Sử dụng học máy nghiên cứu mô hình XY 2 chiều tổng quát

Khóa luận tốt nghiệp

Nguyễn Bá Nghĩa

Ngày 28 tháng 12 năm 2019

Sơ lược đề tài

Học máy phát triển mạnh mẽ trong thập kỷ qua nhờ sự bùng nổ của dữ liệu lớn, học máy có tác động mạnh trong các lĩnh vực khoa học và công nghệ. Vật lý chất rắn không đứng ngoài xu hướng này. Tính từ 2015, số lượng của các bài báo xuất bản sử dụng học máy trong vật lý, cụ thể trong vật lý chất rắn, gia tăng đáng kể. Hiện tại có 1 lượng lớn ứng dụng học máy trong vật lý chất rắn: Phân loại pha cho mô hình spin cổ điển hoặc mô hình mạng lượng tử, tìm trạng thái cơ bản (ground state) của hệ lượng tử, tăng tốc mô phỏng Monte Carlo.

Trong đề tài này, chúng tôi nghiên cứu việc sử dụng học máy nghiên cứu mô hình XY 2 chiều tổng quát. Chúng tôi trình bày 2 vấn đề được của việc phân loại pha của mô hình xy 2 chiều tổng quát. Thứ nhất liệu học máy có phân loại pha tốt hơn các phương pháp khác không. Thứ hai xem xét tại sao nó lại phân loại như vậy, nếu hiệu suất nó tốt hơn thì nó đã dùng đặc trưng gì để phân loại và liệu đặc trưng đó có tương ứng với một đại lượng vật lý nào không.

Lời cảm ơn

Tôi xin chân thành cảm ơn Trường Đại học Khoa học Tự nhiên Hà Nội, viện nghiên cứu Phenikaa, Đại học Phenikaa đã cho phép tôi thực hiện đồ án tại trường Trường Đại học Khoa học Tự nhiên. Xin cảm ơn viện nghiên cứu Phenikaa và Đại học Phenikaa về sự hỗ trợ và giúp đỡ trong suốt quá trình tôi làm đồ án.

Tôi xin chân thành cảm ơn TS. Đặng Thế Hùng và các thầy bộ môn Tin vật lý Đại học KHTN đã hướng dẫn tôi hết sức tận tình và chu đáo về mặt chuyên môn để tôi có thể thực hiện và hoàn thành đồ án.

Tôi xin cảm ơn Ban Giám hiệu Trường Đại học Khoa học Tự nhiên và các thầy cô đã giúp đỡ và động viên tôi trong suốt quá trình nghiên cứu và học tập.

Tôi xin bày tỏ lòng biết ơn sâu sắc đến các thầy phản biện, các thầy trong hội đồng chấm đồ án đã đồng ý đọc duyệt và góp các ý kiến quý báu để tôi có thể hoàn chỉnh đồ án này và định hướng nghiên cứu trong tương lai.

Cuối cùng tôi xin gửi lời cảm ơn chân thành tới gia đình và bạn bè, những người đã động viên khuyến khích tôi trong suốt thời gian tôi tham giao nghiên cứu và thực hiện đồ án này.

Mục lục

Chương 1

Giới thiệu

1.1 Mô hình XY

Giải Nobel cho vật lý năm 2016 đã được trao cho David Thouless, Michael Kosterliz và Duncan Haldane cho những khám phá lý thuyết về sự chuyển pha topo và pha topo của vật chất [1]. Đó là chuyển pha được khám phá bởi 2 trong số họ và được gọi là chuyển pha Kosterliz-Thouless. Chuyển pha KT đặc biệt vì nó xảy ra trong hệ 2 chiều (trên các lớp vật chất mỏng hoặc bề mặt tinh thể) và có tính chất topo. Để hiểu được các tính chất, đặc điểm của nó, trước hết chúng ta sẽ xem 1 vài tính chất cơ bản của chuyển pha.

Chuyển pha thông thường là chuyển pha của sự tan của đá hoặc sự sôi của nó. Chuyển pha thông thường có tham số trật tự (order parameter), chuyển pha KT khác với chuyển pha thông thường , nó không có order parameter. Bên dưới chúng ta cùng xét 1 mô hình chuyển pha điển hình có order parameter là Ising model

