

TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC QUỐC GIA TP.HCM
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG



BÀI TẬP LỚN HỌC KÌ 202

Giới thiệu và thực nghiệm so sánh một số hàm kích hoạt sử dụng trong các mạng học sâu

GVHD: TS. Phạm Việt Cường (pvcuong@hcmut.edu.vn)

Lớp: L01, Nhóm: 17

Nhóm sinh viên thực hiện: Nguyễn Thành Trung - 1814515

Hoàng Đình Toản - 1814379

Dào Minh Triết - 1814426

Lời cảm ơn

Chúng em - nhóm thực hiện bài tập lớn này muốn bày tỏ lòng biết ơn chân thành tới giáo viên hướng dẫn của nhóm trong môn học *Trí tuệ nhân tạo trong điều khiển*, là thầy **TS. Phạm Việt Cường** - “giảng viên bộ môn Điều Khiển Tự Động khoa Điện–Điện Tử” đã trực tiếp và gián tiếp chỉ dạy những kiến thức cần thiết không chỉ về môn học, mà là còn những kinh nghiệm trong phương pháp học tập và nghiên cứu. Tuy thời gian được trực tiếp làm việc cùng thầy trên lớp cho môn này không được nhiều do ảnh hưởng từ dịch bệnh COVID-19. Nhưng đã từng có cơ hội được thầy dẫn dắt ở những môn học khác, nhóm chúng em ai cũng dễ dàng nhận thấy được sự tâm huyết, sự nhiệt tình dành cho sinh viên không chỉ riêng gì khoa Điện mà còn cho các khoa khác về vấn đề học tập và làm việc. Dù rằng có thể thầy nghĩ thầy không hiền lành, hài hước hay cho điểm dễ như các giáo/giảng viên khác, nhưng nhóm chúng em đều đồng ý rằng cách làm việc của thầy hiện giờ rất phù hợp cho những sinh viên nào muốn học và quyết tâm học. Điều này thật sự là hết sức cần thiết cho sinh viên thời đại mới. Dẫu cho kết quả thống kê thầy nhận được không tương xứng với những gì thầy đóng góp, nhưng những điều đó không có nghĩa là cách thức giáo dục của thầy là không đúng đắn. Thật khó để tưởng tượng rằng nhóm 17 sẽ tiếp cận và xử lý đề tài bài tập lớn này như thế nào nếu không có những vốn liếng quý báu đó từ thầy. Mong thầy sẽ tiếp tục giữ được ngọn lửa này cho các lứa sinh viên kế cận, để duy trì truyền thống học tập đáng tự hào của trường Đại học Bách Khoa - Đại học Quốc gia Thành phố Hồ Chí Minh.

Chúng em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, ngày 04 tháng 05 năm 2021.

Nhóm Sinh Viên Thực Hiện

Nguyễn Thành Trung

Hoàng Đình Toản

Dào Minh Triết

Tóm tắt nội dung bài tập lớn

Hàm kích hoạt có một vai trò thiết yếu trong những tầng của một mạng học sâu, ảnh hưởng đến việc huấn luyện cũng như hiệu quả của mô hình. Bài tập lớn này thực hiện áp dụng những hàm kích hoạt chính được giới thiệu và thực nghiệm là: Tanh, ReLU, Leaky ReLU, ELU, SELU, GELU và Swish trên nhiều kiến trúc mạng khác nhau từ mạng nơ-ron đa tầng cổ điển đến những kiến trúc tích chập hiện đại gồm AlexNet, SimpleNet và VGG16. Các tập dữ liệu được sử dụng để đánh giá gồm tập MNIST, tập CIFAR-10 và tập CIFAR-100. Trong đó MNIST được thử nghiệm với mạng nơ-ron sâu và cả mạng tích chập đơn giản. Phần về CIFAR-10 và CIFAR-100 được huấn luyện bởi các kiến trúc tích chập hiện đại như đã đề cập. Kết quả thu được cho thấy hàm ELU duy trì được kết quả tốt đối với các mạng nơ-ron đa tầng khi ta tăng số tầng ẩn lên, điều này là không thể với hàm ReLU hay Leaky ReLU mặc dù với những mạng không sâu thì cho kết quả rất xuất sắc. Đối với những kiến trúc tích chập hiện đại, Leaky ReLU vượt trội với các hàm còn lại, kể đến là GELU. Ngoài ra, những kết quả còn cho thấy hàm Tanh là một hàm khá nhanh tìm thấy những điểm cực tiểu địa phương và mắc kẹt ở đó, điều này khiến cho hàm Tanh ở một số kiến trúc không thể học được. Một điều nữa là kỹ thuật alpha dropout không thực sự là một kỹ thuật tốt để áp dụng cho các hàm kích hoạt hiện giờ. Tất cả các quá trình thực nghiệm được tiến hành nhờ vào thư viện Keras trên nền môi trường Python của Google Colab sử dụng Tesla K80 GPU. Toàn bộ mã nguồn, những kết quả thực nghiệm được lưu trữ tại: <https://github.com/dee-ex/EE3063-SEM202-PROJECT>

Mục lục

1 Giới thiệu	1
1.1 Tổng quan	1
1.2 Lý do nghiên cứu	2
2 Giới thiệu các hàm kích hoạt được sử dụng	3
2.1 Những yêu cầu tối thiểu của hàm kích hoạt trong mạng học sâu	3
2.2 Hàm Sigmoid và hàm Tanh	4
2.3 Hàm ReLU (Rectified Linear Unit)	5
2.4 Hàm Leaky ReLU	7
2.5 Hàm ELU (Exponential Linear Unit)	8
2.6 Hàm SELU (Scaled Exponential Linear Unit)	9
2.7 Hàm GELU (Gaussian Error Linear Unit)	10
2.8 Hàm Swish	12
2.9 Thông tin về các hàm kích hoạt được thực nghiệm	13
3 Thực nghiệm so sánh trên tập dữ liệu MNIST	14
3.1 Thực nghiệm thông qua mạng nơ-ron sâu	14
3.1.1 Dữ liệu không nhiễu và khởi tạo trọng số theo chuẩn hoá Lecun	15
3.1.2 Dữ liệu nhiễu Gauss và khởi tạo trọng số theo chuẩn hoá Lecun	22
3.1.3 Dữ liệu nhiễu và không sử dụng kỹ thuật khởi tạo trọng số	29
3.1.4 Tiêu kết thực nghiệm mạng nơ-ron sâu trên tập dữ liệu MNIST	35
3.2 Thực nghiệm thông qua mạng tích chập sâu	35
4 Thực nghiệm so sánh trên tập dữ liệu CIFAR	39
4.1 Thực nghiệm các kiến trúc tích chập trên tập dữ liệu CIFAR-10	39
4.2 Thực nghiệm các kiến trúc tích chập trên tập dữ liệu CIFAR-100	42
5 Tổng kết	46

Danh sách hình

2.1.1	Phân lớp các dữ liệu sử dụng những miền phi tuyến tính [18].	4
2.2.1	Dồ thị hàm số của hàm Sigmoid và đạo hàm của nó.	5
2.2.2	Dồ thị hàm số của hàm Tanh và đạo hàm của nó.	5
2.3.1	Dồ thị hàm số của hàm ReLU và đạo hàm của nó.	6
2.4.1	Dồ thị hàm số của hàm Leaky ReLU ($\alpha = 0.1$) và đạo hàm của nó.	7
2.5.1	Dồ thị hàm số của hàm ELU ($\alpha = 0.3$) và đạo hàm của nó.	9
2.6.1	Dồ thị hàm số của hàm SELU và đạo hàm của nó.	10
2.7.1	Dồ thị hàm số của hàm GELU và đạo hàm của nó.	11
2.8.1	Dồ thị hàm số của hàm Swish ($\beta = 1$) và đạo hàm của nó.	12
3.1.1	Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 2, 3, 4).	18
3.1.2	Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 5, 6, 7).	19
3.1.3	Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 8, 9, 10).	20
3.1.4	Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định so với số tầng ẩn [(e), (f)].	21
3.1.5	Ví dụ về ảnh trước và sau khi thêm nhiễu Gauss từ tập dữ liệu MNIST.	22
3.1.6	Giá trị chính xác và mất mát (nhiễu Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 2, 3, 4).	25
3.1.7	Giá trị chính xác và mất mát (nhiễu Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 5, 6, 7).	26
3.1.8	Giá trị chính xác và mất mát (nhiễu Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 8, 9, 10).	27
3.1.9	Giá trị chính xác và mất mát (nhiễu Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (nhiễu Gauss + chuẩn hoá Lecun) trên tập kiểm định so với số tầng ẩn [(e), (f)].	28
3.1.10	Giá trị chính xác và mất mát (nhiễu) trên tập kiểm định (số tầng ẩn: 2, 3, 4).	31
3.1.11	Giá trị chính xác và mất mát (nhiễu) trên tập kiểm định (số tầng ẩn: 5, 6, 7).	32
3.1.12	Giá trị chính xác và mất mát (nhiễu) trên tập kiểm định (số tầng ẩn: 8, 9, 10).	33
3.1.13	Giá trị chính xác và mất mát (nhiễu) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (nhiễu) trên tập kiểm định so với số tầng ẩn [(e), (f)].	34
3.2.1	Giá trị chính xác và mất mát trên tập kiểm định (số tầng tích chập: 1, 6, 12).	37
3.2.2	Giá trị chính xác và mất mát trên tập kiểm định (số tầng tích chập: 18, 24) [(a) - (d)]; Giá trị chính xác và mất mát trên tập kiểm định so với số tầng tích chập [(e), (f)].	38



4.1.1	Giá trị chính xác và mất mát trên tập kiểm định CIFAR-10 (AlexNet, SimpleNet, VGG16).	41
4.2.1	Giá trị chính xác và mất mát trên tập kiểm định CIFAR-100 (AlexNet, SimpleNet, VGG16).	44
4.2.2	Giá trị chính xác Top-{3, 5} trên tập kiểm định CIFAR-100 (AlexNet, SimpleNet, VGG16).	45

Danh sách bảng

3.1.1	Giá trị chính xác (%) của các hàm kích hoạt (chuẩn hoá Lecun) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.	16
3.1.2	Giá trị mất mát của các hàm kích hoạt (chuẩn hoá Lecun) tương ứng với các mô hình trên tập dữ liệu MNIST.	16
3.1.3	Số epoch cần thiết để 80% giá trị mất mát (chuẩn hoá Lecun) bằng giá trị mất mát cuối cùng.	17
3.1.4	Giá trị trung bình và phương sai của các hàm kích hoạt (chuẩn hoá Lecun) trong 11 mô hình thử nghiệm.	17
3.1.5	Giá trị chính xác (%) của các hàm kích hoạt (nhiều Gauss + chuẩn hoá Lecun) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.	22
3.1.6	Giá trị mất mát của các hàm kích hoạt (nhiều Gauss + chuẩn hoá Lecun) tương ứng với các mô hình trên tập dữ liệu MNIST.	23
3.1.7	Số epoch cần thiết để 80% giá trị mất mát (nhiều Gauss + chuẩn hoá Lecun) bằng giá trị mất mát cuối cùng.	23
3.1.8	Giá trị trung bình và phương sai của các hàm kích hoạt (nhiều Gauss + chuẩn hoá Lecun) trong 11 mô hình thử nghiệm.	24
3.1.9	Giá trị chính xác (%) của các hàm kích hoạt (nhiều) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.	29
3.1.10	Giá trị mất mát của các hàm kích hoạt (nhiều) tương ứng với các mô hình trên tập dữ liệu MNIST.	29
3.1.11	Số epoch cần thiết để 80% giá trị mất mát (nhiều) bằng giá trị mất mát cuối cùng.	29
3.1.12	Giá trị trung bình và phương sai của các hàm kích hoạt (nhiều) trong 11 mô hình thử nghiệm.	30
3.1.13	Giá trị trung bình và phương sai của các hàm kích hoạt trung bình sau 3 lần thử nghiệm với 11 mô hình khác nhau.	35
3.2.1	Giá trị chính xác (%) của các hàm kích hoạt tương ứng với các mô hình mạng tích chập với số lớp tích chập khác nhau trên tập dữ liệu MNIST.	36
3.2.2	Giá trị mất mát của các hàm kích hoạt tương ứng với các mô hình mạng tích chập với số lớp tích chập khác nhau trên tập dữ liệu MNIST.	36
4.1.1	Giá trị chính xác (%) và mất mát của các hàm kích hoạt tương ứng với các kiến trúc mạng tích chập hiện đại trên tập dữ liệu CIFAR-10.	40
4.1.2	Giá trị trung bình của các hàm kích hoạt trong 3 kiến trúc tích chập hiện đại trên tập dữ liệu CIFAR-10.	40
4.2.1	Giá trị mất mát và Top-{1, 3, 5} (%) trên tập kiểm định và kiểm tra CIFAR-100 (AlexNet, SimpleNet, VGG16).	42
4.2.2	Giá trị trung bình của các hàm kích hoạt trong 3 kiến trúc tích chập hiện đại trên tập dữ liệu CIFAR-100.	43

Chương 1

Giới thiệu

1.1 Tổng quan

Học sâu - một phần nhỏ của học máy, đã và đang len lỏi vào trong đời sống hiện đại của chúng ta ngày nay. Rất nhiều ứng dụng giúp cải thiện cuộc sống con người đã được ra đời nhờ vào những nghiên cứu về học sâu. Đơn cử như chúng ta sử dụng Google Dịch để chuyển đổi một đoạn văn bản từ một ngôn ngữ chúng ta mong muốn sang một ngôn ngữ khác. Google Maps giúp chúng ta tìm được đường đi, dẫn chúng ta đi những đường tối ưu. Google Mail gợi ý cho chúng ta những đoạn văn mà chúng ta nên ghi theo, phân loại thư rác. Nhiều công ty công nghệ như Facebook cũng sử dụng học sâu để đưa ra những gợi ý quảng cáo phù hợp theo đúng nhu cầu chúng ta quan tâm thông qua các hành vi được ghi lại. Ô-tô tự lái của Tesla đã khiến cho nhiều người kinh ngạc khi việc vừa ngủ vừa lái xe không còn là một điều quá đỗi đieber khùng như trong phim viễn tưởng nữa. Đó là ở những nước tiên tiến, học sâu ở Việt Nam ta cũng không hề thua kém khi rào cản kiến thức đã bị xoá nhòa trong thời đại thế giới phẳng bây giờ. Nhận diện khuôn mặt trong việc điểm danh, chấm công. Gợi ý sản phẩm mua hàng cho những khách hàng dựa vào thông tin khách hàng cung cấp. Những ứng dụng gần gũi thúc đẩy công nghiệp hoá - hiện đại hoá đang được áp dụng rất nhiều dựa học máy cũng như học sâu.

Sức mạnh của học sâu có thể nói chủ yếu là việc kết hợp nhiều tầng ẩn, mỗi tầng ẩn gồm nhiều đơn vị ẩn. Sau quá trình tối ưu sử dụng lan truyền ngược, ta có được một hệ thống mạng nơ-ron sâu giúp ta xử lý được những bài toán khó khăn. Một trong những thành phần không thể thiếu trong những mạng học sâu này là các hàm kích hoạt phi tuyến. Với mỗi tầng ẩn thứ l , đầu ra của đơn tầng ẩn đó có thể được viết dưới dạng toán học là $\mathbf{a}^{(l)} = f^{(l)}(\mathbf{z}^{(l)}) = f^{(l)}(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})$. Thời khởi điểm những mạng sâu, hai hàm được sử dụng phổ biến là Sigmoid và Tanh. Một thời gian sau, một trong những bước đột phá về việc sử dụng hàm kích hoạt được Hinton và các cộng sự của mình giới thiệu đó là hàm ReLU (Rectified Linear Unit) - một hàm được sử dụng rộng rãi cho tới tận bây giờ sau 9 năm ra mắt. Hàm ReLU đã khắc phục một vài yếu điểm của những hàm Sigmoid hoặc Tanh, nhưng cũng không phải là không có nhược điểm. Một trong những nhược điểm được biết đến nhiều nhất đó là chết ReLU, khi ép những giá trị đầu vào âm thành giá trị 0. Nhiều năm sau, rất nhiều những hàm kích hoạt khác đã được đề xuất để có thể cải thiện cũng như chỉ ra những hạn chế của hàm ReLU, có thể kể đến là Leaky ReLU, ELU, SELU, GELU. Hàm Swish, được định nghĩa $f(x) = x \text{Sigmoid}(\beta x)$, được đưa ra và đã chứng minh được sự mạnh mẽ của mình so với những họ hàm LU.

Trong bài tập lớn này, những hàm kích hoạt được nêu tên sẽ được giới thiệu, phân tích một vài điểm mạnh và yếu. Sau cùng, ta sẽ áp dụng những hàm trên trong một số kiến trúc cũng như trên những tập dữ liệu khác nhau, để có một cái nhìn khái quát về hiệu xuất của các hàm

khi được dùng để kích hoạt các giá trị ở những tầng ẩn của một mạng học sâu.

1.2 Lý do nghiên cứu

Việc lựa chọn một hàm kích hoạt phù hợp với từng kiến trúc, cơ sở phần cứng, không chỉ giúp cải thiện tốc độ huấn luyện cho mô hình, mà còn là hỗ trợ các thuật toán tối ưu tìm được cực tiểu tốt nhất. Lý do là vì các hàm kích hoạt khác nhau sinh ra các hàm măt măt (hàm mục tiêu) khác nhau dẫn đến việc tối ưu nó cũng khác nhau dù dùng chung một thuần toán tối ưu với các thông số giống nhau. Vì thế, khảo sát và đánh giá những hàm kích hoạt khác nhau với những kiến trúc khác nhau là cần thiết. Từ đó có thể đưa ra những gạch đầu dòng tiêu chí quan trọng cho những người kỹ sư, nhà nghiên cứu chọn lựa cho mình những hàm kích hoạt phù hợp trong kiến trúc mô hình của họ.

Chương 2

Giới thiệu các hàm kích hoạt được sử dụng

2.1 Những yêu cầu tối thiểu của hàm kích hoạt trong mạng học sâu

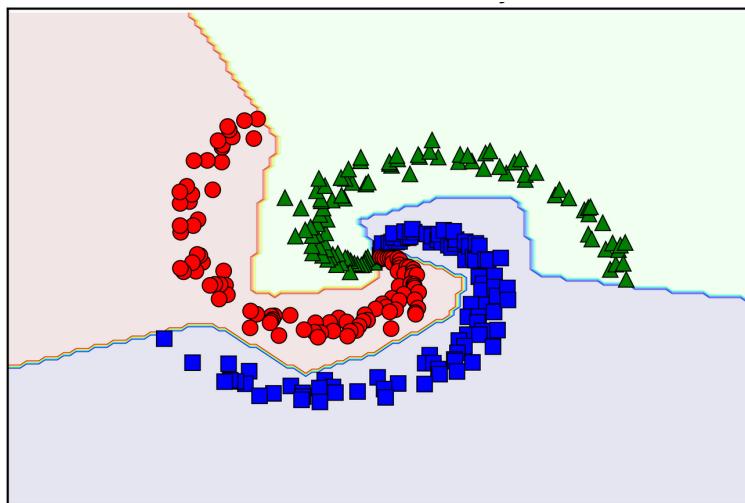
Những hàm kích hoạt được giới thiệu đều là những hàm kích hoạt phi tuyến. Lý do hàm kích hoạt không được phép là một hàm tuyến tính có thể nhìn ở nhiều góc độ. Nếu ta sử dụng các hàm kích hoạt tại một tầng ẩn nào đó là một hàm tuyến tính, tầng này và tầng tiếp theo có thể rút gọn thành một tầng. Vì hợp của các hàm tuyến tính là một hàm tuyến tính. Giả sử ta có một mạng nơ-ron chỉ gồm 2 tầng ẩn đều sử dụng hàm kích hoạt là một hàm tuyến tính $f(\mathbf{x}) = \mathbf{x}$. Ta dễ dàng suy ra được như sau:

$$\begin{aligned}\mathbf{a}^{(1)} &= f(\mathbf{z}^{(1)}) = f(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{a}^{(2)} &= f(\mathbf{z}^{(2)}) = f(\mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)}) \\ &= f(\mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \\ &= f\left(\underbrace{\mathbf{W}^{(2)}\mathbf{W}^{(1)}}_{\mathbf{W}^*}\mathbf{x} + \underbrace{\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}}_{\mathbf{b}^*}\right) \\ &= f(\mathbf{W}^*\mathbf{x} + \mathbf{b}^*) = \mathbf{W}^*\mathbf{x} + \mathbf{b}^*\end{aligned}\tag{2.1}$$

Theo như 2.1 thì đầu ra của tầng ẩn thứ 2 vẫn chỉ là một tổ hợp tuyến tính của \mathbf{x} ban đầu. Thế thì khi một mạng nơ-ron sâu có tất cả các tầng ẩn đều sử dụng những hàm kích hoạt tuyến tính, ta có thể rút gọn những tầng ẩn này đi mà chỉ có tầng đầu vào và tầng đầu ra. Điều này làm cho mô hình không thể hiện được tính sâu, thứ làm nên sức mạnh của những mạng nơ-ron.

Ở một góc nhìn đơn giản, bản thân những hàm tuyến tính chỉ có thể làm việc được với những bài toán đơn giản. Nếu chỉ sử dụng những hàm kích hoạt tuyến tính thì ta cũng chỉ có thể đà ra một lời giải đơn giản. Mà rõ ràng, những bài toán được áp dụng trong mạng học sâu không chỉ có những bài toán đơn giản mà còn là những bài toán rất phức tạp. Những hàm tuyến tính không thể nào đáp ứng được nhu cầu bài toán đưa ra, chẳng hạn như hình 2.1.1, nếu chỉ dùng những hàm tuyến tính để phân lớp thì ta sẽ không bao giờ có một lời giải tốt. Do đó, hàm kích hoạt được yêu cầu phải là một hàm phi tuyến.

Nền móng của mạng nơ-ron sâu là mô hình perceptron [17], với hàm kích hoạt được sử dụng



Hình 2.1.1: Phân lớp các dữ liệu sử dụng những miền phi tuyến tính [18].

là hàm dấu.

$$\text{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \quad (2.2)$$

Dù rằng là hàm phi tuyến, nhưng thực tế lại không được sử dụng, vì đạo hàm tại hầu hết các điểm bằng 0 (trừ tại gốc toạ độ không có đạo hàm). Việc đạo hàm bằng 0 này khiến cho các thuật toán dựa trên gradient không hoạt động. Trong khi việc tối ưu các mạng học sâu sử dụng lan truyền ngược lại dựa trên gradient.

2.2 Hàm Sigmoid và hàm Tanh

Hàm Sigmoid, hay còn biết đến với cái tên là hàm logistic. Hàm được định nghĩa toán học như sau:

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

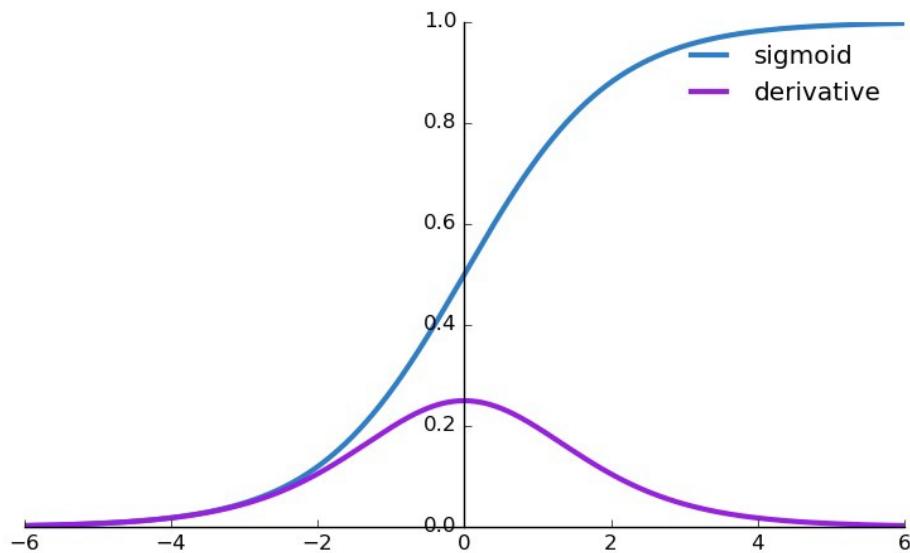
Nhìn vào 2.2.1 ta thấy rằng: nếu đầu vào lớn, hàm số sẽ cho đầu ra gần với 1; nếu đầu vào nhỏ (rất âm), hàm số sẽ cho đầu ra gần với 0. Trước đây, hàm kích hoạt này được sử dụng rất nhiều vì có đạo hàm rất đẹp (xem hình 2.2.1). Nhưng những năm gần đây, hàm số này ít khi được sử dụng để làm hàm kích hoạt cho các tầng ẩn. Thay vào đó, hàm Sigmoid được sử dụng ở tầng đầu ra khi đầu ra là các giá trị nhị phân hoặc biểu diễn xác suất.

