ẢNH HƯỞNG CỦA HÀM KÍCH HOẠT ĐẾN MÔ HÌNH MẠNG NƠRON TÍCH CHẬP

VĨNH ANH NGHIÊM QUÂN, NGUYỄN LÊ TRUNG THÀNH
ĐINH THỊ DIỆU MINH, TRẦN HOÀI NHÂN
Khoa Tin học, Trường Đại học Sư phạm, Đại học Huế

Tóm tắt: Mạng nơron tích chập (CNN) ngày càng được sử dụng phổ biến trong xử lý hình ảnh nói chung và phân lớp hình ảnh nói riêng. Để cải thiện hiệu năng của mạng, việc tinh chỉnh các siêu tham số (hyper-parameters) là cần thiết. Trong bài báo này, chúng tôi đề cập đến tầm quan trọng của việc lựa chọn hàm kích hoạt phù hợp khi huấn luyện CNN. Chúng tôi so sánh các hàm kích hoạt cơ bản và các hàm kích hoạt được đề xuất trong một số nghiên cứu gần đây. Để đánh giá mức ảnh hưởng của chúng đến hiệu năng của CNN, chúng tôi tiến hành các thí nghiệm với hai mô hình, một đơn giản một phức tạp lần lượt trên hai tập dữ liệu hình ảnh phổ biến MNIST và CIFAR-10. Cách thức tiến hành thực nghiệm và các tiêu chí đánh giá được tham khảo dựa trên DAWNBench với một số thay đổi nhỏ. Kết quả cho thấy hàm kích hoạt ReLU và các biến thể của nó đem lại độ chính xác cao sớm hơn các dạng hàm kích hoạt khác, mặc dù ưu thế về tổng thời gian huấn luyện là không đáng kể.

Từ khóa: CNN, hàm kích hoạt.

1. MỞ ĐẦU

Hiện nay, phương pháp học sâu (deep learning) là một trong những phương pháp nhận được nhiều sự quan tâm từ cộng đồng nghiên cứu về học máy (machine learning). Tuy học sâu không đồng nghĩa với mạng nơron nhân tạo (artificial neural networks – ANN) nhưng khi nhắc đến thuật toán học sâu, người ta chủ yếu quan tâm đến các mạng nơron nhân tạo được triển khai trên diện rộng, đặc biệt là trong các bài toán về nhận dạng hình ảnh.

Kể từ năm 2012, sau thành công của mạng nơron tích chập (convolutional neural networks – CNN) trong các cuộc thi nhận dạng hình ảnh diện rộng, các mô hình mạng này ngày càng được nâng cấp để giải quyết các bài toán ngày một phức tạp hơn. Ngoài các hàm kích hoạt (activation function) cơ bản như hàm tiếp tuyến hyperbol tanh hay hàm logistic, ngày càng nhiều hàm kích hoạt mới được đề xuất như ReLU[1], LeakyReLU[2], PReLU[3], ELU[4], SELU[5].

Do số lượng nơron và các liên kết nơron trong các mô hình mạng nơron hiện đại dao động từ con số vài ngàn đến vài triệu, quá trình huấn luyện của các mạng này tiêu tốn rất nhiều thời gian cũng như đòi hỏi khá cao về cấu hình phần cứng. Việc tối ưu hóa cấu trúc mô hình mạng, mà điển hình là việc lựa chọn một hàm kích hoạt thích hợp, do đó là rất cần thiết.

Tuy hiện nay đã có nhiều hàm kích hoạt mới để người xây dựng mô hình có thể chọn lựa, nhưng tính hiệu quả của các hàm này thường chỉ được minh chứng qua một mô hình đặc thù được đề cập đến trong bài báo của một nhóm nghiên cứu. Vì thiếu sự so sánh về hiệu

Tạp chí Khoa học, Trường Đại học Sư phạm, Đại học Huế
ISSN 1859-1612, Số 03(51)/2019: tr. 125-134
Ngày nhận bài: 27/12/2018; Hoàn thành phản biện: 05/01/2019; Ngày nhận đăng: 28/12/2018

năng của các hàm kích hoạt này trong cùng một mô hình nên người xây dựng mô hình thường có xu hướng chọn một hàm kích hoạt ngẫu nhiên hoặc thử nhiều hàm kích hoạt khác nhau rồi chọn hàm giúp huấn luyện mô hình nhanh nhất. Điều này dẫn đến việc lãng phí thời gian và tài nguyên.