1.1.1 Mô hình Ising và sự phá vỡ đối xứng tự phát

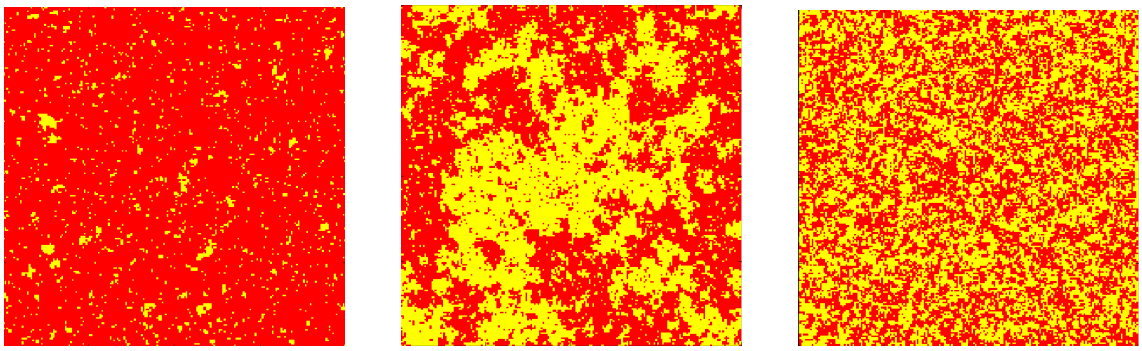
Xét 1 mô hình nhỏ của chuyển pha là mô hình Ising, được định nghĩa như sau. Xem xét 1 lưới bình thường, ví dụ 1 mạng vuông 2 chiều. Mỗi vị trí trên mạng, đặt 1 spin, spin này có thể chỉ lên hoặc xuống. Mỗi spin tương tác với 4 lân cận gần nhất (tương tác trên dưới trái phải). Chúng ta dùng $s = +1$ và $s = -1$ để biểu thị spin lên hay xuống tương ứng. Mỗi cặp của spin s và spin s' lân cận đóng góp cho năng lượng của hệ 1 lượng $-Jss'$, trong đó J là hằng số cặp và nó thường được chọn là số dương (J âm sẽ gây ra phản sắt từ thay vì sắt từ)

Trạng thái năng lượng thấp nhất của hệ trong mô hình Ising thu được bởi căn chỉnh tất cả spin lên +1 hoặc tất cả xuống -1. Lưu ý rằng : hệ có đối xứng như sau : năng lượng của hệ sẽ ko thay đổi nếu chúng ta đổi dấu toàn bộ spin ở bất kỳ cấu hình ngẫu nhiên nào. Ở nhiệt độ thấp, sự đối xứng này là vỡ tự phát: các spin chủ yếu dọc theo 1 hướng hoặc lên hoặc xuống (như trong sắt từ). Ở nhiệt độ cao, không có sự chiếm ưu thế của lên hoặc xuống tất cả spin và hệ biểu hiện như hệ thuận từ.

Hình 1.1 là 3 cấu hình điển hình của 200x200 Ising model thu được từ mô phỏng Monte Carlo ở $T < Tc$, $T \approx Tc$ và $T > Tc$ (từ trái qua phải)

Tóm lại, mô hình Ising có chuyển pha từ pha sắt từ nhiệt độ thấp (nơi đối xứng lên xuống là phá vỡ tự phát) đến pha thuận từ ở nhiệt độ cao. Do đó tồn tại một nhiệt độ chuyển pha Tc. Chuyển pha trong Ising model, và trong tất cả các hệ cơ

bản được được nghiên cứu trước đó, trước công trình của Kosterlitz và Thouless, biểu hiện qua sự phát vỡ đố xứng tự phát.