Một hàm tương tự thường sử dụng và mang lại hiệu quả tốt hơn chính là hàm Tanh:

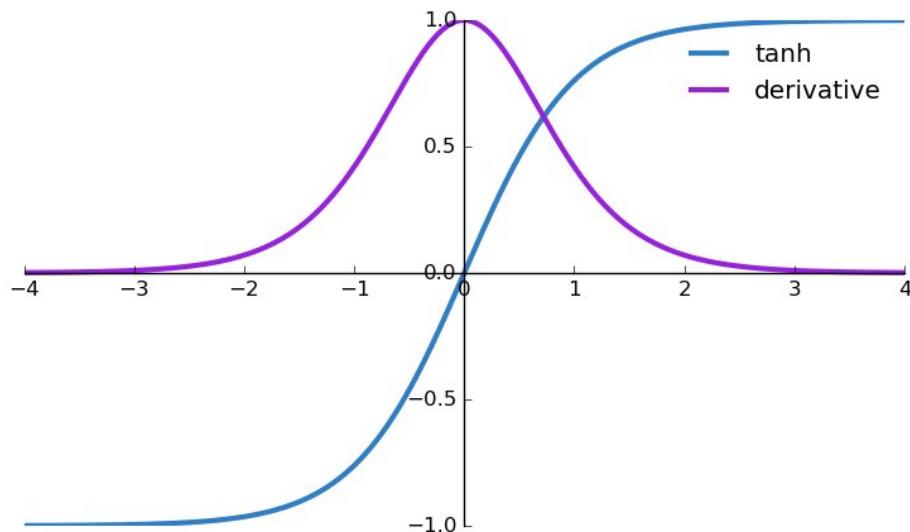
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.4)$$

Hàm số này có tính chất đầu ra chạy từ -1 đến 1, khiến cho nó có tính chất tâm 0 thay vì chỉ dương như hàm Sigmoid (xem hình 2.2.2). Tính chất tâm 0 này giúp cho mô hình có thể làm việc hiệu quả được với những tập dữ liệu chưa được chuẩn hóa tốt [3].

Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn, đạo hàm của cả Sigmoid và Tanh tại điểm giá trị đó rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với



Hình 2.2.1: Đồ thị hàm số của hàm Sigmoid và đạo hàm của nó.



Hình 2.2.2: Đồ thị hàm số của hàm Tanh và đạo hàm của nó.

đơn vị ẩn đang xét sẽ gần như không được thay đổi khi sử dụng công thức cập nhật theo thuật toán gradient khi tiến hành lan truyền ngược. Hiện tượng này được gọi là hiện tượng biến mất gradient.

2.3 Hàm ReLU (Rectified Linear Unit)

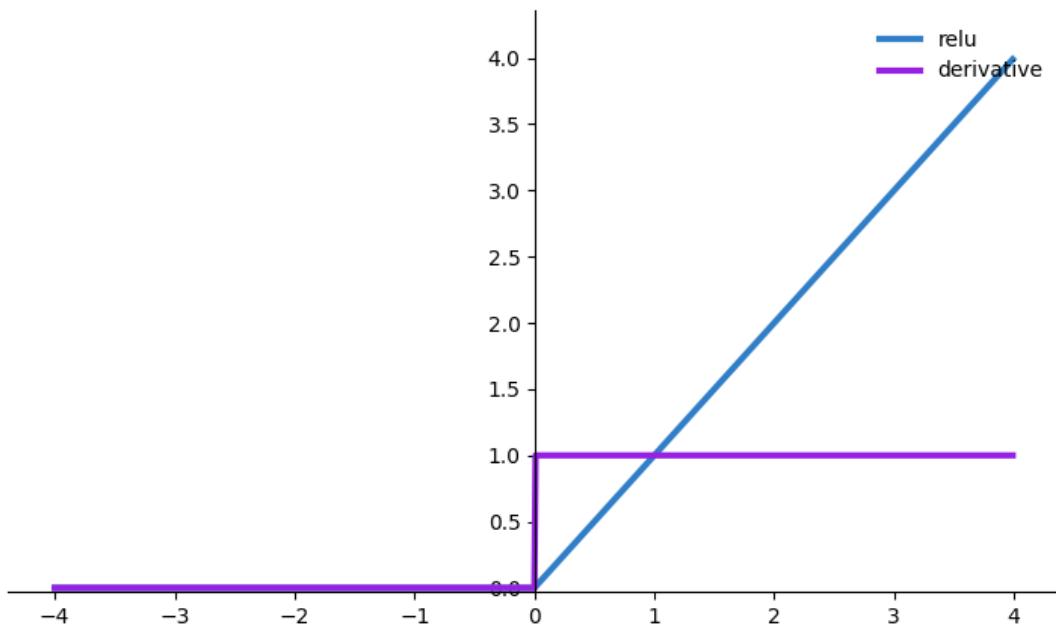
Đây là hàm [13] được sử dụng rộng rãi nhất trong các mô hình học sâu vì tính đơn giản của nó (xem hình 2.3.1). Tuy đơn giản, nhưng không hề đồng nghĩa với ít hiệu quả. Hàm kích hoạt

này có công thức toán học là:

$$\text{ReLU}(x) = \mathcal{R}(x) = \max(0, x) \quad (2.5)$$

Một cách nôm na, ta có thể miêu tả hàm ReLU như sau:

- Nếu đầu vào nhỏ hơn hoặc bằng 0, thì đầu ra là 0.
- Nếu đầu vào lớn hơn 0, thì đầu ra là chính nó, tức đầu ra đúng bằng đầu vào.



Hình 2.3.1: Đồ thị hàm số của hàm ReLU và đạo hàm của nó.

Ta có thể thấy, hàm ReLU thực sự rất đơn giản trong tính toán. Nhìn vào tưởng chừng không khác mấy so với một hàm tuyến tính thông thường. Hay thay, điều đơn giản này lại khắc phục được hiện tượng biến mất gradient mà hàm Sigmoid và Tanh mắc phải. Để có thể thấy rõ điều đó, ta sẽ xem xét đạo hàm của hàm ReLU. Hiển nhiên, ta không khó để ta tính được:

$$\frac{d\mathcal{R}}{dx} = \max(\text{sgn}(x)) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (2.6)$$

Cũng như chính hàm ReLU, đạo hàm của nó cũng hết sức đơn giản:

- Nếu đầu vào nhỏ hơn hoặc bằng 0, thì đầu ra là 0.
- Nếu đầu vào lớn hơn 0, thì đầu ra là 1.

Vậy nên sẽ không xuất hiện chuyện khi đầu vào quá lớn, đạo hàm tại giá trị đó sẽ rất gần 0, khiến cho việc cập nhật giá trị tham số mô hình diễn ra chậm. ReLU được chứng minh là đã giúp việc huấn luyện mạng nơ-ron đa tầng nhanh hơn rất nhiều so với lại hàm Tanh.

Nhưng hàm này lại gây ra một trở ngại khác, chính là chết ReLU. Mọi đầu vào nhỏ hơn hoặc bằng 0 đều sẽ được đưa về 0. Chuyện gì sẽ xảy ra khi ta có nhiều đầu vào nhỏ hơn 0? Hiển nhiên là kết quả tệ và thậm chí là tệ hơn so với biến mất gradient, khi giá trị tham số sẽ

không có cách nào thay đổi và mãi như vậy do đạo hàm của những giá trị đầu vào này đều là giá trị 0. Một số chứng minh thực nghiệm cho thấy, việc này có thể được khắc phục bằng cách tăng số lượng đơn vị ẩn [14]. Tuy nhiên, việc này cần phải cẩn thận khi Lu Lu [10] đã chứng minh lý thuyết được rằng một mạng học sâu với hàm kích hoạt ReLU sẽ chết khi độ sâu lên đến vô hạn.

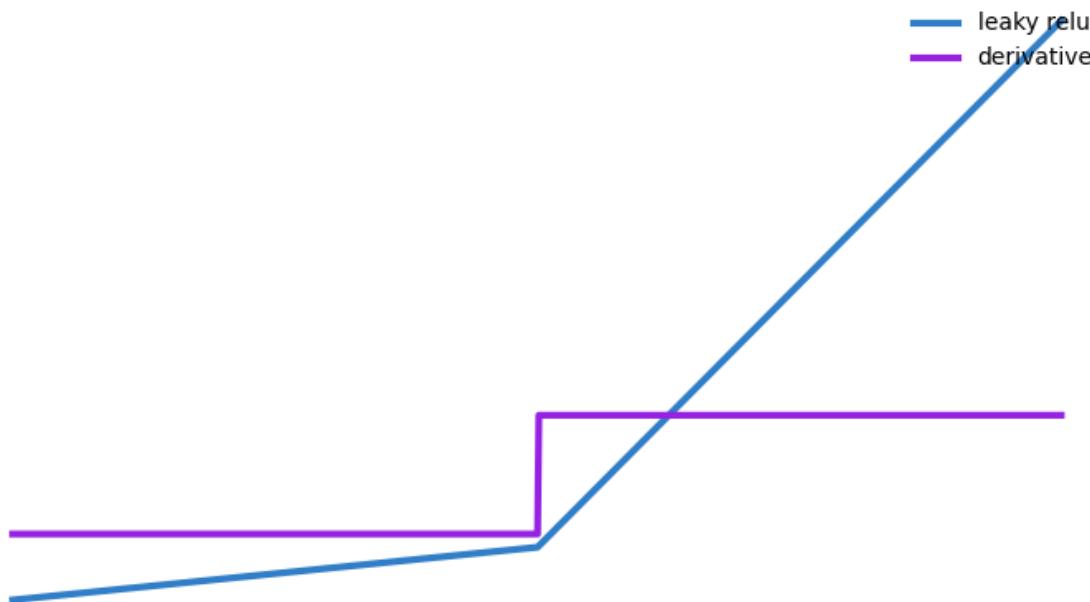
Ưu điểm dễ dàng nhận ra của RELU đó chính là việc tính toán đơn giản, giúp cho chi phí tính toán sẽ đơn giản hơn những hàm sử dụng hàm mũ. Đồng thời hàm này cũng khắc phục được hiện tượng biến mất gradient. Tuy thế, phần âm đưa hết về 0 của hàm lại gây nên hiện tượng chết ReLU.

Theo kinh nghiệm của những kỹ sư đi trước, khi xây dựng một mạng nơ-ron đa tầng, hàm kích hoạt ReLU nên được thử đầu tiên, vì nó nhanh cho kết quả và thường hiệu quả trong nhiều trường hợp.

2.4 Hàm Leaky ReLU

Đây là một biến thể của ReLU [11] mà có khả năng khắc phục hiện tượng chết ReLU. Công thức toán học miêu tả hàm này như sau:

$$\text{Leaky ReLU}(x) = \mathcal{R}_L(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}; \alpha \in (0, 1) \quad (2.7)$$



Hình 2.4.1: Đồ thị hàm số của hàm Leaky ReLU ($\alpha = 0.1$) và đạo hàm của nó.

Nửa khoảng giá trị $(0, +\infty)$, hai hàm ReLU và Leaky ReLU hoàn toàn giống nhau. Duy chỉ phần nửa đoạn $(-\infty, 0]$, hàm Leaky ReLU đã có một chút sự thay đổi thay vì chỉ là giá trị 0 như ReLU. Bây giờ, những giá trị âm sẽ có giá trị là αx thay vì giá trị 0. Đạo hàm của Leaky ReLU cũng vì thế mà sẽ là:

$$\frac{d\mathcal{R}_L}{dx} = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases} \quad (2.8)$$

Nhờ như vậy, những đầu vào nhỏ hơn 0 bây giờ sẽ không bị đưa hết về 0 mà sẽ là α . Điều này có nghĩa là, hiện tượng chết ReLU đã được xử lý.

Việc lựa chọn giá trị α cho thực sự hiệu quả cũng là một vấn đề không hề đơn giản. Nếu giá trị $\alpha = 0$, Leaky ReLU chính là ReLU. Còn nếu $\alpha = 1$, thì Leaky ReLU sẽ trở thành một hàm tuyến tính, nên không ai sử dụng Leaky ReLU với $\alpha = 1$. Thông thường, α sẽ được chọn là 0.01, tức 1 phần 100. Cũng có một số mô hình chọn giá trị α lớn hơn một chút, khoảng từ 0.1 tới 0.3. Đã có nhiều cải thiện được đề xuất cho Leaky ReLU và cũng đã có những kết quả khả quan trên một số thực nghiệm. Có thể liệt kê qua như việc học hệ số α này trong quá trình huấn luyện với hàm đề xuất là PReLU (Parametric Rectified Linear Unit). Hoặc là ngẫu nhiên chọn hệ số α dựa vào một biến ngẫu nhiên theo một phân phối đều - RReLU (Randomized Leaky Rectified Linear Unit).

Có thể nói, Leaky ReLU đã thừa hưởng được những đặc điểm tốt của ReLU, ngoài ra còn khắc phục được hiện tượng chết ReLU của phiên bản gốc. Nhưng một điều khá rắc rối là việc chọn hệ số α cần phải được thực hiện một cách hợp lý, hoặc phải dùng thuật toán học, nếu không dễ gây tác dụng không đáng có.

2.5 Hàm ELU (Exponential Linear Unit)

Cũng như Leaky ReLU, hàm ELU [2] được đưa ra để khắc phục điểm yếu của ReLU. Một hệ số α sẽ được chọn giống như cách chọn cho hàm Leaky ReLU, đôi khi sẽ là chọn bằng kinh nghiệm, và có lúc sẽ là học trong quá trình huấn luyện. Nhưng ELU không còn giữ được tính đơn giản tính toán như Leaky ReLU hay ReLU tuy hình dáng không thay đổi nhiều (xem hình 2.5.1). Công thức toán học của ELU có thể được biểu diễn như sau:

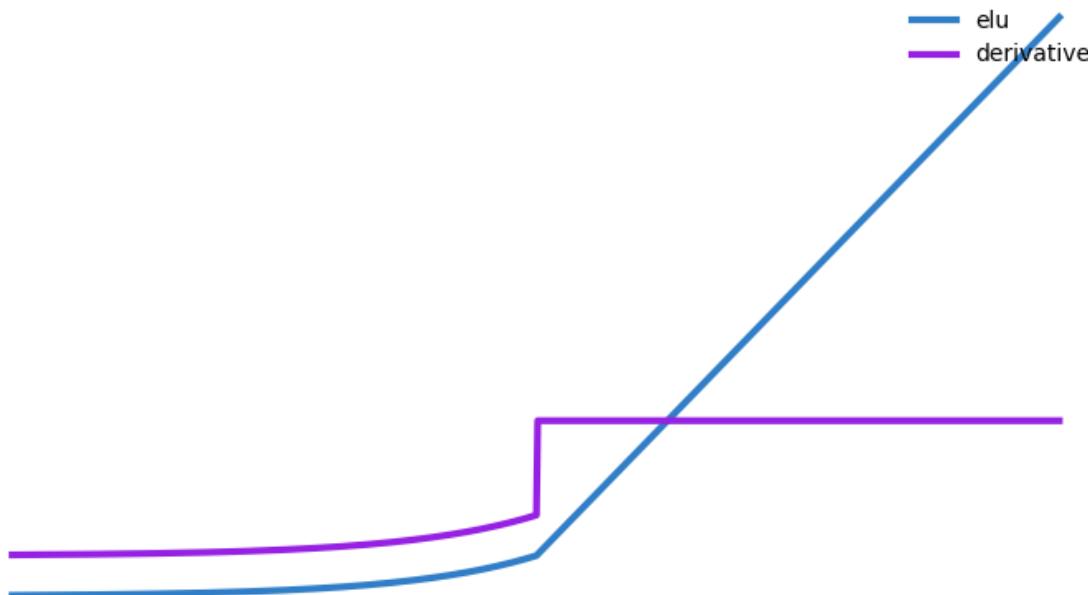
$$\text{ELU}(x) = \mathcal{E}(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}; \alpha \in (0, 1) \quad (2.9)$$

Nếu đầu vào lớn hơn 0, đầu ra bằng đúng đầu vào. Còn với những đầu vào bé hơn hoặc bằng 0, chúng ta sẽ có một giá trị chỉ là gần với 0. Khác so với ReLU, khi ReLU sẽ đưa hết tất cả về 0. Leaky ReLU thì sẽ đưa trị tuyệt đối của giá trị đó nhỏ lại theo hệ số α . Những thay đổi của ELU cũng không làm mất đi tính chất của những họ hàm LU, dùng để giải quyết biến mất gradient. Nếu như vậy, thì ELU có tránh được việc chết ReLU như Leaky ReLU làm được? Ta hãy cùng xem đạo hàm của ELU:

$$\frac{d\mathcal{E}}{dx} = \begin{cases} 1, & x > 0 \\ \alpha e^x, & x \leq 0 \end{cases} = \begin{cases} 1, & x > 0 \\ \alpha(e^x - 1) + \alpha, & x \leq 0 \end{cases} = \begin{cases} 1, & x > 0 \\ \mathcal{E}(x) + \alpha, & x \leq 0 \end{cases} \quad (2.10)$$

Lý do ta nên biểu diễn đạo hàm phía âm của hàm ELU dưới dạng $\mathcal{E}(x) + \alpha$ thay vì αe^x (2.10) là để tái sử dụng lại giá trị $\mathcal{E}(x)$ thay vì phải tính lại giá trị e^x .

Trở lại với việc rằng liệu ELU có tránh được việc chết ReLU hay không, ta cần chú ý vào phần phía âm, tức đại lượng $\mathcal{E}(x) + \alpha$. Rõ ràng rằng dù đầu vào của $\mathcal{E}(x)$ có âm như thế nào, thì giá trị nhận về cũng là một số âm gần với 0. Khi cộng với một α lớn vừa phải, giá trị này sẽ có xu hướng lớn hơn 0 một tí, vừa đủ để thoát khỏi được hiện tượng chết ReLU. Khác với Leaky ReLU, hàm ELU vẫn có thể có một vài đơn vị ẩn bị chết, tức không cập nhật được tham số nếu đầu vào rất âm. Vì khi đó $\mathcal{E}(x)$ không còn quá gần 0, khi cộng vào với α , sẽ có xu hướng rất sát với 0 như Sigmoid hoặc Tanh. Hoặc có thể hiểu là do việc $x \rightarrow -\infty$ sẽ suy ra được $e^x \rightarrow 0$.



Hình 2.5.1: Đồ thị hàm số của hàm ELU ($\alpha = 0.3$) và đạo hàm của nó.

Hàm ELU cũng có được những ưu điểm của Leaky ReLU, tuy vậy việc tính toán lúc này sẽ phức tạp hơn một chút vì có đại lượng mũ.

2.6 Hàm SELU (Scaled Exponential Linear Unit)

Hàm này [7] là một trong những hàm khá mới (xem hình 2.6.1). Việc cài đặt hàm này để sử dụng trong thực tiễn một cách hiệu quả cũng cần phải có nhiều công sức hơn so với các hàm đã giới thiệu. Theo hướng dẫn, các trọng số cũng như bias của mạng cần phải được khởi tạo với LeCun Normal [4]. Trong trường hợp mạng có sử dụng kỹ thuật dropout, thì cần phải sử dụng Alpha Dropout. Tác giả khi đề xuất hàm này, đã tính toán 2 giá trị quan trọng cho hàm này, chính là α và λ :

$$\alpha \approx 1.6732632423543772848170429916717 \quad (2.11)$$

$$\lambda \approx 1.0507009873554804934193349852946 \quad (2.12)$$

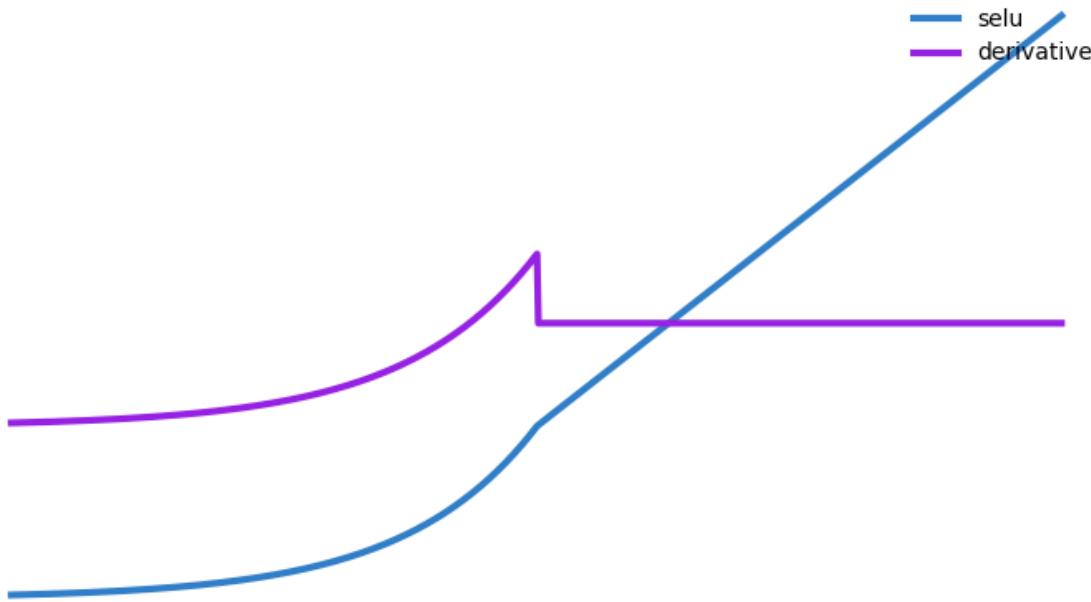
Không giống như Leaky ReLU hay ELU, đây là những giá trị đã được các tác giả tính toán trước và áp dụng cho mọi mô hình. Giống như cái tên của mình, đây là một hàm tỷ lệ với hàm ELU, chính xác là tỷ lệ với hệ số λ (2.12):

$$\text{SELU}(x) = \mathcal{E}_S(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \quad (2.13)$$

Đạo hàm của SELU cũng không khác ELU là bao:

$$\frac{d\mathcal{E}_S}{dx} = \lambda \frac{d\mathcal{E}}{dx} = \begin{cases} 1, & x > 0 \\ \mathcal{E}_S(x) + \alpha, & x \leq 0 \end{cases} \quad (2.14)$$

Bây giờ, nếu đầu vào lớn hơn 0 thì giá trị đầu ra cần phải tỷ lệ theo một hệ số λ thành là λx hay vì chính bằng x . Phần miền nhỏ hơn hoặc bằng 0, tương tự một hàm ELU với giá trị α (2.11) đã chọn nhưng giá trị sẽ được tỷ lệ với λ .



Hình 2.6.1: Đồ thị hàm số của hàm SELU và đạo hàm của nó.

Chọn một số α để cài đặt cho hàm ELU, và tỷ lệ với một hằng số λ , điều gì khiến hàm này trở nên đặc biệt? Lý thuyết mà nói, hàm SELU là hàm có khả năng tự chuẩn hoá đầu ra nếu được cài đặt đúng như những gì đã yêu cầu. Thay vì phải thêm một tầng chuẩn hoá đầu ra như một số các hàm khác, thì khi sử dụng SELU, bước chuẩn hoá này đã được hàm SELU tự động hoàn thành trong nội bộ. Điều này giúp giảm thời gian huấn luyện của mô hình sử dụng hàm SELU.

Về bản chất, SELU cũng vẫn là một dạng tỷ lệ của ELU, nên cách nó khắc phục biến mất gradient và chết ReLU không khác gì ELU. Nếu đúng như lý thuyết, thì SELU lợi thế hơn ELU ở điểm có khả năng tự chuẩn hoá, bớt cồng kềnh cho mạng sử dụng.

