Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Ngành: Khoa học máy tính

Môn học: Các kĩ thuật học sâu và ứng dụng - CS431.P11

ADOPT: Modified Adam Can Converge with Any β_2 with the Optimal Rate

Giảng viên hướng dẫn:

GV. Nguyễn Vinh Tiệp

Sinh viên thực hiện:

Trịnh Thị Lan Anh	22520083
Nguyễn Ấn	22520019
Vương Dương Thái Hà	22520375
Lê Văn Giáp	22520363

Thành phố Hồ Chí Minh, 2024

Mục lục

source code here!

1 Giới thiệu

1.1 Tổng quan

Trong lĩnh vực học sâu, việc lựa chọn thuật toán tối ưu hóa đóng vai trò quan trọng trong quá trình huấn luyện mô hình. Một trong những thuật toán phổ biến nhất là Adam [Kingma and Ba, 2014], được biết đến với khả năng điều chỉnh tốc độ học theo từng tham số dựa trên các moment bậc nhất và bậc hai. Tuy nhiên, Adam cũng tồn tại hai điểm yếu chính [Zhou et al., 2018]:

- **Phụ thuộc vào tham số** β_2: Giá trị của β_2 (thông thường được đặt khoảng 0.999) ảnh hưởng đáng kể đến hiệu quả của thuật toán. Điều này gây khó khăn khi áp dụng Adam cho các bài toán khác nhau, đặc biệt là trong những trường hợp giá trị β_2 không tối ưu.
- **Sự phụ thuộc của gradient hiện tại vào ước lượng moment bậc hai**: Việc sử dụng ước lượng moment bậc hai để điều chỉnh gradient đôi khi làm giảm hiệu quả hội tụ, đặc biệt trong các bài toán phức tạp hoặc có noise cao.

Nhằm khắc phục các hạn chế trên, phương pháp Adopt (*ADaptive gradient method with the OPTimal convergence rate*) [Zhang et al., 2024] đã được đề xuất. Adopt cải thiện hiệu suất hội tụ của Adam và loại bỏ sự phụ thuộc quá mức vào β_2. Thuật toán này đạt được tốc độ hội tụ ổn định và tối ưu ở mức $O\left(\frac{1}{\sqrt{T}}\right)$, bất kể giá trị của β_2 được lựa chọn.

Sự ra đời của Adopt đánh dấu một bước tiến trong các phương pháp tối ưu hóa, cung cấp một giải pháp hiệu quả hơn cho các bài toán học sâu phức tạp. Với khả năng khắc phục điểm yếu của Adam, Adopt được kỳ vọng sẽ trở thành một công cụ hữu ích trong việc huấn luyện các mô hình lớn.

1.2 Lý do chọn đề tài

Tầm quan trọng của tối ưu hóa

- Tối ưu hóa hàm mục tiêu là một thành phần quan trọng trong học sâu, đóng vai trò cốt lõi trong việc huấn luyện mô hình. Quá trình tối ưu hóa không chỉ ảnh hưởng đến thời gian huấn luyện mà còn quyết định chất lượng và khả năng tổng quát hóa của mô hình khi áp dụng vào dữ liệu mới.

- Với sự phát triển không ngừng của công nghệ và dữ liệu, các mô hình học sâu ngày càng phức tạp, đòi hỏi số lượng lớn tham số và tài nguyên tính toán. Trong bối cảnh này, việc lựa chọn thuật toán tối ưu hóa phù hợp trở thành yếu tố sống còn. Các thuật toán không chỉ cần đảm bảo khả năng hội tụ đến nghiệm tốt nhất mà còn phải hoạt động hiệu quả với các mô hình lớn, độ phức tạp cao và dữ liệu đa dạng.

$\Longrightarrow$ Chính vì vậy, tối ưu hóa không chỉ là một bài toán kỹ thuật mà còn mang tính chiến lược trong việc phát triển các ứng dụng học sâu hiện đại.

Đề xuất giải pháp mới

- Để giải quyết những hạn chế của các thuật toán hiện tại (đặc biệt là Adam), giải pháp mới mang tên phương pháp Adopt (*ADaptive gradient method with the OPTimal convergence rate*) [Zhang et al., 2024] đã được đề xuất.

- Phương pháp Adopt tập trung vào hai cải tiến chính:

  - Tăng tốc độ hội tụ: Adopt tối ưu hóa cách tính moment bậc hai, giúp thuật toán hội tụ được chứng minh là $O\left(\frac{1}{\sqrt{T}}\right)$, ngay cả trong những trường hợp dữ liệu có nhiễu cao hoặc không ổn định. Điều này giảm đáng kể thời gian huấn luyện và tăng hiệu suất trong các bài toán lớn.
  - Cải thiện sự ổn định trong quá trình huấn luyện: Adopt loại bỏ sự phụ thuộc quá mức vào tham số β_2, vốn là điểm yếu lớn của Adam. Bằng cách này, Adopt đảm bảo sự ổn định trong mọi giai đoạn của quá trình tối ưu hóa, đặc biệt khi mô hình phức tạp hoặc dữ liệu không đồng nhất.

$\Longrightarrow$ Với những cải tiến này, Adopt được kỳ vọng sẽ không chỉ khắc phục điểm yếu của Adam mà còn mở ra một hướng đi mới cho các phương pháp tối ưu hóa hiện đại. Giải pháp này mang lại tiềm năng lớn cho các ứng dụng học sâu trong nhiều lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, và học tăng cường.

2 Các phương pháp tối ưu trước đó

2.1 SGD

Stochastic Gradient Descent (SGD) được giới thiệu lần đầu tiên vào những năm 1960 bởi Robbins và Monro trong một bài báo có tiêu đề *"A Stochastic Approximation Method"* [Robbins and Monro, 1951]. Phương pháp này đã trở thành một công cụ cốt lõi trong lĩnh vực học sâu và trí tuệ nhân tạo nhờ khả năng tối ưu hóa nhanh chóng và hiệu quả trên các tập dữ liệu lớn.

SGD là một biến thể của phương pháp Gradient Descent (GD) truyền thống. Thay vì tính gradient trên toàn bộ tập dữ liệu, SGD tính gradient dựa trên *minibatch* (một phần nhỏ của dữ liệu huấn luyện) tại mỗi bước cập nhật tham số.

Phương trình cập nhật SGD:

$$w_{t+1} \leftarrow w_t - \eta g(t)$$

w_t : Giá trị tham số tại bước t.

η : Learning rate (tốc độ học).

$g(t)$: Gradient ước lượng trên minibatch tại bước t.

Ưu điểm:

- Tốc độ huấn luyện nhanh hơn: Do chỉ sử dụng một phần nhỏ dữ liệu thay vì toàn bộ tập dữ liệu, mỗi bước cập nhật tham số của SGD diễn ra nhanh hơn.
- Hội tụ trong trường hợp hàm mục tiêu lồi: SGD đảm bảo tính hội tụ khi hàm mất mát có dạng lồi.
- Khả năng tổng quát hóa tốt hơn: Với gradient noise trong quá trình cập nhật, SGD có khả năng tránh được các cực tiểu cục tốt hơn so với GD truyền thống.

Nhược điểm

- Phụ thuộc vào learning rate η: Việc lựa chọn tốc độ học phù hợp là rất quan trọng. Nếu η quá lớn, quá trình huấn luyện có thể không hội tụ. Ngược lại, nếu η quá nhỏ, tốc độ hội tụ sẽ rất chậm.
- Phụ thuộc vào điểm khởi tạo: SGD có thể hội tụ đến các nghiệm khác nhau tùy thuộc vào vị trí khởi tạo ban đầu của tham số.
- Ảnh hưởng bởi gradient noise: Gradient tính trên minibatch chứa sai số ngẫu nhiên (noise) so với gradient thực sự trên toàn bộ tập dữ liệu, làm cho đường đi của SGD không ổn định và có thể dao động quanh cực tiểu.