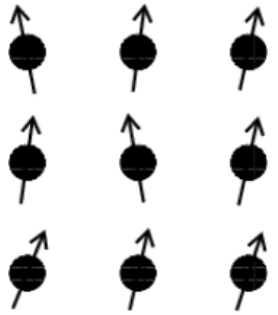


(a) $T < T_c$ (b) $T \approx T_c$ (c) $T > T_c$

Hình 1.1: Hình ảnh của ising model trước, gần và sau chuyển pha

1.1.2 Mô hình XY và sự rời ra của cặp vortex và anti-vortex

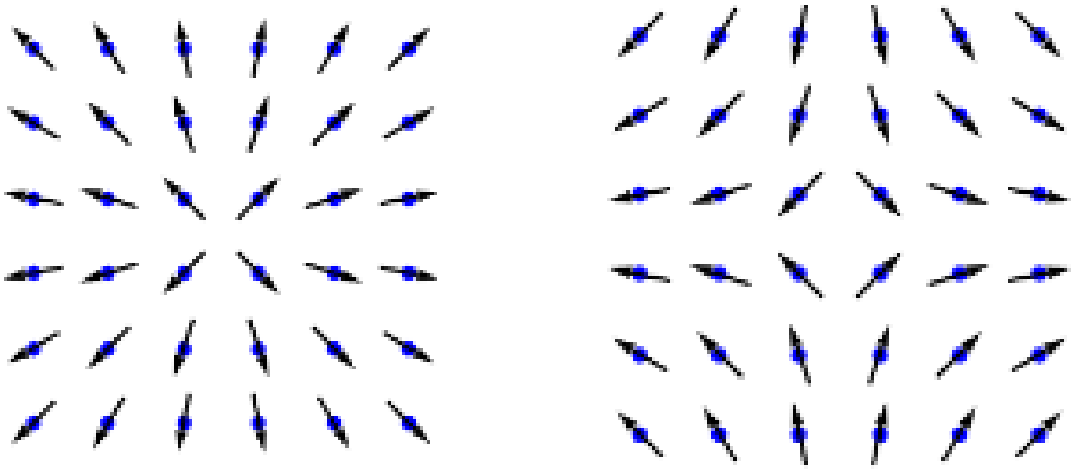
Với mô hình Ising, order parameter là độ từ hóa. Chuyển pha KT thì không có order paremeter. Mô hình XY là một dạng tổng quá hóa từ mô hình Ising, trong đó spin có thể chỉ các hướng ngẫu nhiên (Hình 1.2) ở trên mạng hình vuông phẳng trên một mặt phẳng, không chỉ có 2 hướng lên hoặc xuống. Sự tương tác giữa những spin lân cận bây giờ là tích vô hướng $Js^{\rightarrow}s'^{\rightarrow} = -Jcos(\delta)$, trong đó δ là góc được tạo thành giữa 2 spin và chúng ta giả sử rằng các spin có độ dài là 1 đơn vị. Khi xét trong không gian 2 chiều, chúng ta mong muốn nó cư xử như mô hình Ising chuẩn: 1 pha thuận từ nhiệt độ cao và 1 pha sắt từ nhiệt độ thấp. Nhưng điều này không xảy ra! . Có một định lý trong cơ học thống kê, định lý Mermin-Wagner nói rằng :"Sự đối xứng liên tục không thể là phá vỡ tự phát trong 2 chiều". Mô hình XY có đối xứng liên tục vì nếu chúng ta biểu diễn 1 quay toàn cục của 1 góc 0 của tất cả spin của 1 cấu hình spin bất kỳ, chúng ta không thay đổi năng lượng của hệ. Do đó định lý Mermin-Wagner loại trừ khả năng của pha sắt từ. Điều đó có nghĩa là mô hình XY không có chuyển pha ? Không hẳn vậy. Kosterlitz và Thouless chỉ ra rằng mô hình XY có chuyển pha topo mà không có sự xuất hiện của từ hóa tự phát.



Hình 1.2: Trong mạng vuông xy-model, mỗi spin có thể chỉ hướng bất kỳ

Để hiểu những gì đang xảy ra, chúng ta cần thảo luận hình thức cụ thể của các spin trong hệ : các xoáy (vortex). Hình 1.3 chỉ ra 1 vortex (bên trái) và 1 anti-vortex (bên phải). Các khuyết tật có cấu trúc topo này (topological defect) cần nhiều năng

lượng để hình thành . Tuy nhiên người ta có thể thấy rằng các gặp vortex-antivortex liên kết với nhau tốn chỉ 1 năng lượng hữu hạn. Ở nhiệt độ thấp các cặp vortex tách rời nhau ra, ở nhiệt độ cao xảy ra hiện tượng ghép cặp giữa vortex và antivortex. Đó là chuyển pha Kosterlitz-Thouless, là sự chuyển pha khi vortex-antivortex có kết cặp với nhau khong



(a) vortex (b) anti-vortex

Hình 1.3: Hình ảnh của vortex và anti-vortex

Nhiều hệ 2 chất rắn chiều với tính chất đối xứng tương tự mô hình xy có 1 chuyển pha Kosterlitz-Thouless (KT). Ví dụ chuyển pha của He siêu lỏng của màng mỏng là chuyển pha KT [2].

1.1.3 Mô hình XY tổng quát

Hamilton của hệ:

$$H = -J \sum_{\langle ij \rangle} [\Delta \cos(\theta_i - \theta_j) + (1 - \Delta) \cos(q\theta_i - q\theta_j)], \tag{1.1}$$