Để khắc phục những hạn chế trên, trong nghiên cứu này, nhóm chúng tôi áp dụng các hàm kích hoạt đã nêu cho một số bài toán có sử dụng mô hình mạng nơron tích chập phổ biến nhất hiện nay, đồng thời so sánh ảnh hưởng của từng hàm kích hoạt đến hiệu năng của mô hình trong từng bài toán. Việc xác định được hàm kích hoạt phù hợp sẽ giúp người dùng có được cái nhìn bao quát hơn khi xây dựng mô hình giải quyết các bài toán tương tự.

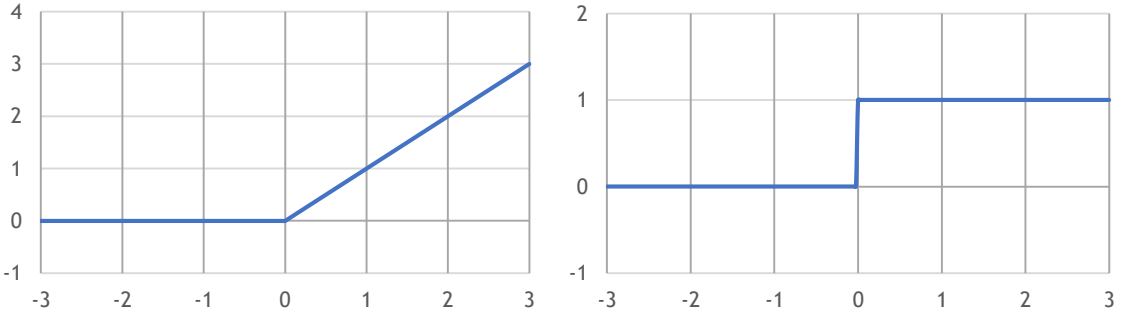
Cấu trúc tiếp theo của bài viết được trình bày như sau: mục 2 trình bày về các hàm kích hoạt; mục 3 về mô hình sẽ được khảo sát, mục 4 mô tả thực nghiệm và mục 5 là phần kết luận.

2. CÁC HÀM KÍCH HOẠT

2.1. Hàm ReLU

Được đề xuất trong [2], Rectified Linear Unit (ReLU)[1] được định nghĩa:

$$\mathrm{ReLU}(x) = \max(0, x)$$



Hình 1. *Hàm ReLU và đạo hàm*

Đây là hàm không tuyến tính và không có đạo hàm tại $x = 0$. Tuy nhiên trong tính toán, đạo hàm của ReLU tại $x = 0$ được ngầm định bằng 0 hoặc 1.

ReLU đơn giản về mặt tính toán hơn các hàm logistic / sigmoid hay tanh do không phải sử dụng các phép toán lũy thừa. Theo [1], ReLU có đạo hàm bằng 1 khi nơron được kích hoạt nên giúp tránh được hiện tượng vanishing gradient thường gặp trong các hàm sigmoid hay tanh.

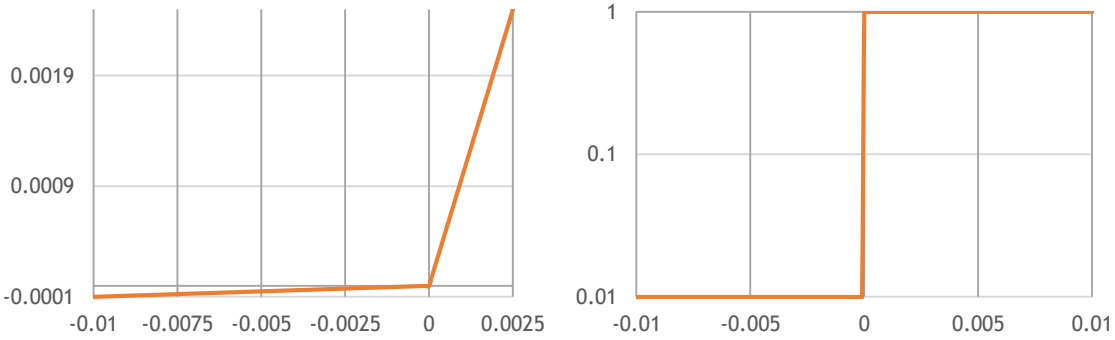
Tuy nhiên theo [2], nhược điểm của hàm này là khi nơron không được kích hoạt, gradient sẽ bằng 0. Điều này dẫn đến việc một nơron có nguy cơ không bao giờ được kích hoạt do các giải thuật tối ưu mạng nơron bằng gradient sẽ không điều chỉnh trọng số của một nơron nếu nơron đó không kích hoạt ngay từ đầu.

