tuần tự truyền thống. Từ đó rút ra những điểm mạnh cũng như điểm cần phải phát triển trong tương lai khi mở rộng mô hình học máy song song cũng như phát triển các công nghệ xử lý phân tán.

# **Bài toán**

Việc huấn luyện các mạng nơ-ron sâu (Deep Neural Networks) là một công việc tương đối phức tạp vì một vài lý do sau:

- **Số lượng tham số lớn:** Các mô hình hiện đại thường có hàng triệu tham số cần được tối ưu hóa, đòi hỏi nhiều tài nguyên tính toán.

- **Dữ liệu khổng lồ:** Để mô hình học hiệu quả, cần lượng dữ liệu huấn luyện lớn, đôi khi vượt quá khả năng lưu trữ và xử lý của một máy tính đơn lẻ.

- **Thời gian huấn luyện lâu:** Thời gian huấn luyện trên CPU truyền thống có thể kéo dài từ hàng giờ đến hàng tuần, đặc biệt với các mô hình lớn như VGGNet.

- **Giới hạn tài nguyên:** Một số mô hình hoặc dữ liệu không thể vừa với bộ nhớ của một máy tính đơn, làm phát sinh nhu cầu phân tán mô hình hoặc dữ liệu.

Vì vậy nhóm đã tìm hiểu và áp dụng các phương pháp nhằm tăng tốc quá trình huấn luyện nhằm rút ngắn thời gian cũng như tối ưu hóa việc huấn luyện trên hệ thông phân tán nhiều lõi xử lý. Từ đó nhóm đã tìm ra được phương pháp sau:

- **Song song hóa:** Tận dụng khả năng tính toán song song của GPU hoặc nhiều lõi CPU để xử lý đồng thời nhiều dữ liệu.

- **Thuật toán SGD cải tiến:** Phân tích và tối ưu hóa các biến thể của Stochastic Gradient Descent (SGD) như Parallel SGD, ADMM, và Downpour SGD, tập trung vào các bài toán:

* Chi phí tính toán (computation cost).
* Chi phí truyền thông (communication cost).
* Tối ưu hóa trong môi trường không đồng bộ.

Ngoài ra, còn một phương pháp nữa có thể áp dụng trong thực tế đó là sử dụng các **hệ thống phân tán** bằng cách chia nhỏ mô hình hoặc dữ liệu trên nhiều máy tính, sử dụng các thuật toán cập nhật tham số đồng bộ (synchronous) hoặc bất đồng bộ (asynchronous) để giảm chi phí giao tiếp giữa các máy.

# **Thuật toán**

Nhóm quyết định tập trung vào việc sử dụng thuật toán Parallel SGD là biến thể của thuật toán Stochastic Gradient Descent (SGD) được tối ưu cho các tác vụ xử lý song song. Nhóm sử dụng mạng neuron fully connected để thực hiện việc phân loại ảnh cho bộ dữ liệu MNIST dựa trên thuật toán đã nêu trên.

Thuật toán Stochastic Gradient Descent (SGD) là một thuật toán tối ưu hóa thường được sử dụng để huấn luyện các mô hình học máy, đặc biệt trong các trường hợp liên quan đến tập dữ liệu lớn hoặc mạng nơ-ron. Đây là một biến thể của thuật toán Gradient Descent tiêu chuẩn.

* **Gradient Descent (GD):** Gradient Descent là một thuật toán nhằm giảm thiểu hàm mất mát J(θ) bằng cách di chuyển các tham số θ theo hướng ngược lại của gradient: θ:=θ−η∇J(θ)

+ **Ưu điểm**: Chính xác, đặc biệt đối với các tập dữ liệu nhỏ.

+ **Nhược điểm**: Chậm khi áp dụng trên tập dữ liệu lớn, vì cần tính gradient dựa trên toàn bộ dữ liệu ở mỗi bước.

* **Stochastic Gradient Descent (SGD):** SGD cải tiến Gradient Descent bằng cách cập nhật tham số sau khi tính gradient dựa trên một mẫu dữ liệu ngẫu nhiên thay vì toàn bộ tập dữ liệu: θ:=θ−η∇J(θ;x(i),y(i))

Trong đó:

+ (x(i),y(i)): Một mẫu dữ liệu duy nhất được chọn ngẫu nhiên từ tập huấn luyện.

+ η: Tốc độ học (learning rate).

**Ưu điểm:**

**- Nhanh hơn:** Không cần duyệt toàn bộ dữ liệu trong mỗi lần cập nhật.

**- Khả năng tránh local minima:** Do tính ngẫu nhiên, SGD có thể thoát khỏi các điểm cực tiểu cục bộ (local minima) và hội tụ đến điểm cực tiểu toàn cục hoặc điểm yên ngựa.

**- Phù hợp với dữ liệu lớn:** Rất hữu ích khi làm việc với các tập dữ liệu lớn hoặc trực tuyến.

**Nhược điểm:**

**- Biến động lớn:** Các cập nhật có thể không ổn định, dẫn đến đường đi zigzag xung quanh điểm tối ưu.

**- Hội tụ chậm:** Cần điều chỉnh tốt tốc độ học η\etaη, nếu không có thể dẫn đến hội tụ kém hoặc không hội tụ.

SGD thường được sử dụng để huấn luyện các mô hình như:

- Mạng nơ-ron (Neural Networks)

- Hồi quy logistic (Logistic Regression)

- SVM (Support Vector Machines)

# **Thuật toán song song**

# **Kết quả**

# **Kết luận**