2.7 Hàm GELU (Gaussian Error Linear Unit)

Được sử dụng nhiều gần đây trong kiến trúc Transformers - Google's BERT và OpenAI's GPT-2. Tuy ra đời cũng không phải là quá mới, khi 2016 đã được trình làng [6], nhưng mãi ít năm gần đây mới được chú ý tới. Mô tả toán học của hàm này như sau:

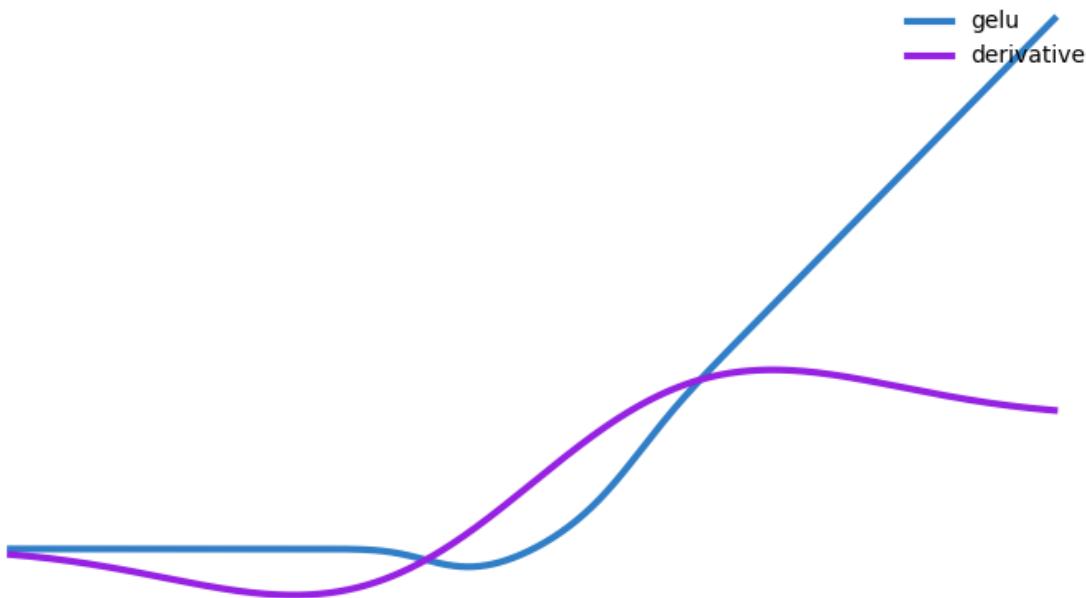
$$GELU(x) = \mathcal{G}(x) = 0.5x \left\{ 1 + \tanh \left[\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right] \right\} \quad (2.15)$$

Không có gì là giống với những hàm LU trước, khi bây giờ GELU là tổ hợp của nhiều hàm cộng lại: hàm Tanh, hàm đa thức, những hệ số vô tỉ. Nếu để ý kĩ, hàm Tanh sẽ cho giá trị gần với 1 khi x dương lớn, như vậy hàm GELU sẽ có giá trị đầu ra đúng gần bằng với đầu vào như là một hàm Linear Unit. Muốn có cái nhìn chính xác của điều trên, ta quan sát những giá trị giới hạn và phép tương đương sau:

$$\lim_{x \rightarrow +\infty} \sqrt{\frac{2}{\pi}} (x + 0.044715x^3) = +\infty \quad (2.16)$$

$$\lim_{x \rightarrow +\infty} 0.5(1 + \tanh(x)) = 0.5(1 + 1) = 1 \quad (2.17)$$

$$\mathcal{G}(x) \sim 0.5x(1 + 1) = x, x \rightarrow +\infty \quad (2.18)$$



Hình 2.7.1: Đồ thị hàm số của hàm GELU và đạo hàm của nó.

Trường hợp x âm lớn, hàm Tanh sẽ cho ta giá trị gần với -1 khiến cho kết quả của hàm GELU có giá trị gần với 0. Đại lượng âm vô cùng x sẽ được khử với giá trị 0 của $1 + \tanh$. Để hiểu rõ chỗ này, ta cần một số kiến thức giải tích cơ bản về chuỗi xấp xỉ tương đương Taylor-Maclaurin:

$$\frac{1}{1+x} = \sum_{n=0}^{\infty} (-1)^n x^n = 1 - x + x^2 - x^3 + \dots \quad (2.19)$$

Từ đó, ta có thể tương đương hàm Tanh như sau:

$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} = (e^{2x} - 1) \cdot \frac{1}{e^{2x} + 1} \\ &= (e^{2x} - 1) (1 - e^{2x} + \dots) \sim -1 + e^{2x}, x \rightarrow -\infty \end{aligned} \quad (2.20)$$

Dựa vào 2.20 vừa khai triển, ta suy ra được:

$$\tanh \left[\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right] \sim \tanh(kx^3) \sim -1 + e^{2kx^3}, x \rightarrow -\infty \quad (2.21)$$

Theo đó, ta sẽ có được kết quả giới hạn:

$$\begin{aligned} \lim_{x \rightarrow -\infty} \mathcal{G}(x) &= \lim_{x \rightarrow -\infty} 0.5x \left\{ 1 + \tanh \left[\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right] \right\} \\ &= \lim_{x \rightarrow -\infty} 0.5x \left\{ 1 - 1 + e^{2kx^3} \right\} \\ &= \lim_{x \rightarrow -\infty} 0.5x e^{2kx^3} \\ &= \lim_{x \rightarrow +\infty} -\frac{0.5x}{e^{2kx^3}}; e^x \gg x, x \rightarrow +\infty \\ &= 0 \end{aligned} \quad (2.22)$$

Không quá gian nan để tìm ra đạo hàm của GELU, bằng đạo hàm của hàm hợp ta có:

$$\begin{aligned} \frac{d\mathcal{G}}{dx} &= 0.5 \tanh(0.0356774x^3 + 0.797885x) \\ &\quad + \frac{0.0535161x^3 + 0.398942x}{\cosh^2(0.0356774x^3 + 0.797885x)} + 0.5 \end{aligned} \quad (2.23)$$

Trong đó, hàm Cosh được định nghĩa như sau:

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \quad (2.24)$$

Cosh là một vô cùng lớn khi $x \rightarrow \infty$, nên là:

$$\lim_{x \rightarrow \infty} \frac{x^3}{\cosh(x^3)} = 0 \quad (2.25)$$

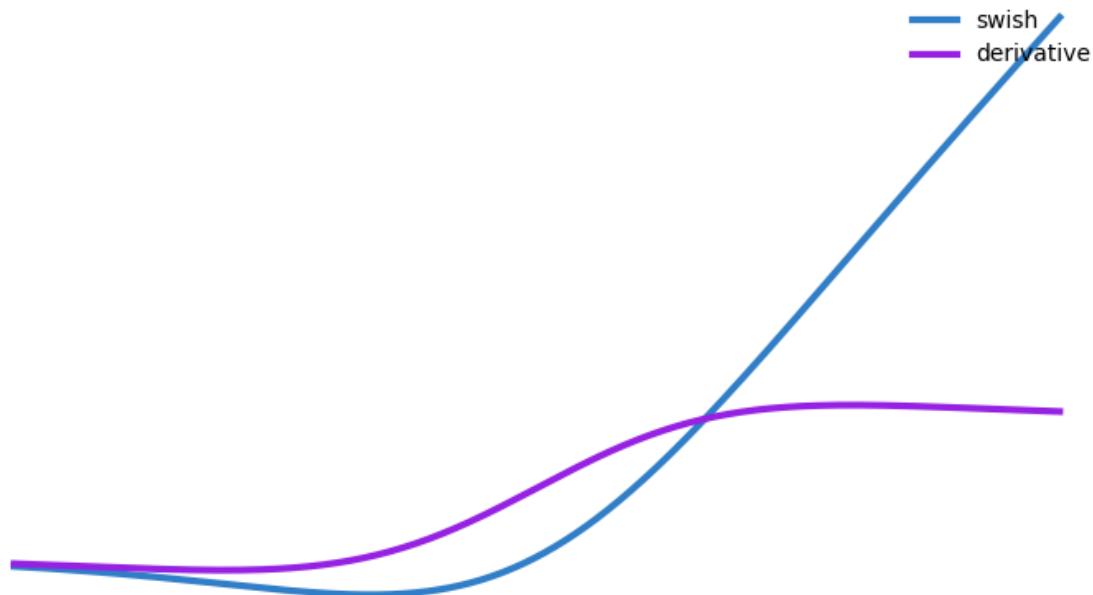
$$\Rightarrow \frac{d\mathcal{G}}{dx} \sim 0.5 \tanh(x^3) + 0.5 \sim 0.5 + 0.5 = 1, x \rightarrow +\infty \quad (2.26)$$

$$\Rightarrow \frac{d\mathcal{G}}{dx} \sim 0.5 \tanh(x^3) + 0.5 \sim -0.5 + 0.5 = 0, x \rightarrow -\infty \quad (2.27)$$

Nhìn sơ qua thì có vẻ công thức của GELU và đạo hàm của nó khá phức tạp, tuy nhiên khi được biểu diễn đồ thị lại là có hình ảnh cong mượt mà khá đẹp (xem hình 2.7.1). Điều này khác với những ReLU, Leaky ReLU, ELU hay SELU khi đồ thị hàm số bị gãy và đạo hàm không được liên tục (đạo hàm của ELU chỉ liên tục khi và chỉ khi $\alpha = 1$).

Tuy đã được đề xuất từ lâu, nhưng cũng như SELU, nghiên cứu về hiệu xuất hàm này trong thực tiễn vẫn chưa nhiều nên cần nhiều thực nghiệm để đưa ra kiểm chứng.

2.8 Hàm Swish



Hình 2.8.1: Đồ thị hàm số của hàm Swish ($\beta = 1$) và đạo hàm của nó.

Không kinh khủng như GELU, Swish [15] có công thức đơn giản hơn khi là một hàm tuyến tính bội với một hàm Sigmoid. Đó cũng là lí do mà hàm này còn có cái tên khác là SiLU (Sigmoid Linear Unit):

$$\text{Swish}(x) = \mathcal{S}(x) = x\sigma(\beta x) = \frac{x}{1 + e^{-\beta x}}; \beta \in \mathbb{R} \quad (2.28)$$

Giống với các hàm LU khác (trừ Leaky ReLU), Swish không bị chặn trên nhưng lại bị chặn dưới. Swish có hình dạng khá giống với GELU khi nó rất mượt mà không gãy và trông còn có vẻ mượt hơn (Xem hình 2.8.1).

Có 2 trường hợp đặc biệt của hệ số β . Với β được chọn là 0, hàm Swish gần giống như một hàm tuyến tính vì $\sigma(0x) = 0.5$. Nếu $\beta \rightarrow \infty$, hàm Swish giờ đây trở thành một hàm ReLU do $\lim_{\beta \rightarrow -\infty} \sigma(\beta x) = 0$, $\lim_{\beta \rightarrow +\infty} \sigma(\beta x) = 1$. Swish nằm giữa 2 thái cực là tuyến tính và ReLU biến thiên nhờ vào hệ số β .

Đạo hàm của hàm Swish tính được theo đạo hàm hàm hợp như sau:

$$\begin{aligned} \frac{d\mathcal{S}}{dx} &= \sigma(\beta x) + x \cdot \beta \sigma'(\beta x) (1 - \sigma(\beta x)) \\ &= \sigma(\beta x) + \beta x \sigma'(\beta x) - \beta x \sigma^2(\beta x) \\ &= \beta x \sigma'(\beta x) + \sigma(\beta x) - \beta x \sigma^2(\beta x) \\ &= \beta \mathcal{S}'(x) + \sigma(\beta x) (1 - \beta \mathcal{S}(x)) \end{aligned} \quad (2.29)$$

Đạo hàm của Swish có hình dáng gần giống với GELU và liên tục trên toàn miền (Xem hình 2.8.1). Về mặt lý thuyết, có vẻ Swish khá tương tự với GELU và cũng chưa có nhiều thực nghiệm trên các mô hình khác nhau của hàm này nên cũng chưa thể đưa ra nhận xét nào quá đặc biệt.

2.9 Thông tin về các hàm kích hoạt được thực nghiệm

Trong các phần thực nghiệm sau, những hàm được đem vào những mô hình để đánh giá là Sigmoid, Tanh (tanh), ReLU (\mathcal{R}), Leaky ReLU - sẽ được gọi tắt là Leaky ($\mathcal{R}_L(\alpha = 0.2)$), ELU ($\mathcal{E}(\alpha = 1)$), SELU ($\mathcal{E}_S = \lambda \mathcal{E}$), GELU (\mathcal{G}) và cuối cùng là Swish ($\mathcal{S}(\alpha = 1)$).

Chương 3

Thực nghiệm so sánh trên tập dữ liệu MNIST

Thực nghiệm đầu tiên sẽ là trên tập dữ liệu MNIST¹, xem xem những hàm kích hoạt khác nhau sẽ hoạt động thế nào trong bài toán phân loại chữ số viết tay từ 0 tới 9. Tập dữ liệu gồm 60,000 ảnh huấn luyện và 10,000 ảnh kiểm tra. Mỗi ảnh là một ảnh xám có kích thước $28 \times 28 \times 1$, có điểm ảnh có giá trị ban đầu cao nhất là 255 và thấp nhất là 0, sau được chuẩn hoá lại về khoảng giá trị $[0, 1]$ bằng cách chia giá trị tất cả các điểm ảnh cho 255. Trong quá trình huấn luyện, không sử dụng bất cứ kỹ thuật làm giàu dữ liệu nào. Một lưu ý là, tập ảnh dữ liệu kiểm tra sẽ được dùng để kiểm định trong quá trình huấn luyện, tức ta sẽ bỏ qua quá trình kiểm tra và ta có thể xem như giá trị kiểm định cuối cùng là giá trị kiểm tra nếu muốn.

Các thông số của mạng như chiều sâu của mạng, kiểu khởi tạo trọng số, kích thước batch, tốc độ học, và thuật toán tối ưu đều có một ảnh hưởng nhất định lên quá trình huấn luyện một mô hình mạng học sâu. Một số ảnh hưởng được đưa ra thử nghiệm để đánh giá như là việc tăng độ sâu của một mạng nơ-ron, thay đổi cách khởi tạo trọng số, thêm nhiều vào dữ liệu, sử dụng kiến trúc mạng tích chập. Đối với việc tăng độ sâu của mạng, kỹ thuật của mạng dư thừa không được áp dụng do cách thức trên sinh ra để xử lý việc tăng độ sâu của mạng, phần nào giảm đi tính đúng đắn về hiệu quả của các hàm kích hoạt.

3.1 Thực nghiệm thông qua mạng nơ-ron sâu

Bao gồm 11 kiến trúc được đưa ra, mỗi kiến trúc có số lượng tầng ẩn lần lượt tăng từ 2 cho tới 12. Mỗi ảnh đầu vào sẽ được làm phẳng thành một vector có số chiều 728 (chính là $28 \times 28 \times 1 = 728$) tương ứng với số đơn vị ở tầng đầu vào. Tầng đầu ra sẽ là 10 đơn vị ẩn tương ứng với 10 lớp từ 0 tới 9. Các tầng ẩn sẽ có số lượng đơn vị ẩn như sau:

- 2 tầng ẩn: $512 \rightarrow 128$.
- 3 tầng ẩn: $512 \rightarrow 256 \rightarrow 128$.
- 4 tầng ẩn: $512 \rightarrow 256 \rightarrow 128 \rightarrow 64$.
- 5 tầng ẩn: $512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32$.
- 6 tầng ẩn: $512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16$.
- 7 tầng ẩn: $512(\times 2) \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16$.

¹THE MNIST DATABASE of handwritten digits, đường dẫn: <http://yann.lecun.com/exdb/mnist/>

- 8 tầng ẩn: $512(\times 2) \rightarrow 256(\times 2) \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16$.
- 9 tầng ẩn: $512(\times 2) \rightarrow 256(\times 2) \rightarrow 128(\times 2) \rightarrow 64 \rightarrow 32 \rightarrow 16$.
- 10 tầng ẩn: $512(\times 2) \rightarrow 256(\times 2) \rightarrow 128(\times 2) \rightarrow 64(\times 2) \rightarrow 32 \rightarrow 16$.
- 11 tầng ẩn: $512(\times 2) \rightarrow 256(\times 2) \rightarrow 128(\times 2) \rightarrow 64(\times 2) \rightarrow 32(\times 2) \rightarrow 16$.
- 12 tầng ẩn: $512(\times 2) \rightarrow 256(\times 2) \rightarrow 128(\times 2) \rightarrow 64(\times 2) \rightarrow 32(\times 2) \rightarrow 16(\times 2)$.

Sau mỗi tầng dày đặc, một dropout có tỷ lệ 50% sẽ được áp dụng, riêng với tầng ẩn cuối cùng của mỗi kiến trúc, tỷ lệ dropout giảm xuống 25%.

Khi huấn luyện các mô hình với những hàm kích hoạt khác nhau, để cho công bằng thì khi cài đặt đều sử dụng chung một hàm tối ưu, tốc độ học và cùng một kiểu khởi tạo trọng số. Kỹ thuật dropout cũng giống nhau nếu không có lưu ý gì.

3.1.1 Dữ liệu không nhiễu và khởi tạo trọng số theo chuẩn hoá Lecun

Tất cả được khởi tạo trọng số theo chuẩn hoá Lecun, riêng đối với mô hình áp dụng hàm kích hoạt SELU sẽ sử dụng kỹ thuật alpha dropout với tỷ lệ bằng với tỷ lệ dropout của các hàm kích hoạt khác. Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 3.1.1, 3.1.2, 3.1.3, 3.1.4. Chi tiết về giá trị được cho ở bảng 3.1.1 và bảng 3.1.2. Ở phần bảng giá trị không có thống kê của Sigmoid, do hàm này có kết quả tệ khá rõ rệt (sẽ được nêu rõ ngay sau đây).

Với mạng nơ-ron không quá sâu - 2 tầng ẩn (hình 3.1.1(a), 3.1.1(b)), các hàm dễ dàng đạt được độ chính xác cao (tất cả đều trên 92%) và giá trị mất mát thấp (tất cả đều dưới 0.40). Hàm Sigmoid không được tích cực giống như các hàm còn lại khi thua thiệt hẳn về tốc độ hội tụ khi mãi gần 20 epoch mới đạt được mất mát dưới giá trị 1. Tuy các hàm ngoài Sigmoid chưa thấy được sự khác biệt rõ rệt, nhưng có vẻ như SELU là người yếu thế nhất trong nhóm này khi giá trị mất mát tuy hội tụ khá sớm. Chỉ sau 20 epoch (bảng 3.1.3), 80% giá trị mất mát đã bằng giá trị mất mát cuối cùng. Nhưng kết quả mất mát cuối cùng không quá ẩn tượng và gần bằng so với Sigmoid.

Ngay khi tăng lên 3 tầng ẩn (hình 3.1.1(c), 3.1.1(d)), Sigmoid đã cho thấy sự lép vê rõ rệt. Không chỉ thế, SELU cũng có vẻ sẽ chung số phận với Sigmoid khi số lượng tầng ẩn tiếp tục tăng. Dù cho cuối cùng SELU vẫn đạt giá trị chính xác trên 92% nhưng giá trị mất mát đã tiệm cận 0.50 (chính xác là 0.49).

Sigmoid chính thức bỏ cuộc khóc cuộc đua khi tới số tầng ẩn là 5 (hình 3.1.2(a), 3.1.2(b)), gần như mô hình với hàm này không thể học được. SELU tuy vẫn còn có khả năng học, nhưng cho kết quả rất tệ như Sigmoid những mô hình ban đầu khi giá trị mất mát đạt tới 1.67. Dù cho vẫn giữ được độ chính xác gần 90%, nhưng khi tăng thêm số tầng ẩn thì có khả năng cao sẽ bị giảm đi thấp rất nhiều. Ngoài ra, ta thấy rằng hàm ELU và Tanh có vẽ hội tụ nhanh hơn so với các hàm còn lại, và chính xác là như thế khi ELU chỉ cần đúng 3 epoch để 80% mất mát hiện tại bằng mất mát cuối. Với Tanh thì con số này là 24 epoch. Kế đến là Leaky và RELU ngay sau (cùng 27 epoch), Swish (38) và GELU (35) ở nhóm sau cùng. Giá trị chính xác và mất mát của các hàm kích hoạt sau 50 epoch vẫn chưa có sự chênh lệch đáng kể.

Dộ sâu được nâng lên 7 tầng ẩn (hình 3.1.2(e), 3.1.2(f)), Leaky đã bỏ lại RELU chung với nhóm của GELU và Swish ở khoảng 15 epoch đầu tiên. Mãi khi về cuối, RELU cũng như GELU và Swish mới bám được với nhóm dẫn đầu gồm có ELU, Tanh và Leaky. Lúc này, vẫn chưa có sự chênh lệch quá đáng kể về giá trị chính xác cũng như mất mát. Duy chỉ có Swish có vẻ đuối

sức so với những hàm còn lại khi giá trị chính xác duy trì được $\approx 95\%$ giảm xuống 92.03%. SELU lúc này đã không còn có khả năng tiếp tục khi giá trị chính xác giảm thảm hại xuống 37.25% và giá trị mất mát là 7.02. So với lúc 5 tầng ẩn thì đã giá trị chính xác đã giảm đi 50%, giá trị mất mát thì tăng lên 5.35.

Kết quả khi số tầng ẩn tăng lên là 9 (hình 3.1.3(c), 3.1.3(d)) cho thấy chỉ có Leaky, Elu và Tanh là đủ sức để đi tiếp những mô hình sâu hơn khi vẫn giữ được giá trị chính xác trên 90%. Trong nhóm này, ELU có vẻ nổi trội hơn một chút so với hai hàm còn lại ở những epoch đầu. Dẫu thế, những kết quả về cuối cho thấy ELU vẫn chưa thể tách nhóm khi giá trị chính xác vẫn chỉ đang xấp xỉ Leaky. Nhóm còn lại gồm RELU, GELU và Swish thì chỉ có RELU và GELU tiếp tục khi Swish cũng cho thấy sự hụt hơi của mình (chính xác: 57.32%, mất mát: 1.21). Tuy nhiên ở ngay độ sâu kế tiếp - 10 tầng ẩn (hình 3.1.3(e), 3.1.3(f)), GELU cũng như Swish đã không còn có khả năng học được nữa khi giá trị chính xác cũng như mất mát đã bão hòa ở mức 10.60% và 2.30.

Ở mô hình cuối cùng - 12 tầng ẩn (hình 3.1.4(c), 3.1.4(d)), Leaky lúc này cũng không còn bám được với nhóm đầu dù là những epoch cuối khi giá trị tụt xuống 62.84% và mất mát 1.08. Nhìn vô địch ở mô hình này là ELU với chiến thắng khá thuyết phục khi ta thấy có cách biệt rõ rệt giữa ELU so với Tanh và cụ thể là giá trị chính xác $79.24\% > 76.97\%$, giá trị mất mát $0.45 < 0.6$. Tốc độ hội tụ của ELU cũng nhanh hơn rõ rệt khi sau 16 epoch đã đạt được giá trị cần thiết so với Tanh cần tới 26 epoch.

	2	3	4	5	6	7	8	9	10	11	12
tanh	93.87	94.77	95.15	94.86	94.55	94.87	94.61	94.55	94.66	94.64	76.97
\mathcal{R}	96.87	97.13	97.39	97.32	96.66	96.16	95.33	79.99	79.18	42.32	37.99
\mathcal{R}_L	96.22	96.82	97.12	96.78	96.70	97.02	97.01	96.39	96.64	87.16	62.84
\mathcal{E}	94.79	95.76	95.99	95.92	95.43	95.84	95.92	95.92	95.67	95.45	79.24
\mathcal{E}_S	92.85	92.68	91.47	87.25	65.71	37.25	30.75	27.47	10.79	10.87	10.60
\mathcal{G}	96.03	96.28	96.29	96.38	95.05	94.69	92.58	80.23	10.60	10.60	10.60
\mathcal{S}	95.13	95.51	95.39	95.38	93.07	92.03	88.86	57.32	10.60	10.60	10.60

Bảng 3.1.1: Giá trị chính xác (%) của các hàm kích hoạt (chuẩn hoá Lecun) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.