2.2. Hàm Leaky ReLU

Leaky Rectified Linear Unit (Leaky ReLU)[2] là một biến thể của ReLU. Leaky ReLU được định nghĩa như sau:

$$\text{Leaky ReLU} = \begin{cases} x & x > 0 \\ 0.01x & x \le 0 \end{cases}$$

Dễ thấy hàm Leaky RELU thay thế phần âm của ReLU bằng một hàm tuyến tính với hệ số góc nhỏ cố định (0.01).



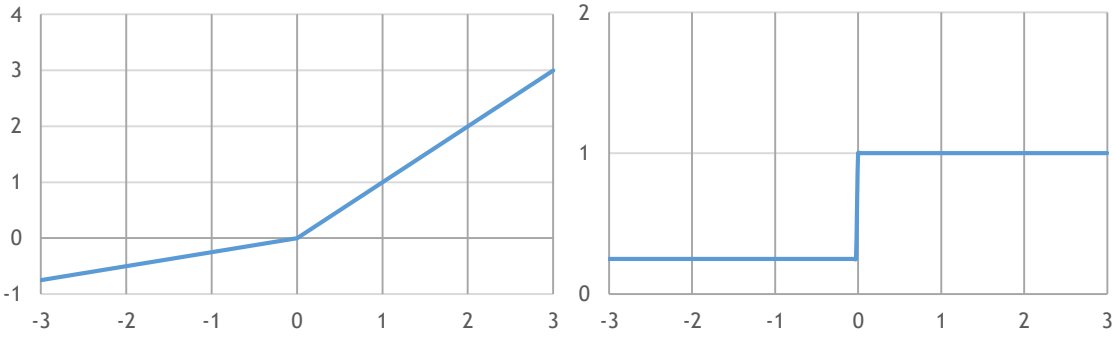
Hình 2. *Hàm Leaky ReLU và đạo hàm (trục Y của đạo hàm theo thang logarit)*

2.3. Hàm PReLU

Hàm PReLU hay còn gọi là Parametric ReLU[3] được định nghĩa:

$$\text{PReLU} = \begin{cases} x & x > 0 \\ ax & x \le 0 \end{cases} \text{ hay } \text{PReLU} = \max(0, x) + a\min(0, x)$$

Đây là dạng tổng quát của hàm Leaky ReLU, hệ số góc a của phần âm có giá trị khởi điểm là 0.25 và được cập nhật trong quá trình huấn luyện mô hình. Khi a = 0.01, PReLU trở thành Leaky ReLU.



Hình 3. *Hàm PReLU và đạo hàm*

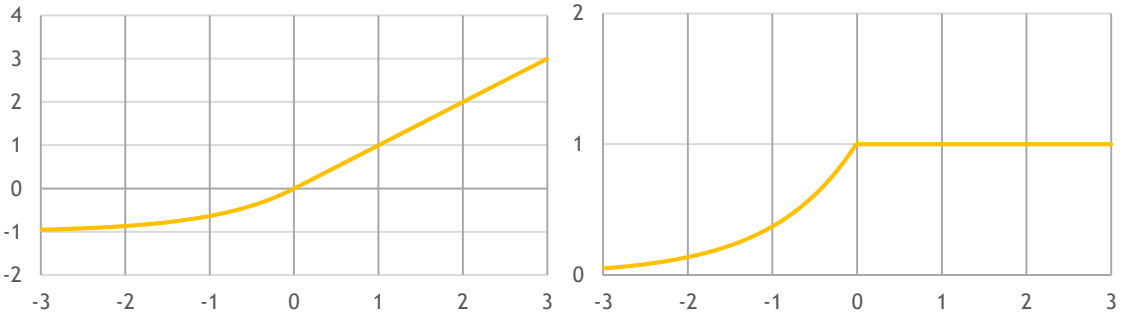
2.4. Hàm ELU

Hàm ELU (Exponential Linear Unit)[4] được định nghĩa như sau:

