

Huấn luyện mạng nơ-ron nhiều tầng ẩn bằng thuật toán Adam

Nhóm sinh viên thực hiện:

- Nguyễn Ngọc Lan Như - 1712644
- Hoàng Minh Quân – 1712688

Giáo viên hướng dẫn: Th.S. Trần Trung Kiên

Mục lục

1. Giới thiệu đề tài
2. Kiến thức nền tảng
3. Thuật toán Adam
4. Thí nghiệm
5. Tổng kết

1.

Giới thiệu đề tài

Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



Giới thiệu đề tài

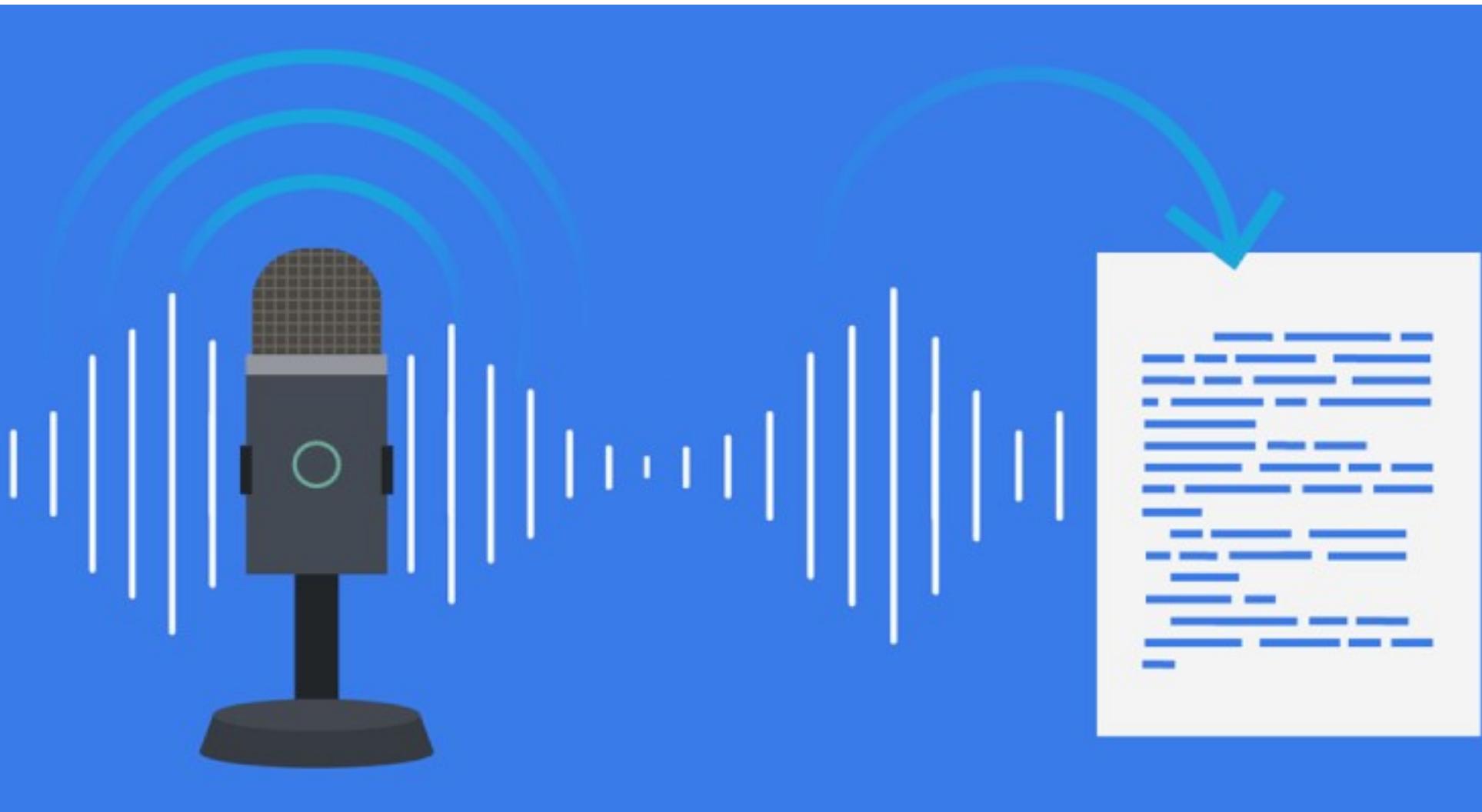
Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?

MACHINE TRANSLATION



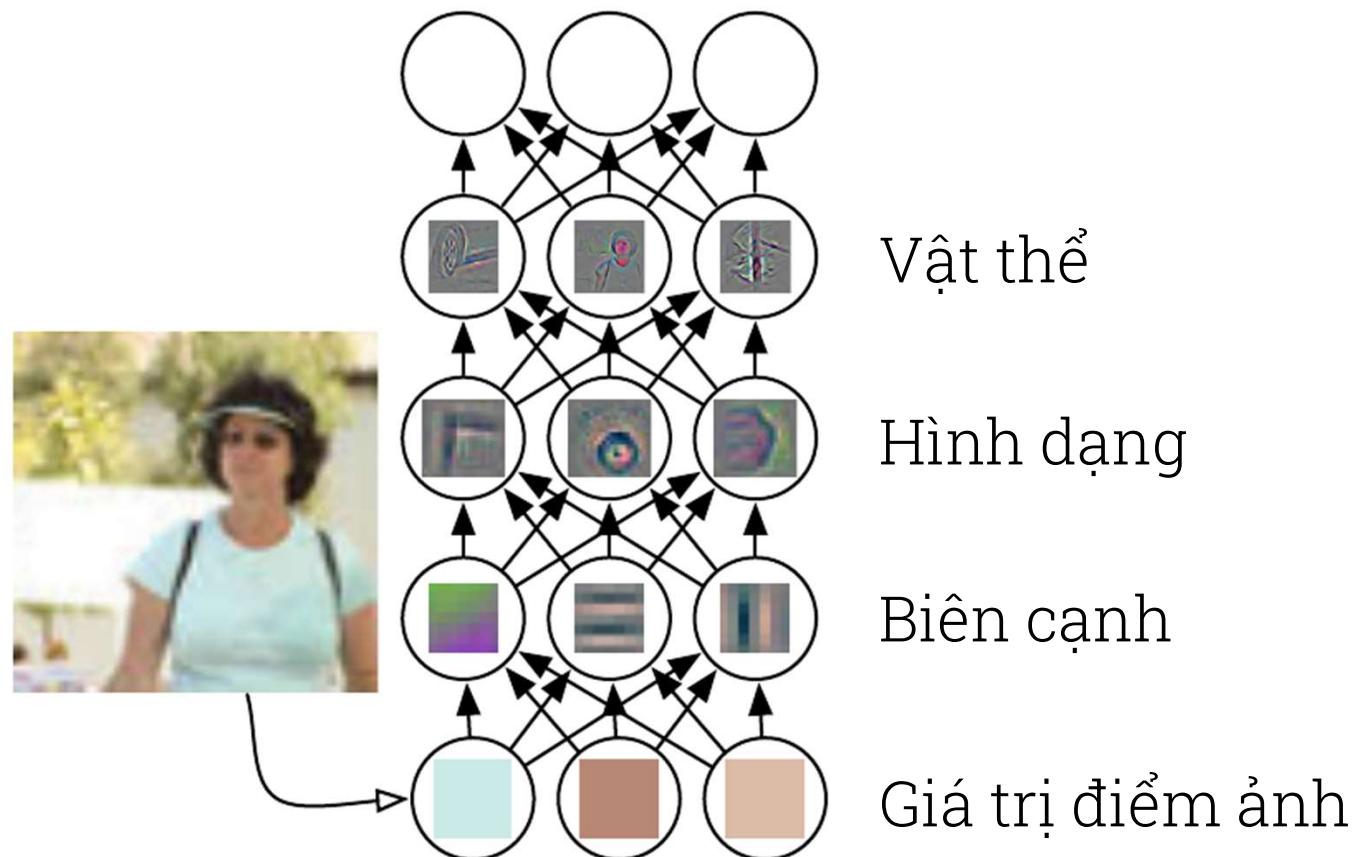
Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



Giới thiệu đề tài

Tại sao lại sử dụng mạng nơ-ron nhiều tầng ẩn?



*Quá trình mạng nơ-ron trích xuất các đặc trưng
của dữ liệu*

Giới thiệu đề tài

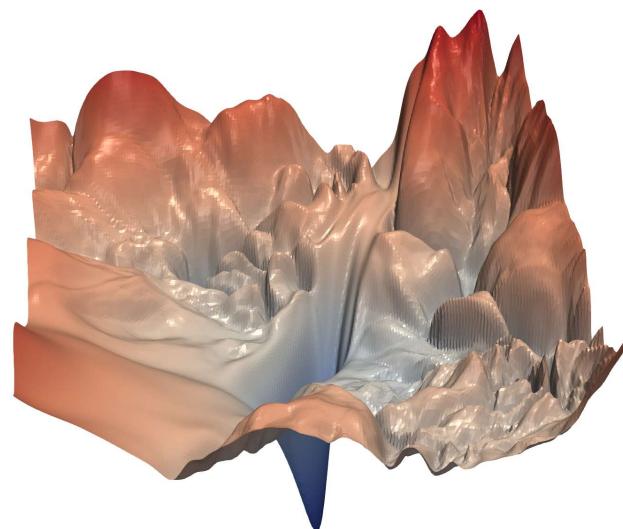
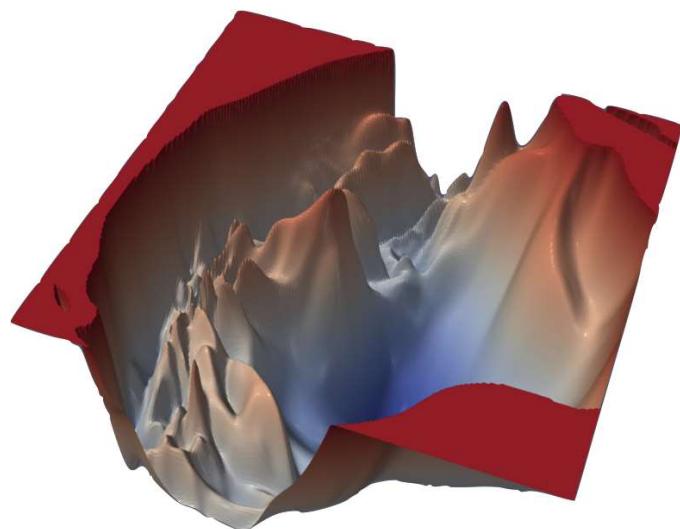
Bài toán huấn luyện mạng nơ-ron nhiều tầng ẩn

- **Input:** Hàm chi phí với tham số là các trọng số của mạng nơ-ron nhiều tầng ẩn. Hàm chi phí cho biết sự sai lệch giữa kết quả dự đoán của mạng nơ-ron so với giá trị đúng trên tập dữ liệu huấn luyện, hay *độ lỗi*.
- **Output:** Bộ trọng số của mạng nơ-ron nhiều tầng ẩn cho độ lỗi là nhỏ nhất, hoặc đủ nhỏ.

Giới thiệu đề tài

Bài toán huấn luyện mạng nơ-ron nhiều tầng ẩn

- Các giá trị của hàm chi phí có thể được biểu diễn ở dạng bề mặt lõi.
- Quá trình huấn luyện cũng là quá trình di chuyển trên bề mặt lõi.

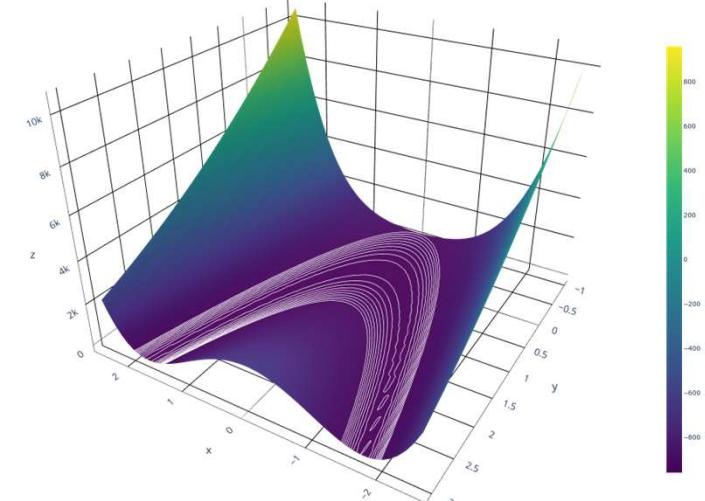


Bề mặt lõi của mô hình ResNet-110 (trái) và ResNet-56 (phải)

Giới thiệu đề tài

Thách thức

- Bề mặt lõi phức tạp [1]:
 - Nhiều cực tiểu địa phương.
 - Nhiều điểm yên ngựa.
 - Nhiều vùng bằng phẳng.
 - Nhiều **rãnh hẹp**.



Dạng bề mặt rãnh hẹp

[1] Yann Dauphin et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", NIPS 2014.

Giới thiệu đề tài

Nghiên cứu liên quan

- Phương pháp truyền thống: GD/SGD.
 - Gặp khó khăn khi di chuyển trong vùng rãnh hẹp.
- Đề xuất những phương pháp cải tiến khác.