	2	3	4	5	6	7	8	9	10	11	12
tanh	0.21	0.19	0.17	0.19	0.22	0.21	0.23	0.24	0.26	0.27	0.60
\mathcal{R}	0.11	0.09	0.09	0.11	0.16	0.28	0.33	0.53	0.57	1.21	1.39
\mathcal{R}_L	0.13	0.11	0.10	0.12	0.13	0.13	0.13	0.17	0.19	0.37	1.08
\mathcal{E}	0.19	0.15	0.14	0.15	0.17	0.16	0.16	0.16	0.18	0.20	0.45
\mathcal{E}_S	0.38	0.49	0.81	1.67	2.79	7.02	7.55	4.86	2.30	2.30	2.30
\mathcal{G}	0.14	0.12	0.12	0.13	0.20	0.22	0.34	0.72	2.30	2.30	2.30
\mathcal{S}	0.17	0.16	0.15	0.16	0.29	0.36	0.46	1.21	2.30	2.30	2.30

Bảng 3.1.2: Giá trị mất mát của các hàm kích hoạt (chuẩn hoá Lecun) tương ứng với các mô hình trên tập dữ liệu MNIST.

	2	3	4	5	6	7	8	9	10	11	12	μ
tanh	19	22	30	24	26	27	30	30	26	27	26	26
\mathcal{R}	33	34	31	27	38	36	34	31	33	23	27	32
\mathcal{R}_L	32	31	32	27	33	31	31	34	39	46	27	33
\mathcal{E}	26	33	32	28	28	30	32	31	30	26	16	28
\mathcal{E}_S	20	22	16	3	1	1	1	1	1	1	1	X
\mathcal{G}	32	35	35	35	38	43	42	45	1	1	1	X
\mathcal{S}	33	32	37	38	40	39	43	43	1	1	1	X

Bảng 3.1.3: Số epoch cần thiết để 80% giá trị mất mát (chuẩn hoá Lecun) bằng giá trị mất mát cuối cùng.

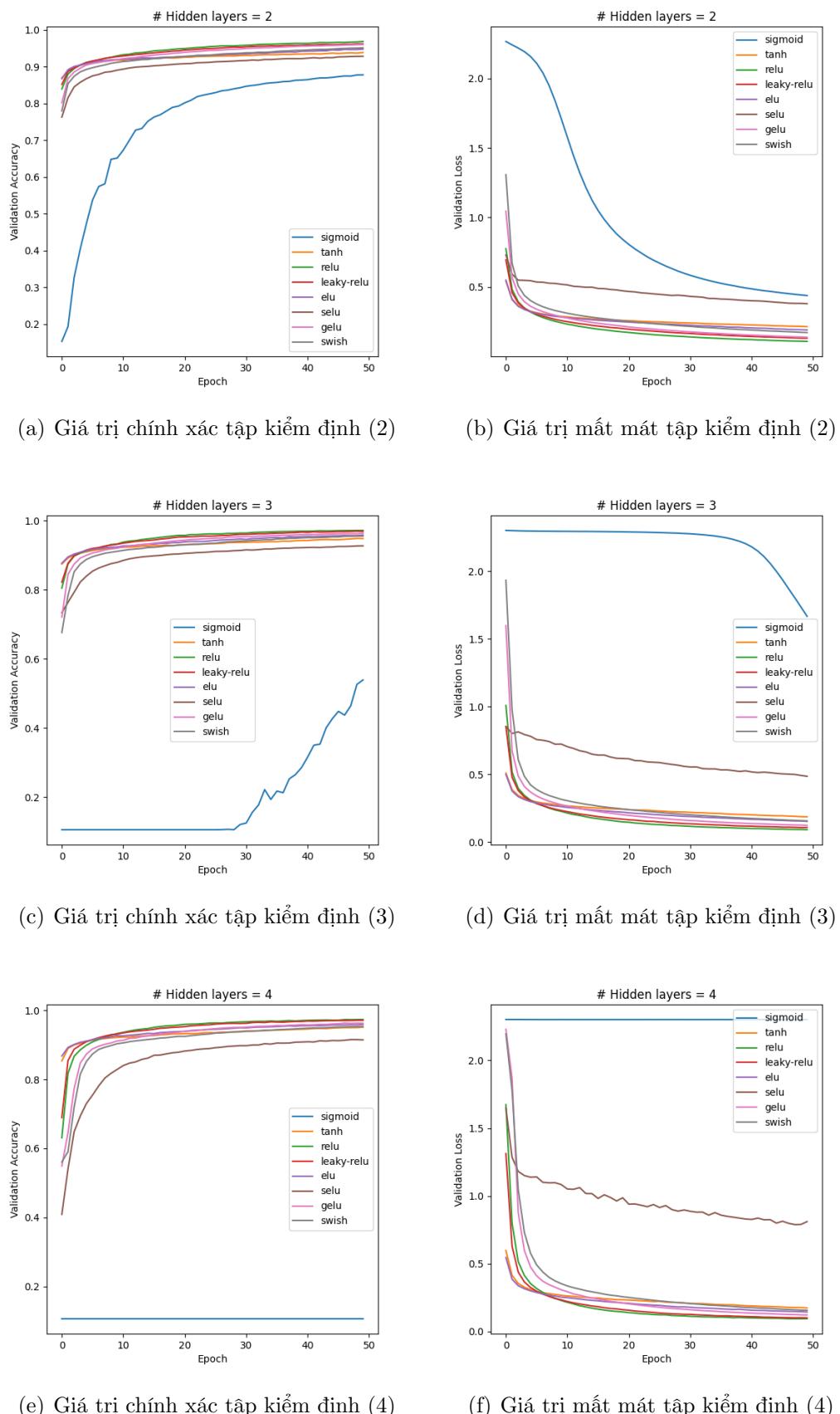
	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{chính xác}} (\%)$	93.05	83.31	92.77	94.18	50.7	70.85	67.68
$\sigma_{\text{chính xác}}^2 (\%)$	0.0026	0.0456	0.0097	0.0022	0.1152	0.1380	0.1330
$\mu_{\text{mất mát}}$	0.25	0.44	0.24	0.19	2.95	0.81	0.9
$\sigma_{\text{mất mát}}^2$	0.0131	0.1907	0.0748	0.0069	5.5967	0.8616	0.8204

Bảng 3.1.4: Giá trị trung bình và phương sai của các hàm kích hoạt (chuẩn hoá Lecun) trong 11 mô hình thử nghiệm.

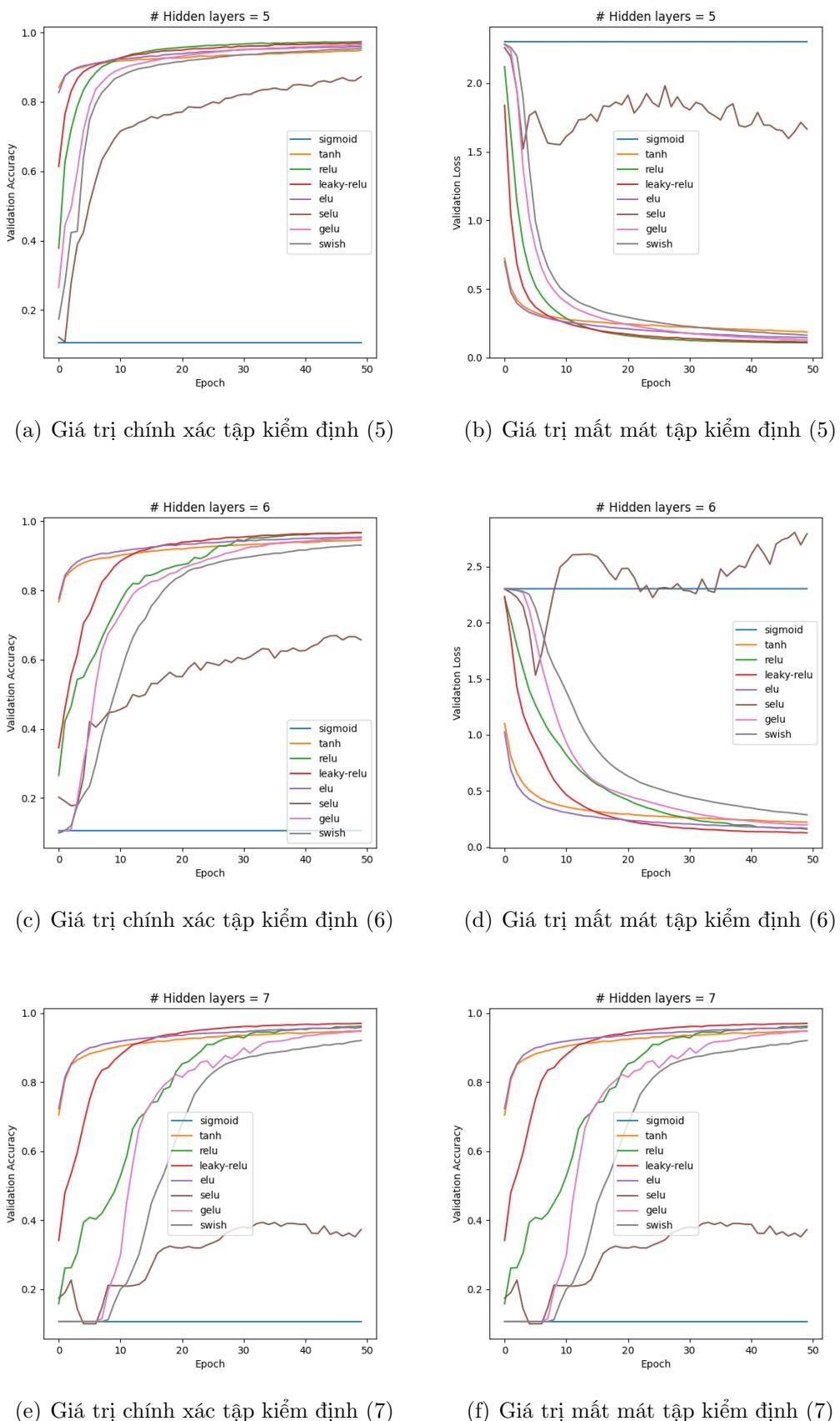
Dộ sâu của mạng tăng thì ảnh hưởng nhiều tới giá trị chính xác hơn là so với giá trị mất mát khi giá trị chính xác có những góc gãy bớt tù hơn so với giá trị mất mát (hình 3.1.4(e), 3.1.4(f)). Theo bảng 3.1.1 và bảng 3.1.2, ở những mạng chưa quá sâu, RELU là hàm có kết quả tốt nhất. Khi mạng sâu hơn một chút thì Leaky là hàm dẫn đầu, nhưng khi mạng trở nên phức tạp hơn với khoảng 9 tầng ẩn thì Leaky không còn đạt được phong độ tốt nữa mà thay vào đó là ELU. Có 2 hàm rất bền bỉ từ những mạng đầu tiên là ELU và Tanh. Trong đó, Tanh luôn tìm được điểm cực tiểu sớm nhất (bảng 3.1.3) bất kể độ sâu của mạng. Nhưng điều này cũng chưa cho thấy Tanh thực sự tốt hơn RELU hay Leaky, lí do có thể là vì Tanh chỉ đang tìm được hố cực tiểu nông, trong khi RELU hay Leaky tìm thấy một hố cực tiểu sâu hơn để có giá trị tốt hơn. Mãi khi mạng quá sâu, RELU và Leaky không thể tìm được hố cực tiểu tốt nữa, thì lúc đó hố cực tiểu của Tanh nghiêm trở thành một giá trị tốt hơn.

Theo như bảng 3.1.4, có 3 hàm đạt được giá trị chính xác trung bình $\mu_{\text{chính xác}}$ trong 11 mô hình trên 90% lần lượt từ thấp tới cao là Leaky, Tanh và ELU. Phương sai về chính xác $\sigma_{\text{chính xác}}^2$ và mất mát $\sigma_{\text{mất mát}}^2$ của Leaky cao hơn so với Tanh với ELU là do ở những tầng cuối thì thông số của Leaky thay đổi nhiều khi giá trị chính xác giảm đáng kể dù cho mất mát tăng không quá căng thẳng.

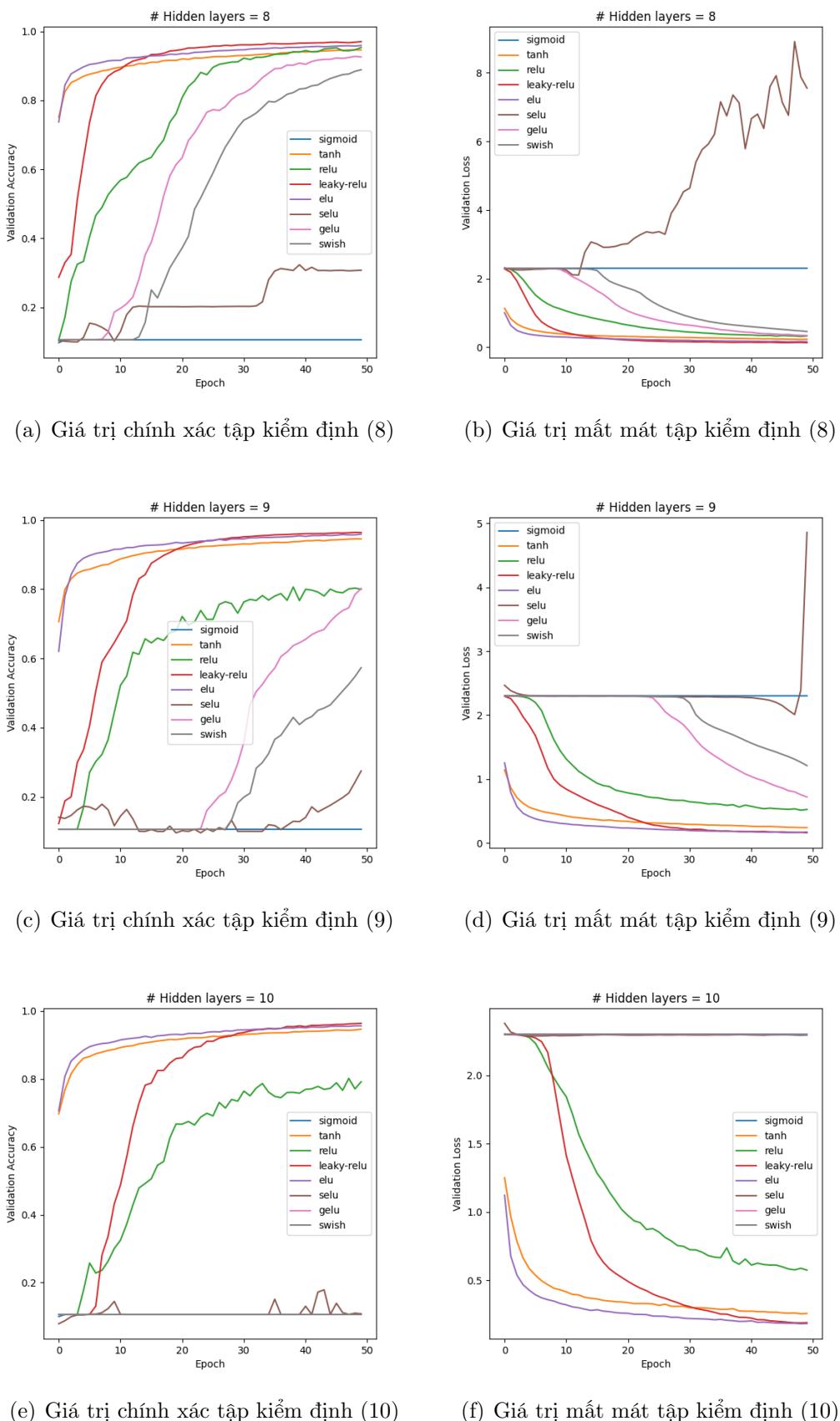
Hàm Sigmoid thực sự không phù hợp để huấn luyện cho những mạng học sâu đúng với nhưng suy tính lý thuyết, vì thế các thực nghiệm tiếp theo sẽ không có hàm Sigmoid.



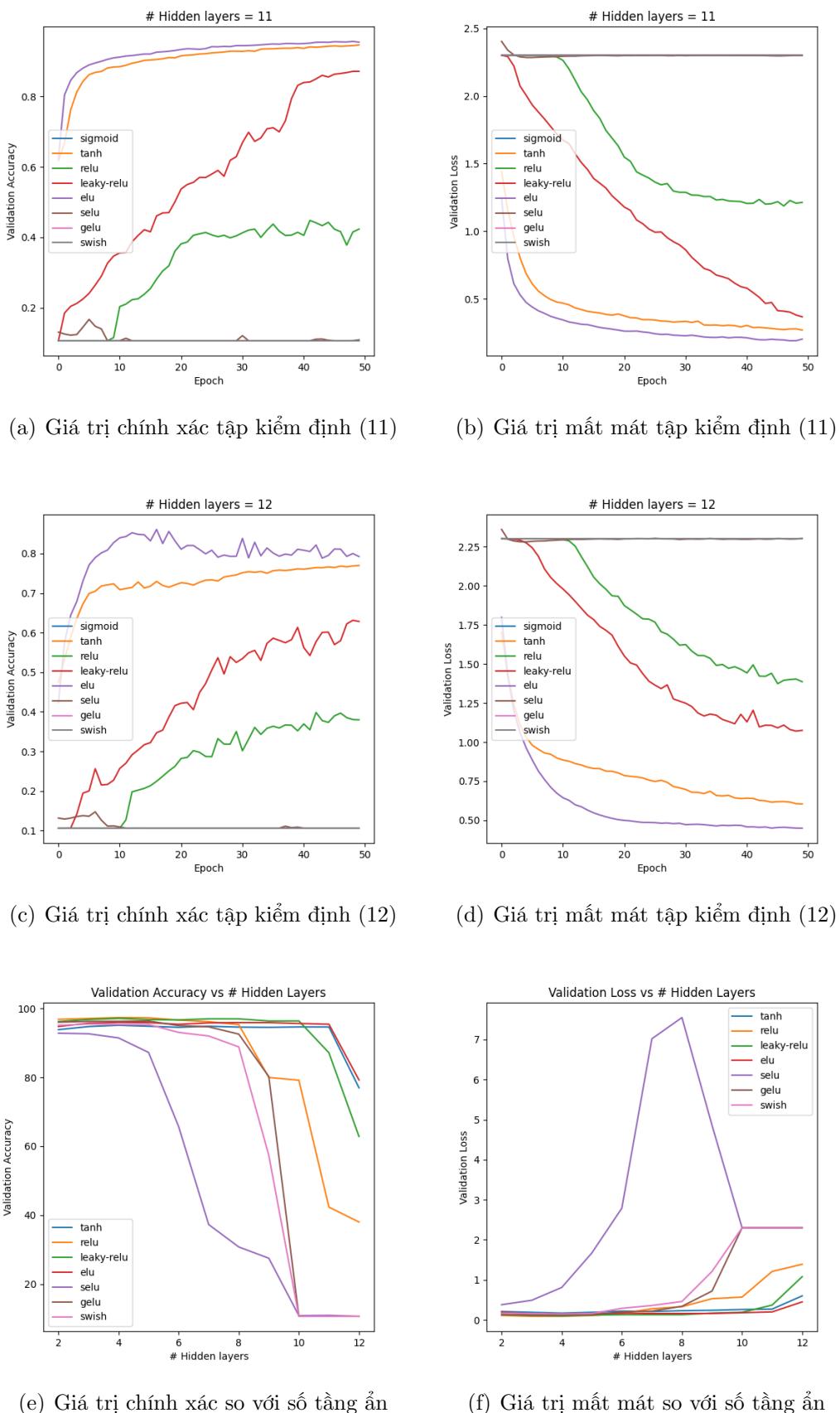
Hình 3.1.1: Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 2, 3, 4).



Hình 3.1.2: Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 5, 6, 7).



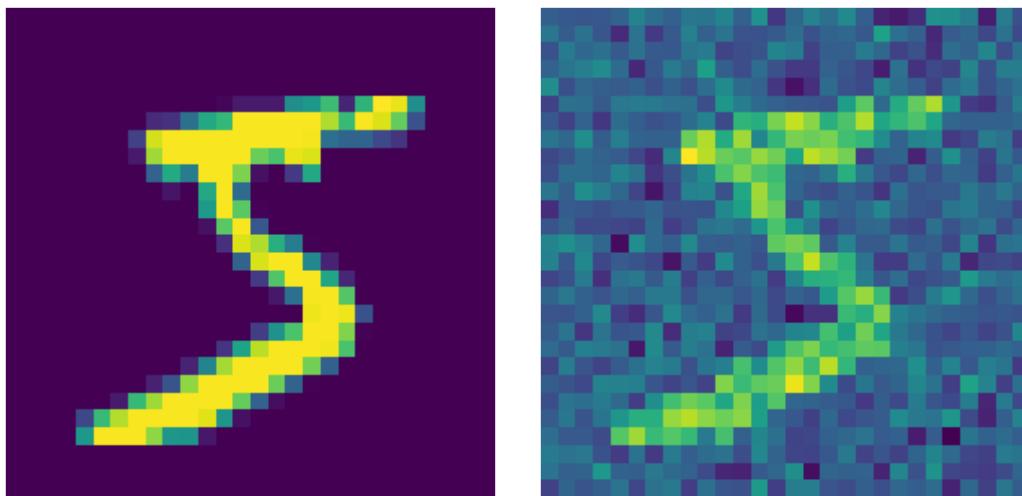
Hình 3.1.3: Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 8, 9, 10).



Hình 3.1.4: Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (chuẩn hoá Lecun) trên tập kiểm định so với số tầng ẩn [(e), (f)].

3.1.2 Dữ liệu nhiễu Gauss và khởi tạo trọng số theo chuẩn hoá Lecun

Dữ liệu bây giờ được tạo nhiễu theo hàm Gauss có độ lệch chuẩn $\sigma = 0.2$. Hình 3.1.5 cho ta thấy sự khác biệt giữa mẫu nguyên thuỷ và mẫu khi có nhiễu tác động vào. Khoảng giá trị bây giờ của các phần tử trong mẫu nhiễu có thể âm (tức < 0) và cũng có thể lớn hơn 1. Như trong ảnh 3.1.5(b) có phần tử điểm ảnh có giá trị nhỏ nhất và lớn nhất lần lượt là -0.5876574 và 1.4352913 .



(a) Ảnh chữ số 5 trong tập dữ liệu MNIST (b) Ảnh chữ số 5 khi thêm nhiễu Gauss ($\sigma = 0.2$) trong tập dữ liệu MNIST

Hình 3.1.5: Ví dụ về ảnh trước và sau khi thêm nhiễu Gauss từ tập dữ liệu MNIST.

Cách thức huấn luyện vẫn được áp dụng như phần 3.1.1, tuy nhiên lần này có một chút khác biệt là kỹ thuật alpha dropout sẽ không được áp dụng cho SELU mà sẽ sử dụng cách dropout bình thường với cùng tỷ lệ như các hàm kích hoạt khác. Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 3.1.6, 3.1.7, 3.1.8, 3.1.9. Chi tiết về giá trị được cho ở bảng 3.1.5 và bảng 3.1.6.

	2	3	4	5	6	7	8	9	10	11	12
tanh	92.14	93.10	93.40	93.09	92.83	93.35	92.98	92.74	92.86	92.13	67.80
\mathcal{R}	95.46	95.88	95.97	95.78	94.57	93.83	91.21	86.59	42.95	40.26	45.08
\mathcal{R}_L	94.83	95.18	95.55	95.14	94.63	95.04	95.10	94.01	93.08	73.42	54.34
\mathcal{E}	93.03	94.23	94.52	94.43	93.86	94.30	94.13	93.99	93.98	93.94	84.06
\mathcal{E}_S	91.46	93.10	93.76	93.66	93.32	94.12	94.31	93.06	94.03	93.77	68.40
\mathcal{G}	95.02	95.38	95.48	95.17	93.31	93.55	87.75	70.12	10.60	10.60	10.60
\mathcal{S}	94.08	94.42	94.42	94.02	92.34	88.00	80.94	46.26	21.84	10.60	10.60

Bảng 3.1.5: Giá trị chính xác (%) của các hàm kích hoạt (nhiễu Gauss + chuẩn hoá Lecun) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.