$\Longrightarrow$ SGD là phương pháp tối ưu hóa phổ biến trong học sâu nhờ vào tốc độ và khả năng tổng quát hóa. Tuy nhiên, để SGD hoạt động hiệu quả, cần có chiến lược lựa chọn learning rate thích hợp và các kỹ thuật cải tiến như Momentum [Smith, 2017], RMSProp [Hinton et al., 2012] và Adam [Kingma and Ba, 2014] để khắc phục nhược điểm của nó.

2.2 SGD with Momentum

SGD with Momentum là một cải tiến của phương pháp Stochastic Gradient Descent (SGD) được giới thiệu lần đầu tiên vào những năm 1980 bởi Leslie N. Smith trong một bài báo có tiêu đề *"Cyclical Learning Rates for Training Neural Networks"* [Smith, 2017]. Mặc dù bài báo này không trực tiếp liên quan đến Momentum trong SGD, nó đã gợi ý ý tưởng về việc sử dụng "momentum" để tăng tốc độ hội tụ và cải thiện hiệu suất của thuật toán.

Trong phương pháp này, động lượng được giữ lại từ các lần cập nhật trước nhằm giảm thiểu hiện tượng dao động trong quá trình huấn luyện và giúp mô hình tiến nhanh hơn về phía cực tiểu của hàm mất mát.

SGD với Momentum là một cải tiến của Stochastic Gradient Descent (SGD) nhằm tăng tốc độ hội tụ và giải quyết các vấn đề trong bài toán phi lồi. Ý tưởng chính của Momentum là lưu giữ "động lượng" (*momentum*) từ các lần cập nhật trước đó để tránh dao động quá nhiều và tăng tốc quá trình huấn luyện.

Phương trình cập nhật SGD with mometum:

$$v_t \leftarrow pv_{t-1} + \eta g(t)$$
$$w_t \leftarrow w_t - v_t$$

v_t : Velocity (động lượng) tại bước t.

w_t : Tham số mô hình tại bước t.

p : Hệ số momentum, thường trong khoảng $[0.9, 0.99]$.

η : Tốc độ học (learning rate).

$g(t)$: Gradient ước lượng tại bước t.

Ưu điểm:

- Hội tụ nhanh hơn Với việc lưu giữ động lượng từ các lần cập nhật trước, SGD với Momentum giúp mô hình tiến nhanh hơn về phía cực tiểu.
- Giải quyết bài toán phi lồi: Momentum giúp giảm thiểu hiện tượng dao động xung quanh các cực tiểu cục, đặc biệt trong các hàm mất mát phi lồi.

Nhược điểm:

- Mất nhiều thời gian để đạt được điểm tối ưu: Do giữ lại "động lượng" từ các bước trước, SGD với Momentum có thể đi qua điểm tối ưu và cần thêm thời gian để điều chỉnh.
- Phụ thuộc vào hệ số Momentum: Việc lựa chọn hệ số p không phù hợp có thể dẫn đến việc hội tụ không ổn định hoặc quá chậm.

$\Longrightarrow$ SGD với Momentum là một cải tiến mạnh mẽ của SGD, giúp tăng tốc độ hội tụ và cải thiện hiệu suất trong các bài toán có hàm mất mát phức tạp. Tuy nhiên, để đạt được kết quả tối ưu, cần lựa chọn cẩn thận các siêu tham số như tốc độ học η và hệ số Momentum p.

2.3 AdaGrad

AdaGrad (Adaptive Gradient Algorithm) là một phương pháp tối ưu hóa dựa trên gradient, được đề xuất lần đầu tiên bởi John Duchi, Elad Hazan, and Yoram Singer trong bài báo *"Adaptive Subgradient Methods for Online Learning and Stochastic Optimization"* năm 2011 [Duchi et al., 2011]. Phương pháp này đặc biệt có ích trong các bài toán như xử lý ngôn ngữ tự nhiên, nơi mà một số từ hiếm gặp trong tập dữ liệu cần được cập nhật với tốc độ học chậm hơn để tránh các cập nhật không chính xác và thất bại trong việc hội tụ.

AdaGrad là phương pháp tối ưu hóa dựa trên gradient với ý tưởng chính là điều chỉnh tốc độ học (*learning rate*) cho từng tham số dựa trên tích lũy gradient theo thời gian. Điều này giúp AdaGrad

hoạt động tốt hơn trong các bài toán có đặc trưng hiếm.

Phương trình cập nhật AdaGrad:

$$v_{t+1} \leftarrow v_t + g(t)^2$$
$$w_{t+1} \leftarrow w_t - \frac{\eta}{\sqrt{v_{t+1}} + \varepsilon} g(t)$$

v_t : Tổng bình phương gradient tại bước t (moment bậc hai).

w_t : Tham số mô hình tại bước t.

η : Tốc độ học ban đầu (learning rate).

ε : Số rất nhỏ để tránh chia cho 0 (thường $\varepsilon \approx 10^{-8}$).

$g(t)$: Gradient của hàm mất mát tại bước t.

Ưu điểm

- **Tự điều chỉnh tốc độ học:** AdaGrad tự động điều chỉnh tốc độ học cho từng tham số dựa trên gradient tích lũy. Điều này giúp các tham số có gradient lớn giảm tốc độ học nhanh hơn, trong khi các tham số có gradient nhỏ vẫn được cập nhật với tốc độ lớn hơn.
- **Hiệu quả cho đặc trưng hiếm:** AdaGrad hoạt động tốt trong các bài toán mà một số đặc trưng xuất hiện rất ít (ví dụ: xử lý ngôn ngữ tự nhiên).

Nhược điểm

- **Vanishing gradient:** Tốc độ học giảm dần theo thời gian do tổng bình phương gradient tích lũy tăng lên, khiến mô hình có thể hội tụ quá sớm và chậm lại.
- **Không giải quyết bài toán phi lồi:** AdaGrad không hiệu quả trong các bài toán tối ưu hóa phi lồi phức tạp.

$\Longrightarrow$ AdaGrad là phương pháp tối ưu hóa đơn giản và hiệu quả cho các bài toán có đặc trưng hiếm. Tuy nhiên, nhược điểm lớn nhất là tốc độ học giảm quá nhanh, gây ra hiện tượng *vanishing gradient*. Các biến thể của AdaGrad như RMSProp [Hinton et al., 2012] và Adam [Kingma and Ba, 2014] ra đời nhằm khắc phục hạn chế này.

2.4 RMSprop

RMSprop là một phương pháp tối ưu tốc độ học tập (leaning rate) được đề xuất lần đầu bởi Geoff Hilton vào năm 2012 [Hinton et al., 2012] trong bài giảng trực tuyến trên nền tảng Coursera của ông ấy.

Nguồn gốc của ý tưởng tạo ra RMSprop là bởi vì Adagrad cộng dồn tổng bình phương của gradient vào vector trạng thái . Kết quả là, do không có phép chuẩn hóa, vẫn tiếp tục tăng tuyến tính không ngừng trong quá trình hội tụ của thuật toán. Từ đó dẫn đến tốc độ học giảm quá nhanh, làm quá trình hội tụ bị chậm hoặc dừng sớm. Từ đó RMSProp ra đời như một bản vá đơn giản để tách rời tốc độ định thời ra khỏi tốc độ học thay đổi theo tọa độ (coordinate-adaptive).