<ij> là các cặp chỉ số của các spin lân cận

J là hằng số cặp tương tác, được set giá trị là 1

q là số nguyên dương, mỗi giá trị của q sẽ đại diện cho 1 mô hình cụ thể

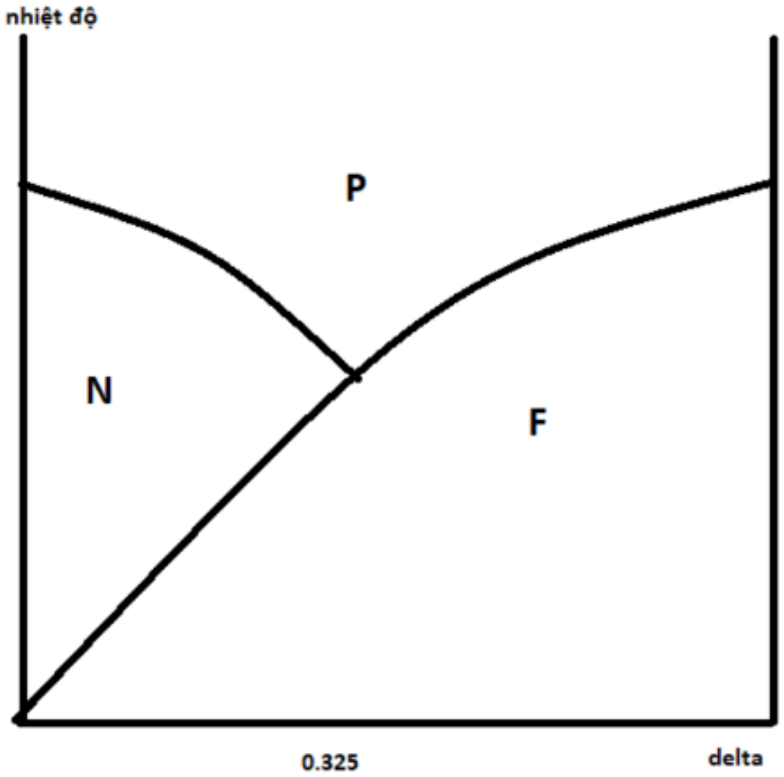
Trong phương trình 1.1 Số hạng đầu tiên bên vế phải đại diện cho tương tác ferromagnetic. Số hạng thứ hai bên vế phải đại diện cho tương tác nematic.

Giữ các mô hình cổ điển khác nhau, mô hình XY tổng quát với $q = 2$ là trường hợp thú vị. Trường hợp này có 2 loại tương tác cạnh tranh nhau: tương tác sắt từ (tương tác sắt từ có xu hướng sắp xếp các spin trong cùng 1 hướng) và tương tác nematic (tương tác nematic có xu hướng sắp xếp các spin cùng phương); 2 tương tác được điều khiển bởi tham số Δ, Δ chạy từ 0 đến 1. Vì cạnh tranh giữa 2 tương tác, giản đồ pha phức tạp hơn mô hình XY nguyên gốc.

Trong bài báo này [3], điểm ba trong giản đồ pha được xác định tại $\Delta \approx 0.325$.

3 pha của hệ trong hình 1.4

- P: trạng thái các spin hỗn loạn



Hình 1.4: Giản đồ pha cho xy model tổng quát. Trong đó trục x là các giá trị Δ, trục y là nhiệt độ

- F: sắt từ: các spin sắp xếp cùng chiều
- N: nematic: các spin sắp xếp cùng phương

Trong khóa luận này, chúng tôi khảo sát $q = 2$ cho mô hình XY tổng quát bằng các dùng học máy cho quá trình phân loại pha. Chúng ta chú ý rằng mô hình đã được nghiên cứu cùng nhau với những vấn đề khác, tập trung của tác giả là gia tăng hiệu suất của học máy cho model, với đề xuất đặc trưng nhất định (feature engineering) từ đầu vào cho học máy. Tất nhiên, công việc của chúng tôi cũng tập trung vào các khía cạnh khác của model, chúng tôi xem xét mạng neural được huấn luyện ở vùng điểm ba và từ đó, xem xét hiệu suất của phương pháp deep learning. Từ tài liệu [4], chúng tôi đã có kiến thức nhất định về việc khó khăn mô phỏng Monte Carlo cho mô hình này tại khu vực xung quanh điểm ba. Mô phỏng Monte Carlo gặp khó khăn, cần nhiều thời gian để hội tụ, nguyên nhân do cạnh tranh giữa 2 tương tác. Từ đó, chúng tôi cố gắng giải quyết 2 câu hỏi chính:

Có phải hiệu suất của deep learning tốt hơn mô phỏng monte-carlo truyền thống

Và nếu có, tại sao nó tốt hơn, nói theo 1 cách khác, đại lượng vật lý nào mà mạng neural đã học để hiệu suất nó tốt hơn

1.2 Học máy và học sâu

1.2.1 Học máy

Học máy (Machine learning) là công việc dùng các thuật toán để phân tích dữ liệu (data), học từ các dữ liệu đó, và sau đó đưa ra quyết định hoặc dự đoán về dữ liệu mới