Ở những kiến trúc đầu tiên, chẳng hạn khi độ sâu tầng ẩn là 2 (hình 3.1.6(a), 3.1.6(b)), chưa có sự phân hoá rõ ràng giữa các hàm. Nếu tính chi ly, thì ta có 2 cái tên quen thuộc ở những lượt đầu là RELU và Leaky có độ chính xác cao nhất (95.46% và 94.83%). Trong nhóm

	2	3	4	5	6	7	8	9	10	11	12
tanh	0.26	0.23	0.22	0.25	0.28	0.28	0.30	0.31	0.34	0.37	0.77
\mathcal{R}	0.15	0.13	0.15	0.17	0.31	0.36	0.49	0.59	1.35	1.30	1.48
\mathcal{R}_L	0.17	0.15	0.16	0.17	0.23	0.21	0.22	0.34	0.38	0.78	1.24
\mathcal{E}	0.23	0.19	0.18	0.19	0.23	0.21	0.22	0.24	0.25	0.27	0.51
\mathcal{E}_S	0.29	0.24	0.21	0.21	0.24	0.23	0.23	0.25	0.26	0.28	0.72
\mathcal{G}	0.16	0.15	0.15	0.17	0.29	0.28	0.57	0.94	2.30	2.30	2.30
\mathcal{S}	0.20	0.18	0.19	0.21	0.30	0.48	0.65	1.50	2.02	2.30	2.30

Bảng 3.1.6: Giá trị mất mát của các hàm kích hoạt (nhiều Gauss + chuẩn hoá Lecun) tương ứng với các mô hình trên tập dữ liệu MNIST.

có độ chính xác $\approx 95\%$ thì có GELU (95.02%) - hàm không cho kết quả không khả quan ở phần thực nghiệm trước (phần 3.1.1).

Khi số tầng ẩn là 5 (hình 3.1.7(a), 3.1.7(a)), chất lượng độ chính xác của các hàm kích hoạt vẫn chưa phân nhóm khi tất cả đều xấp xỉ khoảng 94-95%. Hàm Tanh có tốc độ hội tụ khá nhanh khi chỉ cần 15 epoch để đạt ngưỡng (bảng 3.1.7). Dẫu vậy, rõ ràng là nhận định hàm Tanh tìm được hổ cực tiểu nông là một nhận định hợp lý khi sau 50 epoch, giá trị mất mát của hàm này không thay đổi nhiều và đứng ở 0.25, trở thành hàm có mất mát cao nhất trong lục này.

	2	3	4	5	6	7	8	9	10	11	12	μ
tanh	10	20	20	15	19	23	16	22	24	31	15	20
\mathcal{R}	29	31	24	27	29	36	33	41	18	26	16	28
\mathcal{R}_L	29	30	26	27	28	26	28	29	40	35	30	30
\mathcal{E}	20	29	29	27	25	25	25	25	28	23	30	26
\mathcal{E}_S	4	16	22	22	18	23	22	22	24	26	9	20
\mathcal{G}	34	33	33	32	38	40	46	46	1	1	1	X
\mathcal{S}	31	33	31	33	39	39	44	41	1	1	1	X

Bảng 3.1.7: Số epoch cần thiết để 80% giá trị mất mát (nhiều Gauss + chuẩn hoá Lecun) bằng giá trị mất mát cuối cùng.

Tới 7 tầng ẩn (hình 3.1.7(e), 3.1.7(f)), Swish là cái tên đầu tiên có vẻ không còn đủ khả năng để đi tiếp. Tuy RELU và GELU có màn thể hiện rất tốt ở gian đoạn đầu, nhưng cũng như lần thử nghiệm trước (phần 3.1.1), khi số tầng ẩn khá sâu thì đã giảm hiệu quả của 2 hàm này. Leaky tuy hội tụ khá chậm so với nhóm đầu - nhóm gồm Tanh, ELU và SELU khi phải mất tới 36 epoch mới đạt ngưỡng, nhưng về cuối lại có một kết quả chính xác tốt khi vẫn trên 90% và cao hơn cả hàm Tanh ($93.83\% > 92.92\%$). Một điểm khá bất ngờ là khi không áp dụng alpha dropout, SELU đạt kết quả rất tốt. Không chỉ có giá trị chính xác cao, mất mát thấp mà việc hội tụ cũng rất nhanh khi đạt ngưỡng sau 23 epoch, ngang với hàm dễ đạt ngưỡng như Tanh.

Hình 3.1.8(c) và hình 3.1.8(d) cho thấy rằng GELU không còn có thể duy trì được phong độ như RELU đang làm. Tuy là RELU có vẻ không phù hợp cho những mạng sâu, nhưng giá

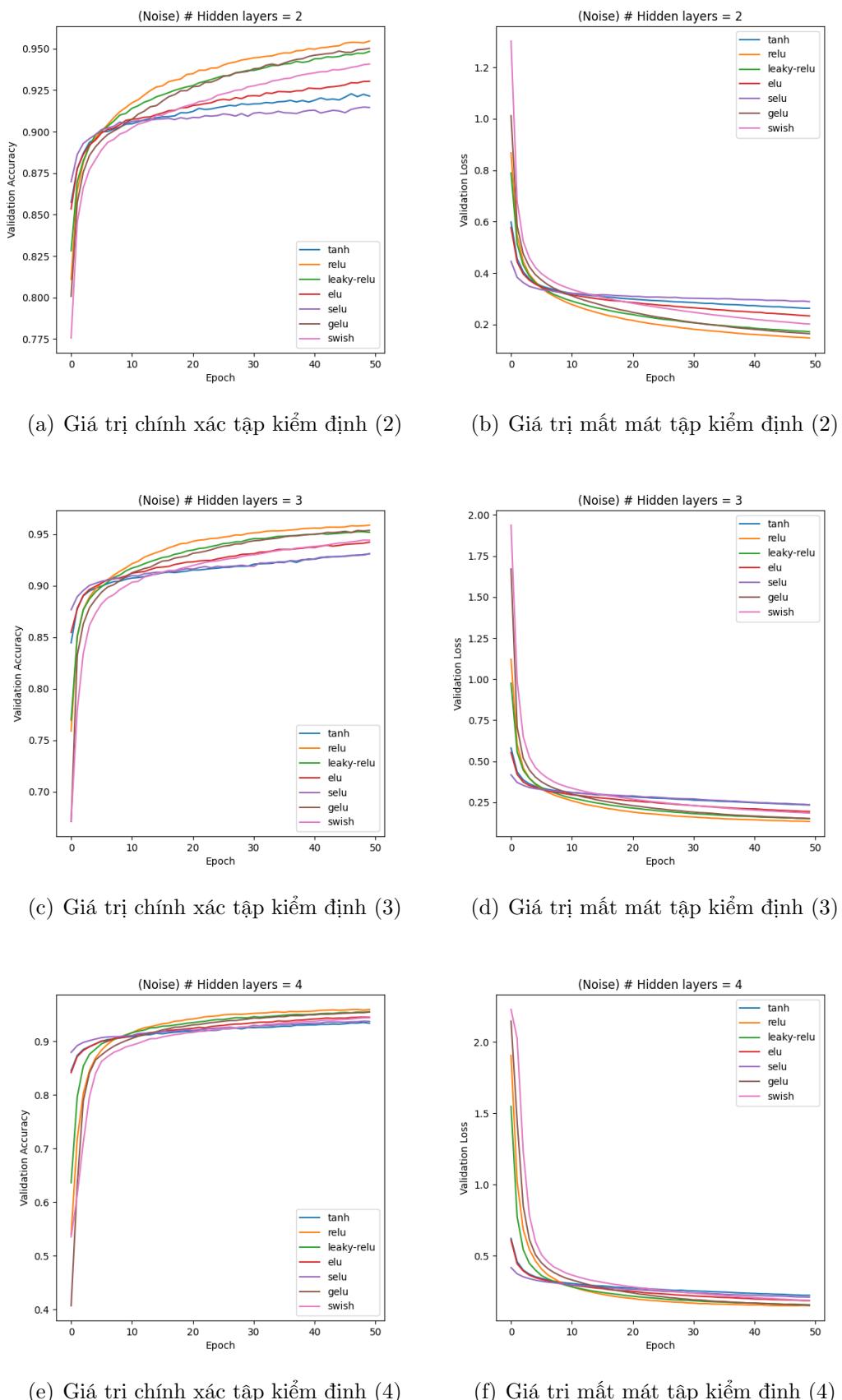
trị chính xác và mất mát không ở mức quá tệ (chính xác 86.59% và mất mát 0.59). Leaky giờ đây đã tụt xuống ngang với hàm Tanh, dẫn đến việc ta thấy Leaky hội tụ rất lâu, và phải tới 41 epoch mới đạt ngưỡng. ELU và người anh em SELU của mình nổi trội với độ chính xác $94\% \pm 0.01\%$. Ngoài ra 2 hàm trên, còn Tanh và Leaky là hai hàm có giá trị chính xác trên tập kiểm định trên 90% sau 50 epoch. Điều này được kéo dài và kết thúc ngay khi số tầng ẩn tăng lên 11 (hình 3.1.9(a), 3.1.9(b)) khi giá trị chính xác Leaky lúc này chỉ còn 73.42%, trong khi 3 hàm Tanh, ELU và SELU vẫn trên 90%.

Ở kiến trúc cuối cùng - 12 tầng ẩn (hình 3.1.9(c), 3.1.9(d)), có 3 nhóm tách biệt khá rõ rệt (không tính những hàm không còn khả năng học là GELU và Swish). Nhóm kết quả thấp nhất là RELU và Leaky với độ chính xác tầm khoảng 50% (45.05% và 54.34%) và mất mát cao hơn 1.0 (1.48 và 1.24). Nhóm giữa là Tanh và SELU với mất mát xấp xỉ nhau với mất mát là 0.77 và 0.72, giá trị chính xác của cả hai $\approx 68\%$. Nhóm có kết quả tốt nhất và cũng chỉ duy nhất ELU khi giá trị chính xác là 84.06% (vẫn trên 80%).

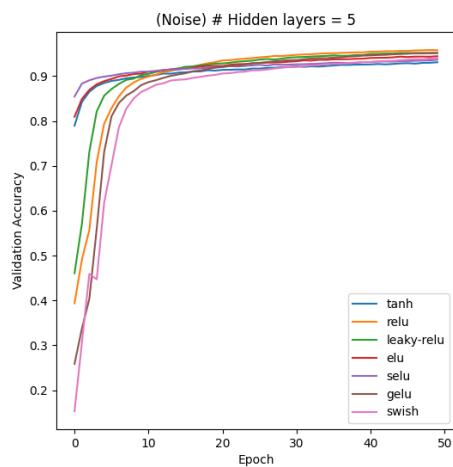
	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{chính xác}} (\%)$	90.58	79.78	89.12	93.13	91.27	68.87	66.14
$\sigma_{\text{chính xác}}^2 (\%)$	0.0052	0.0522	0.0159	0.0008	0.0053	0.1321	0.1188
$\mu_{\text{mất mát}}$	0.33	0.59	0.37	0.25	0.29	0.88	0.94
$\sigma_{\text{mất mát}}^2$	0.0215	0.2526	0.1068	0.0072	0.0194	0.8121	0.7380

Bảng 3.1.8: Giá trị trung bình và phương sai của các hàm kích hoạt (nhiều Gauss + chuẩn hoá Lecun) trong 11 mô hình thử nghiệm.

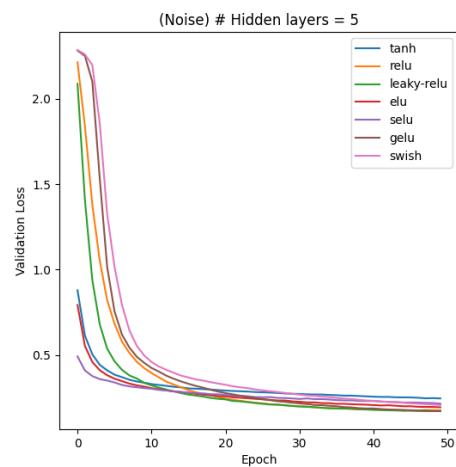
Kết quả của lần thực nghiệm này không có quá nhiều khác biệt (hình 3.1.9(e), 3.1.9(f)) so với lần thử nghiệm khi tập dữ liệu không có nhiễu (phần 3.1.1). Hàm RELU và Leaky vẫn là những hàm nổi trội với giá trị chính xác cao trong những mô hình có độ sâu vừa phải. Về hàm Tanh, vẫn là một hàm không thể hiện được sức mạnh ở những mô hình đầu như RELU hay Leaky khi nhanh chóng tìm một hố sâu cực tiểu. Tuy vậy, điều này lại vẫn là lợi thế khi tăng ẩn trong các mạng được nâng lên, đó là lí do mà Tanh có $\mu_{\text{chính xác}} = 90.58\%$ sau 11 mô hình. ELU vẫn là hàm có kết quả tốt nhất với $\mu_{\text{chính xác}} = 93.13\%$ và $\mu_{\text{mất mát}} = 0.25$. SELU cũng có kết quả rất khả quan sau khi loại bỏ đi kỹ thuật alpha dropout cho mô hình có hàm kích hoạt này khi có $\mu_{\text{chính xác}} = 91.27\%$ (chỉ thua mỗi ELU). Một ẩn tượng nữa của SELU là nó cũng có tốc độ hội tụ khá nhanh, ngang với Tanh. Dẫu thế, vẫn chưa khẳng định là SELU cũng chỉ tìm hố cực tiểu nồng như Tanh vì ngoại trừ 12 tầng ẩn, những mô hình còn lại SELU có giá trị mất mát cách một hàm cho kết quả tốt như ELU không quá 0.03.



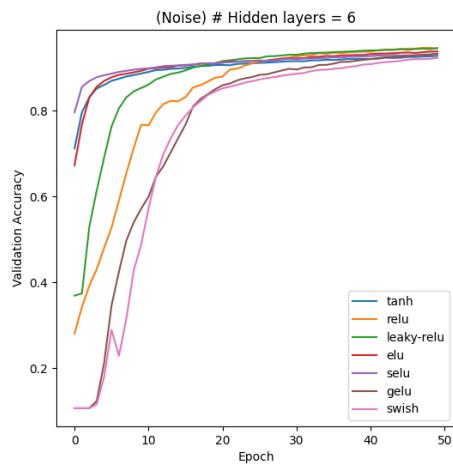
Hình 3.1.6: Giá trị chính xác và mất mát (nhiều Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 2, 3, 4).



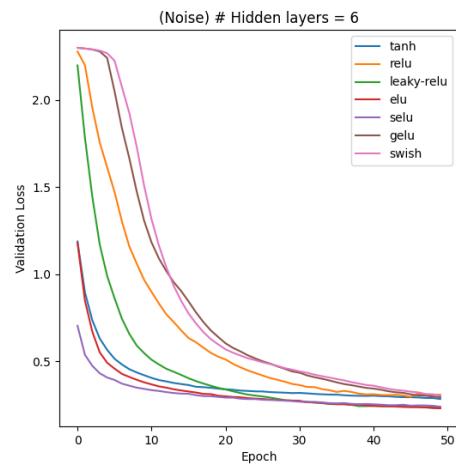
(a) Giá trị chính xác tập kiểm định (5)



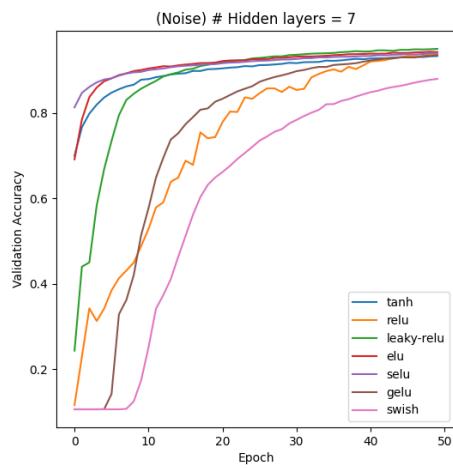
(b) Giá trị mất mát tập kiểm định (5)



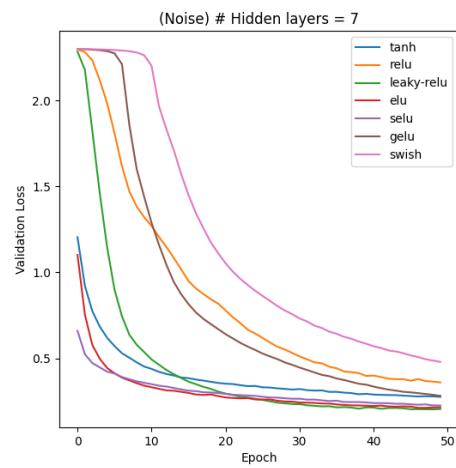
(c) Giá trị chính xác tập kiểm định (6)



(d) Giá trị mất mát tập kiểm định (6)

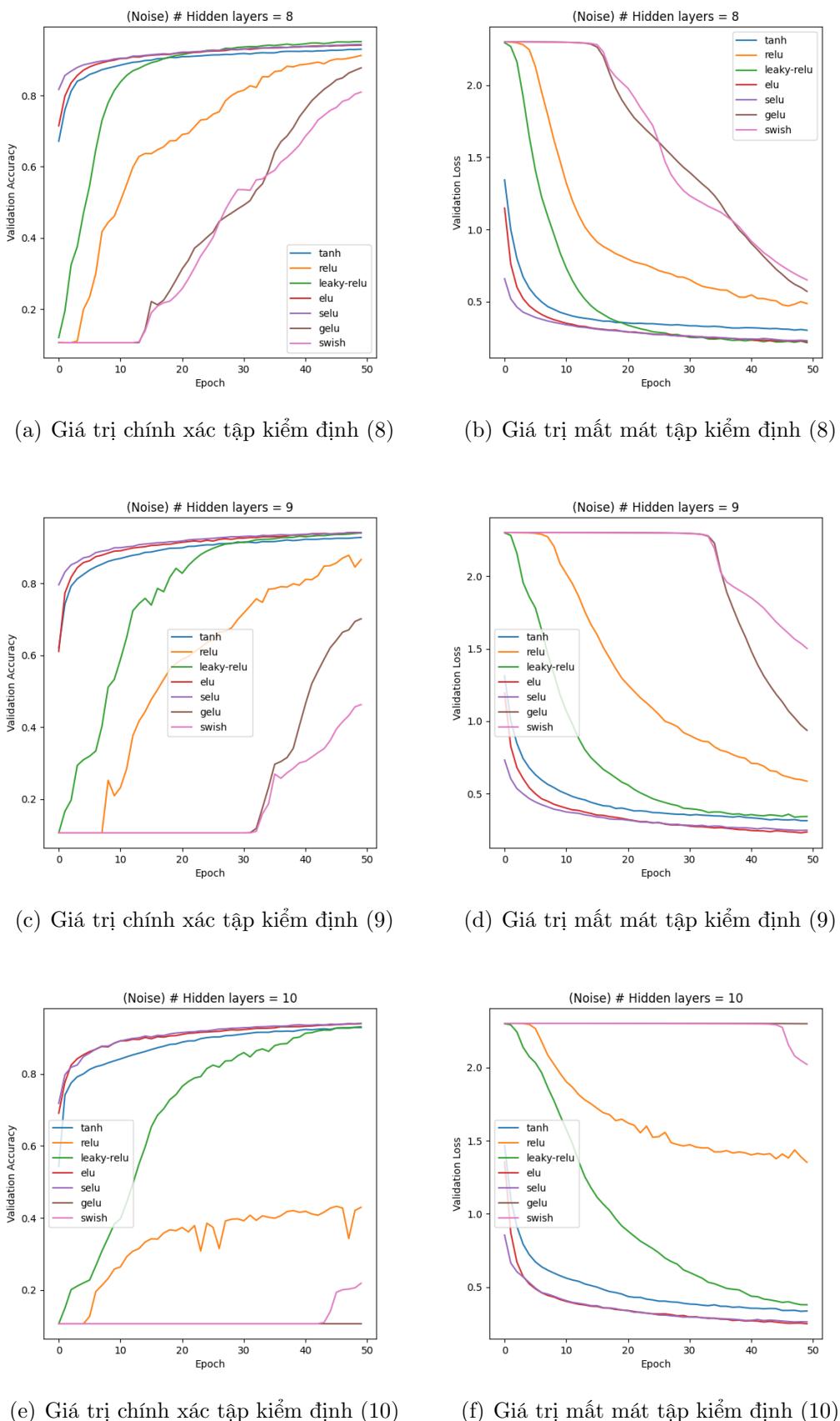


(e) Giá trị chính xác tập kiểm định (7)

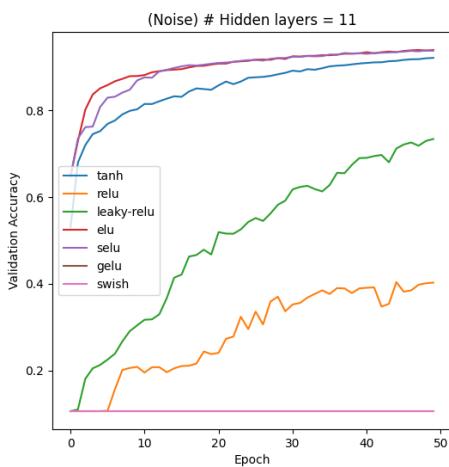


(f) Giá trị mất mát tập kiểm định (7)

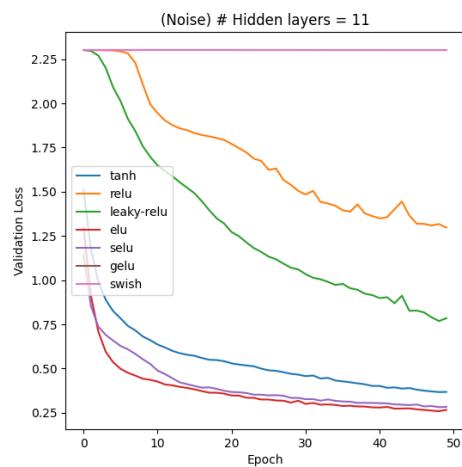
Hình 3.1.7: Giá trị chính xác và mất mát (nhiều Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 5, 6, 7).



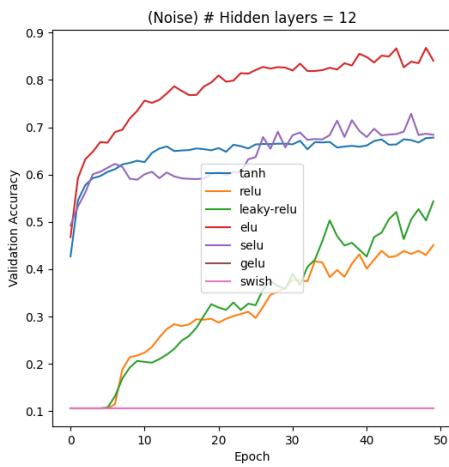
Hình 3.1.8: Giá trị chính xác và mất mát (nhiều Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 8, 9, 10).



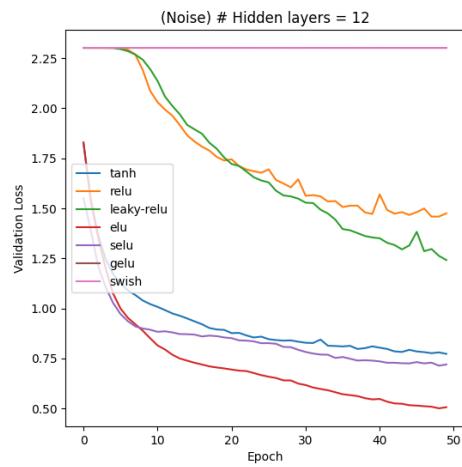
(a) Giá trị chính xác tập kiểm định (11)



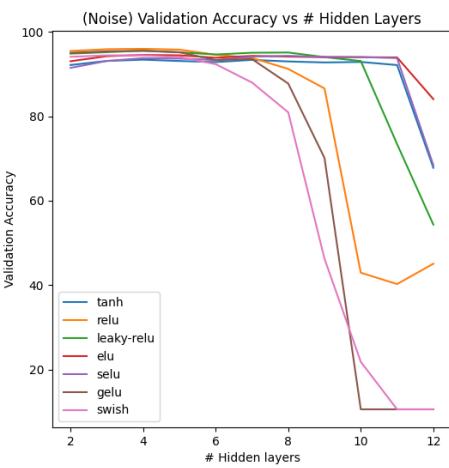
(b) Giá trị mất mát tập kiểm định (11)



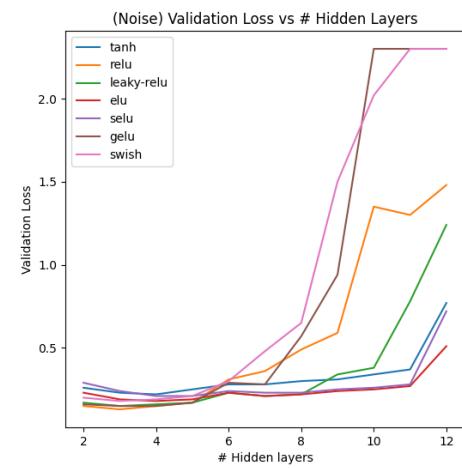
(c) Giá trị chính xác tập kiểm định (12)



(d) Giá trị mất mát tập kiểm định (12)



(e) Giá trị chính xác so với số tầng ẩn



(f) Giá trị mất mát so với số tầng ẩn

Hình 3.1.9: Giá trị chính xác và mất mát (nhiều Gauss + chuẩn hoá Lecun) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (nhiều Gauss + chuẩn hoá Lecun) trên tập kiểm định so với số tầng ẩn [(e), (f)].