Phương trình cập nhật RMSprop:

$$v_{t+1} = \beta v_t + (1-\beta) g_t^2$$

v_{t+1} : Ước lượng moment bậc hai.
β : Hệ số giảm của moment bậc hai.
v_t : Moment bậc hai tại bước t.
g_t : Gradient tại bước t.

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\varepsilon + v_{t+1}}} g_t$$

w_{t+1} : Trọng số sau khi cập nhật.
w_t : Trọng số tại bước t.
η : Tốc độ học (learning rate).
v_{t+1} : Ước lượng moment bậc hai (từ công thức trên).
ε : Số nhỏ để tránh chia cho 0.
g_t : Gradient tại bước t.

Ưu điểm:

- **Điều chỉnh tốc độ học tập:** `RMSprop` sử dụng trung bình tích lũy bình phương gradient để điều chỉnh tốc độ học cho từng tham số. Làm cho thuật toán phù hợp hơn với các bài toán có dữ liệu đạo hàm biến thiên lớn và nhỏ. Mặt khác còn giúp tốc độ học ổn định hơn, tránh tình trạng tốc độ học giảm nhanh về 0 sau các vòng lặp [Hinton et al., 2012].
- **Hiệu quả trong mạng nơ-ron hồi quy:** Sử dụng trung bình tích lũy còn giúp mạng hồi quy học được các mối quan hệ dài hạn mà không bị ảnh hưởng bởi gradient quá lớn hoặc mất thông tin từ gradient nhỏ[De et al., 2018]. Ngoài ra InceptionV4 (Szegedy et al., 2016) [Szegedy et al., 2016] đã sử dụng RMSProp làm thuật toán tối ưu hóa chính, khẳng định hiệu quả của thuật toán này trong bài toán phức tạp .

Nhược điểm:

- **Phụ thuộc vào tham số:** Tham số β vừa là một ưu điểm khắc phục nhược điểm của phương pháp trước, nhưng cũng là một điểm yếu của thuật toán. Bởi vì độ hiệu quả RMSprop phụ thuộc vào cách chọn β. Nếu không cẩn thận, β không phù hợp sẽ khiến bài toán hội tụ chậm hoặc không ổn định. [De et al., 2018]
- **Không đảm bảo lý thuyết về hội tụ:** Dauphin et al. (2015) đã thừa nhận rằng RMSprop thiếu cơ sở lý thuyết đầy đủ, mặc dù nó hoạt động hiệu quả trong thực ngiệm và cho kết quả cao. [Dauphin et al., 2015]

Từ các điều trên, có thể thấy RMSprop là một thuật toán tối ưu hóa để huấn luyện các mạng nơ-ron nhân tạo bằng cách điều chỉnh tốc độ học thích nghi thông qua trung bình động của bình phương gradient. Từ đó giúp hội tụ nhanh và ổn định trong các bài toán không lồi như RNN hay LSTM. Mặc dù phụ thuộc vào tham số hyperparameter và thiếu cơ sở lý thuyết chặt chẽ, RMSProp vẫn được ưa chuộng nhờ tính hiệu quả và dễ sử dụng, đặc biệt trong các ứng dụng học sâu thực tiễn ở thời điểm ra mắt.

2.5 Adam

Adam (Adaptive Moment Estimation) là thuật toán tối ưu có thể được triển khai rộng rãi trong nhiều ứng dụng học sâu khác nhau như thị giác máy tính và xử lý ngôn ngữ tự nhiên hiện nay. Adam được giới thiệu lần đầu tiên tại một hội nghị nổi tiếng dành cho các nhà nghiên cứu học sâu có tên là International Conference on Learning Representations (ICLR) 2014 bởi Kingma và Ba (2014) [Kingma and Ba, 2014].

Phương pháp của `Adam` được thiết kế bằng cách kết hợp các ưu điểm của của hai phương pháp phổ biến trước đó: `AdaGrad` [Duchi et al., 2011] hoạt động tốt với moment thưa và `RMSprop` [Hinton

et al., 2012], phương pháp hoạt động tốt trong các thiết lập trực tuyến (on-line) và không tĩnh (non stationary)

Phương trình cập nhật của Adam:

Tính momentum
Tính ước lượng moment bậc 2

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(g_t)^2$$

Hiệu chỉnh bias

$$\hat{m}_{t+1} = m_{t+1}/(1-\beta_1^{t+1})$$
$$\hat{v}_{t+1} = v_{t+1}/(1-\beta_2^{t+1})$$

Cập nhật trọng số

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon}\hat{m}_{t+1}$$

Ưu điểm:

- **Xử lý tốt gradient thưa trên tập dữ liệu nhiễu:** `Adam` kết hợp giữa `Adagrad` (phù hợp với gradient thưa) và RMSProp (ổn định với gradient nhiễu). Việc thuật toán có thể tự động điều chỉnh tốc độ học cho từng tham số dựa trên trung bình động của gradient và bình phương gradient, Adam đã được chứng minh hoạt động tốt trên dữ liệu rời rạc và nhiễu, đặc biệt trong các bài toán học sâu phức tạp. [Kingma and Ba, 2014]
- **Giá trị tham số mặc định hoạt động tốt trên hầu hết các bài toán:** Adam chỉ yêu cầu tinh chỉnh ba tham số chính: tốc độ học η, hệ số trung bình động bậc 1 (β_1) và bậc 2 (β_2). Các giá trị mặc định được đề xuất ($\eta = 0.01, \beta_1 = 0.9, \beta_2 = 0.999$) thường hoạt động tốt trên nhiều bài toán khác nhau, giảm đáng kể công sức tinh chỉnh.
- **Hiệu quả về mặt tính toán:** Adam có độ phức tạp tính toán tương đương với SGD nhưng đạt tốc độ hội tụ nhanh hơn nhờ khả năng kết hợp thông tin từ hai moment của gradient. Điều này giúp Adam tiết kiệm thời gian huấn luyện đáng kể trên các bài toán lớn.[Liu et al., 2019]
- **Hoạt động tốt trên tập dữ liệu lớn:** Sự kết hợp giữa RMSProp và Momentum giúp nó ổn định hơn và hội tụ nhanh trên dữ liệu quy mô lớn.Adam đã được thử nghiệm thành công trên các bài toán huấn luyện mô hình học sâu lớn với hàng triệu tham số [Kingma and Ba, 2014]

Nhược điểm:

- **Không hội tụ tới nghiệm tối ưu trong một số trường hợp:** `Adam` có thể không hội tụ tới nghiệm tối ưu toàn cục trong một số bài toán phi lồi. Điều này xảy ra do Adam sử dụng trung bình động của bình phương gradient (v_t) và gradient (m_t) để điều chỉnh tốc độ học. Tuy nhiên, v_t có thể bị ảnh hưởng bởi gradient noise, làm sai lệch tốc độ học và khiến Adam không hội tụ chính xác tới nghiệm tối ưu toàn cục trong một số bài toán phi lồi. De et al. [2018]
- **Phụ thuộc vào siêu tham số β_2:** Giá trị β_2 có thể không phù hợp với một vài bài toán, do đó cần chọn thủ công để đảm bảo được hội tụ tốt nhất

Nhìn chung, Adam là một thuật toán tối ưu hóa tiên tiến, kết hợp các ưu điểm của RMSProp và AdaGrad. Từ đó giúp cải thiện đáng kể tốc độ hội tụ và độ ổn định trong các bài toán, phù hợp với cho các vấn đề tối ưu khổng lồ của bài toán học sâu. Phương pháp này đơn giản, dễ sử dụng và sử dụng ít bộ nhớ hơn. Tuy nhiên, phương pháp này vẫn còn tồn đọng một số nhược điểm chí mạng như: không đảm bảo hội tụ đúng trong một số trường hợp, phụ thuộc vào tham số β_2. Sau này, các cải tiến `AMSGrad` và `AdamShift` đã được phát triển để khắc phục những hạn chế này và mang lại hiệu quả tốt hơn.