Giới thiệu đề tài

Bài báo tìm hiểu

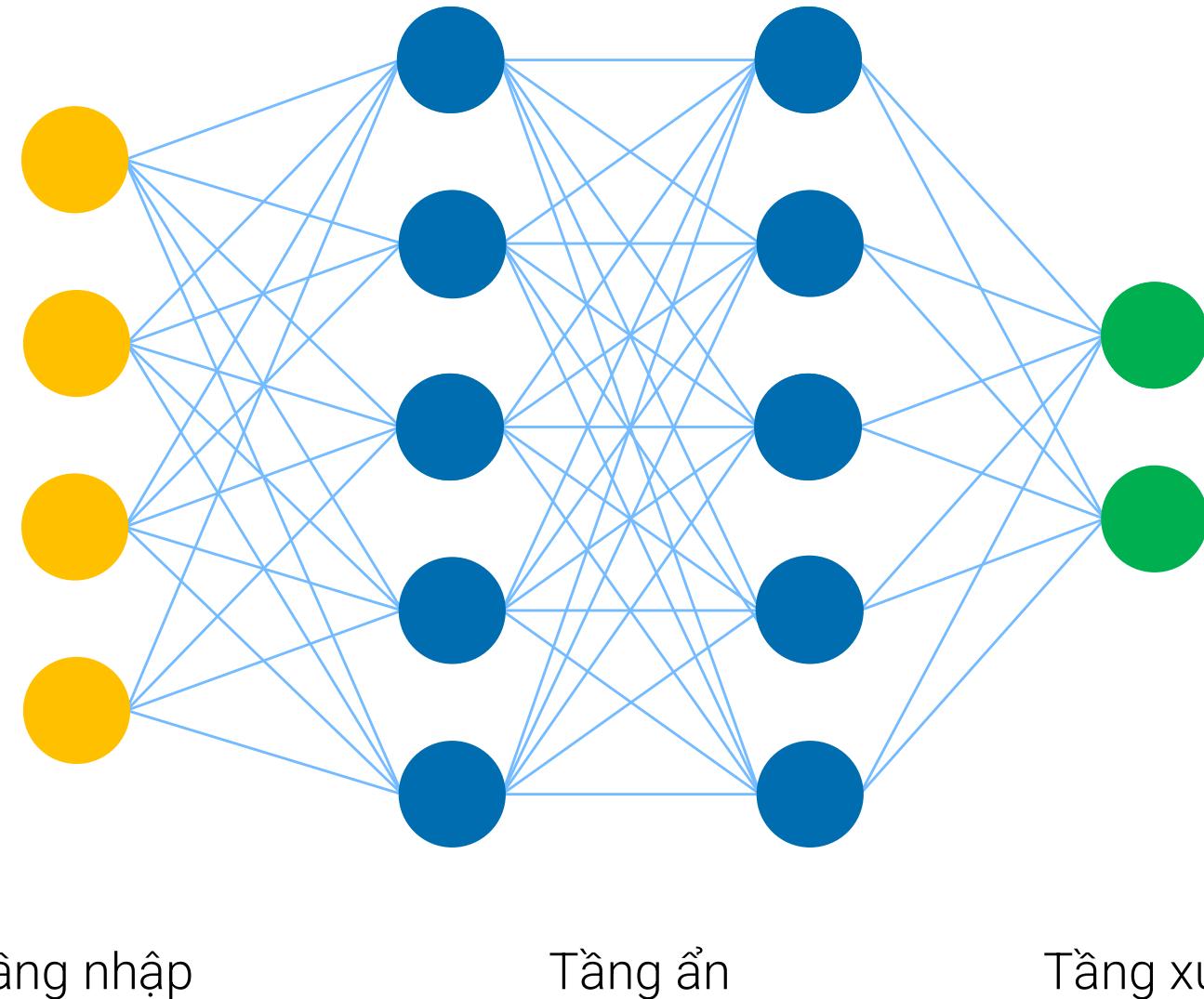
- "Adam: A method for stochastic optimization" của nhóm tác giả từ Google và Đại học Toronto (ICLR 2015). Số lượt trích dẫn: 77928.
 - Là phương pháp nổi bật trong các cải tiến của phương pháp huấn luyện mạng nơ-ron nhiều tầng ẩn.
 - Hoạt động tốt tại các vùng rãnh hẹp.
 - Tăng tốc quá trình huấn luyện mạng nơ-ron nhiều tầng ẩn.

2.

Kiến thức nền tảng

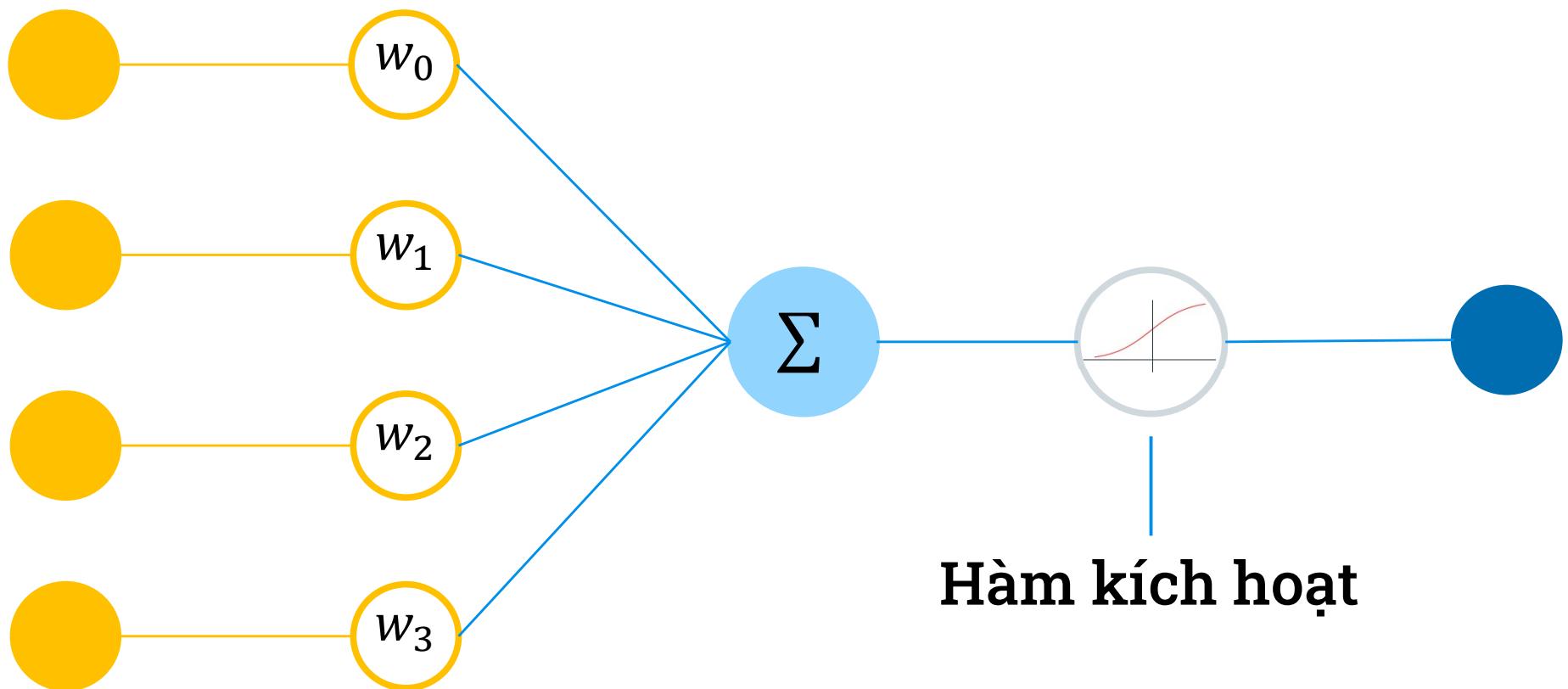
Kiến thức nền tảng

Mạng nơ-ron nhiều tầng ẩn



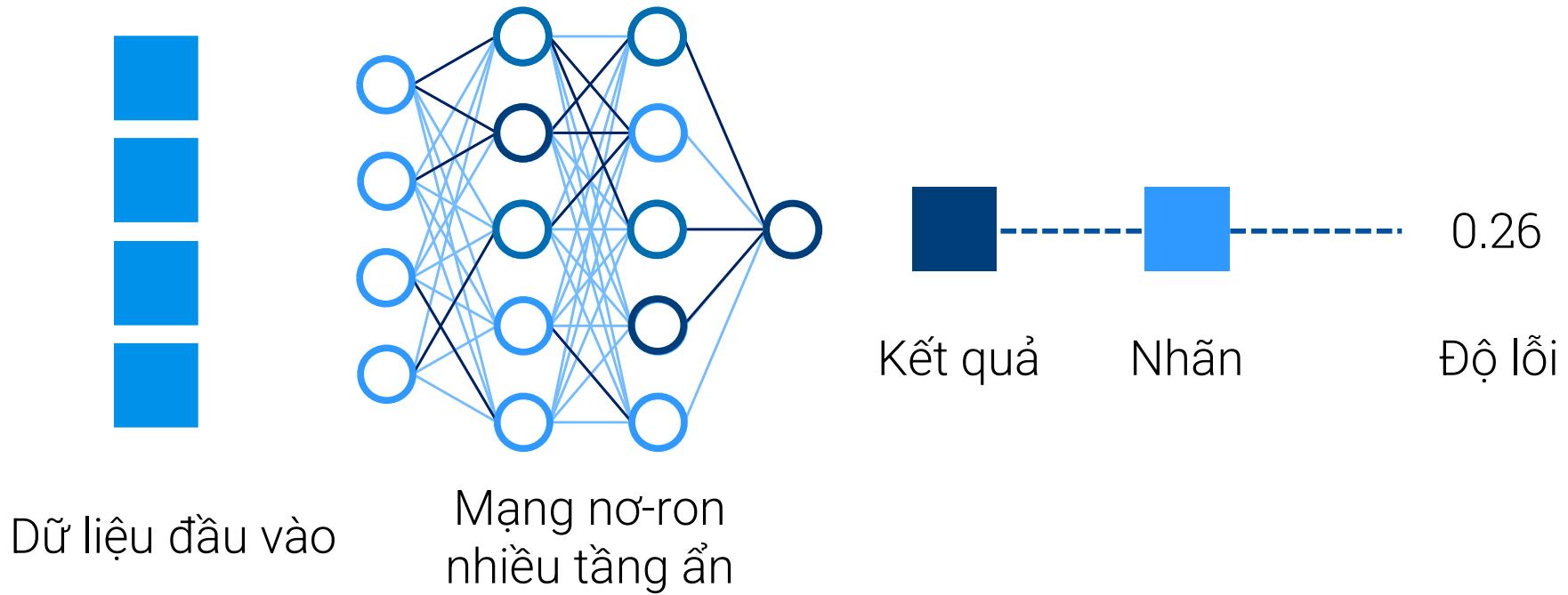
Kiến thức nền tảng

Mạng nơ-ron nhiều tầng ẩn



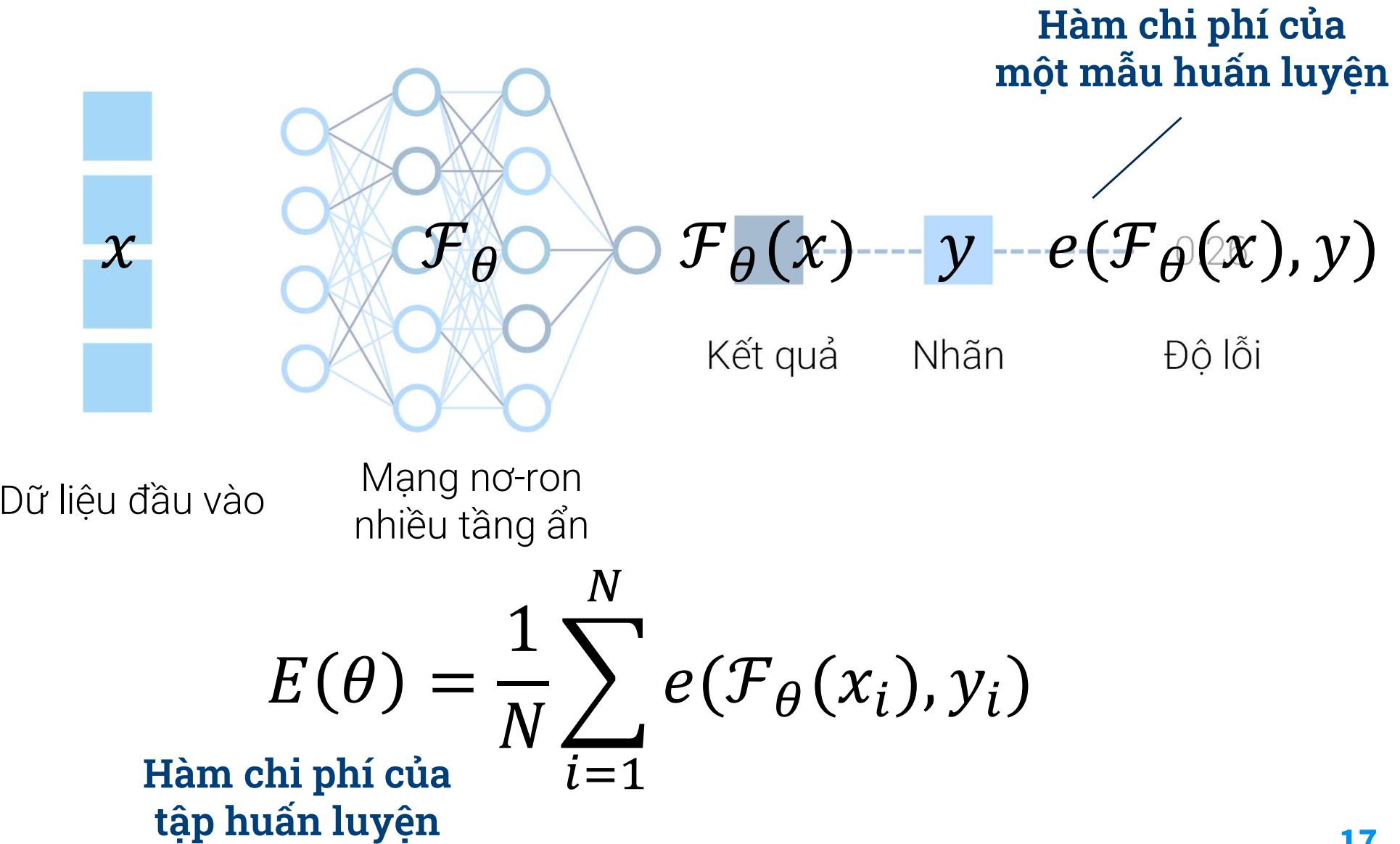
Kiến thức nền tảng

Huấn luyện mạng nơ-ron



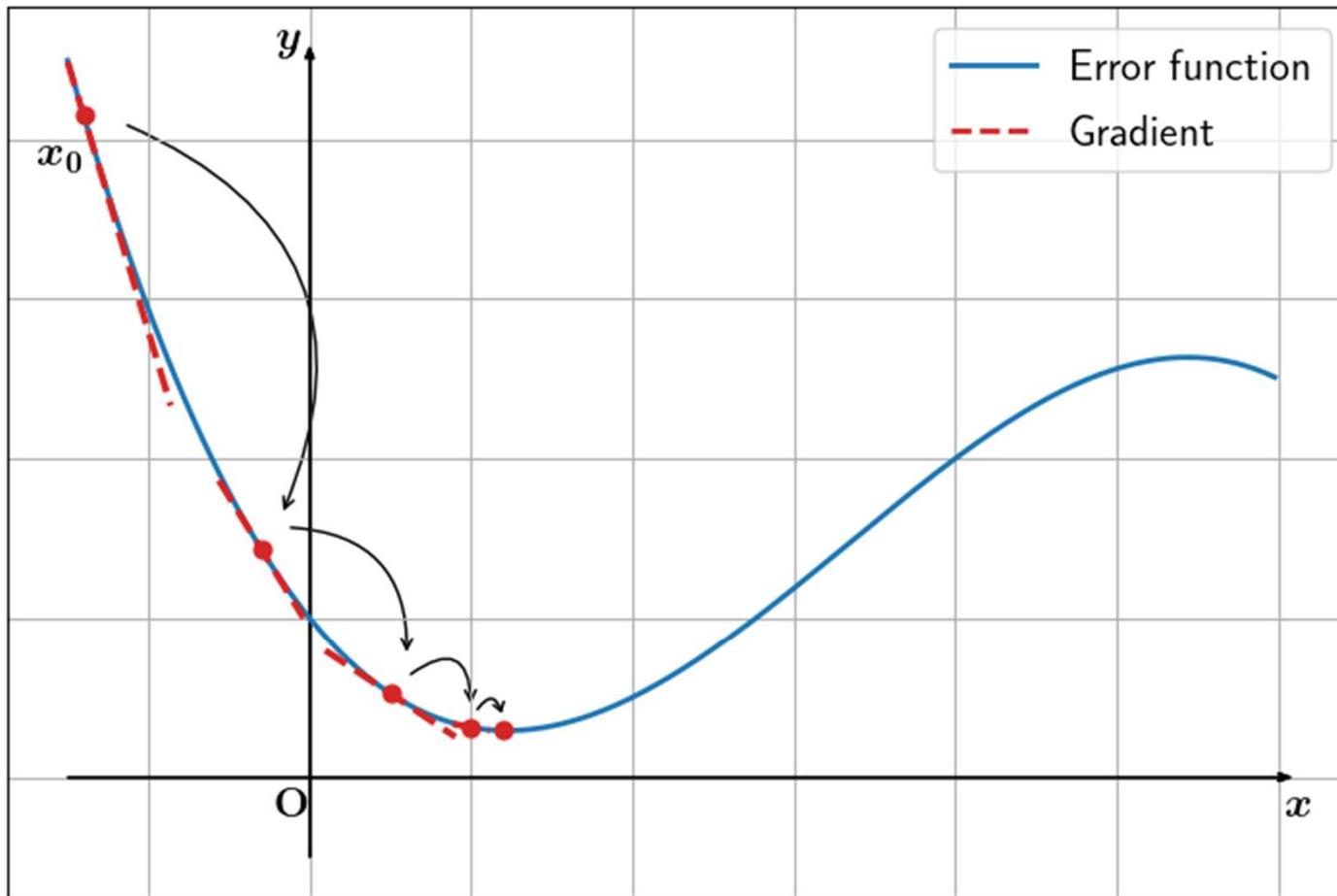
Kiến thức nền tảng

Huấn luyện mạng nơ-ron



Kiến thức nền tảng

Gradient Descent



Minh họa thuật toán Gradient Descent.