3.1.3 Dữ liệu nhiễu và không sử dụng kỹ thuật khởi tạo trọng số

Quá trình thực nghiệm lần này gần như không thay đổi so với phần 3.1.2, chỉ có một thay đổi nhỏ đó là không khởi tạo trọng số bằng chuẩn Lecun mà sẽ để ngẫu nhiên. Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 3.1.10, 3.1.11, 3.1.12, 3.1.13. Chi tiết về giá trị được cho ở bảng 3.1.9 và bảng 3.1.10.

	2	3	4	5	6	7	8	9	10	11	12
tanh	92.47	93.17	93.57	93.38	92.98	93.27	93.03	93.04	92.73	91.93	84.08
\mathcal{R}	95.35	95.74	95.94	95.72	94.03	90.69	92.37	90.34	76.84	38.36	31.90
\mathcal{R}_L	94.67	95.42	95.20	95.46	94.93	95.20	95.07	94.90	93.56	89.71	73.43
\mathcal{E}	92.32	94.23	94.51	94.31	93.94	94.32	94.71	94.35	94.05	93.95	86.58
\mathcal{E}_S	92.35	93.87	94.08	93.98	93.86	94.18	94.42	94.65	93.86	84.70	82.53
\mathcal{G}	95.03	95.45	95.48	95.47	93.61	94.07	93.18	90.57	83.99	46.11	10.60
\mathcal{S}	94.19	94.35	94.61	94.35	93.05	92.55	88.78	88.49	71.93	10.60	10.60

Bảng 3.1.9: Giá trị chính xác (%) của các hàm kích hoạt (nhiễu) tương ứng với các số tầng ẩn của mô hình trên tập dữ liệu MNIST.

	2	3	4	5	6	7	8	9	10	11	12
tanh	0.26	0.23	0.22	0.24	0.29	0.29	0.30	0.31	0.34	0.38	0.52
\mathcal{R}	0.15	0.14	0.15	0.18	0.33	0.42	0.48	0.55	0.87	1.43	1.5
\mathcal{R}_L	0.18	0.15	0.16	0.16	0.21	0.20	0.23	0.23	0.36	0.54	0.83
\mathcal{E}	0.23	0.19	0.18	0.19	0.22	0.21	0.22	0.23	0.25	0.26	0.46
\mathcal{E}_S	0.27	0.21	0.20	0.21	0.23	0.22	0.23	0.24	0.27	0.41	0.51
\mathcal{G}	0.16	0.15	0.15	0.16	0.27	0.24	0.31	0.40	0.58	1.53	2.30
\mathcal{S}	0.20	0.18	0.18	0.19	0.26	0.29	0.45	0.47	0.95	2.30	2.30

Bảng 3.1.10: Giá trị mất mát của các hàm kích hoạt (nhiễu) tương ứng với các mô hình trên tập dữ liệu MNIST.

	2	3	4	5	6	7	8	9	10	11	12	μ
tanh	10	19	21	23	19	22	19	24	27	32	32	23
\mathcal{R}	30	27	25	24	32	34	36	40	33	19	16	29
\mathcal{R}_L	27	30	26	29	28	25	30	27	35	36	40	30
\mathcal{E}	19	27	27	26	27	24	22	25	28	31	21	25
\mathcal{E}_S	6	25	21	21	22	22	22	24	24	22	29	22
\mathcal{G}	32	30	31	30	38	37	36	40	43	44	1	X
\mathcal{S}	28	31	31	33	37	37	39	43	44	1	1	X

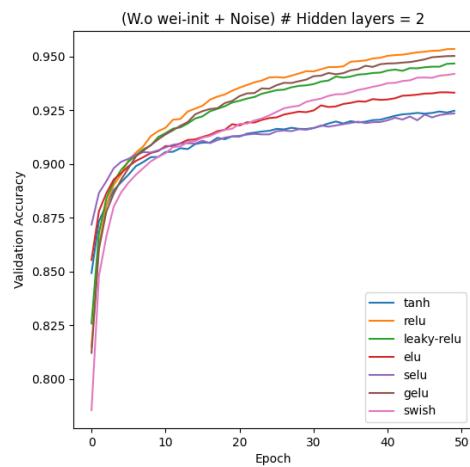
Bảng 3.1.11: Số epoch cần thiết để 80% giá trị mất mát (nhiễu) bằng giá trị mất mát cuối cùng.

Nhìn chung, kết quả này không quá khác biệt so với phần 3.1.2 (hình 3.1.13(e), 3.1.13(f)). Những mô hình đầu tiên luôn là RELU, sau đến Leaky và cuối cùng là ELU. Trong nhóm những hàm có kết quả chính xác cao nhất ngoài ELU còn có sự góp mặt của SELU và Tanh. GELU nhìn chung cho kết quả khá tương đương với RELU ở những tầng đầu tiên giống như phần trước, và lần này duy trì được điều này lâu hơn cả RELU khi từ số lượng tầng ẩn là 5 thì mất mát của GELU luôn thấp hơn RELU. Cho tới khi số tầng ẩn là 11 và 12 thì lúc này mất mát của GELU mới tăng đột biến và cao hơn của RELU.

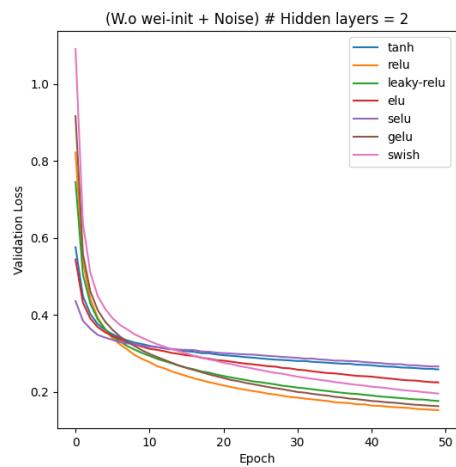
	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{chính xác}} (\%)$	92.15	81.57	82.50	93.48	92.04	81.23	75.77
$\sigma^2_{\text{chính xác}} (\%)$	0.0052	0.0522	0.0159	0.0008	0.0053	0.1321	0.1188
$\mu_{\text{mất mát}}$	0.31	0.56	0.29	0.24	0.27	0.57	0.71
$\sigma^2_{\text{mất mát}}$	0.0064	0.2252	0.0401	0.0055	0.0087	0.4447	0.6111

Bảng 3.1.12: Giá trị trung bình và phương sai của các hàm kích hoạt (nhiều) trong 11 mô hình thử nghiệm.

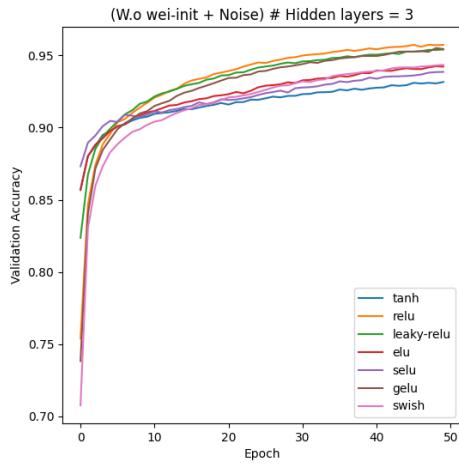
Ngoài hàm GELU khi cải thiện $\mu_{\text{chính xác}}$ của mình từ 68.87% lên 81.23% (tăng 12.36%) và $\mu_{\text{mất mát}}$ giảm từ 0.88 xuống 0.57 (giảm 0.31). Swish cũng là một cái tên có sự cải thiện rõ rệt khi cũng tăng $\mu_{\text{chính xác}} = 66.14\%$ lên 75.77% (tăng 9.23%) và cũng giảm $\mu_{\text{mất mát}} = 0.94$ xuống 0.71 (giảm 0.23). Khả năng học của GELU và Swish tuy không tốt ở những mạng sâu hơn 10 tầng ẩn nhưng vẫn là có khả năng học thay vì hoàn toàn không thể học được như lúc sử dụng chuẩn hoá Lecun để khởi tạo trọng số. Không chỉ GELU và Swish mà những hàm khác cũng có kết quả trung bình tích cực hơn khi so sánh bảng 3.1.12 với bảng 3.1.8. Điều này cho thấy, khởi tạo trọng số theo chuẩn hoá Lecun không phù hợp với các hàm kích hoạt khi huấn luyện trên tập MNIST.



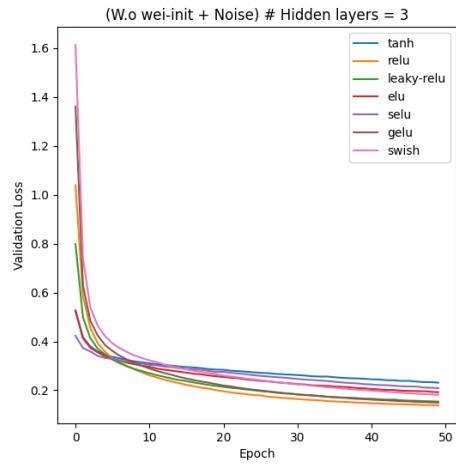
(a) Giá trị chính xác tập kiểm định (2)



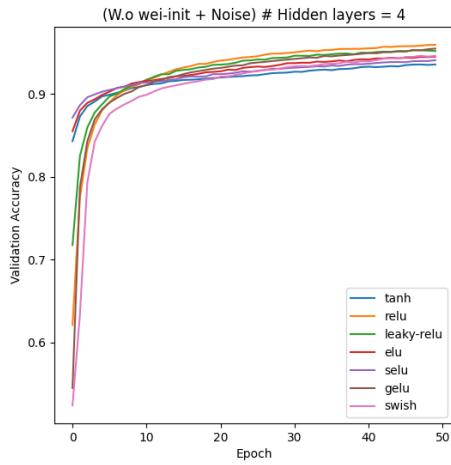
(b) Giá trị mất mát tập kiểm định (2)



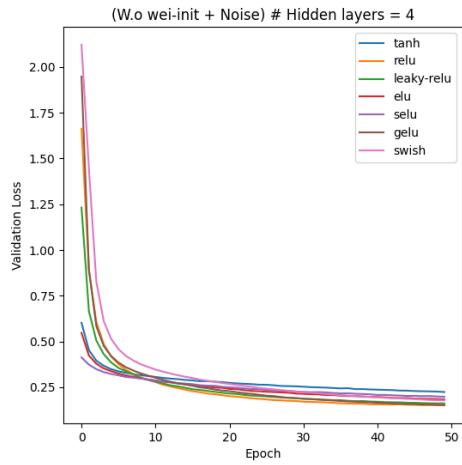
(c) Giá trị chính xác tập kiểm định (3)



(d) Giá trị mất mát tập kiểm định (3)

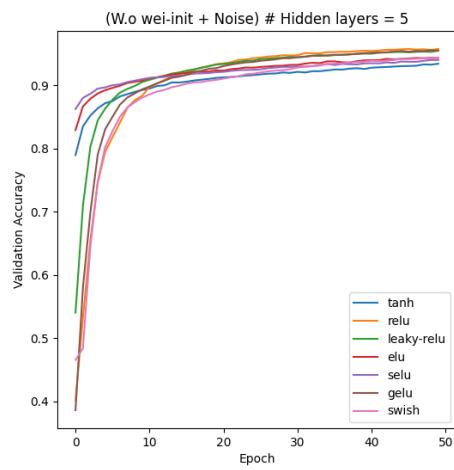


(e) Giá trị chính xác tập kiểm định (4)

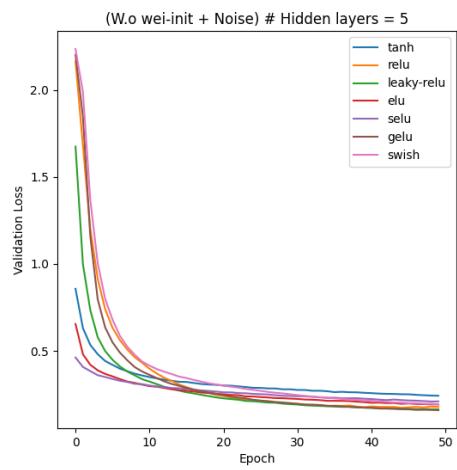


(f) Giá trị mất mát tập kiểm định (4)

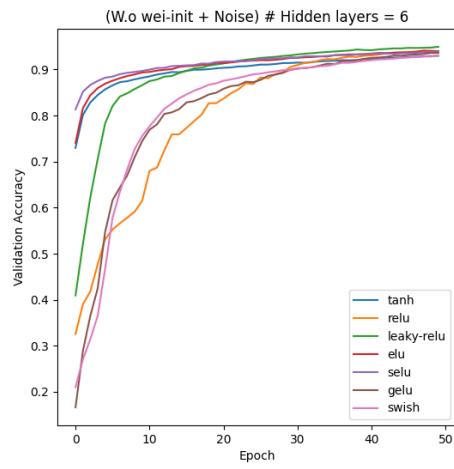
Hình 3.1.10: Giá trị chính xác và mất mát (nhiều) trên tập kiểm định (số tầng ẩn: 2, 3, 4).



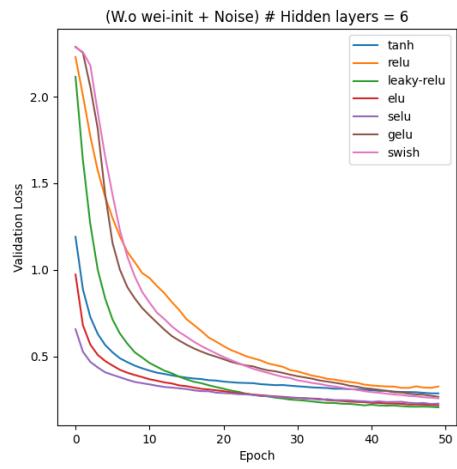
(a) Giá trị chính xác tập kiểm định (5)



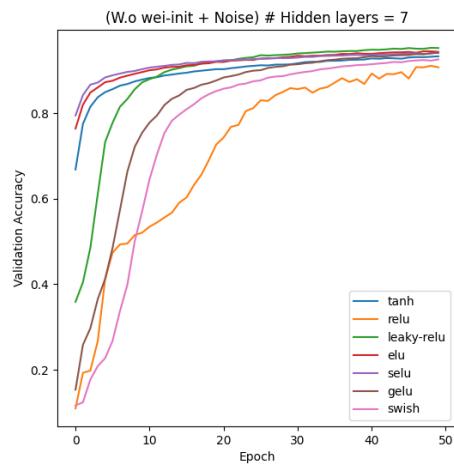
(b) Giá trị mất mát tập kiểm định (5)



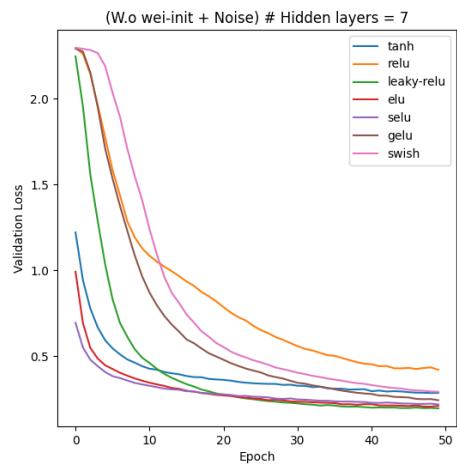
(c) Giá trị chính xác tập kiểm định (6)



(d) Giá trị mất mát tập kiểm định (6)

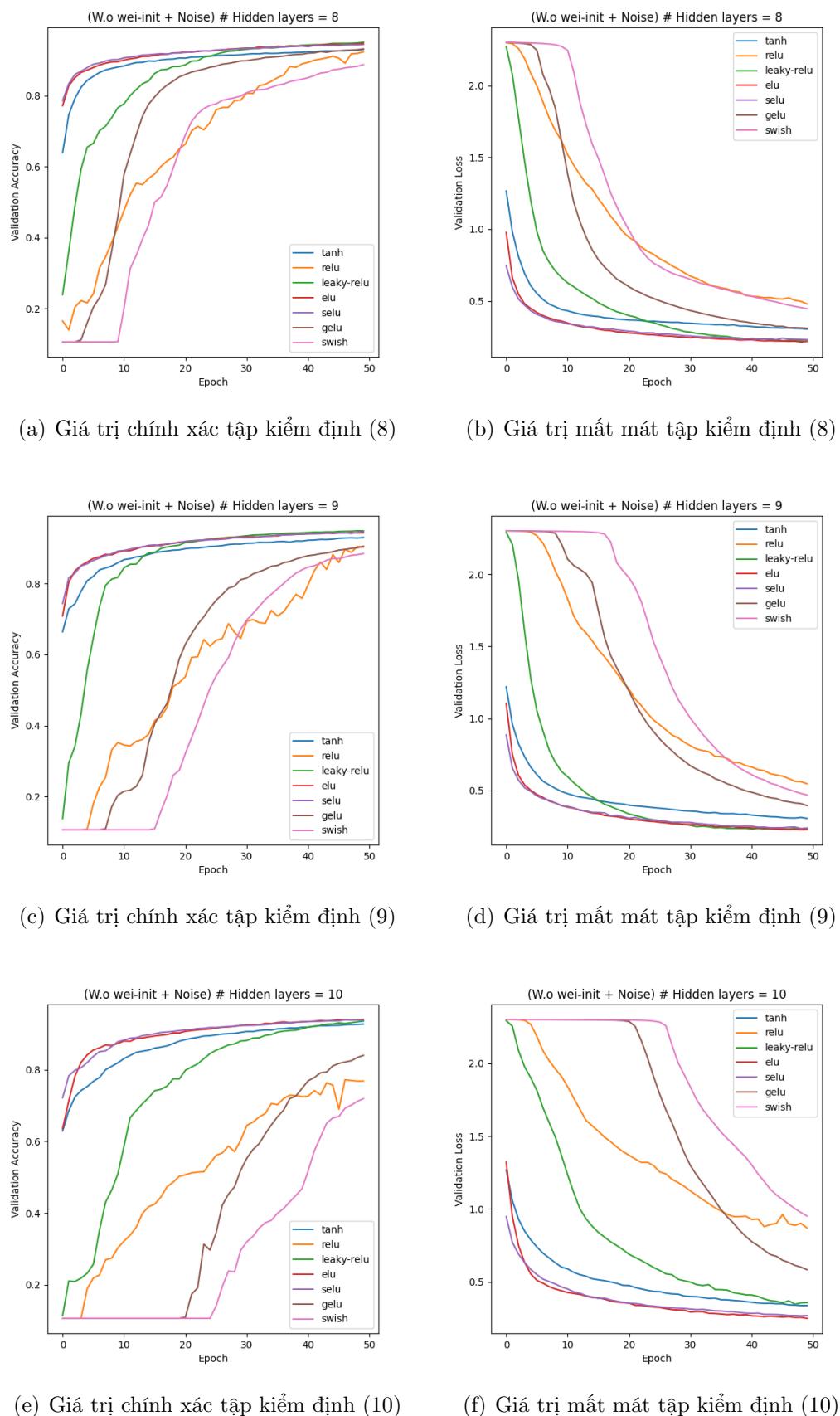


(e) Giá trị chính xác tập kiểm định (7)

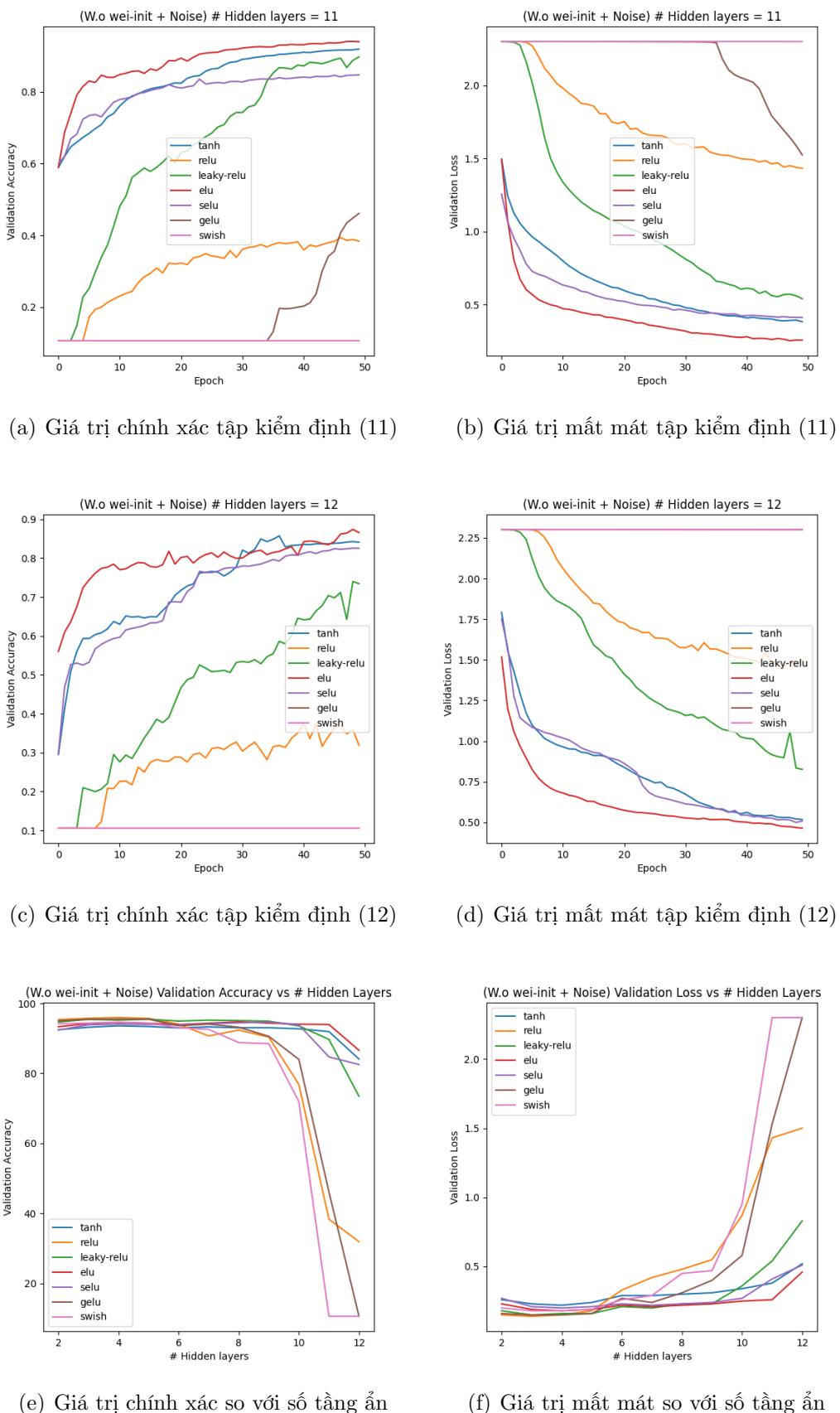


(f) Giá trị mất mát tập kiểm định (7)

Hình 3.1.11: Giá trị chính xác và mất mát (nhiều) trên tập kiểm định (số tầng ẩn: 5, 6, 7).



Hình 3.1.12: Giá trị chính xác và mất mát (nhiều) trên tập kiểm định (số tầng ẩn: 8, 9, 10).



Hình 3.1.13: Giá trị chính xác và mất mát (nhiều) trên tập kiểm định (số tầng ẩn: 11, 12) [(a) - (d)]; Giá trị chính xác và mất mát (nhiều) trên tập kiểm định so với số tầng ẩn [(e), (f)].