2.6 AMSGrad

AMSGrad được giới thiệu bởi Sashank J. Reddi và cộng sự trong bài báo "On the Convergence of Adam and Beyond" (ICLR 2018) [Zhou et al., 2018]. AMSGrad được phát triển để giải quyết nhược điểm của Adam khi không hội tụ đúng trong một số bài toán tối ưu hóa phi lồi.

Adam tính toán trung bình động của gradient và bình phương gradient (m_t, v_t) nhưng trong quá trình cập nhật, $v_t$có thể dao động mạnh do gradient nhiễu. Điều này khiến tốc độ học không ổn định và dẫn đến sai lệch trong hội tụ. AMSGrad khắc phục vấn đề này bằng cách giữ giá trị lớn nhất trong quá khứ của v_t thay vì sử dụng giá trị hiện tại.

Phương trình cập nhật AMSGrad:

AMSGrad giữ nguyên các bước tính moment bậc 1 và bậc 2 như Adam, nhưng sử dụng $\hat{v_t}$, là giá trị lớn nhất của moment bậc 2 trong các bước trước đó, từ đó đảm bảo rằng tốc độ học giảm đều và không bị ảnh hưởng bởi gradient nhiễu:

$$\hat{y_t} = max(\hat{v_{t-1}}, v_t)$$

Ưu điểm: Từ công thức được sửa đổi, có thể thấy v_t luôn tăng qua các bước cập nhật, đảm bảo tốc độ học tập giảm dần và hội tụ ổn định hơn. Từ đó giải quyết vấn đề hội tụ trong Adam, đặc biệt là các bài toán phi lồi.
Nhược điểm: Vì giá trị của moment bậc 2 luôn tăng, bài toán hội tụ chậm hơn, từ đó tăng chi phí tính toán

2.7 AdaShift

`AdaShift` được đề xuất bởi Zhou và cộng sự trong bài báo "AdaShift: Decorrelation and Convergence of Adaptive Learning Rate Methods" (2019) [Chen et al., 2018]. Phương pháp này mở rộng `Adam` để xử lý các vấn đề về độ trễ (shift) trong gradient, đặc biệt trong các mô hình dựa trên dữ liệu tuần tự như RNN hoặc LSTM.

Tác giả chỉ ra rằng các thuật toán tối ưu như `Adam` có thể gặp vấn đề vì có một mối tương quan không phù hợp giữa gradient và thuật ngữ mô men thứ hai trong Adam, dẫn đến một gradient lớn có khả năng làm cho bước nhảy nhỏ trong khi một gradient nhỏ có thể làm cho bước nhảy lớn. `AdaShift` thực hiện cân bằng gradient theo thời gian bằng cách điều chỉnh moment bậc 1 và bậc 2 cho phù hợp với các thay đổi trong không gian gradient.

Phương trình cập nhật AdaShift:

AdaShift điều chỉnh moment dựa trên độ trễ gradient như sau:
Tính momentum *Tính ước lượng momen bậc 2*

$$m_t^{(n)} = \beta_1 m_{t-1}^{(n)} + (1-\beta_1) g_{t-n} \qquad v_t^{(n)} = \beta_2 v_{t-1}^{(n)} + (1-\beta_2) g_{t-n}^2$$

Cập nhật trọng số

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^{(n)}} + \epsilon} m_t^{(n)}$$

Ưu điểm: Hiệu quả hơn trên các mô hình tuần tự như RNN và LSTM khi có độ trễ gradient.

Nhược điểm:

- Tăng độ phức tạp về mặt tính toán do cần quản lý gradient có độ trễ.
- Hiệu quả phụ thuộc vào cách chọn độ trễ n.

3 Phương pháp tối ưu Adopt

3.1 Bối cảnh ra đời của Adopt

Mặc dù đã ra đời được một khoảng thời gian dài, nhưng Adam [Kingma and Ba, 2014] vẫn là phương pháp tối ưu hoá được sử dụng phổ biến nhất trong Machine learning và Deep Learning hiện tại. Bất chấp những nhược điểm hiện hữu của phương pháp tối ưu này, đã được chứng minh và nêu rõ bởi *Reddi et al. (2018)* trong bài báo có tựa đề *"On the Convergence of Adam and Beyond"* [Zhou et al., 2018]. Trong đó hai nhược điểm nổi bật nhất là:

- Sự phụ thuộc giữa ước lượng moment bậc 2 (v_t) và giá trị gradient hiện tại (g_t): Việc ước lượng moment bậc 2 (v_t) được tính trực tiếp dựa trên gradient hiện tại (g_t) làm cho Adam cần giả định mạnh về Gradient Noise (G) để đảm bảo thuật toán có thể hội tụ đúng. Trên thực tế, đây là một yếu tố không thể đảm bảo được và làm hạn chế tính linh hoạt của bài toán.
  $\Rightarrow$ Dẫn đến việc bài toán sẽ không hội tụ đúng trong một số trường hợp.

- Sự phụ thuộc vào việc chọn siêu tham số β_2: Siêu tham số β_2 thường được chọn là 0.9 và 0.99, có nghĩa là lấy rất ít giá trị Gradient hiện tại (g_t) để đảm bảo (v_t) không quá lớn hay không bị ảnh hưởng nhiều bởi Gradient Noise (G). Vì thế khi chọn siêu tham số β_2 nhỏ như 0.1 hay 0.5, thì Adam gần như không thể hội tụ.
  $\Rightarrow$ Dẫn đến việc phụ thuộc nhiều vào việc chọn siêu tham số β_2 phù hợp, mà đây là việc chúng ta không thể đảm bảo được trong thực tế.

Đã có những nổ lực để khắc phục những nhược điểm trên của Adam, nhưng chưa đạt được hiệu quả tốt và không phù hợp với thực nghiệm. Trong đó có hai phương pháp nổi bậc là AMSGrad và AdaShift:

- AdaShift [Chen et al., 2018]: ý tưởng của AdaShift là giảm sự phụ thuộc của v_t và g_t bằng cách dùng v_{t-n}. Điều này giúp Adam hội tụ đúng về mặc lý thuyết, nhưng trong thực tế thì làm tăng độ phực tạp của thuật toán, giá trị n cần phải thử nghiệm nhiều lần để tìm ra giá trị phù hợp.

- AMSGrad [Zhou et al., 2018]: Mặc dù AMSGrad khắc phục điểm yếu không hội tụ đúng trong một số trường hợp của Adam bằng cách đảm bảo rằng learning rate (α) không bị thay đổi đột ngột và giúp thuật toán hội tụ ổn định hơn. Nhưng trong thực tế, Adam vẫn hoạt động tốt hơn AMSGrad, điều này xảy ra khi sự thay đổi learning rate linh hoạt của Adam có lợi hơn cho việc tối ưu hóa.

Deep learning đang ngày càng phát triển và nhiều model, tập dữ liệu có độ phức tạp cao được ra đời, từ đó hàm tối ưu cũng trở nên rắc rối hơn. Vì thế việc đề ra được phương pháp mới khắc phục được nhược điểm về mặt hội tụ trong lý thuyết mà vẫn giữ được ưu điểm linh hoạt trong thực nghiệm của Adam đang là một đề tài nghiên cứu được quan tâm hiện tại. Nhưng những nổ lực trước đó vẫn chưa đạt được thành công mong muốn.