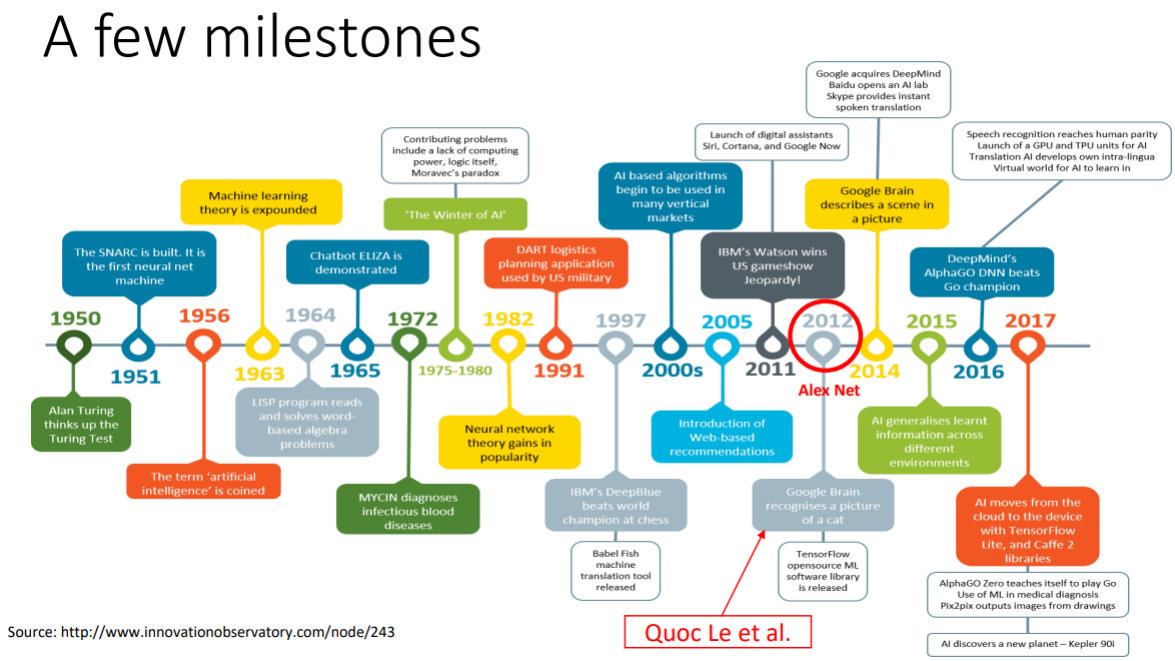
Machine learning không phải là khái niệm mới, nó đã có những thời kỳ đóng băng và 10 năm trở lại đây nó trỗi dậy nhờ sự bùng nổ dữ liệu lớn (các quá tình phát triển của machine learning trong hình 1.5). Học máy có tác động mạnh trong các lĩnh vực khoa học và công nghệ. Vật lý chất rắn không đứng ngoài xu hướng này. Tính từ 2015, số lượng của các bài báo xuất bản sử dụng học máy trong vật lý, cụ thể trong vật lý chất rắn, gia tăng đáng kể. Hiện tại có 1 lượng lớn ứng dụng học máy trong vật lý chất rắn : Phân loạt pha cho mô hình spin cổ điển hoặc mô hình mạng lượng tử, tìm trạng thái cơ bản (ground state), tăng tốc mô phỏng monte carlo, tiếp tục phân tích và giải các mô hình [5]. Học máy cũng dùng nhiều trong kết hợp với tính toán ab initio cho phân tích tính chất của vật liệu thực và dự đoán vật liệu mới với thuộc tính thú vị [6].

Cụ thể các bước dùng machine learning có thể hình dung là

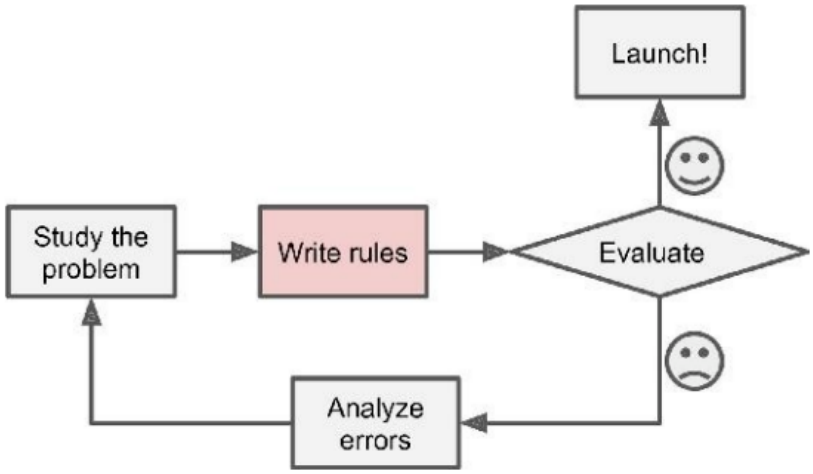
- Viết thuật toán
- Máy tính sẽ thực thi thuật toán trên các tập dữ liệu cụ thể (train dataset)
- Sau đó máy tính có thể làm 1 vài tác vụ với dữ liệu chưa được sử dụng trước đó (test data)