Kiến thức nền tảng

Gradient Descent

$$g_t = \nabla_{\theta} E(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta \cdot g_t$$

- Sử dụng gradient của **cả** tập dữ liệu để xác định hướng đi có sự thay đổi lớn nhất.
- Thời gian tính gradient lâu khi tập dữ liệu lớn.
- Chỉ thực hiện một bước cập nhật cho mỗi lần duyệt qua tập dữ liệu.

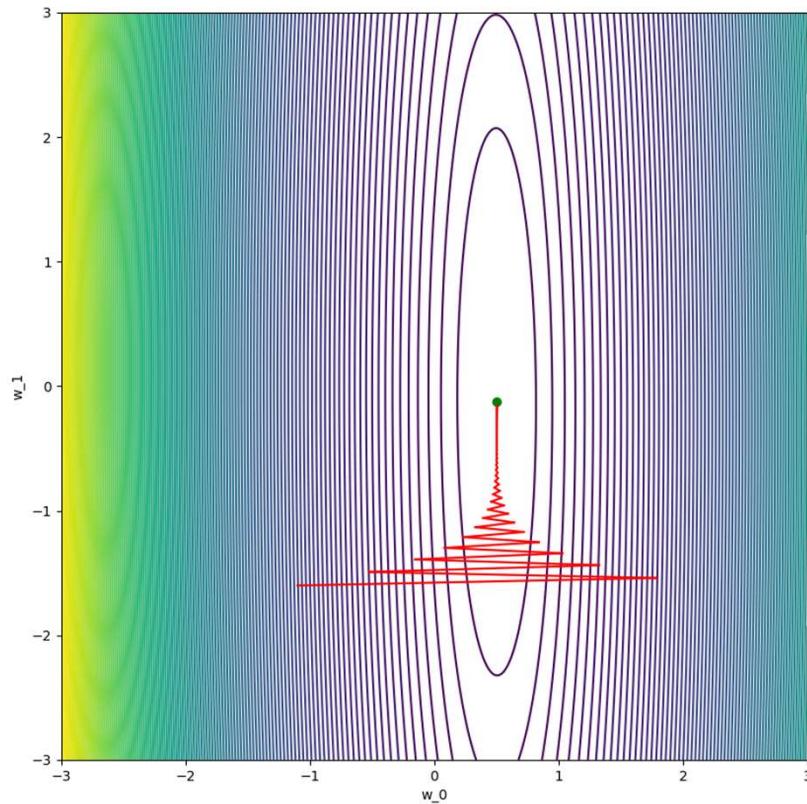
Kiến thức nền tảng

Stochastic Gradient Descent

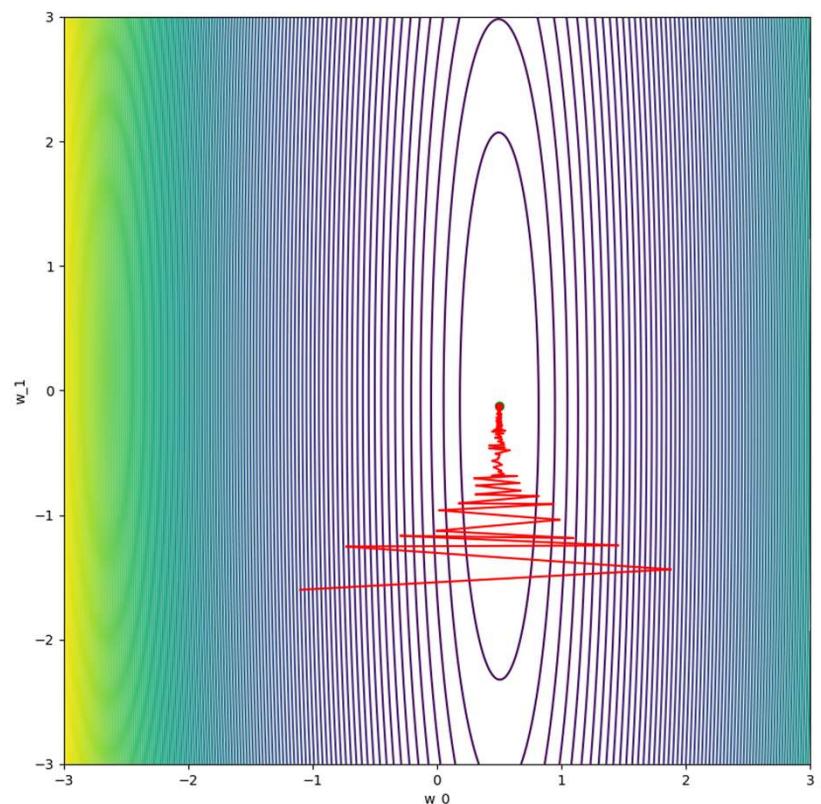
- Sử dụng gradient của một tập con của dữ liệu để xấp xỉ hướng của gradient trên cả tập dữ liệu.
- Thực hiện được nhiều bước cập nhật trong một lần duyệt qua tập dữ liệu.

Kiến thức nền tảng

Stochastic Gradient Descent



Gradient Descent



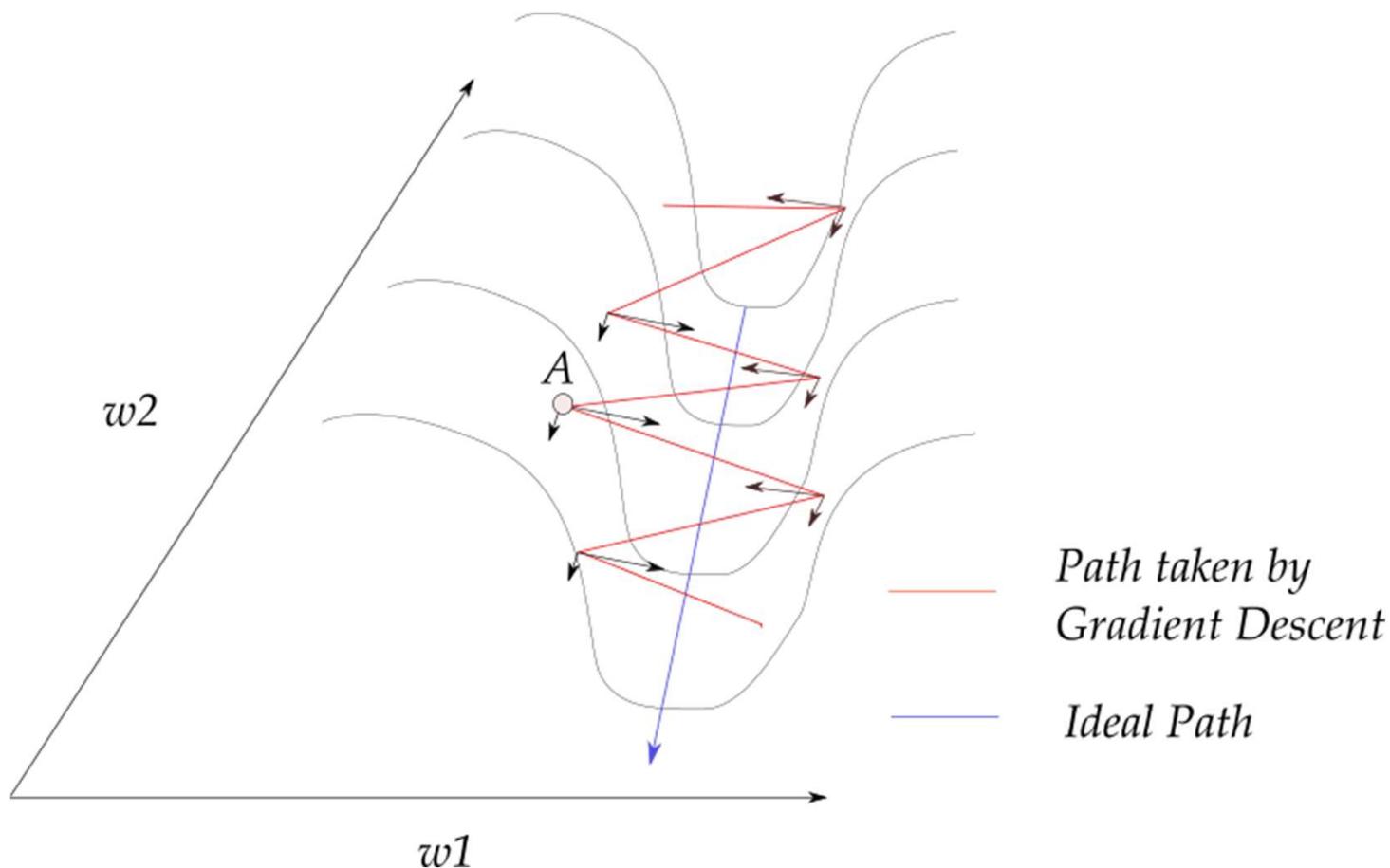
Stochastic Gradient Descent

3.

Thuật toán Adam

SGD với Momentum

Tăng tốc SGD



Minh họa cho hướng đi của gradient trong rãnh hẹp.

SGD với Momentum

Tăng tốc SGD

- Tăng độ lớn cập nhật khi chiều gradient ổn định.
- Giảm độ lớn cập nhật khi chiều gradient thay đổi liên tục.

SGD với Momentum

Tăng tốc SGD

- SGD:

$$g_t = \nabla_{\theta} E(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta \cdot g_t$$

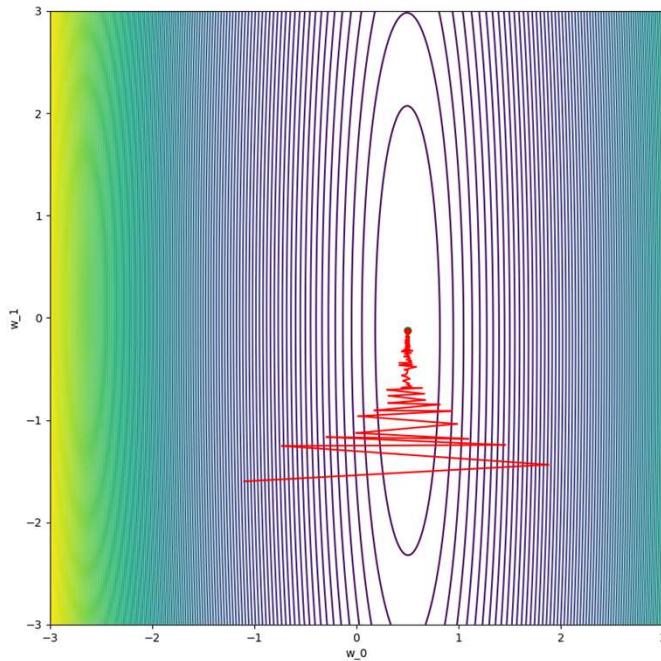
- SGD với Momentum:

$$g_t = \nabla_{\theta} E(\theta_t)$$
$$m_t = \sum_{i=1}^t \beta^{t-i} \cdot g_i \quad (\beta \in (0, 1))$$
$$\theta_{t+1} = \theta_t - \eta \cdot m_t$$

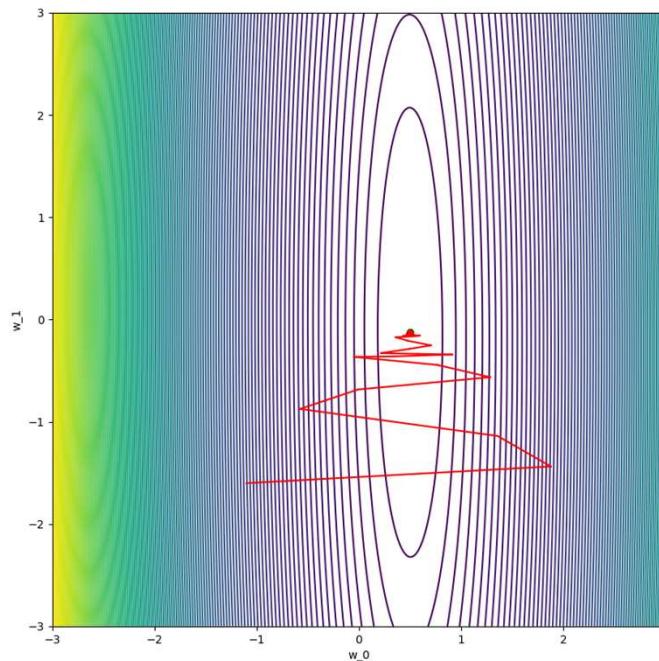
SGD với Momentum

Tăng tốc SGD

- Xấp xỉ gradient của cả tập dữ liệu tốt hơn.
- Giảm dao động trên trực có độ cong lớn.
- Tăng tốc trên trực bằng phẳng