3.1.4 Tiêu kết thực nghiệm mạng nơ-ron sâu trên tập dữ liệu MNIST

	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{chính xác}} (\%)$	91.93	81.55	91.47	93.60	78.00	73.65	69.86
$\sigma_{\text{chính xác}}^2 (\%)$	0.0028	0.0495	0.0098	0.0012	0.0407	0.1130	0.1167
$\mu_{\text{mất mát}}$	0.30	0.53	0.30	0.23	1.17	0.75	0.85
$\sigma_{\text{mất mát}}^2$	0.0137	0.2228	0.0739	0.0065	1.8749	0.7062	0.7232

Bảng 3.1.13: Giá trị trung bình và phương sai của các hàm kích hoạt trung bình sau 3 lần thử nghiệm với 11 mô hình khác nhau.

Sau khi thử nghiệm mô hình mạng nơ-ron sâu với các chiều sâu khác nhau, tập dữ liệu được làm nhiễu nhờ hàm Gauss, và chọn hay không chọn khởi tạo trọng số theo chuẩn hoá Lecun; ta nhận thấy rằng hàm Tanh là một hàm hội tụ rất nhanh với điểm cực tiểu địa phương của mình. Tuy đây là một yếu điểm nhưng là một lợi thế khi với những mạng có nhiều tầng ẩn thì hàm Tanh vẫn rất nhanh chóng tìm được hổ sâu cực tiểu và duy trì được điều xuyên suốt 11 mô hình. Giá trị $\mu_{\text{chính xác}}$ lên tới 91.93% và $\mu_{\text{mất mát}}$ chỉ với 0.30, cộng thêm việc phương sai σ^2 chính xác và mất mát nhỏ (chỉ cao hơn so với ELU) cho thấy sự ổn định của hàm Tanh khi sử dụng trên tập dữ liệu MNIST. Hàm RELU và Leaky là 2 hàm với những mạng không quá sâu thì cho kết quả rất tốt và là tốt nhất. Với những mô hình sâu hơn thì RELU mới cho thấy sự thua thiệt của mình, duy chỉ có Leaky vẫn có thể hoạt động ở những mạng có số lượng tầng ẩn không quá nhiều. Điều này lí giải cho việc giá trị $\mu_{\text{chính xác}}$ của Leaky là 91.47%, cao hơn so với RELU là 81.55%. Ba hàm SELU, GELU và Swish đều không có kết quả khả quan ở lần đầu thử nghiệm. Hàm SELU ngay sau khi bỏ đi alpha dropout đã cho kết quả ẩn tượng - sát nút với ELU, có thể kỹ thuật alpha dropout là một kỹ thuật không phù hợp để sử dụng cho SELU trong những mạng nơ-ron sâu. Tuy chỉ có $\mu_{\text{chính xác}} = 78.00\%$, nhưng nếu chỉ tính 2 lần (phần 3.1.2 và phần 3.1.3) thì $\mu_{\text{chính xác}} = 91.66\%$, cao hơn cả Leaky. GELU và Swish ban đầu cũng cho những kết quả ẩn tượng, đặc biệt là GELU khi có kết quả khá tương đồng với RELU. Nhưng GELU và Swish nổi trội hơn RELU khi có thể học được sâu hơn so với RELU. ELU là hàm cho kết quả tốt trong mọi điều kiện từ những mạng nông cho tới mạng sâu, sử dụng khởi tạo trọng số theo chuẩn hoá hay không. Với $\mu_{\text{chính xác}} = 93.60\%$ và $\mu_{\text{mất mát}}$ chỉ với 0.23 đã cho thấy hàm này đáng được chú ý hơn để sử dụng cho các mạng nơ-ron sâu khi mà RELU và Leaky cho hiệu xuất thấp đi.

3.2 Thực nghiệm thông qua mạng tích chập sâu

Lần thử nghiệm bao gồm 5 mô hình có số lớp tích chập tăng từ 1 tới 24. Cụ thể, kiến trúc những mô hình được thử nghiệm như sau:

- 1 tầng tích chập: $32C3 \rightarrow P2$.
- 6 tầng tích chập: $32C3(\times 2) \rightarrow P2 \rightarrow 64C3(\times 2) \rightarrow P2 \rightarrow 128C3(\times 2) \rightarrow P2$.
- 12 tầng tích chập: $32C3(\times 4) \rightarrow P2 \rightarrow 64C3(\times 4) \rightarrow P2 \rightarrow 128C3(\times 4) \rightarrow P2$.
- 18 tầng tích chập: $32C3(\times 6) \rightarrow P2 \rightarrow 64C3(\times 6) \rightarrow P2 \rightarrow 128C3(\times 6) \rightarrow P2$.

- 24 tầng tích chập: $32C3(\times 8) \rightarrow P2 \rightarrow 64C3(\times 8) \rightarrow P2 \rightarrow 128C3(\times 8) \rightarrow P2$.

Trong đó, nCk là một lớp tích chập gồm n bộ lọc với kích thước $k \times k$. Thành phần $P2$ là một lớp gộp cực đại với kích thước gộp là 2×2 và sải bước bằng với kích thước gộp. Lớp gộp này có thể hiểu đơn giản là một lớp giảm độ phân giải của đầu vào đi 2 lần, bằng cách loại bỏ đi những giá trị không phải cực đại ở mỗi kích thước 2×2 . Sau khi qua tất cả các lớp tích chập, ta làm phẳng để thu được một véc-tơ nhúng và đưa vào một lớp kết nối dày đặc gồm 1 tầng ẩn có 128 đơn vị ẩn. Sau khi qua tầng ẩn này sẽ tới tầng đầu ra gồm 10 đơn vị ẩn, tương tự với 10 lớp giống như lúc thực nghiệm mạng nơ-ron sâu.

Tập dữ liệu để thực nghiệm và khởi tạo trọng số tương tự như phần 3.1.3, đó là khi thêm nhiễu Gauss có độ lệch chuẩn $\sigma = 0.2$ vào dữ liệu gốc và không sử dụng kỹ thuật khởi tạo trọng số. Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 3.2.1, 3.2.2. Chi tiết về giá trị được cho ở bảng 3.2.1 và bảng 3.2.2. Trong quá trình huấn luyện, các hàm SELU, GELU, Swish từ mô hình có 12 lớp tích chập trở đi đều không có khả năng học (hình 3.2.1(e), 3.2.1(f), 3.2.2(a), 3.2.2(b), 3.2.2(c), 3.2.2(d)). Riêng với hàm SELU cũng đã được thử qua việc dùng alpha dropout và khởi tạo trọng số theo chuẩn hoá Lecun nhưng kết quả vẫn là không thay đổi. Nên để cho ngắn gọn và cô đọng, giá trị các hàm có kết quả tệ không được thống kê trong các bảng.

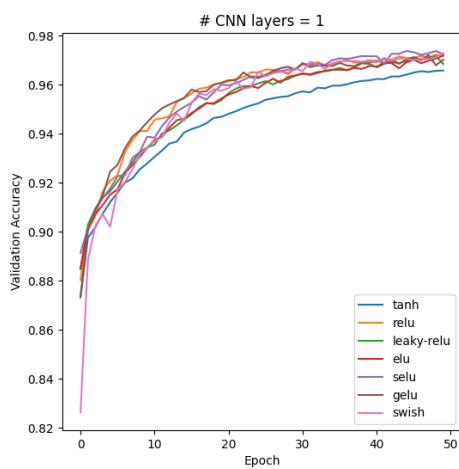
	1	6	12	18	24	μ
tanh	95.58	98.58	98.97	99.05	98.89	98.41
\mathcal{R}	97.18	98.44	98.37	98.24	98.47	98.14
\mathcal{R}_L	96.85	98.49	98.26	98.30	98.65	98.11
\mathcal{E}	97.20	98.27	98.64	98.45	98.77	98.27

Bảng 3.2.1: Giá trị chính xác (%) của các hàm kích hoạt tương ứng với các mô hình mạng tích chập với số lớp tích chập khác nhau trên tập dữ liệu MNIST.

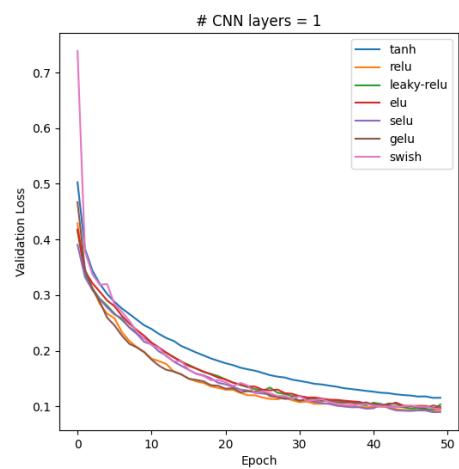
	1	6	12	18	24	μ
tanh	0.12	0.04	0.04	0.03	0.04	0.05
\mathcal{R}	0.09	0.08	0.13	0.15	0.12	0.11
\mathcal{R}_L	0.10	0.08	0.14	0.14	0.11	0.11
\mathcal{E}	0.10	0.08	0.07	0.10	0.09	0.09

Bảng 3.2.2: Giá trị mất mát của các hàm kích hoạt tương ứng với các mô hình mạng tích chập với số lớp tích chập khác nhau trên tập dữ liệu MNIST.

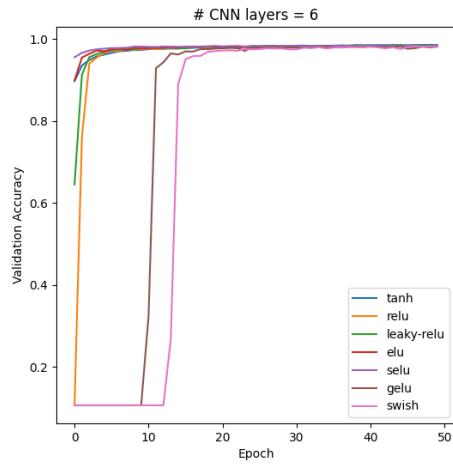
Hiệu quả của tích chập là thực sự lớn khi có giá trị chính xác cho những hàm học được rất cao khi giá trị chính xác của các hàm luôn gần 100%. Cao nhất là hàm Tanh khi đạt giá trị chính xác trung bình sau 5 mô hình lên tới 98.41%. Khoảng cách giá trị chính xác của Tanh cũng không quá chênh lệch so với những hàm khác, khi 4 hàm còn lại cũng đạt giá trị chính xác trung bình trên 98.00%. Về giá trị mất mát, giá trị cao nhất có được là 0.15, nhỏ hơn so với kết quả tốt nhất khi không có nhiễu là 0.19 (bảng 3.1.4). Hàm Tanh cũng là hàm có giá trị mất mát tốt nhất khi mất mát trung bình là 0.05. Có thể với mô hình tích chập như thế này, các hàm mất mát chưa có những hố cực tiểu quá nông so với các hố cực tiểu khác.



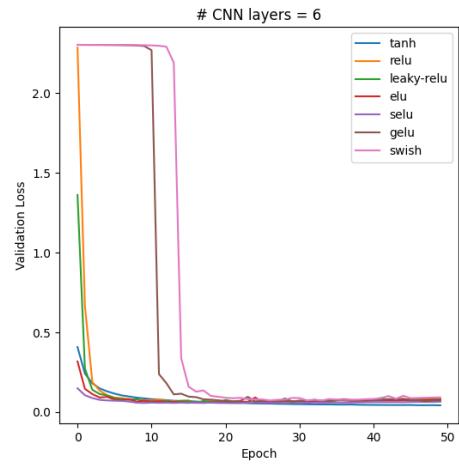
(a) Giá trị chính xác tập kiểm định (1)



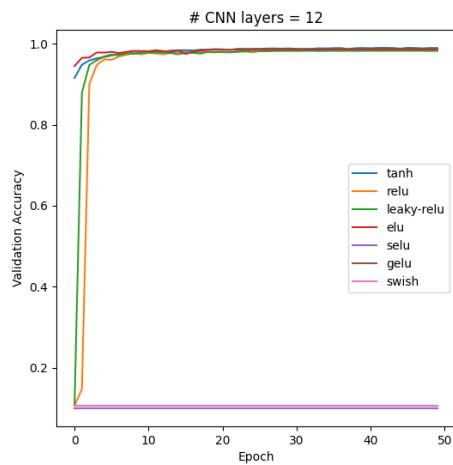
(b) Giá trị mất mát tập kiểm định (1)



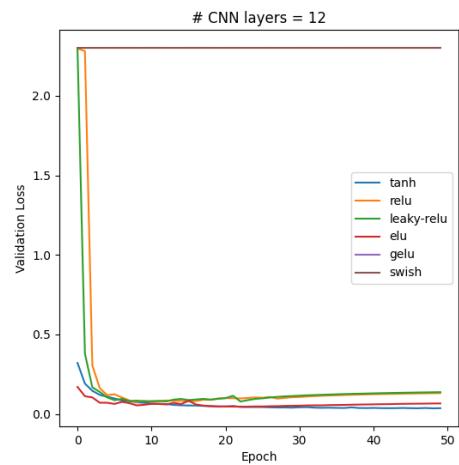
(c) Giá trị chính xác tập kiểm định (6)



(d) Giá trị mất mát tập kiểm định (6)

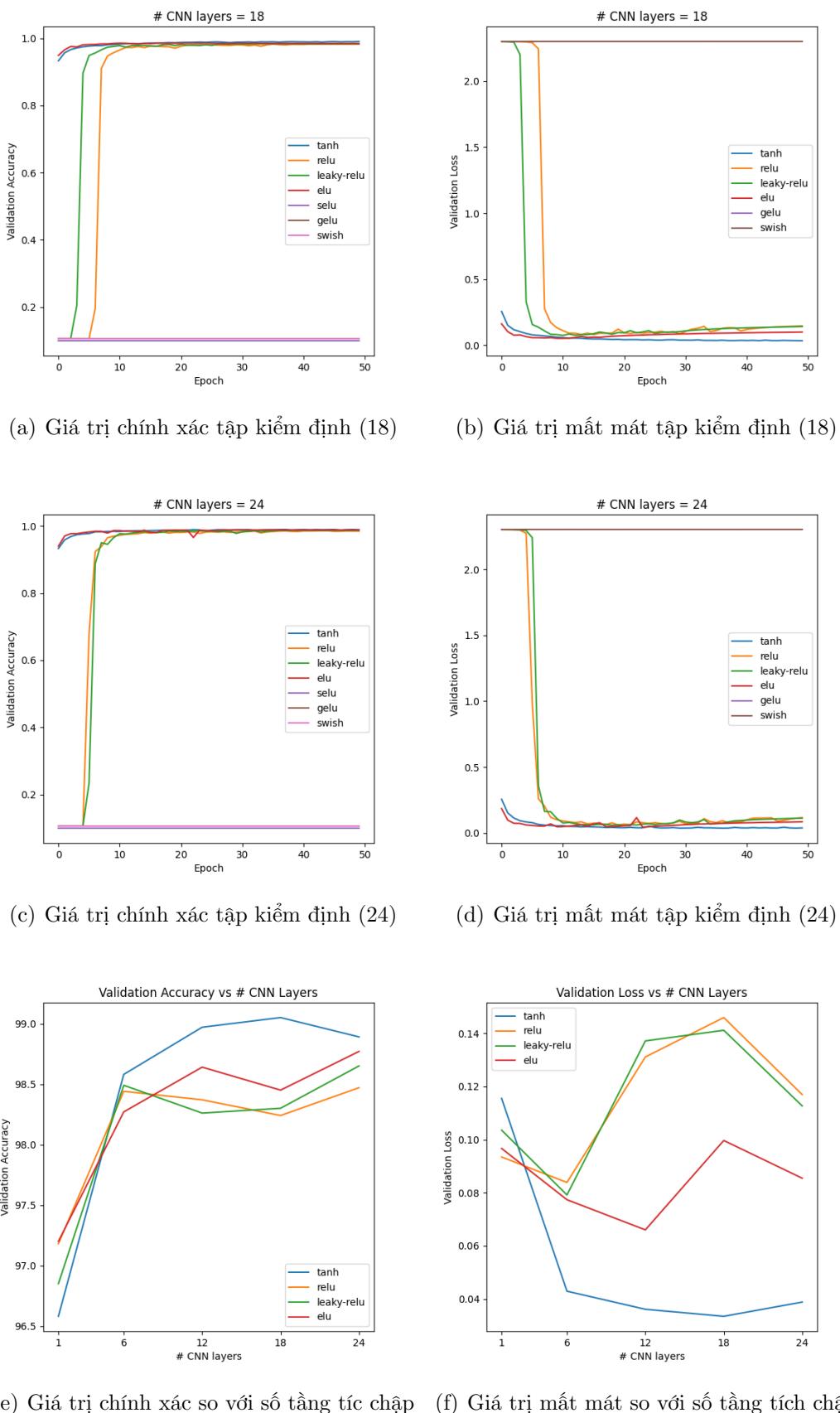


(e) Giá trị chính xác tập kiểm định (12)



(f) Giá trị mất mát tập kiểm định (12)

Hình 3.2.1: Giá trị chính xác và mất mát trên tập kiểm định (số tầng tích chập: 1, 6, 12).



Hình 3.2.2: Giá trị chính xác và mất mát trên tập kiểm định (số tầng tích chập: 18, 24) [(a) - (d)]; Giá trị chính xác và mất mát trên tập kiểm định so với số tầng tích chập [(e), (f)].

Chương 4

Thực nghiệm so sánh trên tập dữ liệu CIFAR

Sau khi thử nghiệm việc sử dụng các hàm kích hoạt khác nhau trên tập dữ liệu MNIST, ta vẫn chưa có được một sự chênh lệch lớn nào để có thể dựa vào đó, lựa chọn một hàm kích hoạt phù hợp với các kiến trúc tích chập. Điều đó khiến ta cần phải thực nghiệm trên một tập dữ liệu khó hơn và với những kiến trúc tích chập khoa học hơn. Vì thế, tập dữ liệu CIFAR¹ được đưa ra huấn luyện và hai tập dữ liệu CIFAR-10 và CIFAR-100 đều sẽ được sử dụng. Cả 2 tập dữ liệu đều có 50,000 dữ liệu huấn luyện và 10,000 dữ liệu kiểm tra. Với 50,000 dữ liệu huấn luyện, ta sẽ tách ra ngẫu nhiên 10,000 cho phần dữ liệu kiểm định (tỷ lệ huấn luyện/kiểm định là 8-2). Tất cả các dữ liệu được làm giàu bằng cách lật đối xứng trực Oy và xoay ảnh một góc nằm trong khoảng $[-2, 2]$ độ.

Trong phần thử nghiệm này, những kiến trúc được sử dụng là những kiến trúc tích chập hiện đại bao gồm: AlexNet [8, 9], SimpleNet [5, 1] và cuối cùng là VGG16 [16, 12]. Tất cả các tầng tích chập, chuẩn hoá batch, dropout đều sẽ được giữ nguyên theo đúng như kiến trúc gốc. Chỉ có phần hàm kích hoạt là sẽ thay đổi tất cả thành hàm kích hoạt đang cần thử nghiệm. Phần kết nối đầy đủ ở cuối sẽ được điều chỉnh sao cho phù hợp với các lớp đầu ra (10 với CIFAR-10 và 100 với CIFAR-100). Những lớp dày của phần kết nối đầy đủ sẽ sử dụng các loại hàm kích hoạt giống như phần tích chập phía trước của mạng, và sẽ chỉ có mỗi tầng đầu ra là sử dụng hàm kích hoạt softmax để lấy xác suất.

4.1 Thực nghiệm các kiến trúc tích chập trên tập dữ liệu CIFAR-10

Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 4.1.1 và chi tiết về giá trị được cho ở bảng 4.1.1.

Về giá trị mất mát trên tập kiểm định, kiến trúc AlexNet cho giá trị giảm dần lượt từ Tanh (1.72) cho với Swish (1.32). Dưới ngay Swish là GELU với giá trị mất mát là 1.33. Còn SimpleNet và VGG16 lại nhận được kết quả thấp nhất của ELU (0.47) và Leaky (2.22). Những giá trị mất mát phản ánh tốt giá trị xác thực trên tập kiểm định và kiểm tra. Cụ thể, GELU đạt giá trị chính xác cao nhất mạng Alexnet với 68.28% trong tập kiểm định và 68.27 trong tập kiểm tra, ngay sát phía sau là Swish với tương ứng 68.01% và 68.69%. ELU và Leaky cho kết quả khá sát nút nhau và nổi trội hơn so với những hàm còn lại. Với mạng SimpleNet thì ELU đạt giá trị tốt nhất với chính xác 84.17% kiểm định và 83.48% kiểm tra. Về phần Leaky

¹The CIFAR-10 and CIFAR-100 dataset, đường dẫn: <https://www.cs.toronto.edu/~kriz/cifar.html>

	Mất mát kiểm định			Chính xác kiểm định (%)			Chính xác kiểm tra (%)		
	Alex	Simple	VGG16	Alex	Simple	VGG16	Alex	Simple	VGG16
tanh	1.72	0.80	3.92	62.64	73.21	10.19	62.24	72.00	10.02
\mathcal{R}	1.53	0.55	2.35	64.98	80.94	79.37	65.01	80.97	79.77
\mathcal{R}_L	1.44	0.49	2.22	64.66	83.30	81.67	64.53	83.29	81.92
\mathcal{E}	1.42	0.47	2.62	64.62	84.17	74.64	65.09	83.48	73.02
\mathcal{E}_S	1.37	0.59	2.43	66.47	80.91	77.21	65.99	79.52	76.49
\mathcal{G}	1.33	0.58	2.31	68.28	80.86	79.97	68.27	80.46	79.23
\mathcal{S}	1.32	0.83	2.33	68.01	73.94	80.11	67.69	74.18	80.31

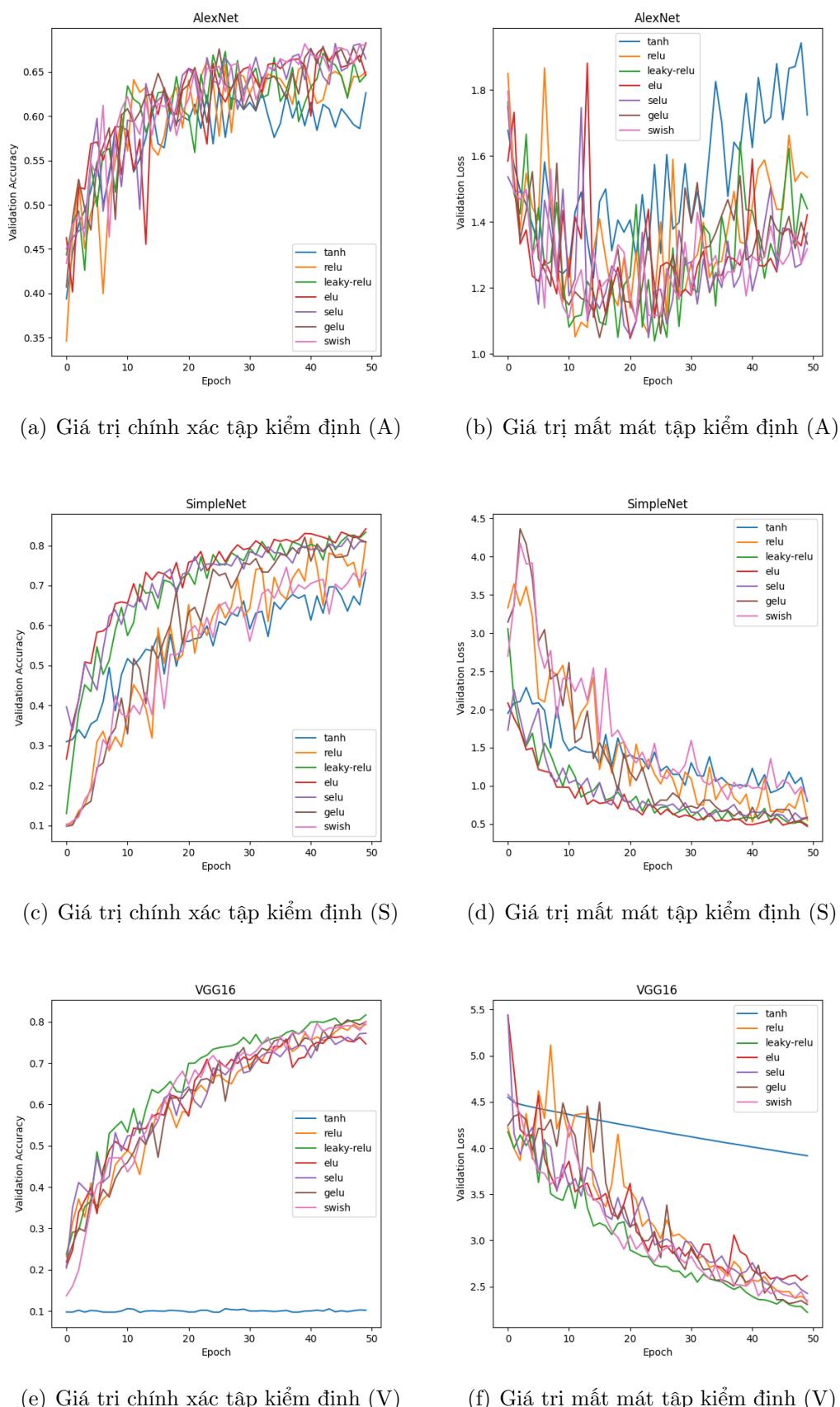
Bảng 4.1.1: Giá trị chính xác (%) và mất mát của các hàm kích hoạt tương ứng với các kiến trúc mạng tích chập hiện đại trên tập dữ liệu CIFAR-10.

là 81.67% kiểm định và 81.92% kiểm tra trong mạng VGG16. Hàm Tanh lần này không hề cho kết quả tốt khi thực nghiệm trên tập MNIST, thay vào đó cho kết quả tệ nhất ở mọi giá trị trong cả 3 kiến trúc và đặc biệt kiến trúc VGG16 thì hàm Tanh mất khả năng học. Ngay cả trong một mạng chưa quá sâu như AlexNet, Tanh cũng có dấu hiệu quá khớp khi sau 20 epoch, mất mát trên tập kiểm tra có dấu hiệu đi lên (hình 4.1.1(b)). Việc tìm được hố cực tiểu nông của Tanh không còn phát huy so như lúc ở tập dữ liệu MNIST, điều này khiến cho việc dùng hàm Tanh để thử nghiệm không phải là một quyết định sáng suốt.