$\Longrightarrow$ Phương pháp Adopt [Zhang et al., 2024] ra đời với triển vọng có thể khắc phục vấn đề hội của Adam với mọi giá trị của siêu tham số β_2.

3.2 Thuật toán Adopt 1

Ý tưởng: Dựa trên ý tưởng từ những phương pháp tối ưu trước đó như AdaShift và AMSGrad là giảm sự phụ thuộc của ước lượng moment bậc 2 (v_t) và giá trị Gradient hiện tại (g_t). Adopt khắc phục nó bằng cách rất đơn giản là thay vì dùng v_t thì dùng v_{t-1} để tính toán cho bước cập nhật hiện tại.

Công thức của Adam:

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t$$

m_{t+1} : Momentum.
β_1 : Hệ số giảm của momentum.
g_t : Gradient tại bước thứ t.

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(g_t)^2$$

v_{t+1} : Ước lượng moment bậc 2.
β_2 : Hệ số giảm của moment bậc 2.

$$w_{t+1} = w_t - \frac{\eta\, \hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon}$$

w_{t+1} : Trọng số sau khi cập nhật.
η : Tốc độ học (learning rate).
$\hat{m}_{t+1}, \hat{v}_{t+1}$: Moment hiệu chỉnh sai lệch.
ϵ : Số nhỏ để tránh chia cho 0.

Nhận xét: Có thể thấy ở công thức trên của Adam, v_t đang được cập nhật trực tiếp bởi g_t, trong khi đó g_t trong thực tế là không ổn định do được tính từ minibatch thay vì toàn bộ tập dữ liệu.

- Nếu g_t lớn đột ngột thì learning rate sẽ nhỏ đột ngột ⇒ Cập nhật trở nên chậm, không ổn định, có thể vanising gradient (hiện tượng tiêu biến gradient khi không có cập nhật).
- Nếu g_t nhỏ đột ngột thì learning rate sẽ lớn đột ngột ⇒ Gây bùng nổ gradient.

$\Longrightarrow$ Adopt dùng v_{t-1} để tính toán cho bước hiện tại bằng cách thay đổi thứ tự cập nhật công thức trên và sinh ra `thuật toán Adopt 1`.

Thuật toán Adopt 1:

- Ràng buộc

$$v_0 \leftarrow g_0 \odot g_0$$

v_0 : Giá trị khởi tạo của moment bậc 2.
g_0 : Gradient tại bước đầu tiên.

$$m_1 \leftarrow \frac{g_1}{\max\{\sqrt{v_0}, \epsilon\}}$$

m_1 : Giá trị khởi tạo của momentum.
ϵ : Số nhỏ để tránh chia cho 0.

- Cập nhật

$$w_t \leftarrow w_{t-1} - \alpha_t m_t$$

w_t : Trọng số tại bước t.
α_t : Tốc độ học.

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t \odot g_t$$

v_t : Moment bậc 2 tại bước t.
β_2 : Hệ số giảm của moment bậc 2.

$$m_{t+1} \leftarrow \beta_1 \cdot m_t + (1-\beta_1) \cdot \frac{g_{t+1}}{\max\{\sqrt{v_t}, \epsilon\}}$$

m_{t+1} : Momentum tại bước $t+1$.
β_1 : Hệ số giảm của momentum.

Kết luận: Sự thay đổi trên đã làm giảm sự phụ thuộc giữa v_t và g_t, thực tế nó giống AdaShift [Chen et al., 2018] với $n = 1$. Thuận toán Adopt 1 đã có thể khắc phục vấn đề hội tụ của Adam về mặt lý thuyết, việc thay thế $\sqrt{v_t^2 + \epsilon^2}$ bằng $max(v_t, \epsilon)$ cũng đem lại kết quả tốt hơn trên thực nghiệm.

3.3 Thuật toán Adopt 2

Vấn đề của thuật toán Adopt 1: Tại những bước đầu tiên, giá trị gradient hiện tại (g_t) có thể trở nên rất nhỏ, đặc biệt với bối cảnh các model có xu hướng khởi tạo các tham số ban đầu là 0. Điều này làm cho momentum (m_{t+1}) sẽ rất lớn, đặc biệt là trong thuật toán Adopt 1 không có hiệu chỉnh

bias như Adam. ⇒ Bùng nổ gradient ở những bước đầu tiên.

$\Longrightarrow$ Thuật toán Adopt 2 sinh ra để khắc phục vấn đề trên bằng cách giới hạn giá trị $\frac{g_{t+1}}{max(v_t,\epsilon)}$ trong một khoảng cố định.

Thuật toán Adopt 2:

- Ràng buộc

$0 \leq \beta_1, \beta_2 < 1,\ \epsilon > 0$ — β_1, β_2 : Hệ số giảm moment bậc 1 và bậc 2. ϵ : Số nhỏ để tránh chia cho 0.

$m_0 \leftarrow 0, \quad v_0 \leftarrow g_0 \odot g_0$ — m_0 : Giá trị khởi tạo của momentum. v_0 : Giá trị khởi tạo của moment bậc 2. g_0 : Gradient tại bước đầu tiên.

- Cập nhật

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \text{Clip}\left(\frac{g_t}{\max\{\sqrt{v_t}, \epsilon\}}, c_t\right)$$

$$w_t \leftarrow w_{t-1} - \alpha_t m_t$$

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t \odot g_t$$

- Trong đó:

$\text{Clip}(x, c)$:	Hàm cắt giá trị x trong khoảng $[-c_t, c_t]$. $Clip(x, c_t) = min(max(a, -c_t), c_t)$
c_t :	Ngưỡng cắt tại bước t. $c_t = c = \Theta(T^{1/4})$ hoặc $c_t = \Theta(t^{1/4})$
α_t :	Tốc độ học tại bước t. $\alpha_t = \Theta(1/\sqrt{t})$
w_t :	Trọng số tại bước t.
m_t :	Momentum tại bước t.
v_t :	Moment bậc 2 tại bước t.
g_t :	Gradient tại bước t.
β_1, β_2 :	Hệ số giảm momentum và bậc 2.
ϵ :	Số nhỏ để tránh chia cho 0.

Kết luận: Với những ràng buộc, cách chọn tham số trên, thuật toán Adopt 2 đã khắc phục được nhược điểm của thuật toán Adopt 1, giúp cập nhật hiệu quả ngay từ những bước đầu. Đồng thời, được chứng minh có tốc hội tụ là $O(1/\sqrt{T})$, với T là tổng số bước lặp.