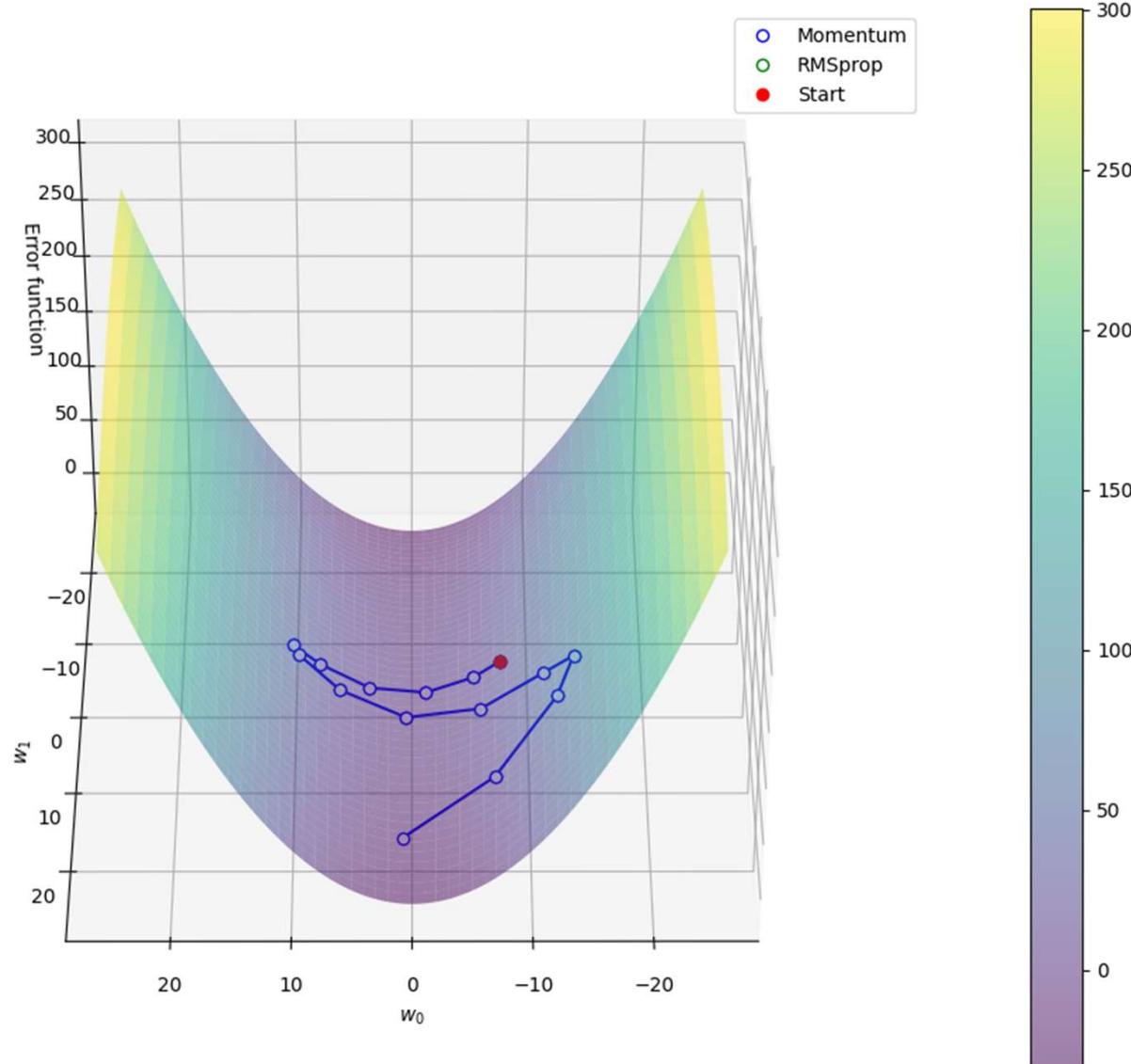
SGD



SGD với Momentum

SGD với Momentum

Momentum trong rãnh rất hẹp



SGD với tỉ lệ học thích ứng

Nguyên lý

- Lấy ý tưởng từ phương pháp bậc hai.
 - Xấp xỉ địa hình xung quanh để tìm bước cập nhật tốt hơn.
- Thích ứng tỉ lệ học cho từng trọng số.
- Cải thiện hướng di chuyển trong trường hợp rãnh rất hẹp.

SGD với tỉ lệ học thích ứng

RMSprop

- Tính đạo hàm theo từng tham số.

$$g_t = \nabla_{\theta} E(\theta_t)$$

- Cập nhật G ở bước hiện tại.

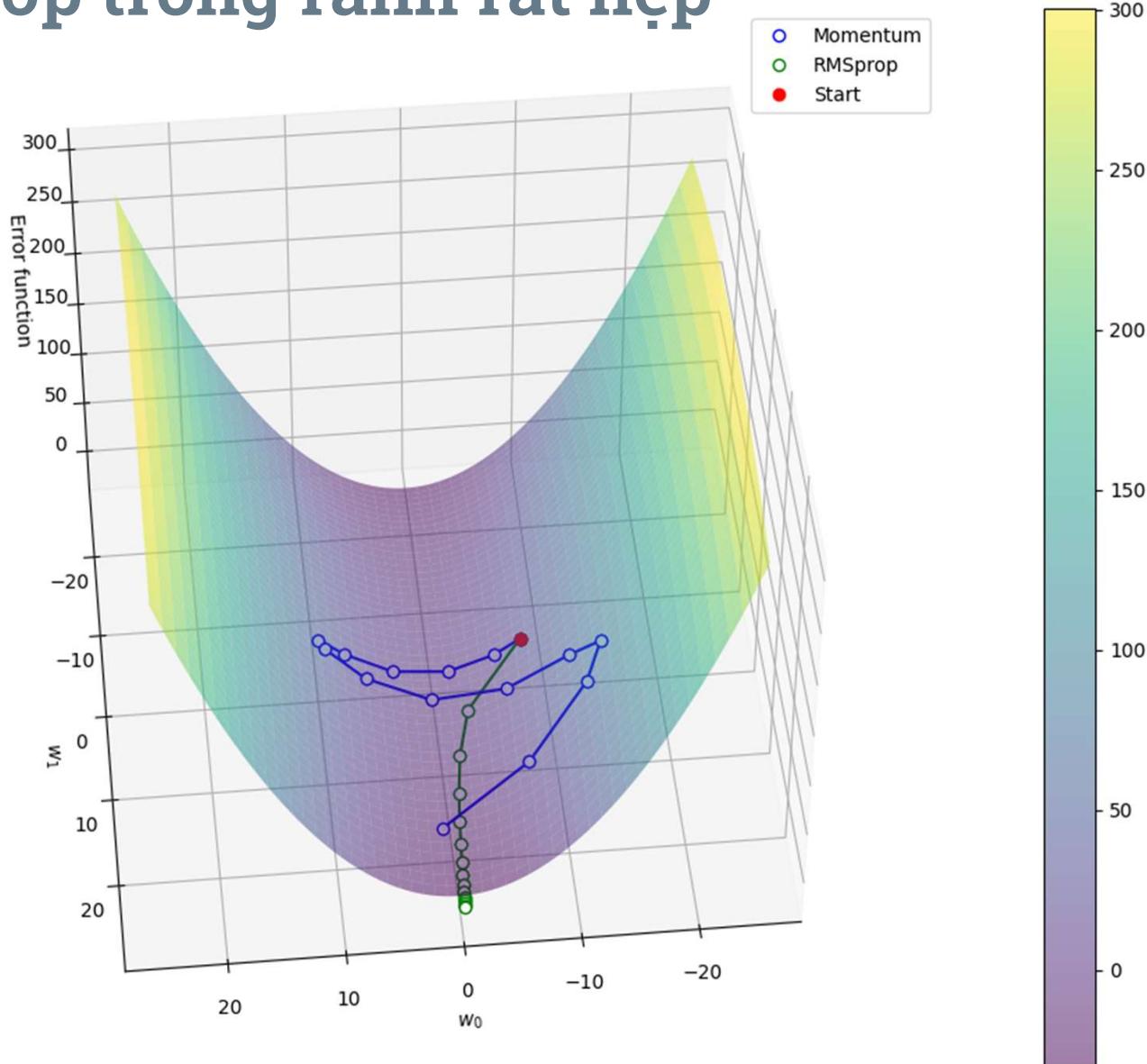
$$G_t = (1 - \gamma) \sum_{i=1}^t \gamma^{t-i} \cdot g_i^2 \quad (\gamma \in (0, 1))$$

- Cập nhật trọng số.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

SGD với tỉ lệ học thích ứng

RMSprop trong rãnh rất hẹp



Thuật toán Adam

Khởi tạo

Khởi tạo 2 biến trung bình chạy \mathbf{m}_t và \mathbf{v}_t

$$\begin{aligned} m_t &= 0 \\ v_t &= 0 \end{aligned}$$

Thuật toán Adam

Các bước thực hiện

Cập nhật m_t và v_t

$$m_t = (1 - \beta) \sum_{i=1}^t \beta^{t-i} g_i$$
$$v_t = (1 - \gamma) \sum_{i=1}^t \gamma^{t-i} \cdot g_i^2$$

Cập nhật **trọng số**

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{m_t}{\sqrt{v_t}} + \epsilon$$

Thuật toán Adam

Kết hợp Momentum và RMSprop

Cập nhật m_t và v_t

$$m_t = (1 - \beta) \sum_{i=1}^t \beta^{t-i} g_i$$
$$v_t = (1 - \gamma) \sum_{i=1}^t \gamma^{t-i} \cdot g_i^2$$

Momentum

$$m_t = \sum_{i=1}^t \beta^{t-i} \cdot g_i$$

RMSprop

$$G_t = (1 - \gamma) \sum_{i=1}^t \gamma^{t-i} \cdot g_i^2$$

Thuật toán Adam

Các bước thực hiện

Cập nhật \mathbf{m}_t và \mathbf{v}_t

$$m_t = (1 - \beta) \sum_{i=1}^t \beta^{t-i} g_i; v_t = (1 - \gamma) \sum_{i=1}^t \gamma^{t-i} \cdot g_i^2$$

Cập nhật **trọng số**

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

Thuật toán Adam

Các bước thực hiện

Cập nhật m_t và v_t

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

Cập nhật **trọng số**

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

Thuật toán Adam

Bias-correction

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

Thực hiện **bias-correction** cho m_t và v_t

$$\widehat{m}_t = \frac{m_t}{(1 - \beta_1^t)}$$

$$\widehat{v}_t = \frac{v_t}{(1 - \beta_2^t)}$$

- Ở các bước đầu tiên, m_t và v_t bị "**bias**" về 0 do khởi tạo.
→ Chia các biến này cho $1 - \beta_1^t$ và $1 - \beta_2^t$ để khôi phục độ lớn bước cập nhật.
- Khi quá trình huấn luyện tiếp tục, t lớn dần.
→ $1 - \beta_1^t$ và $1 - \beta_2^t$ tiến đến 1 → $\widehat{m}_t \cong m_t$ và $\widehat{v}_t \cong v_t$

Thuật toán Adam

Các bước thực hiện

Cập nhật m_t và v_t

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad \text{—— Momentum}$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad \text{—— RMSprop}$$

Thực hiện **bias-correction** cho m_t và v_t

Khôi phục độ lớn
bước cập nhật

$$\widehat{m}_t = m_t / (1 - \beta_1^t)$$
$$\widehat{v}_t = v_t / (1 - \beta_2^t)$$

t tăng dần $\rightarrow \beta^t$ giảm dần
 $\rightarrow 1 - \beta^t$ tiến dần đến 1

Cập nhật **trọng số**

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

4.

Thí nghiệm

Thí nghiệm

Cách thực hiện

- Khởi tạo mạng nơ-ron với các trọng số ngẫu nhiên.
- Lưu các trọng số này làm **điểm xuất phát chung**.
- Với mỗi thuật toán:
 - Nạp lại bộ trọng số đã lưu ở trên cho mạng nơ-ron.
 - Thực hiện tối ưu hóa mạng nơ-ron với số bước xác định.
 - Ghi nhận độ lỗi tại mỗi epoch.

Thí nghiệm

Các thiết lập thí nghiệm

- Sử dụng ngôn ngữ lập trình Python với thư viện NumPy [2] và Pytorch [3].
- Sử dụng GPU NVIDIA RTX 2080, cùng với GPU NVIDIA Tesla T4 và TPU trên nền tảng Google Cloud để tăng tốc độ thực hiện bằng việc tính toán song song.

[2] Charles R. Harris et al., "Array programming with NumPy", Nature 585, 2020.

[3] Adam Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", NIPS 2019.

Thí nghiệm

Các tập dữ liệu

Tập dữ liệu	Kích thước tập huấn luyện	Kích thước tập kiểm thử	Kích thước ảnh/tập từ điển	Số lớp đối tượng
MNIST [4]	60 000	10 000	28x28	10
CIFAR10 [5]	50 000	10 000	32x32	10
ImageNet [6]	1 281 166	50 000	-	1000
Penn Treebank [7]	929 000	82 000	10 000	-

[4] Yann LeCun and Corinna Cortes and CJ Burges, "MNIST handwritten digit database", 2010.

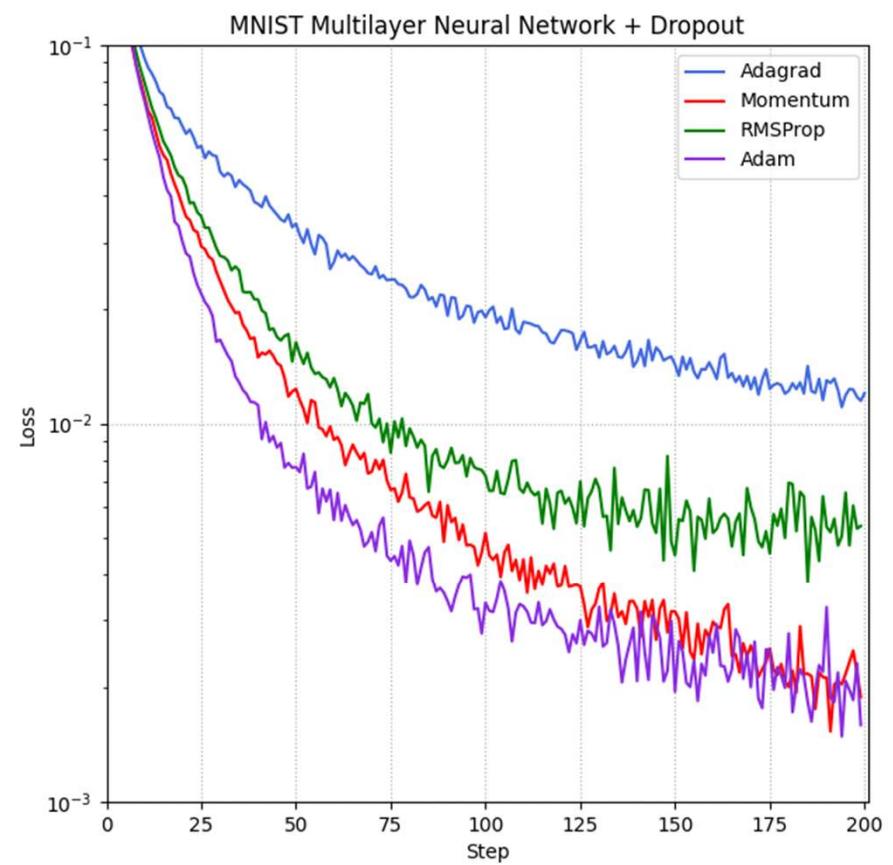
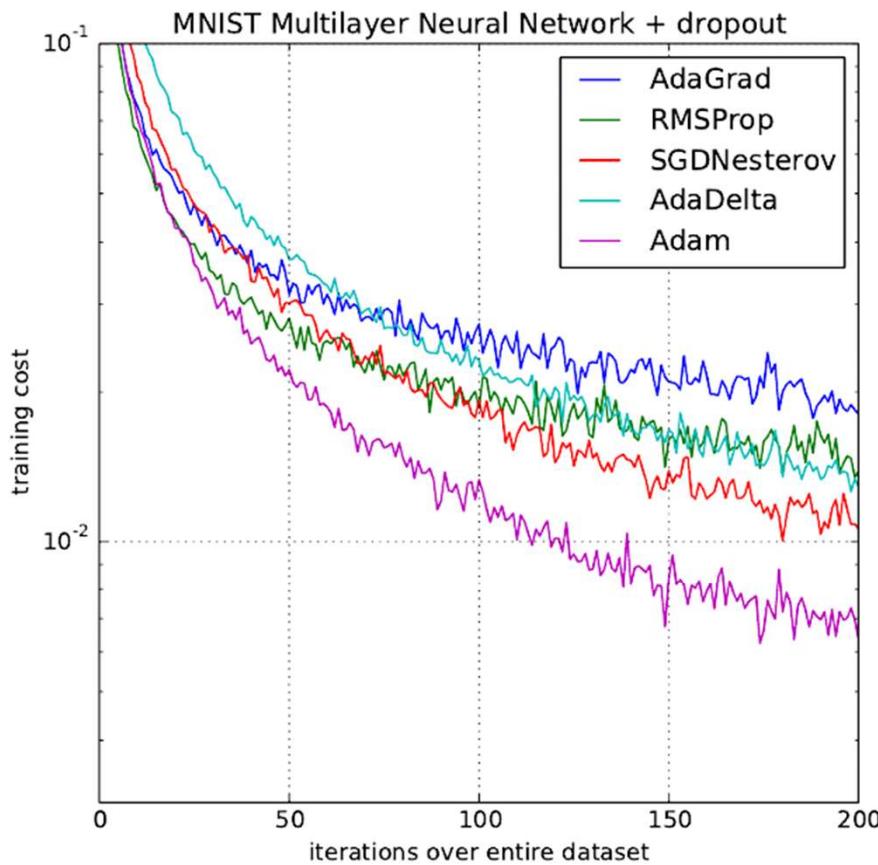
[5] Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", 2009.