	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{mất mát}}$	2.15	1.48	1.38	1.50	1.46	1.41	1.49
$\mu_{\text{chính xác kiểm định}}(\%)$	48.68	75.10	76.54	74.48	74.86	76.37	74.02
$\mu_{\text{chính xác kiểm tra}}(\%)$	48.09	75.25	76.58	73.86	74.00	75.99	74.06

Bảng 4.1.2: Giá trị trung bình của các hàm kích hoạt trong 3 kiến trúc tích chập hiện đại trên tập dữ liệu CIFAR-10.

Không quá bất ngờ khi trung bình của hàm Tanh dưới 50% khi rõ ràng hàm này không có khả năng học (hình 4.1.1(e), 4.1.1(f)). Trung bình của Leaky hơn so với các hàm còn lại ở cả 3 đại lượng: mất mát, chính xác kiểm định và chính xác kiểm tra. Hàm có kết quả tốt nhì là GELU với chênh lệch 0.03 ở giá trị mất mát trung bình $\mu_{\text{mất mát}}$ và giá trị chính xác ở cả 2 tập kiểm tra và kiểm định không quá 0.60%. Phần thực nghiệm này RELU không còn mạnh mẽ hơn so với GELU như khi thử nghiệm trên tập MNIST. Ngoài RELU, hàm ELU cũng không còn được nổi trội, kết quả có được khá cân bằng với Swish khi cả hai đều có giá trị chính xác $\approx 94\%$.



Hình 4.1.1: Giá trị chính xác và mất mát trên tập kiểm định CIFAR-10 (AlexNet, SimpleNet, VGG16).

4.2 Thực nghiệm các kiến trúc tích chập trên tập dữ liệu CIFAR-100

		tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
Mất mát kiểm định	AlexNet	3.83	2.96	3.07	2.76	2.74	3.02	3.02
	SimpleNet	4.33	2.29	1.76	1.84	1.89	1.92	2.14
	VGG16	6.23	4.57	4.27	5.67	5.31	4.60	4.35
Top-1 kiểm định	AlexNet	29.14	31.99	31.58	38.30	38.16	35.14	34.19
	SimpleNet	12.92	40.17	51.92	50.91	51.86	48.58	44.54
	VGG16	1.02	33.27	37.02	26.87	27.93	34.95	34.87
Top-3 kiểm định	AlexNet	47.17	50.95	49.82	56.96	56.60	53.51	52.22
	SimpleNet	25.40	61.53	74.07	72.52	73.17	70.34	65.03
	VGG16	2.81	54.97	59.79	47.61	47.76	56.87	57.35
Top-5 kiểm định	AlexNet	56.08	60.04	59.32	65.43	64.58	61.95	61.26
	SimpleNet	34.46	70.97	81.55	80.70	81.17	78.61	73.85
	VGG16	4.62	65.67	70.09	57.42	57.79	66.95	67.98
Top-1 kiểm tra	AlexNet	29.45	32.68	30.85	38.39	37.46	35.42	34.87
	SimpleNet	12.80	42.25	51.92	51.25	51.77	48.88	44.43
	VGG16	1.06	33.42	37.51	26.25	27.68	35.47	34.96
Top-3 kiểm tra	AlexNet	47.70	51.49	49.46	57.38	56.64	53.71	52.89
	SimpleNet	25.16	64.09	72.57	72.81	72.11	70.09	64.55
	VGG16	3.00	55.48	60.04	45.42	46.69	57.33	57.10
Top-5 kiểm tra	AlexNet	57.04	60.68	59.27	65.12	64.57	62.71	61.49
	SimpleNet	34.10	73.00	80.90	50.55	80.44	78.45	73.15
	VGG16	4.99	65.45	70.58	55.53	56.64	57.12	67.70

Bảng 4.2.1: Giá trị mất mát và Top-{1, 3, 5} (%) trên tập kiểm định và kiểm tra CIFAR-100 (AlexNet, SimpleNet, VGG16).

Số lượng lớp của tập dữ liệu này là 100, khá lớn. Khi này đánh giá chất lượng ngoài giá trị chính xác, tức Top-1 thì ta cũng nên quan tâm đến Top-3 và Top-5 chính xác. Biểu đồ giá trị chính xác và mất mát trên tập kiểm định qua mỗi epoch được cho ở các hình 4.2.1, còn Top-3 và Top-5 chính xác trên tập kiểm định sẽ là hình 4.2.2. Giá trị chi tiết cho các kết quả trên cho ở bảng 4.2.1.

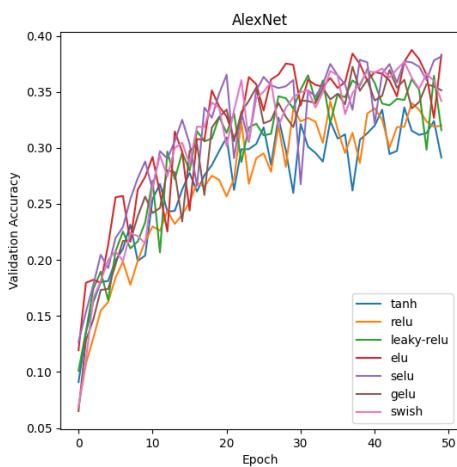
Hàm Tanh thêm một lần nữa khẳng định được rằng nó không phù hợp với các mạng tích chập hiện đại khi rất dễ bị quá khớp (hình 4.2.1(b)), lại không có khả năng học tốt với những mạng tích chập sâu (hình 4.2.1(c) - 4.2.1(f), 4.2.2(b) - 4.2.2(f)). Với những hàm khác thì việc

học diễn ra khá đúng lộ trình khi càng huấn luyện giá trị chính xác càng lên cao và giá trị mất mát càng giảm. Dựa vào bảng 4.2.1, ta thấy rằng các chỉ số đánh giá là sự nổi trội của các hàm Leaky và ELU. Leaky có 13/21 chỉ số đứng đầu, còn ELU là 7/21 chỉ số. Riêng giá trị mất mát thì SELU mới là hàm đạt giá trị nhỏ nhất. Chỉ có 3 hàm là Leaky, ELU và SELU mới có giá trị Top-5 kiểm định trên 80% trong mạng SimpleNet - mạng cho kết quả tốt nhất. Một điều khá bất ngờ là kết quả Top-5 kiểm tra của ELU lại khá tệ, khi chỉ có 50.55% (giảm 30.15%) trong khi 2 hàm vừa nêu vẫn tiếp tục 80% mặc dù đã có giảm sút (dưới 1%).

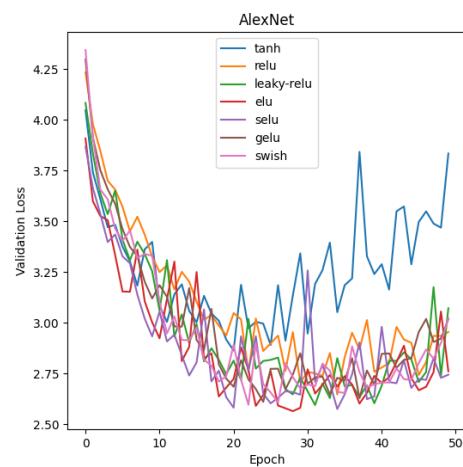
	tanh	\mathcal{R}	\mathcal{R}_L	\mathcal{E}	\mathcal{E}_S	\mathcal{G}	\mathcal{S}
$\mu_{\text{mất mát}}$	4.80	3.27	3.03	3.42	3.31	3.18	3.17
$\mu_{\text{Top-1 kiểm định}}(\%)$	14.36	35.14	40.17	38.69	39.32	39.56	37.87
$\mu_{\text{Top-3 kiểm định}}(\%)$	25.13	55.82	61.23	59.03	59.18	60.24	58.20
$\mu_{\text{Top-5 kiểm định}}(\%)$	31.72	65.56	70.32	67.85	67.85	69.10	67.70
$\mu_{\text{Top-1 kiểm tra}}(\%)$	14.44	36.12	40.09	38.63	38.97	39.92	38.09
$\mu_{\text{Top-3 kiểm tra}}(\%)$	25.29	57.02	60.69	58.54	58.48	60.38	58.18
$\mu_{\text{Top-5 kiểm tra}}(\%)$	32.04	66.38	70.25	67.07	67.22	69.43	67.45

Bảng 4.2.2: Giá trị trung bình của các hàm kích hoạt trong 3 kiến trúc tích chập hiện đại trên tập dữ liệu CIFAR-100.

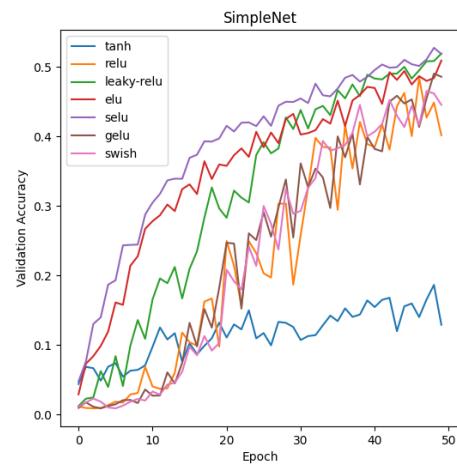
Trong bảng 4.2.2 cho ta thấy rõ ràng khi thực nghiệm trên tập dữ liệu CIFAR-100, Leaky vượt trội hơn khi so sánh với các hàm khác, khi mọi chỉ số đều đứng đầu chứ không chia sẻ bất kỳ vị trí nào. ELU và SELU cho kết quả khá tương tự nhau và cũng ở mức khá tốt, không thua quá nhiều Leaky như là RELU. Tuy không cùng họ ELU, nhưng Swish cũng cho kết quả cũng tương đồng với lại ELU và SELU. Nhưng nếu ta loại bỏ Leaky ra khỏi bảng thì GELU mới là hàm có kết quả vượt trội nhất khi tất cả các giá trị đều đứng đầu. Nhìn chung, với tập dữ liệu CIFAR-100 thì vẫn là Leaky và GELU cho kết quả ưng ý nhất ở các phương diện đánh giá.



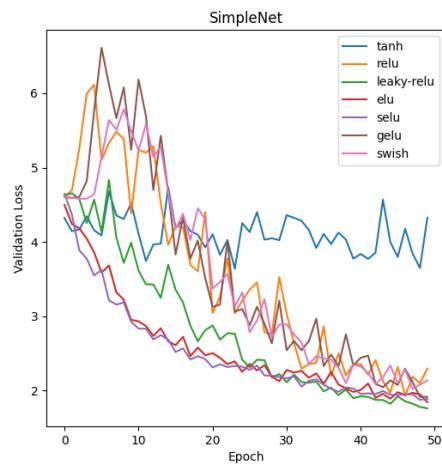
(a) Giá trị chính xác tập kiểm định (A)



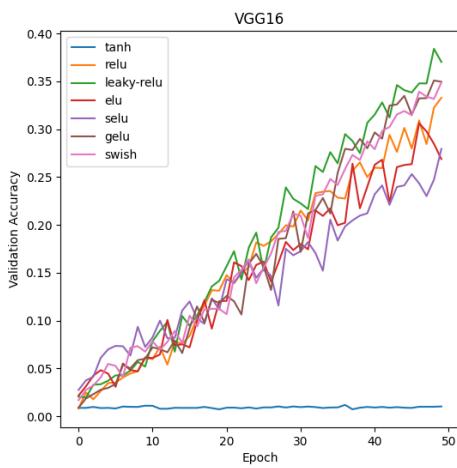
(b) Giá trị mất mát tập kiểm định (A)



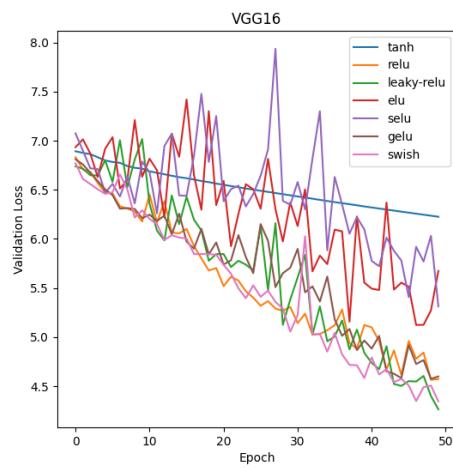
(c) Giá trị chính xác tập kiểm định (S)



(d) Giá trị mất mát tập kiểm định (S)

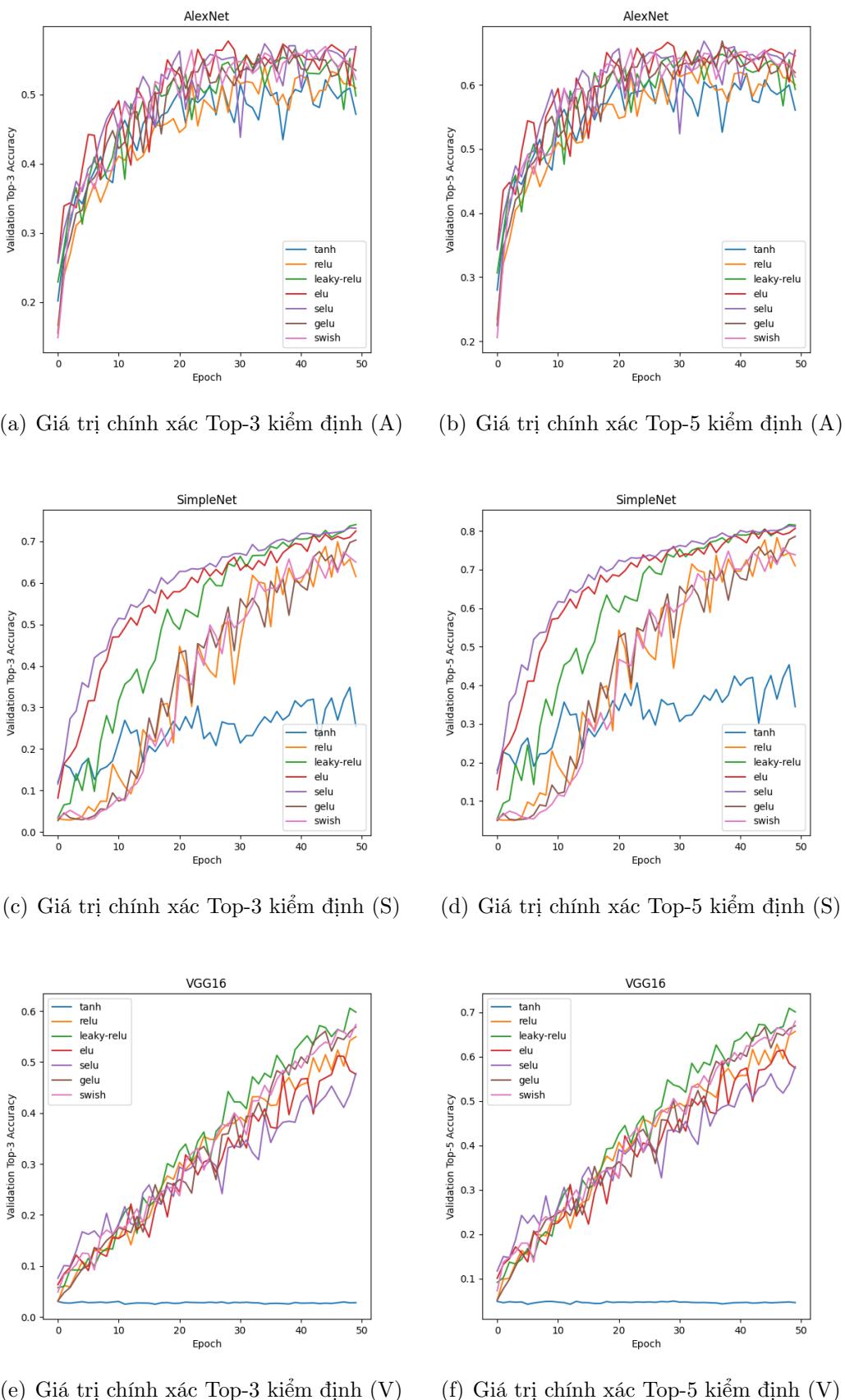


(e) Giá trị chính xác tập kiểm định (V)



(f) Giá trị mất mát tập kiểm định (V)

Hình 4.2.1: Giá trị chính xác và mất mát trên tập kiểm định CIFAR-100 (AlexNet, SimpleNet, VGG16).



Hình 4.2.2: Giá trị chính xác Top-{3, 5} trên tập kiểm định CIFAR-100 (AlexNet, SimpleNet, VGG16).

Chương 5

Tổng kết

Bài tập lớn này đã trình bày những kết quả sau khi thực nghiệm các hàm kích hoạt trên 3 tập dữ liệu MNIST, CIFAR-10 và CIFAR-100, từ những hàm xa xưa như Sigmoid và Tanh. Cho tới những hàm được đề xuất cách đây 10 năm đó lại như ReLU, Leaky ReLU, ELU, SELU, GELU và hàm mới được đề xuất gần đây là Swish. Dưới nhiều điều kiện khác nhau, chẳng hạn như thêm nhiễu vào dữ liệu, áp dụng nhiều kiến trúc học sâu kiểm mạng nơ-ron sâu đơn thuần, đến những mạng tích chập đơn giản với các độ sâu khác nhau và các kiến trúc tích chập hiện đại. Khi thử nghiệm mạng nơ-ron sâu đơn thuần với số lượng tầng ẩn khác nhau thì nhận thấy rằng: Hàm Tanh là một trong những hàm hội tụ khá nhanh về hố sâu cực tiểu và có thể giữ vững được kết quả dù cho tăng độ sâu lên. Hàm RELU và Leaky cho kết quả xuất sắc, tuy nhiên khi số tầng ẩn tăng lên thì kết quả bị giảm xuống rất nhiều. Riêng Leaky thì có thể học với những mạng sâu hơn so với RELU, dù vậy tốc độ hội tụ với những mạng sâu là không ẩn tượng. GELU và Swish có kết quả tương tự với RELU và Leaky tuy nhiên lại nổi trội hơn về tốc độ hội tụ. ELU là một hàm ổn định trong xuyên suốt các mô hình với các độ sâu khác nhau. Trong các mạng nơ-ron sâu cổ điển có nhiều tầng ẩn (khoảng 7 tầng trở lên) nên cần nhắc việc sử dụng ELU thay cho RELU hoặc Leaky để có thể học một cách tốt nhất. Một điều nữa đã được chứng minh là việc sử dụng kỹ thuật alpha dropout thực sự không đem lại kết quả tốt trên tập MNIST, và chuẩn hoá kiểu Lecun cũng phần nào giảm đi khả năng học của các hàm kích hoạt hiện giờ. Về kết quả thực nghiệm sử dụng các kiến trúc mạng tích chập hiện đại khác nhau: Sử dụng hàm kích hoạt là Tanh sẽ không giúp mô hình có khả năng học tốt thậm chí là không học được. Các hàm RELU, SELU và Swish nhìn chung có khả năng học tốt với kết quả khá khả quan. Ẩn tượng hơn một chút là hàm GELU khi cho kết quả tốt hơn nhóm hàm vừa đề cập. Nhưng tốt hơn cả chính là Leaky với sự vượt trội ở mọi chỉ số với những hàm được so sánh cùng.

Trong tương lai, cần có thêm những nghiên cứu để tối ưu hoá các tham số α với những kiến trúc khác nhau cho những hàm đã và đang hoạt động tốt như Leaky và ELU hay thậm chí là SELU, để có thể cho một kết quả khả quan hơn kết quả hiện tại. Thêm vào đó, rõ ràng cách khởi tạo trọng số nếu không cẩn thận cũng làm giảm hiệu quả của các hàm kích hoạt. Do đó cũng cần có phải thực nghiệm những cách khởi tạo trọng số với các hàm để chọn cho mình cách khởi tạo tốt nhất.

Tài liệu tham khảo

- [1] Eric Alcaide. *SimpleNet-Keras*. 2019. URL: <https://github.com/hypnopump/SimpleNet-Keras>.
- [2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2016. arXiv: [1511.07289 \[cs.LG\]](https://arxiv.org/abs/1511.07289).
- [3] dontloo. “Reply: Why are non zero-centered activation functions a problem in backpropagation?” In: <https://stats.stackexchange.com/> (2017). URL: <https://bit.ly/3rzaXLx>.
- [4] Casper Hansen. “Activation Functions Explained - GELU, SELU, ELU, ReLU and more”. In: <https://mlfromscratch.com/> (2019). URL: <https://mlfromscratch.com/activation-functions-explained/>.
- [5] Seyyed Hossein Hasanpour et al. *Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures*. 2018. arXiv: [1608.06037 \[cs.CV\]](https://arxiv.org/abs/1608.06037).
- [6] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2020. arXiv: [1606.08415 \[cs.LG\]](https://arxiv.org/abs/1606.08415).
- [7] Günter Klambauer et al. *Self-Normalizing Neural Networks*. 2017. arXiv: [1706.02515 \[cs.LG\]](https://arxiv.org/abs/1706.02515).
- [8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. 2012.
- [9] Dr. Vaibhav Kumar. “Hands-on Guide To Implementing AlexNet With Keras For Multi-Class Image Classification”. In: <https://analyticsindiamag.com/> (2020). URL: <https://bit.ly/3fciv8q>.
- [10] Lu Lu. “Dying ReLU and Initialization: Theory and Numerical Examples”. In: *Communications in Computational Physics* 28.5 (June 2020), pp. 1671–1706. ISSN: 1991-7120. DOI: [10.4208/cicp.oa-2020-0165](https://doi.org/10.4208/cicp.oa-2020-0165). URL: <http://dx.doi.org/10.4208/cicp.OA-2020-0165>.
- [11] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *Proc. ICML* 30 (2013).
- [12] Sachin Mohan. “Keras Implementation of VGG16 Architecture from Scratch with Dogs Vs Cat Data Set”. In: <https://machinelearningknowledge.ai/> (2020). URL: <https://bit.ly/3BW35Zc>.
- [13] Vinod Nair and Geoffrey E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proc. ICML* 30 (2010), pp. 807–814.
- [14] Andrew Ng. *Activation Function*. Coursera. URL: <https://bit.ly/3x6fWUZ>.
- [15] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: [1710.05941 \[cs.NE\]](https://arxiv.org/abs/1710.05941).

- [16] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: [1409.1556 \[cs.CV\]](https://arxiv.org/abs/1409.1556).
- [17] Vũ Hữu Tiệp. “Lược sử Deep Learning”. In: <https://machinelearningcoban.com/> (2018). URL: <https://machinelearningcoban.com/2018/06/22/deeplearning/>.
- [18] Vũ Hữu Tiệp. “Multi-layer Perceptron và Backpropagation”. In: <https://machinelearningcoban.com/> (2017). URL: <https://machinelearningcoban.com/2017/02/24/mlp/>.