$$\text{ELU} = \begin{cases} x & x > 0 \\ \alpha e^x - \alpha & x \le 0 \end{cases} \text{ hay } \text{ELU}(x) = \max(0, x) + \min(0, \alpha e^x - \alpha))$$

Trong [4], tác giả lấy $\alpha = 1$.

Từ đồ thị của ELU, dễ thấy phần dương là hàm đồng nhất như ReLU còn phần âm trơn dần về giá trị $-\alpha$.



Hình 4. *Hàm ELU và đạo hàm*

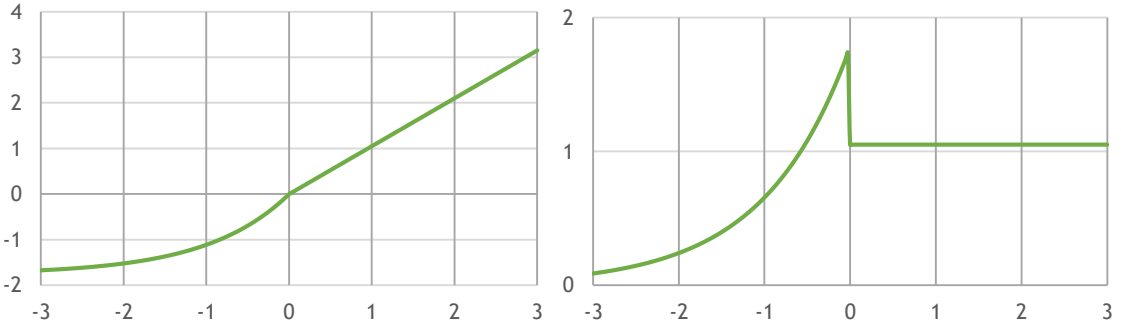
2.5. Hàm SELU

Hàm SELU hay Scaled Exponential Linear Unit[5] được định nghĩa như sau:

$$\text{SELU} = \lambda \begin{cases} x & x > 0 \\ \alpha e^x - \alpha & x \le 0 \end{cases}$$

Trong đó λ và α là hai hằng số được xác định trước. $\lambda \approx 1.0507$ và $\alpha \approx 1.67326$.

Dễ thấy đây là một biến thể của hàm ELU. Giá trị của λ và α được chọn để đảm bảo giá trị trung bình và phương sai của input được bảo toàn giữa hai lớp liên tiếp của mạng nơron.



Hình 5. *Hàm SeLU và đạo hàm*

3. MÔ HÌNH

Trong bài báo này, chúng tôi sử dụng hai mô hình, một đơn giản và một phức tạp, tương ứng với hai tập dữ liệu nhỏ và lớn.

Đối với tập dữ liệu nhỏ, chúng tôi sử dụng một mô hình với hai lớp tính chặp như sau:

Bảng 1. *Kiến trúc mô hình tích chập*

Layer (type)	Output Shape	Param #
Conv2d	[-1, 10, 24, 24]	260
MaxPool2d	[-1, 10, 12, 12]	0
Conv2d	[-1, 20, 8, 8]	5,020
Dropout2d	[-1, 20, 8, 8]	0
MaxPool2d	[-1, 20, 4, 4]	0
Linear	[-1, 50]	16,050
Linear	[-1, 10]	510

Tổng số tham số cần huấn luyện là 21,840. Về cơ bản, mô hình tương tự với LeNet-5. Ở đây, chúng tôi thêm vào lớp dropout để giảm hiện tượng overfitting.

Đối với tập dữ liệu lớn hơn, chúng tôi sử dụng mô hình VGG (Visual Geometry Group)[10] cấu hình A (VGG-11) với 8 lớp tích chặp. Số tham số cần huấn luyện là 9,231,144.