Chứng minh tốc độ hội tụ là $O(1/\sqrt{T})$:

Giả thuyết:

- Hàm $f(\theta)$ là L-smooth:

$$f(\theta_{t+1}) \leq f(\theta_t) + \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + \frac{L}{2}\|\theta_{t+1} - \theta_t\|^2.$$

- Gradient có kỳ vọng không chệch:

$$E[g_t] = \nabla f(\theta_t)$$

- Gradient có phương sai hữu hạn:

$$E[\|g_t - \nabla f(\theta_t)\|^2] \leq \sigma^2.$$

Chứng minh:

- Sử dụng L-smoothness, ta có:

$$f(\theta_{t+1}) \leq f(\theta_t) - \alpha_t \nabla f(\theta_t)^\top \frac{m_t}{\sqrt{v_t} + \epsilon} + \frac{L\alpha_t^2}{2} \left\| \frac{m_t}{\sqrt{v_t} + \epsilon} \right\|^2.$$

- Lấy kỳ vọng cả hai vế, dùng giả thiết không chệch:

$$E[f(\theta_{t+1})] \leq E[f(\theta_t)] - \alpha_t E\left[\|\nabla f(\theta_t)\|^2\right] + \frac{L\alpha_t^2}{2} E\left[\|g_t\|^2\right].$$

- Tổng cộng từ $t = 1$ đến T:

$$\frac{1}{T}\sum_{t=1}^{T} E\left[\|\nabla f(\theta_t)\|^2\right] \leq \frac{f(\theta_1) - f^*}{\alpha_t T} + \frac{L\alpha_t}{2T}\sum_{t=1}^{T} E\left[\|g_t\|^2\right].$$

- Chọn $\alpha_t = \Theta(1/\sqrt{T})$, ta có:

$$\min_{t=1,\ldots,T} E\left[\|\nabla f(\theta_t)\|^{4/3}\right]^{3/2} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right).$$

4 Thực nghiệm

Bao gồm 4 thực nghiệm: toy problem, MLP với MNIST, Resnet18 với Cifar-10, Resnet50 với Imagenet (do hạn chế về mặc tài nguyên nên sữ dụng Resnet50 thay vì SwinTransformer giống với bài báo). Tất cả các thực nghiệm trên được thực hiện dựa trên bài báo [Zhang et al., 2024].

Source code: Github

4.1 Toy problem

Mô tả thực nghiệm:

- Chúng ta sẽ xem xét một vấn đề tối ưu với một hàm mục tiêu $f(\theta) = \theta$ với $\theta \in [-1, 1]$. Ta sẽ có lời giải cho bài toán này là $\theta = 1$. Thông qua quá trình tối ưu, chúng ta sẽ chỉ sử dụng hàm mục tiêu ngẫu nhiên f_t sau:

$$f_t(\theta) = \begin{cases} k^2\theta & \text{với xác suất } \frac{1}{k} \\ -k\theta & \text{với xác suất } 1 - \frac{1}{k} \end{cases}$$

- Khi $k \geq 1$, bởi vì $E[f_t(\theta)] = f(\theta)$ nên đạo hàm ngẫu nhiên $g_t = \nabla f_t(\theta)$ là một ước lượng không chệch của đạo hàm thực ∇f và không phụ thuộc và k. Thực nghiệm dựa theo bái báo [Reddi et al., 2018].

- Ta sẽ đánh giá hiệu suất của 3 thuật toán tối ưu khác nhau là Adam, AMSGrad và ADOPT. Khi k = 1 nó tương ứng với trường hợp không có nhiễu, nơi mà $f_t = f$ với xác suất là 1. Nhưng việc tối ưu sẽ gặp khó khăn hơn khi mà k trở nên lớn hơn, đạo hàm ngẫu nhiên sẽ trở nên nhiễu hơn. Trong thực nghiệm này ta sẽ đặt 2 giá trị k = 10 và k = 50 để kiểm tra độ vững chắc của các thuật toán này trước nhiễu sẽ như thế nào.

Ràng buộc môi trường:

- Tham số β_2:

$$\beta_2 \in [0.1, 0.5, 0.999]$$

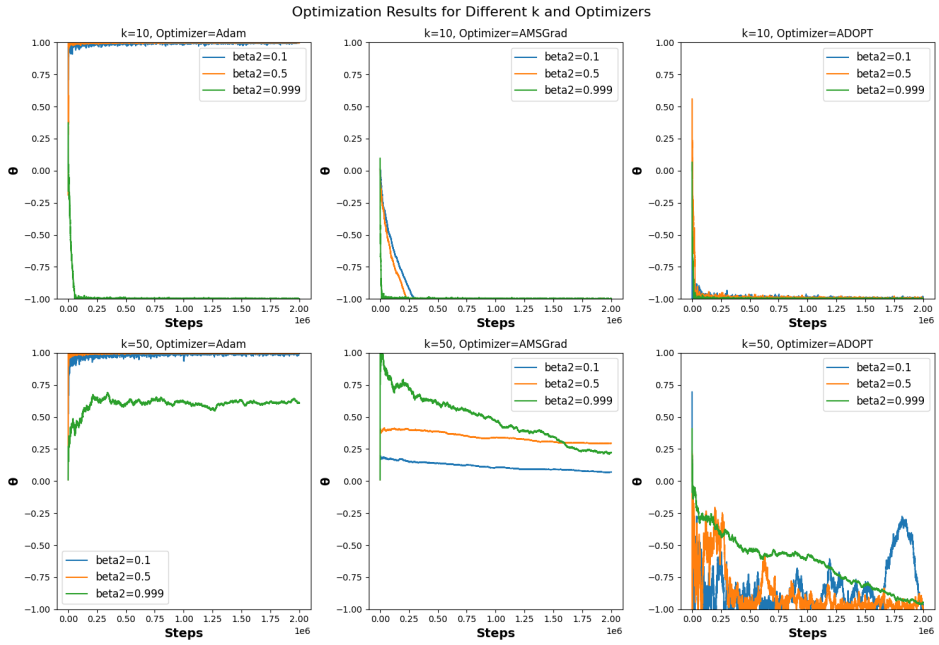
- Tham số β_1:

$$\beta_1 = 0.9$$

- Learning rate:

$$\alpha_t = \frac{0.01}{\sqrt{1 + 0.01t}} \quad \text{với } t \text{ là iteration}$$

Kết quả:



Hình 1: So sánh hiệu suất giữa Adam, AMSGrad và ADOPT trong một bài toán tối ưu hóa lồi một biến đơn giản. Các đồ thị thể hiện sự thay đổi giá trị tham số, và giá trị tham số này sẽ hội tụ về nghiệm $\theta = -1$

Kết luận:

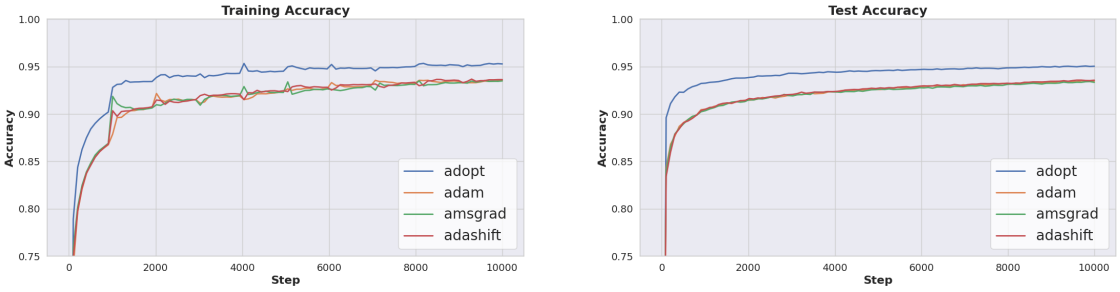
- Trong bài toán tối ưu hóa, kết quả cho thấy Adam không hội tụ trừ khi $\beta_2 = 0.999$ trong khi AMSGrad và ADOPT hội tụ nhanh chóng về nghiệm đúng $\theta = -1$ cho mọi giá trị của β_2.
- Khi $k = 50$, Adam không hội tụ kể cả khi $\beta_2 = 0.999$, điều này cho thấy khi độ nhiễu của gradient lớn, vùng hội tụ của Adam cũng mở rộng, dẫn đến sự phân kỳ.Hơn nữa, tốc độ hội tụ của AMSGrad cũng chậm chậm hơn ADOPT khá nhiều.
- Những kết quả này phù hợp với kết quả nghiên cứu lý thuyết, cho thấy tính ưu việt của ADOPT so với Adam và AMSGrad về tốc độ hội tụ và khả năng lựa chọn siêu tham số.