Vậy machine learning là học từ dữ liệu.Hiện bài toán machine learning chia làm 3 loại:

- Dạng 1: Học có giám sát (supervised learning)



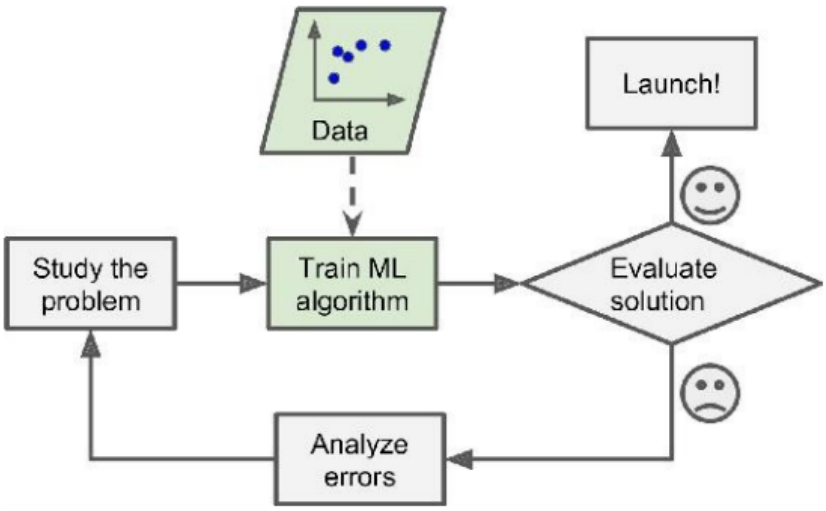
Hình 1.5: Quá trình phát triển của machine learning



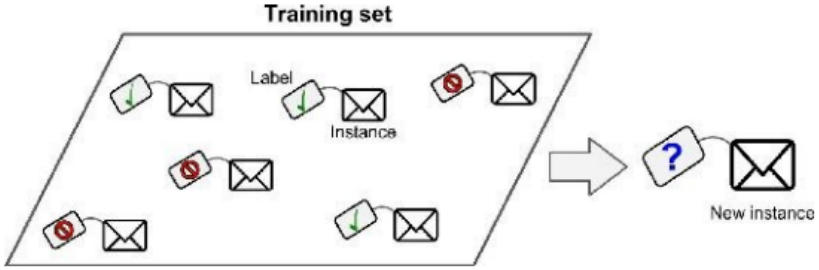
Hình 1.6: Cách tiếp cận truyền thống

- Dạng 2: Học không giám giám sát (unsupervised learning)
- Dạng 3: Học củng cố (reinforcement learning)

Trong đồ án này, chúng ta sẽ dùng dạng 1, học có giám sát. Học có giám sát nghĩa là chúng ta cung cấp dữ liệu gồm sample và label tương ứng. Sau đó ta tiến hành cho máy học tập dữ liệu đó (quá trình này gọi là traing model) để sinh ra 1 model cụ thể. Cuối cùng dùng model để dự đoán label của sample mà nó chưa từng được thấy, chưa từng được học (quá trình này gọi là dùng model đã train để predict dữ liệu test)



Hình 1.7: Cách tiếp cận machine learning



Hình 1.8: Ví dụ về việc học có giám sát trong việc phân loại email là spam hay không

1.2.2 Học sâu

Học sâu (deep learning) là 1 trường nhỏ của học máy (machine learning), nó dùng các thuật toán được lấy cảm hứng từ cấu trúc và các chức năng của mạng lưới thần kinh não người.

Trong học sâu chúng ta vẫn nói về các thuật toán học từ dữ liệu giống như chúng ta đã thảo luận khi nhắc đến học máy. Mô hình thực hiện việc học sẽ dựa trên cấu trúc và chức năng của mạng lưới thần kinh của não.

Trong kỷ nguyên của "dữ liệu lớn"(big data), học sâu với việc dùng các mạng neural network nổi bật là 1 kỹ thuật học máy dùng cho nhiều khía cạnh của đời sống, khoa học và công nghệ. Trong vật lý chất rắn, một trong những ứng dụng cơ bản của deep learning là phân loại các pha khác nhau trong sơ đồ pha cho các mô hình vật lý khác nhau. Phép thử thông thường cho deep learning là một vào mô hình spin cổ điển vì chúng ta đã nghiên cứu mô hình như này nhiều năm và dữ liệu của những mô hình này có thể sinh ra dễ dàng bằng mô phỏng Monte Carlo. Do đó nó cho phép 1 số lượng không giới hạn các mẫu để làm đầu vào cho deep learning. Công trình tiên phong của Carrasquilla và Melko đã chỉ ra sức mạnh của deep learning trong xác định điểm chuyển pha, finite-size scaling như thường được quan sát trong mô phỏng large-scale [7]. Sau đó, có 1 số các bài báo đi theo hướng cho các mô hình khác nhau : vortices và chuyển pha Berezinskii- Kosterlitz-Thouless (BKT) trong mô hình XY, phân tích các tham số deep learning trong mô hình ISing

và mô hình XY... [8]