Bảng 2. *Kiến trúc mô hình VGG-11*

Layer (type)	Output Shape	Param #
Conv2d	[-1, 64, 32, 32]	1,792
BatchNorm2d	[-1, 64, 32, 32]	128
MaxPool2d	[-1, 64, 16, 16]	0
Conv2d	[-1, 128, 16, 16]	73,856
BatchNorm2d	[-1, 128, 16, 16]	256
MaxPool2d	[-1, 128, 8, 8]	0
Conv2d	[-1, 256, 8, 8]	295,168
BatchNorm2d	[-1, 256, 8, 8]	512
Conv2d	[-1, 256, 8, 8]	590,080
BatchNorm2d	[-1, 256, 8, 8]	512
MaxPool2d	[-1, 256, 4, 4]	0
Conv2d	[-1, 512, 4, 4]	1,180,160
BatchNorm2d	[-1, 512, 4, 4]	1,024
Conv2d	[-1, 512, 4, 4]	2,359,808
BatchNorm2d	[-1, 512, 4, 4]	1,024
MaxPool2d	[-1, 512, 2, 2]	0
Conv2d	[-1, 512, 2, 2]	2,359,808
BatchNorm2d	[-1, 512, 2, 2]	1,024
Conv2d	[-1, 512, 2, 2]	2,359,808
BatchNorm2d	[-1, 512, 2, 2]	1,024
MaxPool2d	[-1, 512, 1, 1]	0
AvgPool2d	[-1, 512, 1, 1]	0
Linear	[-1, 10]	5,130

Khi tiến hành khảo sát mô hình, chúng tôi lần lượt thay thế tất cả các hàm kích hoạt sau các lớp tích chặp lẫn lớp kết nối đầy đủ (fully-connected) bằng các hàm đã được đề cập ở mục 2.

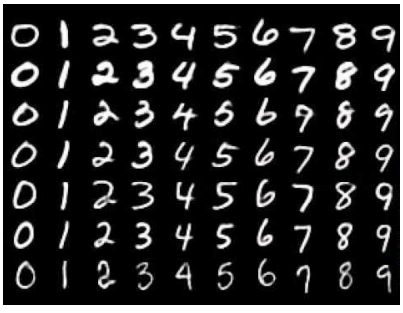
Trong phần tiếp theo, chúng tôi khảo sát sự ảnh hưởng của các hàm kích hoạt trên đến CNN. Các chi tiết của thực nghiệm như tập dữ liệu, thang đo (tiêu chí đánh giá) cũng sẽ được giải thích cụ thể.

4. THỰC NGHIỆM VÀ KẾT QUẢ

4.1. Tập dữ liệu

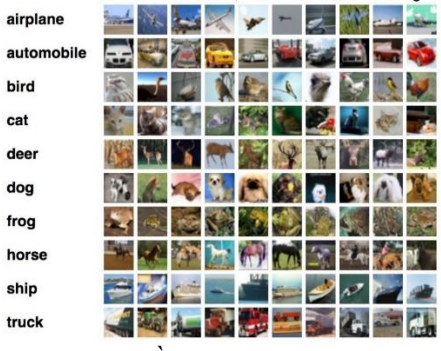
Để đánh giá tác động của các hàm kích hoạt kể trên lên CNN trong bài toán phân lớp hình ảnh, chúng tôi sử dụng hai tập dữ liệu hình ảnh phổ biến là CIFAR-10 và MNIST.

Tập dữ liệu MNIST (Modified National Institute of Standards and Technology)[6] là tập các chữ số viết tay từ 0-9 được số hóa. MNIST bao gồm 60,000 hình ảnh để huấn luyện và 10,000 hình ảnh để kiểm định. Mỗi phần tử của tập là một kí tự số theo hệ màu xám đơn sắc (grayscale) nằm chính giữa một khung hình có kích thước đã được chuẩn hóa (28x28). Nhãn của mỗi phần tử là kí tự số tương ứng trong hình ảnh của phần tử đó.