[6] J. Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database", CVPR 2009.

[7] Mitchell P. Marcus and Mary Ann Marcinkiewicz and Santorini Beatrice, "Building a Large Annotated Corpus of English: The Penn Treebank", Comput. Linguit. 1993.

Thí nghiệm tái tạo

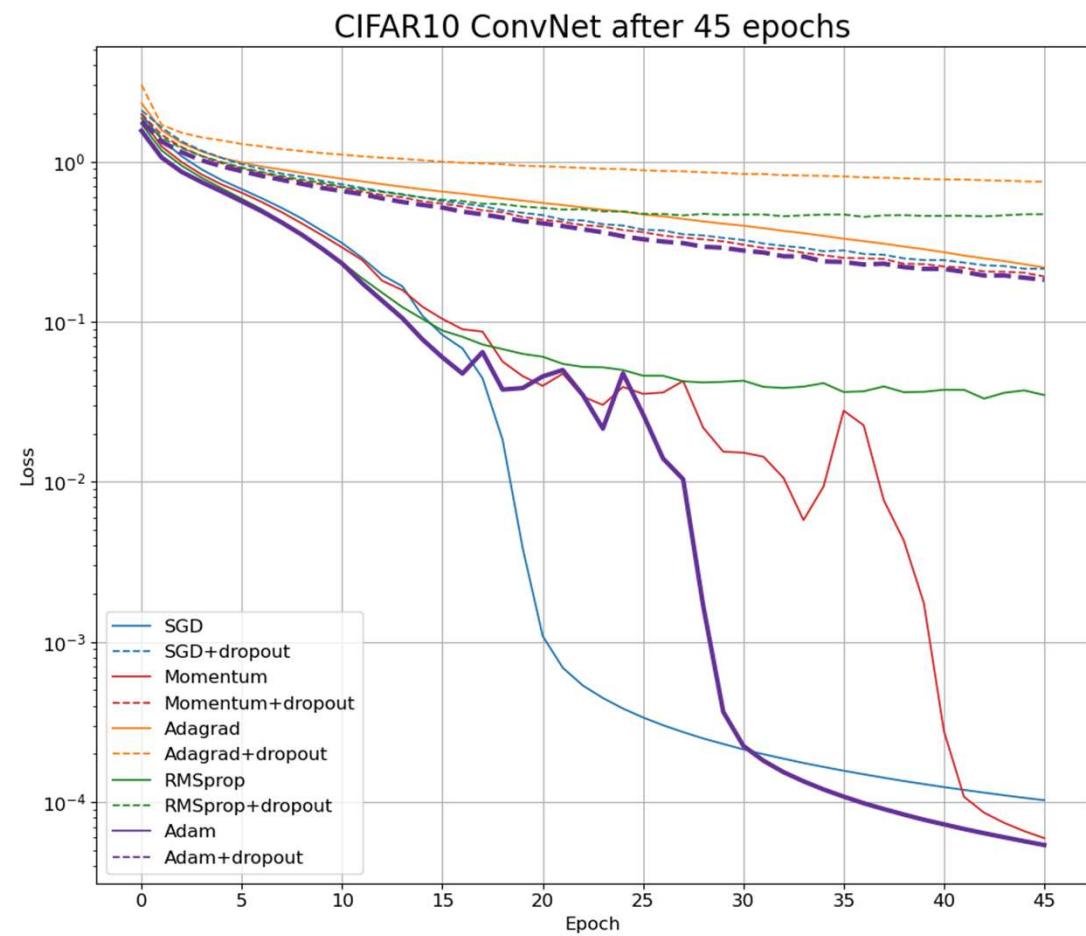
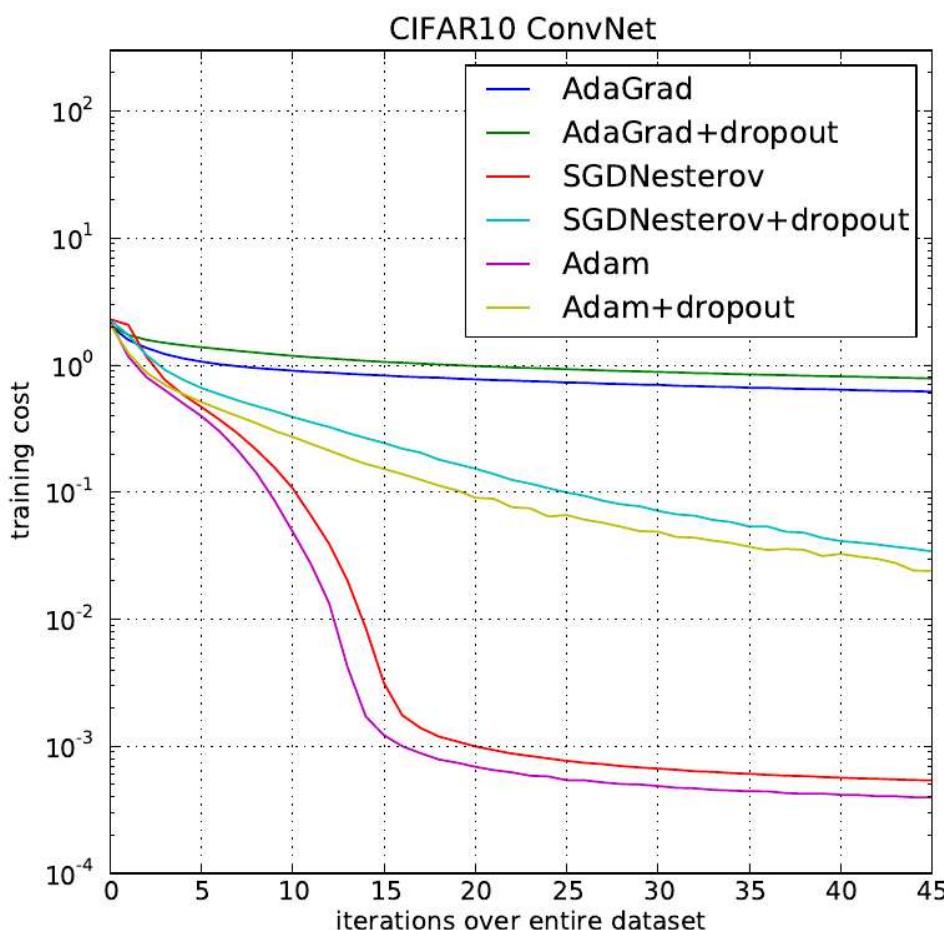
1000fc-1000fc - MNIST



Kết quả thí nghiệm Multi-layer Neural Network của tác giả (trái) [8] và kết quả cài đặt của chúng tôi (phải).

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR 2015.

Thí nghiệm tái tạo c64-c64-c128-1000FC – CIFAR10

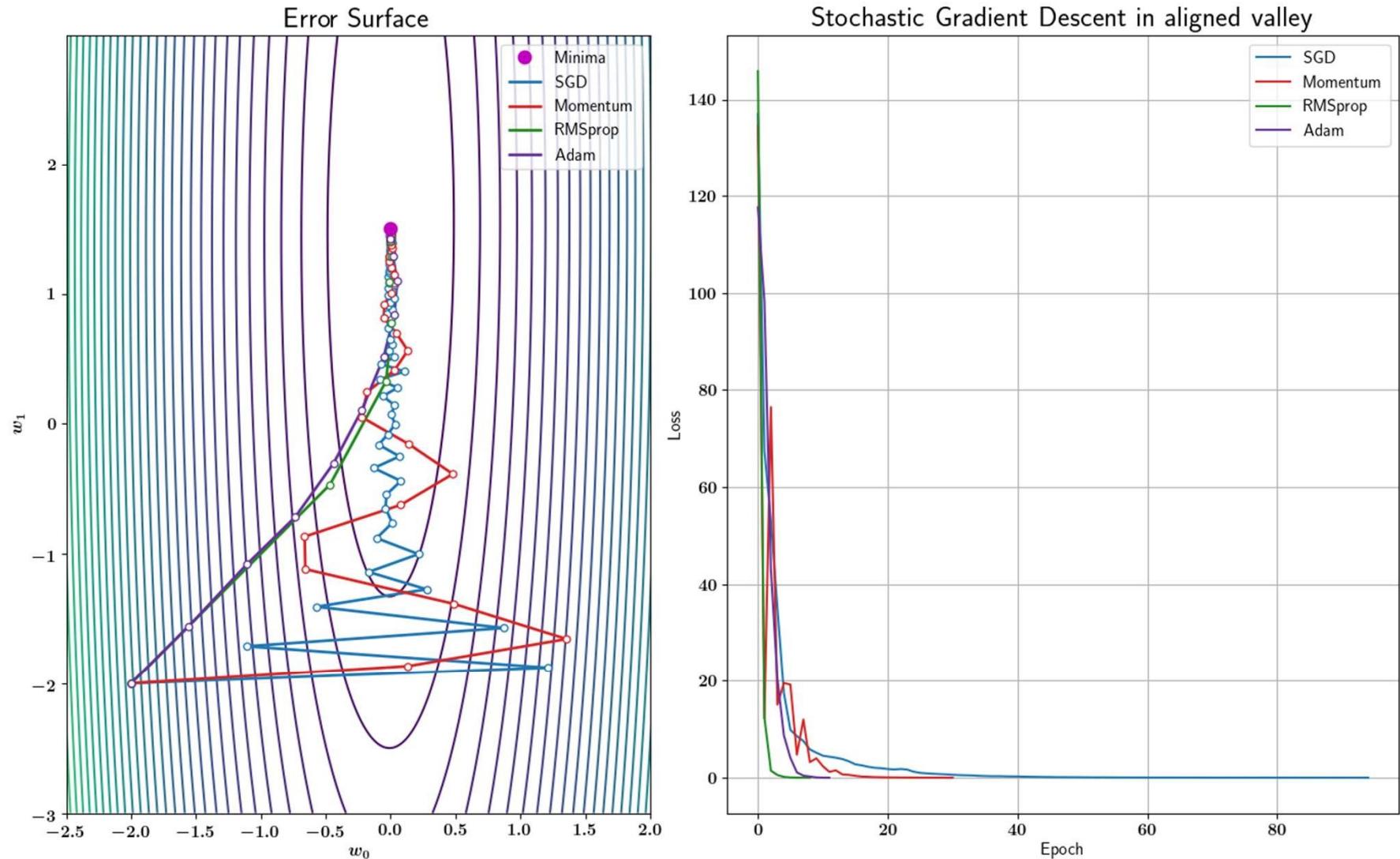


Kết quả thí nghiệm *Convolutional Neural Network* của tác giả (trái) [8] và kết quả của chúng tôi.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR 2015.

Thí nghiệm trực quan hóa

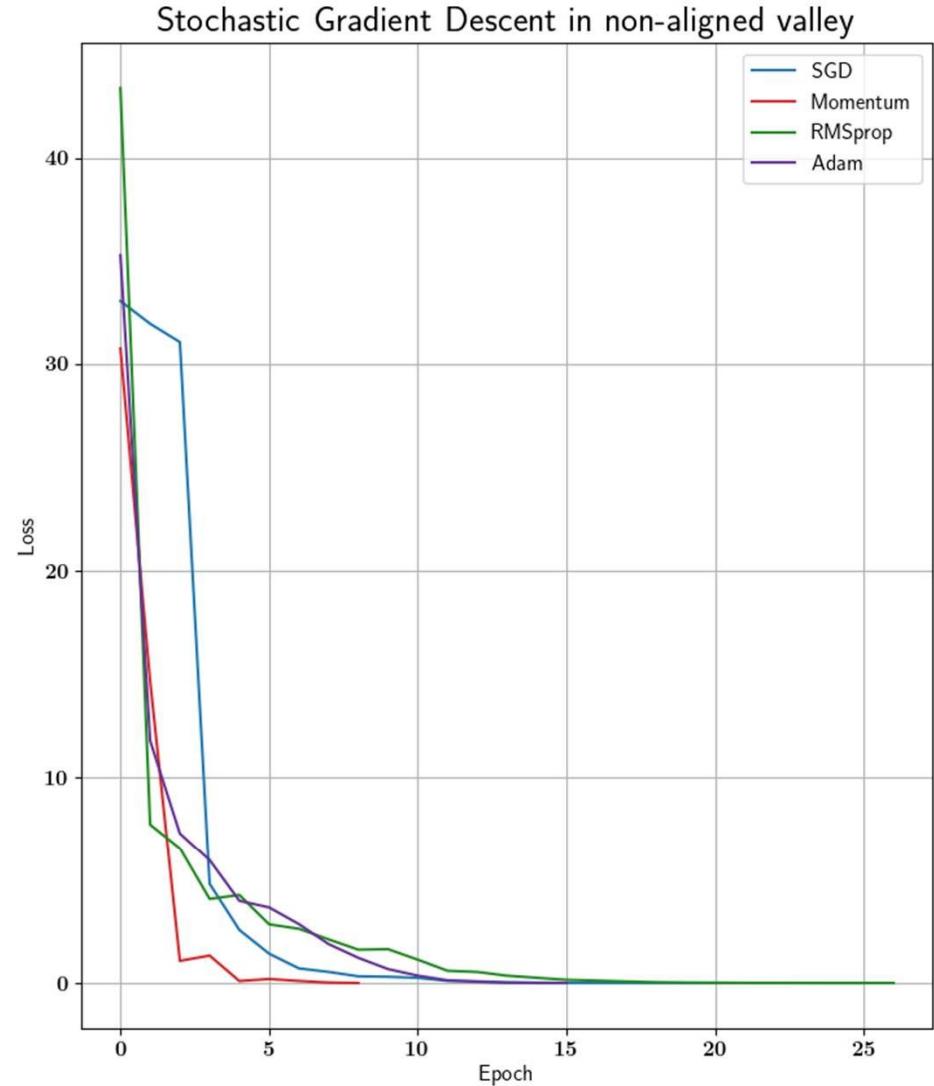
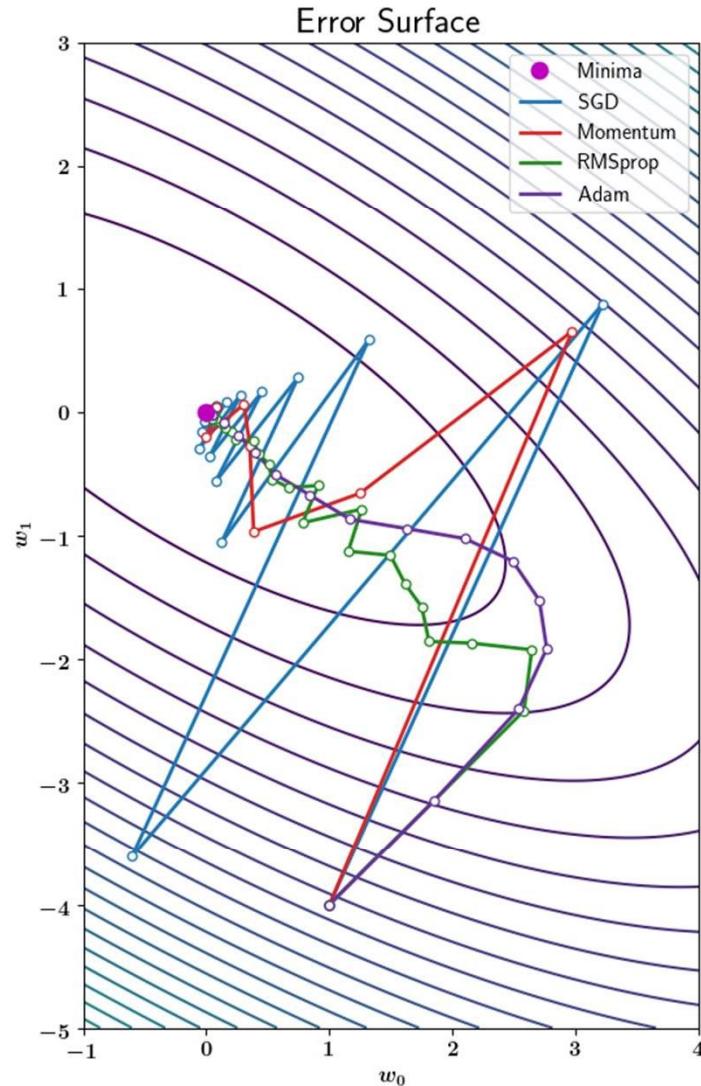
Rãnh hẹp trùng với trọng số