Hiện nay có rất nhiều cấu trúc mạng neuron hiệu quả với nhiều mục đích , tác vụ khác nhau, ta có thể kể đến ở đây:

- Feedforward Neural Network
- Recurrent Neural Network
- Long/Short Term Memory (LSTM)
- Convolution Neural Network (mạng tích chập)

Trong đồ án này chúng ta chỉ dùng 2 mạng neuron Convolution Neural Network và mạng Feed Forward.

1.2.3 Mạng Feedforward (Feedforward Neural Network)

Mạng feedforward hoặc perceprtron nhiều lớp là mô hình quan trọng của deep learning. Mục đích của mạng feedforward là xấp xỉ 1 hàm số nào đó. Ví dụ, trong quá trình phân loại , y=f(x), f là ánh xạ map x với label y. Mạng feedforward định nghĩa 1 ánh xạ y=f(x,delta) và học để điều chỉnh giá trị delta sao cho kết quả được đánh giá tốt nhất.

Model này được gọi là feedforward vì dữ liệu chạy qua mạng lần lượt từ lớp này sang lớp khác. Trên hình 1, multilayer neural network xác định những vật thể khác nhau bởi học đặc điểm khác nhau của mỗi vật thể tại mỗi lớp. vd: ở lớp ẩn thử nhất (hidden layer), lớp này sẽ phát hiện ra các cạnh của vật thể, ở lớp thứ 2, các góc và các đường cong được phát hiện. Giống như trong não của chúng ta, có các vùng các nhau với mục đích giống như các lớp ẩn của mạng.

Chúng ta có thể thấy bằng việc áp dụng neural network chúng ta có thể vượt qua vấn để của phi tuyến. Tức là có thể mô tả bất cứ hàm số nào dùng mạng neural network.

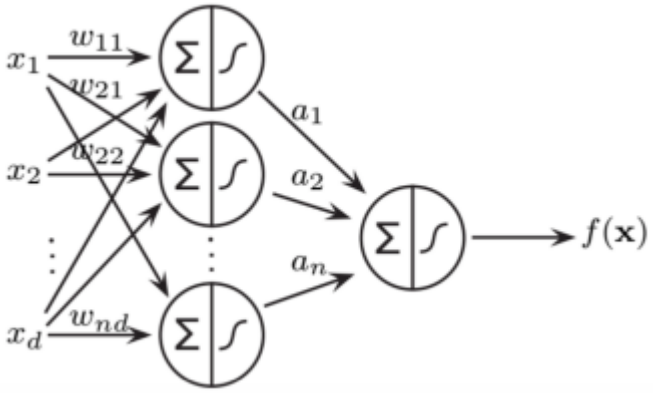
Vd: đầu vào cho 1 network có thể là độ lớn các giá trị pixel từ 1 ảnh chữ viết tay. Chúng ta sẽ cho nework học và điều chỉnh các weights và các bias, từ đó đầu ra của network sẽ phân loại chữ viết tay đó thuộc lớp nào.

Cấu trúc của 1 mạng feedforward gồm 3 phần : lớp đầu vào, lớp ẩn, lớp đầu ra. Lớp bên trái ngoài cùng được gọi là lớp đầu vào, những neuron nằm ở lớp đầu vào được gọi là neuron đầu vào (input neurons). Lớp bên phải ngoài cùng được gọi là lớp đầu ra. Những lớp ở giữa được gọi là hidden layer.

1.2.4 Hàm mất mát (Loss function)

Hàm mất mát và tham số mô hình Mỗi mô hình Machine lerning được mô tả bởi các tham số mô hình (model parameters). Công việc của một thuật toán machine learning là đi tìm các tham số mô hình phù hợp với mỗi bài toán. Việc đi tìm các tham số mô hình có liên quan mật thiệt đến các phép đánh giá. Mục đích của chúng ta là đi tìm các tham số mô hình sao cho các phép đánh giá cho kết quả tốt nhất. Trong bài toán classification ,kết quả tốt có thể hiểu là ít điểm dữ liệu bị phân lớp sai nhất.