Hình 6. *Tập dữ liệu ký tự số MNIST*

Tập dữ liệu CIFAR-10 (Canadian Institute For Advanced Research)[7] bao gồm 60,000 phần tử thuộc 10 lớp khác nhau (mỗi lớp có 6,000 phần tử), trong đó 50,000 phần tử để huấn luyện và 10,000 phần tử để kiểm định. Mỗi phần tử là một hình ảnh màu đa sắc (RGB) có kích thước đã được chuẩn hóa (32x32). Nhãn của 10 lớp phần tử lần lượt là: máy bay, xe hơi, chim chóc, mèo, hươu nai, chó, ếch, ngựa, tàu thủy và xe tải.



Hình 7. *Các lớp phần tử của tập dữ liệu CIFAR-10*

4.2. Thang đo

DAWNBench[8] là một benchmark nhằm đánh giá hiệu năng của mạng nơron sâu. Theo [8], các bài benchmark khác thường chỉ tập trung lấy thời gian huấn luyện một minibatch dữ liệu làm tiêu chi chính mà bỏ qua độ chính xác của mô hình. Các tiêu chí của DAWNBench đảm bảo việc đánh giá một mô hình phải cân nhắc cả thời gian huấn luyện lẫn độ chính xác của mô hình đó.

Để đánh giá ảnh hưởng của các hàm kích hoạt lên hiệu năng, chúng tôi dựa theo hai trong bốn tiêu chí được đề cập trong DAWNBench, bao gồm thời gian huấn luyện mô hình để đạt được một ngưỡng chính xác nhất định và thời gian suy diễn (inference time) hay trong trường hợp này là thời gian phân lớp một mẫu từ mô hình đó sau khi huấn luyện.

Chúng tôi bỏ qua hai tiêu chí còn lại (chi phí thuê phần cứng để huấn luyện và suy diễn) do thực nghiệm được tiến hành trên phần cứng có sẵn.

Ngoài ra, để có cái nhìn tổng quát hơn, chúng tôi cũng đưa thêm một số tiêu chí như tổng thời gian huấn luyện và tần suất lỗi sau khi huấn luyện xong.

4.3. Cấu hình phần cứng và tinh chỉnh mô hình

Thực nghiệm được tiến hành trên máy trạm với cấu hình như sau: Ryzen 5 1600 3.8Ghz (6-core), 16GB RAM , NVIDIA GTX 1060 (6GB VRAM). Do giới hạn về phần cứng cùng độ phức tạp của mô hình, chúng tôi chỉ thực hiện đánh giá việc huấn luyện sau 10 chu kỳ (epoch) đối với tập MNIST và 50 chu kỳ với tập CIFAR-10.

Trong [8], tác giả có đề xuất việc sử dụng thuật toán tối ưu SGD thay cho Adam[9] vì SGD với momentum cho độ chính xác cao hơn nếu số chu kỳ huấn luyện lớn (>100 chu kỳ). Tuy nhiên, điều này cũng đồng nghĩa với việc khi thực nghiệm được tiến hành với số chu kỳ nhỏ, Adam sẽ đem lại mức độ hội tụ ban đầu cao hơn đối với cả tập huấn luyện và kiểm thử. Do đó, chúng tôi quyết định lựa chọn thuật toán tối ưu Adam thay cho SGD.

Ngoài ra, tác giả [8] cũng đưa ra hai kết luận sau về batch size. Thứ nhất, batch size quá lớn hay quá nhỏ đều ảnh hưởng đến độ hội tụ của thuật toán. Thứ hai, batch size lớn đem lại thông lượng (throughput: số lượng hình ảnh xử lý / đơn vị thời gian) tối ưu, giúp giảm thời gian huấn luyện. Cụ thể, trong [8], với batch size là 32, mô hình cần ít chu kỳ nhất để đạt độ chính xác tối đa; trong khi batch size là 256 cho độ chính xác tương đương (chênh lệch <0.5%) nhưng thời gian huấn luyện mô hình giảm đến 1.9 lần. Do batch size lớn cũng yêu cầu cao hơn về mặt phần cứng nên để cân bằng các yếu tố này, chúng tôi chọn batch size là 128.