4.2 MLP với MNIST

Mô tả:

- Để nghiên cứu hiệu quả của ADOPT trong tối ưu hóa không lồi, ta huấn luyện các mạng neural phi tuyến cho các nhiệm vụ phân loại trên bộ dữ liệu MNIST và so sánh hiệu suất giữa ADOPT và các thuật toán tối ưu hóa hiện có, như Adam, AMSGrad và AdaShift.
- Trong thực nghiệm này chúng ta sẽ sử dụng một mô hình Multi-layer perceptron (ML đơn giản với gồm một lớp ẩn với 784 node. Learning rate sẽ là $\alpha_t = \frac{\alpha}{\sqrt{t}}$, α ban đầu sẽ được điều chỉnh trong khoảng $\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$. Thêm vào đó ta áp dụng trọng số giảm dần (weight decay) với giá trị 10^{-4} để ovverfitting. Thực nghiệm này sẽ được thực hiện trên 10K iteration.
- Đối với tập dữ liệu MNIST tôi chia 48K ảnh cho train, 12K ảnh cho val và 10K ảnh cho test. Tập dữ liệu được chia theo $batch = 48$ nên sẽ cho 1000 iteration cho mỗi epoch $\Rightarrow$ Số epochs cho việc train sẽ là 10.

Kết quả:



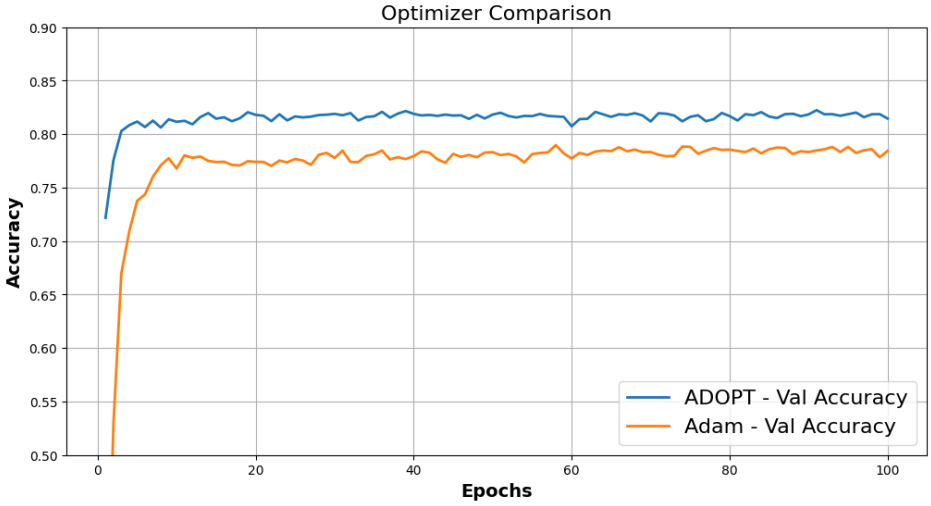
Hình 2: So sánh độ chính xác của mô hình với các thuật toán tối ưu trên tập dữ liệu MNIST

4.3 Resnet18 với CIFAR-10

Mô tả thực nghiệm:

- Ta tiến hành thực nghiệm phân loại ảnh sử dụng các bộ dữ liệu ảnh trong thực tế. Để so sánh hiệu suất của ADOPT và Adam trên nhiệm vụ phân loại ảnh trên tập dữ liệu CIFAR-10 sử dụng mô hình Resnet18 [He et al., 2016a], một mô hình convolutional neural network được sử dụng rộng rãi. Các hyperparameter được sử dụng giống với trường hợp thực nghiệm với bộ dữ liệu MNIST ở trên. Thực nghiệm sẽ được thực hiện trên 100 epochs.
- Về tập dữ liệu CIFAR-10 tôi chia 40K ảnh cho train, 10K ảnh cho val và 10K ảnh cho test.

Kết quả:



Hình 3: So sánh độ chính xác của mô hình với các thuật toán tối ưu trên tập dữ liệu CIFAR-10 với mô hình Resnet18

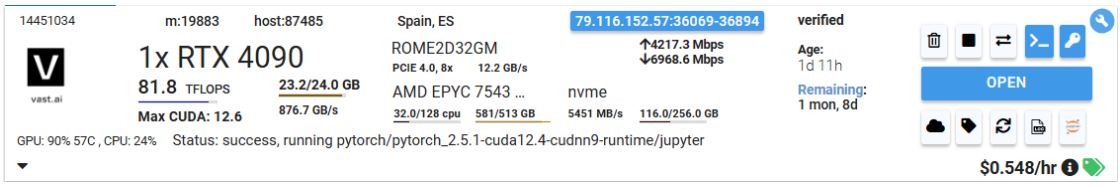
4.4 Resnet50 với IMAGENET

Mô tả thực nghiệm:

- Sau 3 thực nghiệm với các bài toán và bộ dữ liệu nhỏ thì ADOPT cho ra kết quả khá tốt khi đã chứng minh được sự hội tụ tối ưu của bản thân. Vấn đề đặt ra là các thuật toán tối ưu trước đó được kiểm nghiệm trên những bộ dữ liệu lớn và cho ra kết quả rất tốt. Để chứng minh được thuật toán tối ưu ADOPT cũng cho ra kết quả tốt trên những bộ dữ liệu đó, chúng tôi tiến hành thực nghiệm trên một bộ dữ liệu nổi tiếng đó là IMAGENET.
- Bộ dữ liệu chúng tôi sử dụng được lấy từ *ImageNet Large Scale Visual Recognition Challenge 2011* [Russakovsky et al., 2015] bao gồm 1000 lớp, trong đó có 1229118 ảnh trong tập train, 50000 ảnh trong tập validation và 10000 ảnh trong tập test. Tuy nhiên do không đủ tài nguyên để huấn luyện từng đó ảnh, chúng tôi chỉ sử dụng được 40% của tập train tương đương với 491647 ảnh và được chia ra làm tập train và tập val theo tỉ lệ 9:1.
- Thay vì dùng Swin Transformer-tiny cung cấp bởi Torchvision [Paszke et al., 2019] như trên bài báo [Zhang et al., 2024], chúng tôi sữ dụng một mô hình phân loại ảnh phổ biến khác là Resnet50 [He et al., 2016b] (Do hạn chế về mặc tài nguyên). Chúng tôi giữ lại tất cả các lớp của Resnet50, không thay đổi bất kì tham số nào và tiến hành train trên 50 epochs.