*Đường đi, độ lớn bước cập nhật theo từng chiều,
và độ lỗi của các thuật toán.*

Thí nghiệm trực quan hóa

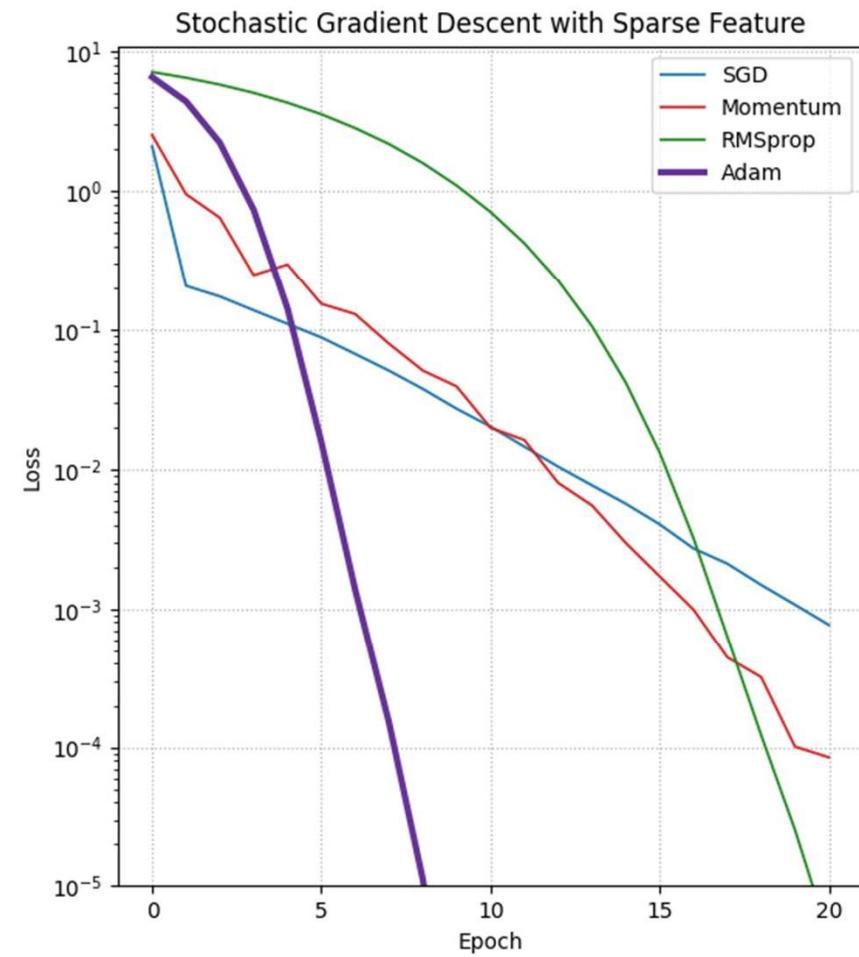
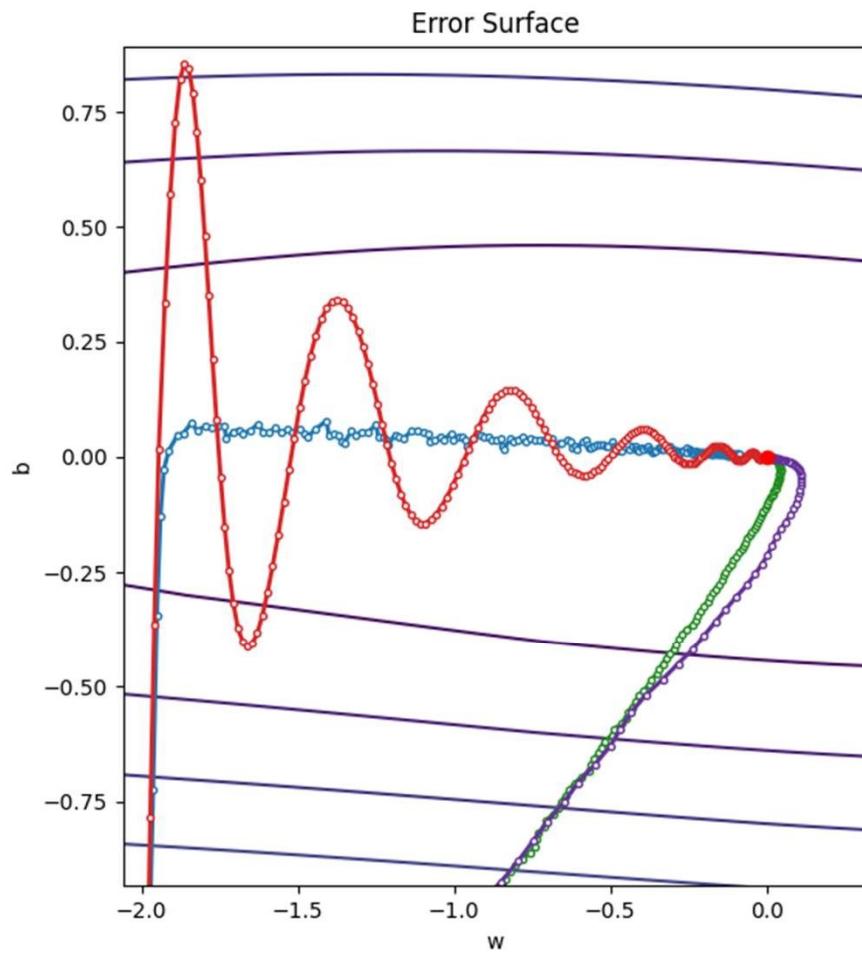
Rãnh hẹp không trùng với trọng số



*Đường đi của các thuật toán trong rãnh hẹp
có hướng không trùng với trọng số.*

Thí nghiệm trực quan hóa

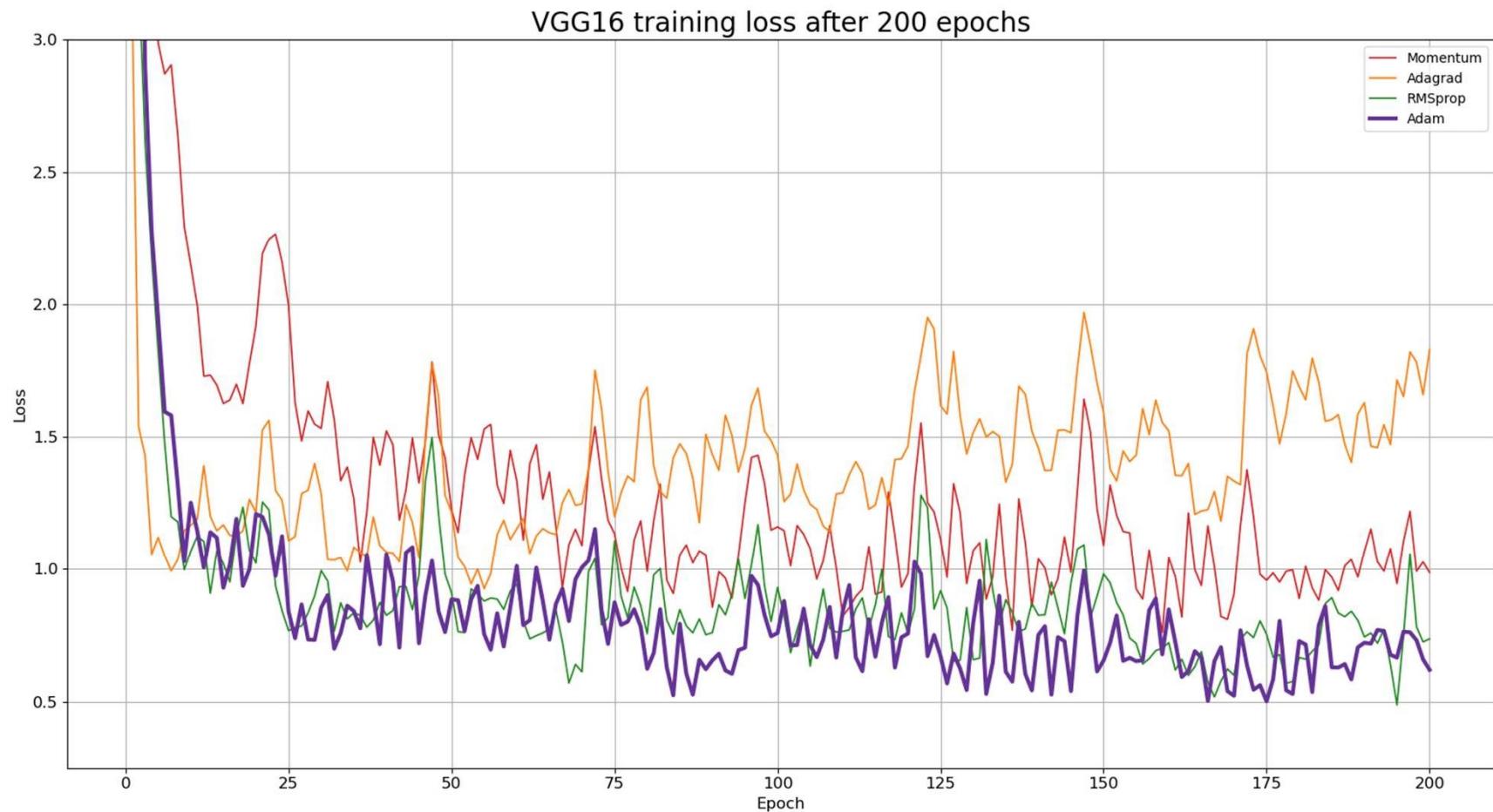
Rãnh rất hẹp do đặc trưng thưa



*Đường đi của các thuật toán trong bề mặt lỗi (a)
và độ lỗi (b) của từng thuật toán trong thí nghiệm đặc trưng thưa.*

Thí nghiệm ngũ cảnh thực tế

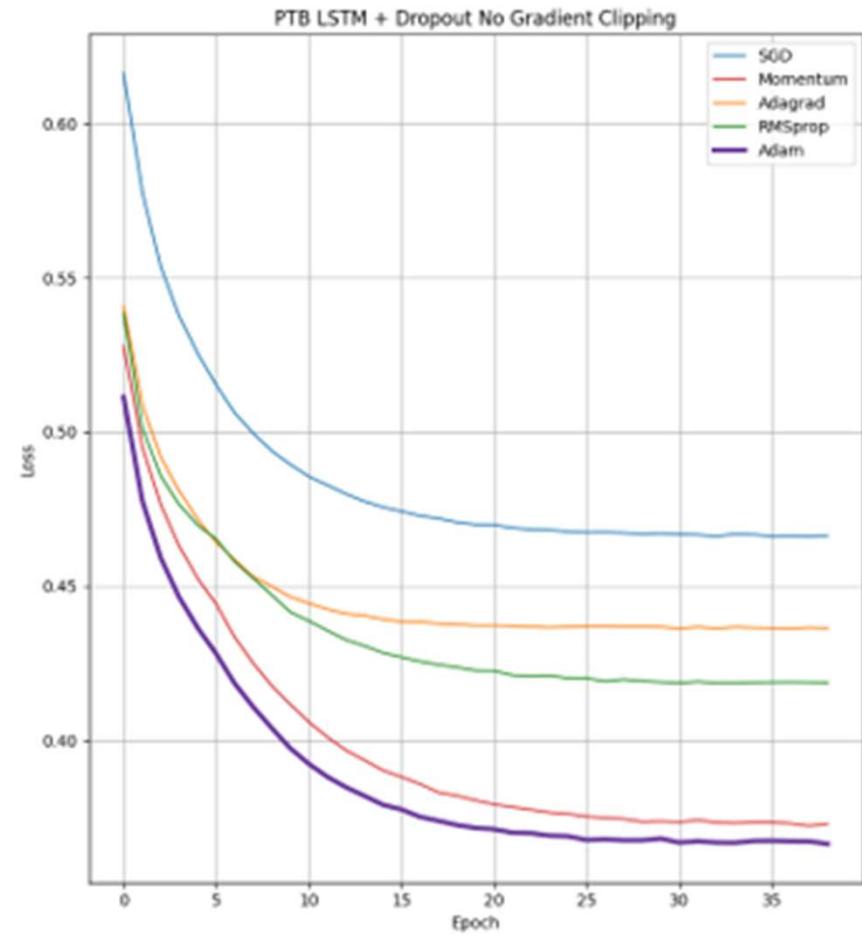
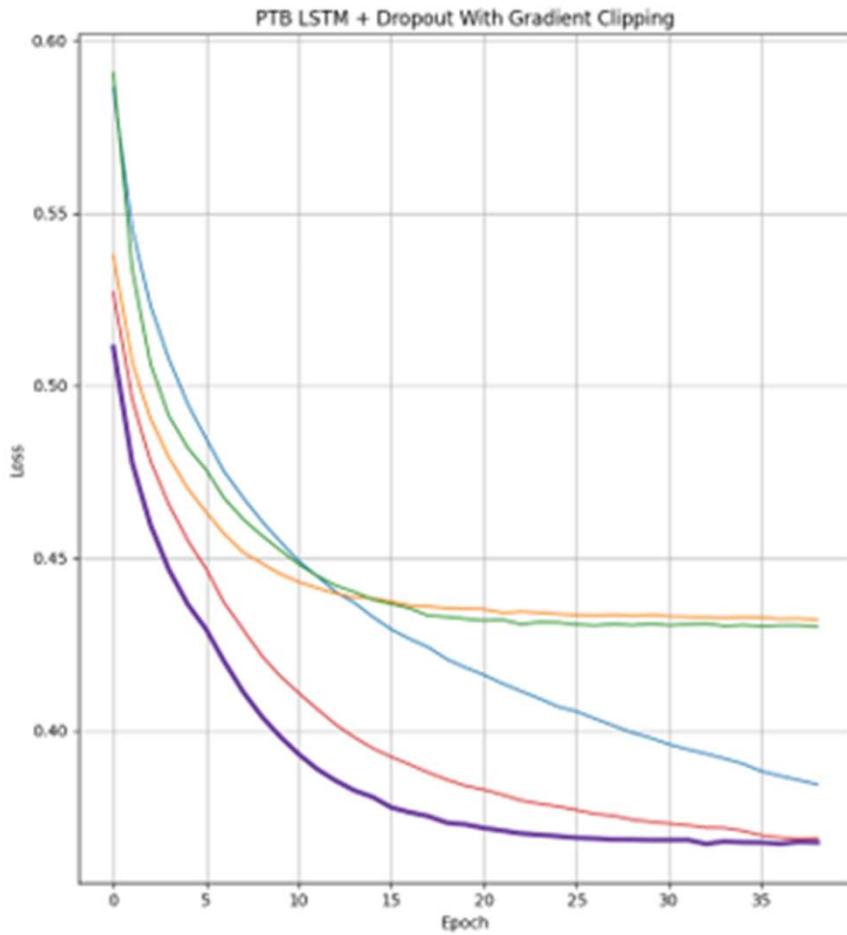
Huấn luyện mô hình VGG16 (16 tầng ẩn)