4.4. Đánh giá kết quả thực nghiệm

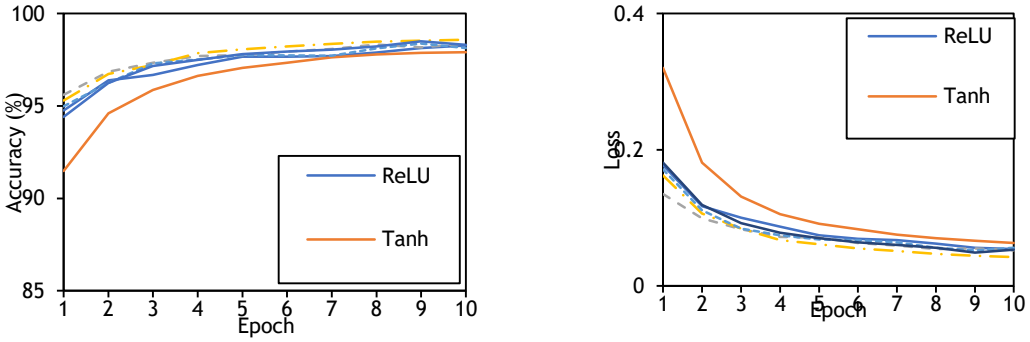
Căn cứ vào các kết quả thể hiện trong Hình 8, 9 và Bảng 3, 4; chúng tôi rút ra một số nhận xét sau:

- Việc huấn luyện với PreLU và Leaky ReLU cho tần suất lỗi thấp (tương ứng với độ chính xác cao) khi kiểm thử, tuy nhiên thời gian huấn luyện của PreLU tương đối cao.

- Hàm Tanh đem lại tần suất lỗi lớn, thời gian huấn luyện và suy diễn thường cao hơn so với các hàm khác. Chúng tôi kiến nghị tránh sử dụng hàm này khi xử lý bài toán phân lớp với CNN.

- Hàm ReLU, SELU và ELU cho tần suất lỗi tương đối thấp với thời gian huấn luyện vừa phải.

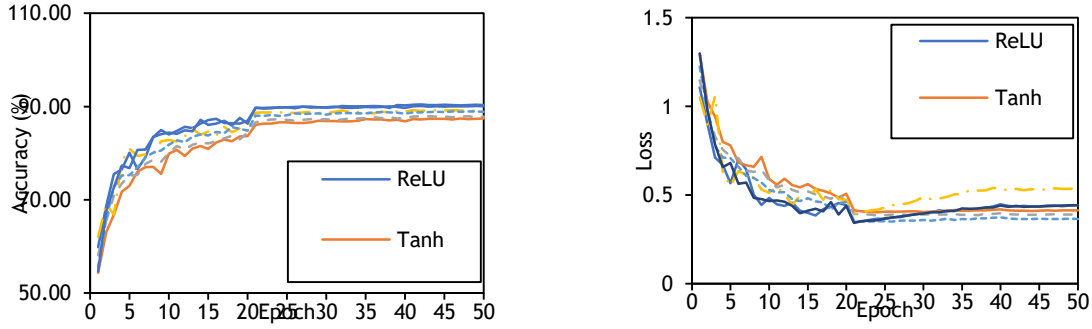
- Hàm Leaky ReLU luôn cho thời gian huấn luyện để đạt ngưỡng chính xác xác định trước ngắn nhất, theo sau là ReLU. Lưu ý ở đây là tuy trong Bảng 3 (MNIST) hàm PReLU cho độ hội tụ tốt nhất, điều này lại không lặp lại trong Bảng 4 (CIFAR-10). Do tính thiếu ổn định này, chúng tôi cho rằng PReLU cần được kiểm chứng thêm bằng một số thực nghiệm khác.



Hình 8. *Độ chính xác và giá trị hàm lỗi của CNN huấn luyện trên tập MNIST*

Bảng 3. *Kết quả thử nghiệm với tập MNIST (các kết quả tốt nhất của mỗi cột được highlight)*

Hàm	Tổng thời gian huấn luyện	Thời gian phân lớp một mẫu (ms)	Chu kỳ kiểm thử đầu tiên đạt độ chính xác 98%	Thời gian huấn luyện để đạt độ chính xác 98%	Tần suất lỗi
ReLU	0:01:48	0.14975	9	0:01:38	1.73%
Tanh	0:01:50	0.15276	N/A	N/A	2.09%
SELU	0:01:48	0.14916	7	0:01:16	1.62%
PReLU	0:01:51	0.14734	5	0:00:56	1.41%
ELU	0:01:48	0.14927	8	0:01:26	1.63%
Leaky ReLU	0:01:48	0.14856	7	0:01:16	1.50%