Môi trường:

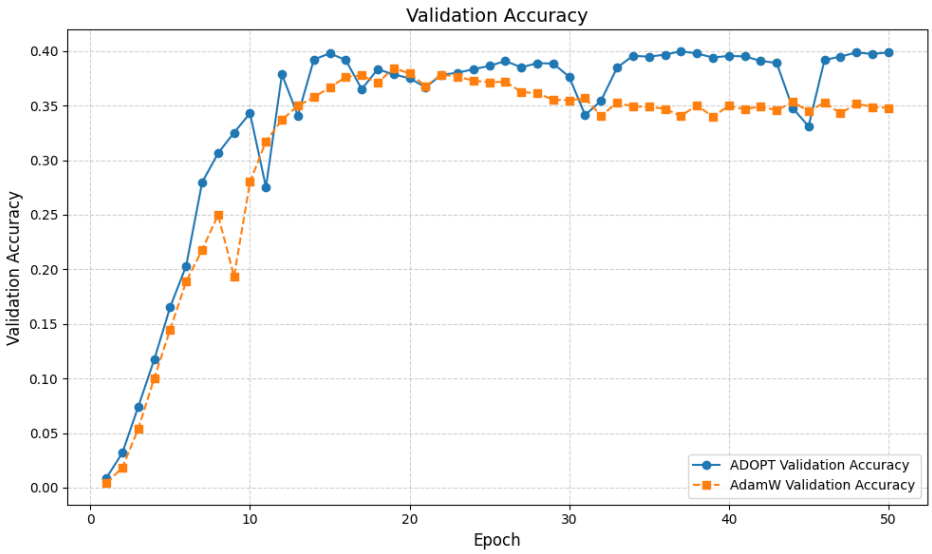
- Cấu hình mà chúng tôi sử dụng bao gồm CPU AMD EPYC 7543 [AMD, 2021], GPU RTX 4090 24G[NVIDIA, 2022]. Trong quá trình huấn luyện, chúng tôi đã sử dụng 116GB ROM, 581GB RAM và 23,2GB GPU RAM, tổng thời gian huấn luyện là 1 ngày 6 giờ.



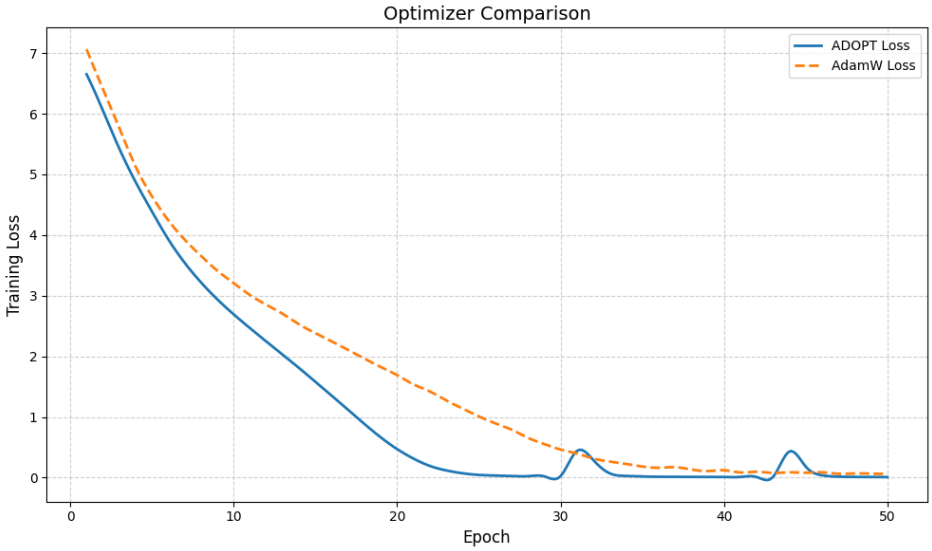
Hình 4: Cấu hình máy dùng để thực nghiệm

- Các cài đặt tham số cần thiết trong quá trình huấn luyện:
  - Learning rate: $\alpha_t = \frac{\alpha_0}{\sqrt{t}}$ với t là iteration và $\alpha_0 = 0.1$
  - Batch size = 256:
  - Kích thước ảnh : $(224, 224, 3)$
  - Weight decay = 0.0001

Kết quả:



Hình 5: So sánh độ chính xác giữa ADOPT và AdamW



Hình 6: So sánh loss theo từng epochs giữa ADOPT và AdamW

Kết luận: Kết quả thu được cho thấy sự biến động rõ rệt của thuật toán ADOPT ở các epoch 32 và 44, điều này có thể là dấu hiệu của quá trình tối ưu hóa không hoàn toàn ổn định trong giai đoạn học, nhưng nó vẫn không làm giảm đi hiệu quả của ADOPT trong việc đạt được điểm hội tụ nhanh hơn so với AdamW. Cụ thể, dù có một số dao động nhất định trong quá trình huấn luyện, ADOPT vẫn cho thấy khả năng tối ưu hóa vượt trội về tốc độ, với sự giảm thiểu loss nhanh chóng trong các epoch đầu. Điều này chứng tỏ rằng ADOPT có thể tìm ra điểm hội tụ với ít bước huấn luyện hơn so với AdamW, giúp tiết kiệm thời gian và tài nguyên tính toán, đồng thời đạt được hiệu quả tương tự hoặc thậm chí tốt hơn.

5 Kết luận

Trong bài nghiên cứu này, chúng tôi đã nêu ra ý tưởng, làm sáng tỏ ưu và nhược điểm của những phương pháp tối ưu phổ biến từ trước đến nay, đặc biệt là Adam. Đồng thời chứng minh độ hiệu quả của giải pháp giải quyết vấn đề đến từ phương pháp tối ưu mới có tên là Adopt. Adopt không chỉ đảm bảo hội tụ với tốc độ tối ưu trên bất kì sự lựa chọn siêu tham số nào, mà còn cho thấy hiệu năng vượt trội trong nhiều ứng dụng thực tiễn.

Các thực nghiệm được thực hiện trên nhiều lĩnh vực của học sâu như phân loại hình ảnh, thị giác máy tính, xử lý ngôn ngữ tự nhiên, ... với các các tập dữ liệu lớn nhỏ và các thuật toán tối ưu phổ biến. Adopt đều cho thấy khả năng hội tụ mạnh mẽ hơn các phương pháp khác khi đạt kết quả cao hơn với thời gian ngắn hơn. Với sự vượt trội này, Adopt hoàn toàn có thể trở thành một ứng cử viên tiềm năng trong tương lai nhằm thay thế các phương pháp tối ưu hiện tại.

6 Bảng phân công

Nhiệm vụ	Nguyễn Ấn	Trịnh Thị Lan Anh	Lê Văn Giáp	Vương Dương Thái Hà
Tìm hiểu paper	X	X	X	X
Làm Slide	X	X	X	X
Thuyết trình		X		
Trả lời câu hỏi	X	X	X	X
Thực nghiệm	X	X	X	X
Báo cáo	X	X	X	X
Mức độ hoàn thành	100%	100%	100%	100%

Tài liệu tham khảo

AMD. Amd epyc 7543 processor, 2021. URL https://www.amd.com/en/products/cpu/amd-epyc-7543. Accessed: 2024-12-23.

Xuyang Chen, Jun Zhu, Bo Zhang, and Le Song. Adashift: Decorrelation and convergence of adaptive learning rate methods. *arXiv preprint arXiv:1810.00143*, 2018.

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*, 2015.

Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016b. doi: 10.1109/CVPR.2016.90.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture slides on optimization methods for deep learning. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012. Accessed: December 17, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Peter Liu, Jason Wei, et al. Rethinking batch normalization in transformers. *OpenReview*, 2019. URL https://openreview.net/forum?id=ryQu7f-RZ. Accessed: December 17, 2024.

NVIDIA. Nvidia geforce rtx 4090 graphics card, 2022. URL https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/. Accessed: 2024-12-23.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

Herbert Robbins and Sathyadeo Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. doi: 10.1214/aoms/1177729586.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://www.image-net.org/challenges/LSVRC/2011/.

Leslie N. Smith. Cyclical learning rates for training neural networks. *arXiv preprint arXiv:1506.01186*, 2017.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

Zhen Zhang, Min Li, Wei Xu, and Yu Wang. Adopt: Modified adam can converge with any β_2 with the optimal rate. *arXiv preprint arXiv:2411.02853v3*, 2024.

Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.