Độ lỗi huấn luyện mô hình VGG16 trên tập dữ liệu ImageNet.

Thí nghiệm ngữ cảnh thực tế

Huấn luyện Mô hình LSTM 35 tầng ẩn



Độ lỗi huấn luyện mô hình ngôn ngữ trên tập Penn Treebank có sử dụng gradient clipping (trái) và không sử dụng gradient clipping (phải).

5.

Tổng kết

Tổng kết

Tổng kết

- Tìm hiểu và cài đặt lại được theo bài báo “Adam: a method for stochastic optimization”.
- Tái tạo lại được các kết quả thí nghiệm công bố trong bài báo.
- Phân tích ưu/nhược điểm của Adam trong nhiều tình huống khác nhau.
- Thí nghiệm trên các mạng nơ-ron nhiều tầng ẩn được sử dụng trong thực tế.

Tổng kết

Hướng phát triển

- Tìm hiểu sự ảnh hưởng của trung bình chạy lên các bước cập nhật.
- Tìm hiểu ảnh hưởng của kích thước minibatch lên quá trình huấn luyện của thuật toán Adam.
- Thực hiện thí nghiệm nhiều lần để giảm thiểu sự ngẫu nhiên của quá trình huấn luyện.
- Tìm hiểu các thuật toán cải tiến từ cơ sở của thuật toán Adam.

Tài liệu tham khảo

- [1] Yann Dauphin *et al.*, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", NIPS 2014.
- [2] Charles R. Harris *et al.*, "Array programming with NumPy", Nature 585, 2020.
- [3] Adam Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", NIPS 2019.
- [4] Yann LeCun and Corinna Cortes and CJ Burges, "MNIST handwritten digit database", 2010.
- [5] Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", 2009.
- [6] J. Deng *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database", CVPR 2009.

Tài liệu tham khảo

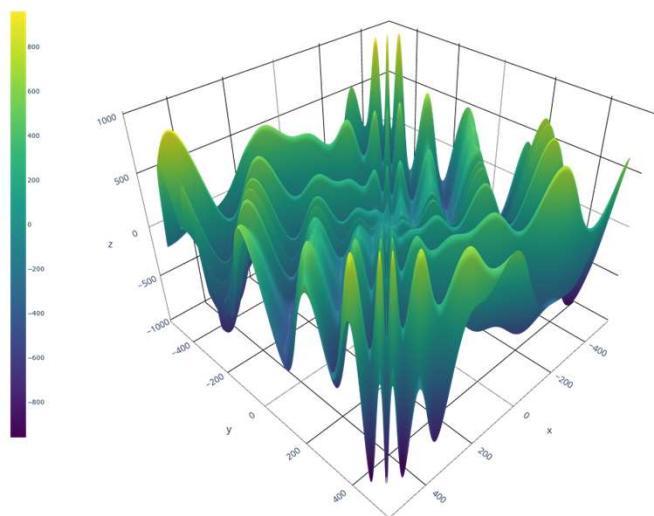
- [7] Mitchell P. Marcus and Mary Ann Marcinkiewicz and Santorini Beatrice, "Building a Large Annotated Corpus of English: The Penn Treebank", Comput. Linguist. 1993.
- [8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR 2015.