Hình 9. *Độ chính xác và giá trị hàm lỗi của mô hình VGG-11 huấn luyện trên tập CIFAR-10*

Bảng 4. *Kết quả thử nghiệm với tập CIFAR-10 (các kết quả tốt nhất của mỗi cột được highlight)*

Hàm	Tổng thời gian huấn luyện	Thời gian phân lớp một mẫu (ms)	Chu kỳ kiểm thử đầu tiên đạt độ chính xác 87%	Thời gian huấn luyện để đạt độ chính xác 87%	Tần suất lỗi
ReLU	0:47:38	0.324384	15	0:14:28	9.91%
Tanh	0:48:30	0.3282	35	0:33:47	12.58%
SELU	0:48:35	0.325202	23	0:22:16	12.07%
PReLU	0:49:25	0.327518	21	0:20:42	10.77%
ELU	0:47:53	0.323012	21	0:20:11	11.07%
Leaky ReLU	0:48:16	0.32579	14	0:13:39	9.57%

5. KẾT LUẬN

Qua bài báo này, chúng tôi đã bước đầu đánh giá được tác động của hàm kích hoạt đến quá trình huấn luyện cũng như suy diễn của mạng nơron. Dựa vào kết quả thực nghiệm, chúng tôi rút ra một số kết luận về việc nên hay không nên sử dụng hàm kích hoạt nào tùy vào trường hợp người dùng muốn ưu tiên giảm thời gian huấn luyện, tăng độ chính xác của mô hình hay cân bằng giữa cả hai. Nhìn chung, hàm ReLU và các biến thể của nó đem lại hiệu năng tốt hơn hàm Tanh trong hầu hết mọi trường hợp.

Ngoài ra, chúng tôi cũng trình bày một số điều chỉnh về siêu tham số và tiêu chí được đề cập trong DAWNBench khi tiến hành khảo sát mô hình trên máy tính cục bộ, bao gồm việc lược qua các tiêu chí đánh giá về chi phí huấn luyện và chi phí suy diễn. Việc triển khai các mô hình này và tiến hành thực nghiệm trên các nền tảng học sâu điện toán đám mây sẽ giúp việc đánh giá được hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

[1] Nair, Vinod, and Geoffrey E. Hinton (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*.

[2] Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." *Proc. icml*. Vol. 30. No. 1. 2013.

[3] He, Kaiming, et al. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*.

[4] Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[5] Klambauer, Günter, et al. (2017). Self-normalizing neural networks. *Advances in Neural Information Processing Systems*.

[6] LeCun, Yann, Corinna Cortes, and C. J. Burges (2010). MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2.

[7] Krizhevsky, Alex, and Geoffrey Hinton (2009). *Learning multiple layers of features from tiny images*. Vol. 1. No. 4. Technical report, University of Toronto.

[8] Coleman, Cody, et al. (2017). DAWNBench: An End-to-End Deep Learning Benchmark and Competition." *Training* 100.101 (2017): 102.
[9] Kingma, Diederik P., and Jimmy Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
[10] Simonyan, Karen, and Andrew Zisserman (2014). Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*.

Title: IMPACT OF ACTIVATION FUNCTIONS IN CONVOLUTION NEURAL NETWORKS

Abstract: Convolution neural networks (CNNs) are becoming more and more widely used in image processing in general and image classification in particular. To improve a CNN's performance, hyperparameter tuning is required. In this paper, we discuss the importance of selecting an appropriate activation function for training CNNs. We compare some basic functions against those proposed in recent studies. To evaluate their impact on CNNs' performance, we conduct various experiments with two CNN models, one simple and another complex in two popular image datasets MNIST and CIFAR-10, respectively. Experimental procedure and evaluation metrics are based on those proposed in DAWNBench with minor modifications. Our results show that ReLU and its variants offer high accuracy earlier than other types, despite having negligible advantage with regards to total training time.

Keywords: CNN, activation function.