**Xin cảm ơn quý thầy cô
đã lắng nghe!**

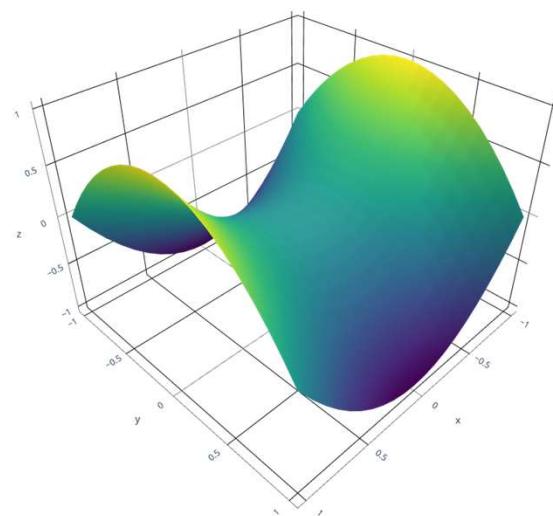
Phụ lục

Phụ lục

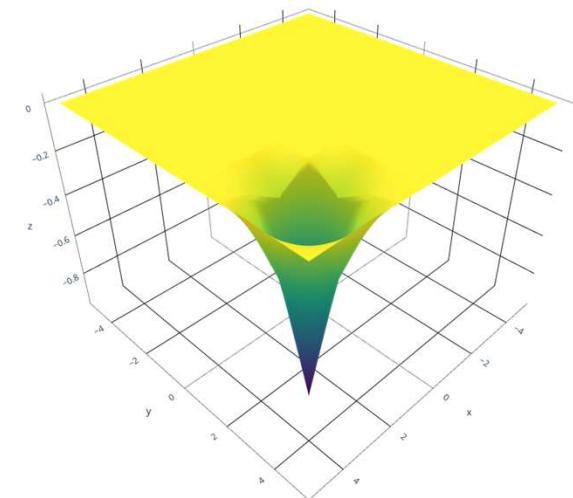
Một số dạng bề mặt lồi khác



Bề mặt lồi phức tạp



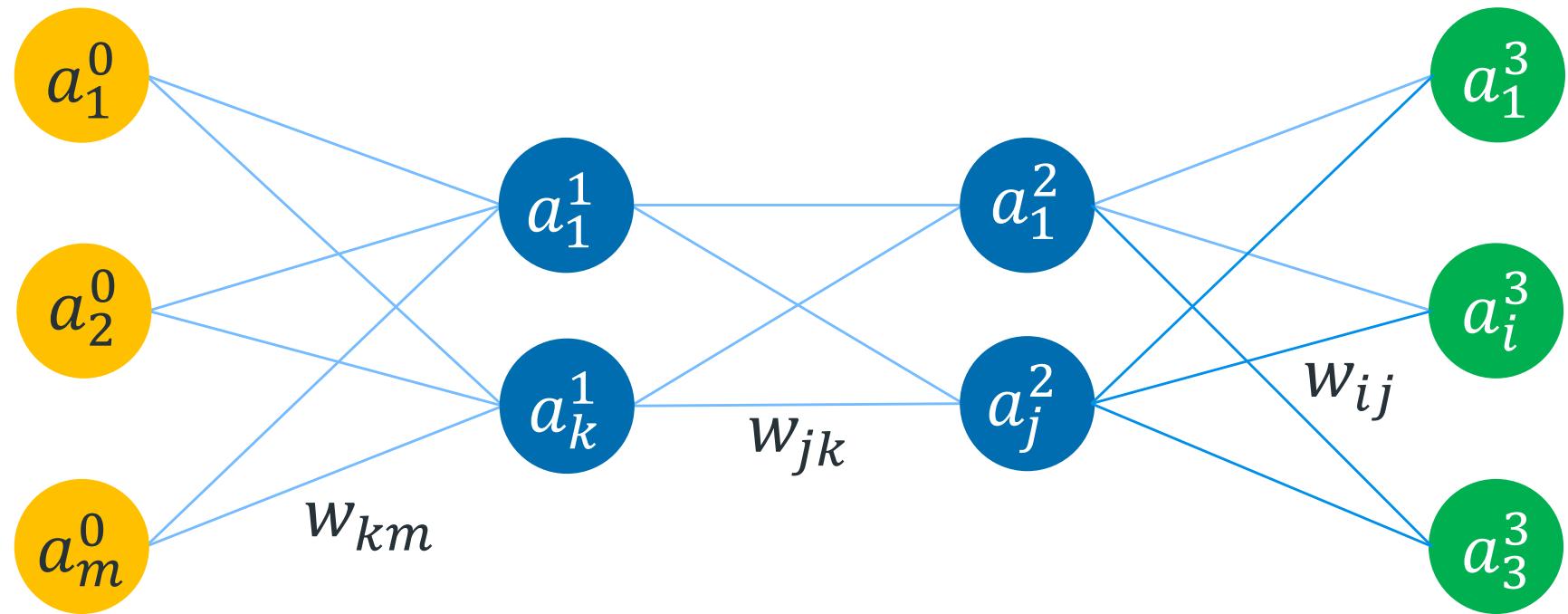
Điểm yên ngựa



Vùng băng phẳng

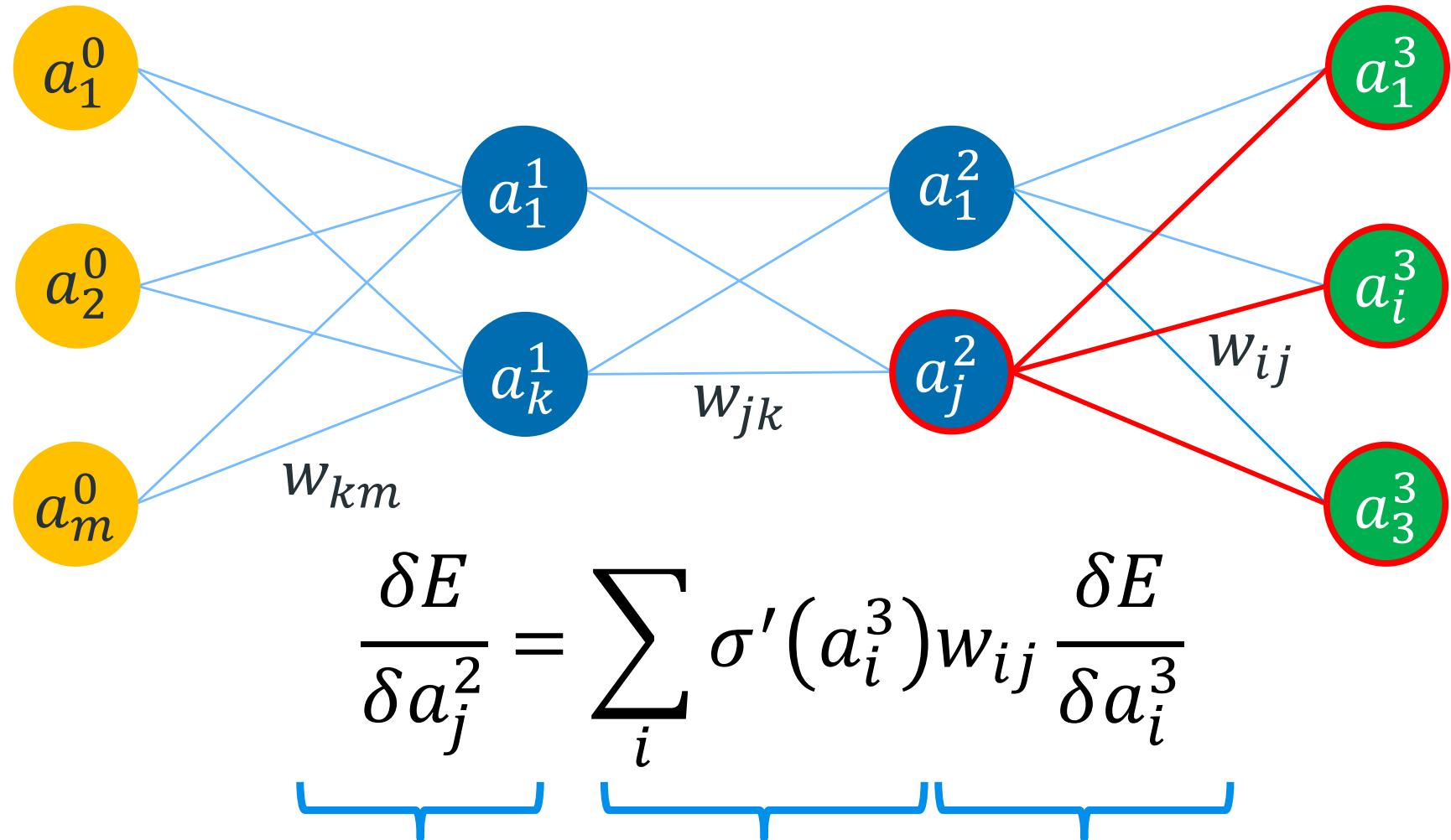
Phụ lục

Thuật toán lan truyền ngược



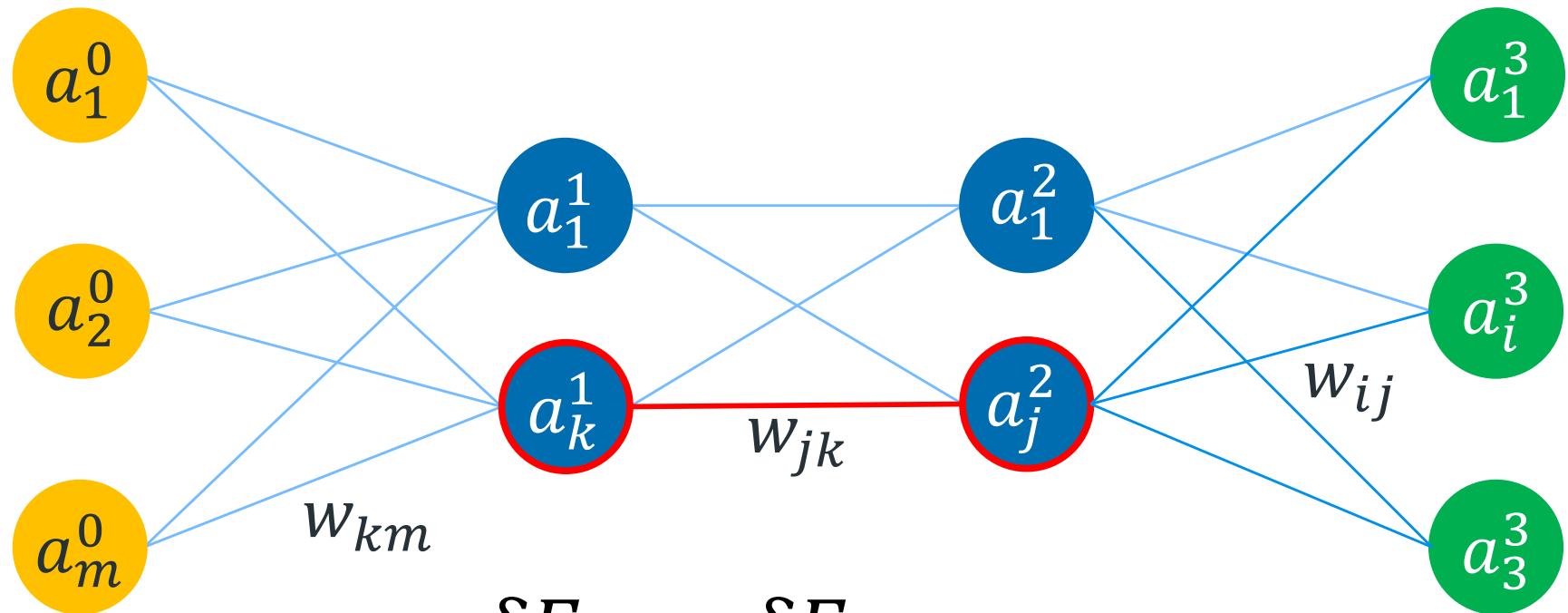
Phụ lục

Thuật toán lan truyền ngược



Phụ lục

Thuật toán lan truyền ngược

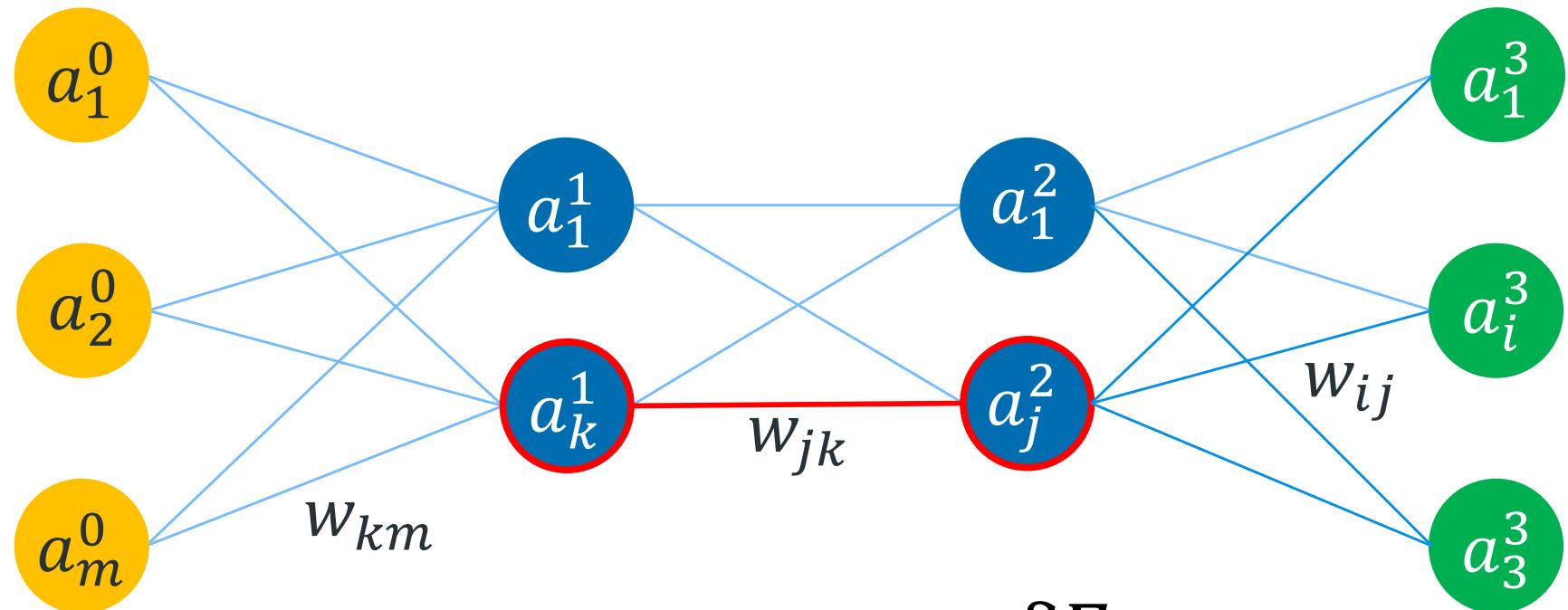


$$\frac{\delta E}{\delta w_{jk}} = \frac{\delta E}{\delta a_j^2} \sigma'(a_j^2) a_k^1$$



Phụ lục

Thuật toán lan truyền ngược



$$w_{jk} = w_{jk} - \eta \frac{\delta E}{\delta w_{jk}}$$

Phụ lục

Tổng quan thuật toán Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Require: α : Stepsize

Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates

Require: $f(\theta)$: Stochastic objective function with parameters θ

Require: θ_0 : Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)

$v_0 \leftarrow 0$ (Initialize 2nd moment vector)

$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)

end while

return θ_t (Resulting parameters)

Phụ lục

Độ lỗi Mean Squared Error (MSE)

- Ý tưởng:
 - Đánh giá khoảng cách giữa giá trị dự đoán với giá trị thật.
 - Thường được sử dụng trong bài toán hồi quy.
 - ❖ Đánh lỗi cao hơn khi giá trị dự đoán chênh lệch nhiều so với giá trị thật.
 - ❖ Giảm độ lỗi rất nhiều khi sự chênh lệch nhỏ dần.

Phụ lục

Độ lỗi Mean Squared Error (MSE)

- Với N điểm dữ liệu, \hat{y}_i là giá trị thứ i dự đoán được, và y_i là giá trị đúng tương ứng:

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

- Ví dụ: với $y = 1$:
 - $\hat{y} = 2.1 \rightarrow \text{Squared Error (SE)} = 1.21$
 - $\hat{y} = 1.9 \rightarrow SE = 0.81$

Phụ lục

Độ lỗi Cross-Entropy

- Ý tưởng:
 - So sánh hai phân phối xác suất.
 - Thường được sử dụng trong bài toán phân lớp.
 - ❖ So sánh phân phối của các lớp đối tượng được dự đoán với phân phối trong tập dữ liệu huấn luyện.

Phụ lục

Độ lỗi Cross-Entropy

- Với C là số lớp đối tượng, \hat{y}_i là tỉ lệ dự đoán được của lớp đối tượng thứ i , và y_i là tỉ lệ đúng tương ứng:

$$CE = - \sum_{i=1}^C y_i \cdot \log(\hat{y}_i)$$

- Ví dụ:
 - $y_i = [[1, 0], [0, 1]]$ và $\hat{y}_i = [[0.75, 0.25], [0.4, 0.6]]$
 - $CE = -[1 \cdot \log(0.75) + 1 \cdot \log(0.6)]$
 - $= 0.7985$

Phụ lục

Xấp xỉ ma trận Hessian

- Tính đạo hàm theo từng tham số.

$$g_t = \nabla_{\theta} E(\theta_t)$$

- Cập nhật G ở bước hiện tại.

$$G_t = \gamma \cdot G_{t-1} + (1 - \gamma) \cdot \text{diag}(g_t \cdot g_t^T)$$

- Cập nhật trọng số.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t$$

Phụ lục

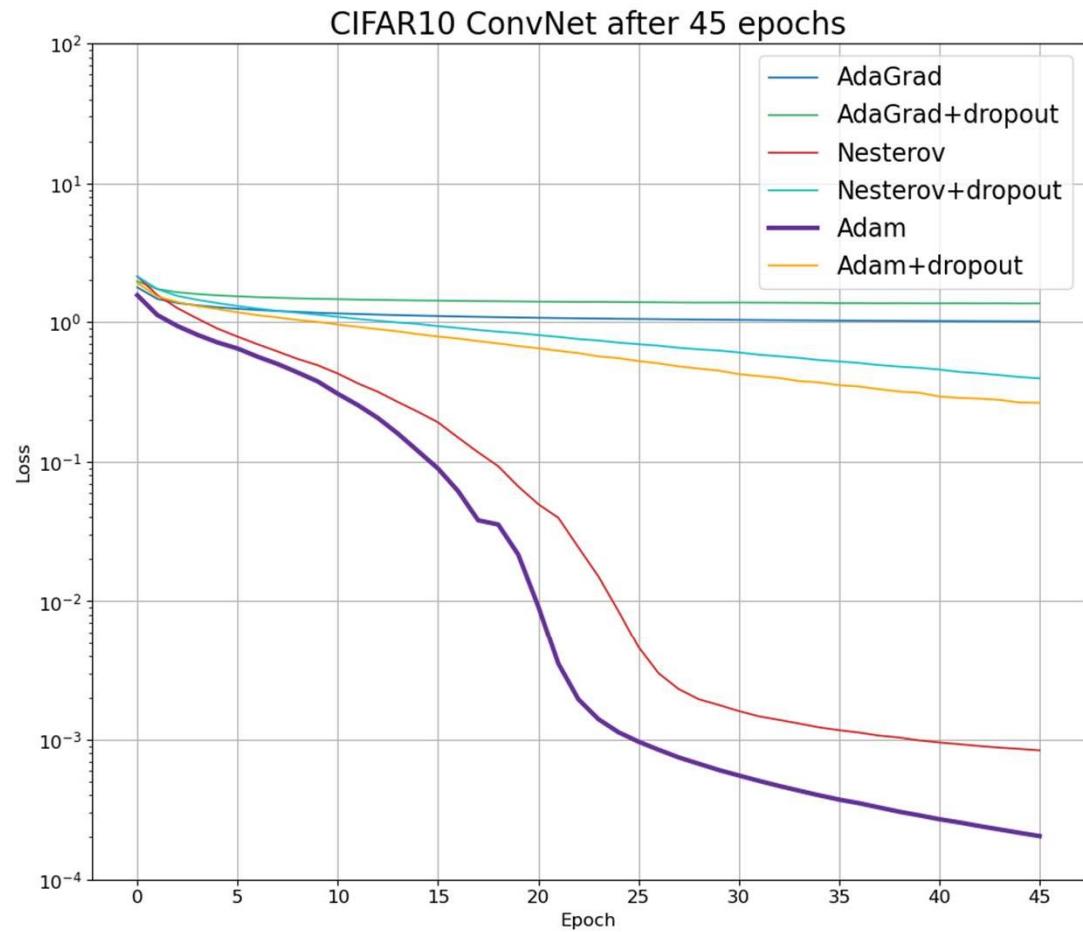
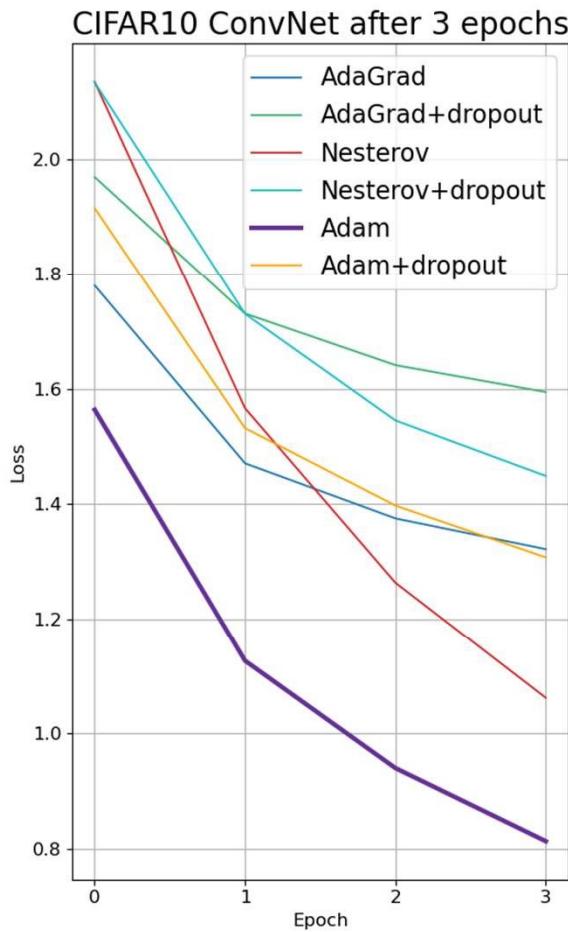
Xấp xỉ ma trận Hessian

$$g_t = \begin{bmatrix} g_{t,1} \\ g_{t,2} \\ g_{t,3} \\ \vdots \\ g_{t,n} \end{bmatrix}, g_t^T = [g_{t,1} \quad \dots \quad g_{t,n}]$$

$$g_t \cdot g_t^T = \begin{bmatrix} g_{t,1} \\ g_{t,2} \\ g_{t,3} \\ \vdots \\ g_{t,n} \end{bmatrix} \cdot [g_{t,1} \quad \dots \quad g_{t,n}] = \begin{bmatrix} g_{t,1}^2 & \cdots & g_{t,1} \cdot g_{t,n} \\ \vdots & \ddots & \vdots \\ g_{t,n} \cdot g_{t,1} & \cdots & g_{t,n}^2 \end{bmatrix}$$

Phụ lục

Tái tạo thí nghiệm Convolutional Neural Network



Kết quả thí nghiệm Convolutional Neural Network với bộ siêu tham số mô phỏng kết quả bài báo.

Phụ lục

Kết quả và thời gian thực hiện thí nghiệm Convolutional Neural Network

Tên thuật toán	Thời gian thực hiện (giây)	Độ lỗi thấp nhất
SGD/SGD+dropout	$6.02 \pm 0.07/6.04 \pm 0.07$	0.0001/0.2139
Momentum/ Momentum+dropout	$6.10 \pm 0.07/6.08 \pm 0.07$	0.00006/0.1914
Adagrad/ Adagrad+dropout	$6.14 \pm 0.09/6.16 \pm 0.09$	0.2178/0.7496
RMSprop/ RMSprop+dropout	$6.16 \pm 0.07/6.19 \pm 0.07$	0.03307/0.4512
Adam/Adam+dropout	$6.19 \pm 0.07/6.23 \pm 0.07$	0.000054/0.1824

*Kết quả và thời gian thực hiện một epoch của các thuật toán
trong thí nghiệm Convolutional Neural Network.*