Quan hệ giữa một phép đánh giá và các tham số mô hình thường được mô tả thông qua một hàm số được gọi là hàm mất mát (loss function). Hàm mất mát này



thường có giá trị nhỏ khi phép đánh giá cho kết quả tốt và ngược lại. Việc đi tìm các tham số mô hình sao cho phép đánh giá trả về kết quả tốt tương đương với việc tối thiểu hóa hàm mất mát. Như vậy, viêc xây dựng một mô hình machine learning chính là việc đi giải một bài toán tối ưu. Quá trình đó có thể được coi là quá trình learning của machine.

Tập hợp các tham số mô hình thường được ký hiệu bằng delta, hàm mất mát của mô hình thường được ký kiệu là L(delta). Bài toán tối thiểu hàm mất mát để tìm tham số mô hình thường được viết dưới dạng

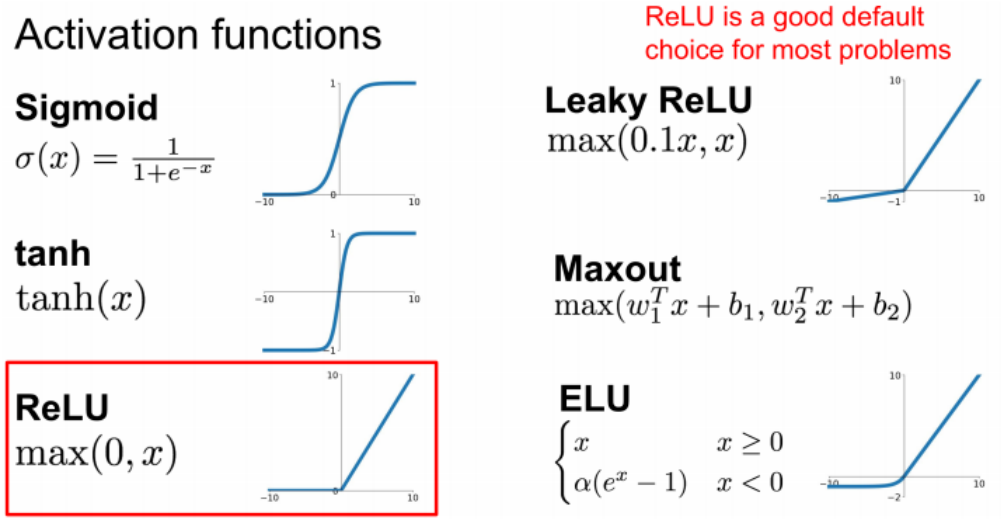
Một hàm số L(δ) bất kỳ có thể có rất nhiều giá trị của của delta để nó đạt giá trị nhỏ nhất, hoặc cũng có thể nó không chặn dưới. Thậm chí việc tìm giá trị nhỏ nhất của một hàm số đôi khi là không khả thi.Trong machine learning cũng như nhiều bài toán thực tê,s việc chỉ cần tìm ra một bộ tham số delta làm cho hàm mất mát đạt giá trị nhỏ nhất, hoặc thậm chú đạt một giá trị cực tiểu, thường mang lại các kết quả khả quan.

Chúng tôi sẽ giới thiệu hàm mất mát cho mục đích huấn luyện mô hình. Tại sao không cố gắng cực đại độ chính xác hóa đầu ra mà lại đi tối thiểu hóa sự sai lệch. Vấn đề là số lượng điểm dữ liệu phân loại đúng không phải là 1 hàm liên tục của các weight và bias của mạng neural. Nhiều khi thay đổi 1 lượng nhỏ của weight và bias không thay đổi số điểm được dự đoán đúng. Điều này tạo khó khăn để xác định cần thay đổi weight và bias như thế nào để tăng hiệu suất của model. Nếu chúng ta dùng các hàm liên tục như hàm mất mát bậc 2 , điều này sẽ dễ dàng để xác định cần thay đổi weight và bias như nào để có được hiệu suất model tốt hơn. Đó là tại sao chúng ta tập tung vào tối thiểu hóa hàm mất mát. Vd 1 hàm loss function tiêu biểu là là mean square error

ở đây, w ký hiệu cho tất cả các weight trong mạng,b là tất cả các bias, n là tổng số điểm huấn luyện, a là vector đầu ra khi x là input, và khi lấy tổng là lấy trên toàn bộ tập huấn luyện. Ta dễ dàng thấy đầu ra của mạng phụ thuộc vào x,w và b.

1.2.5 Hàm kích hoạt (activation function)

Tròn phần này, chúng ta cùng trả lời 1 số câu hỏi như sau : tại sao có nhiều hàm kích hoạt, tại sao hàm này lại tốt hơn hàm khác, làm cách nào chúng lại chọn hàm đó. Đầu tiên chúng ta nhìn vào 1 neuron Y= .. +bias .Bây giờ, giá trị của Y có thể là bất kỳ chạy từ âm vô cùng đến dương vô cùng. Neuron không biết giới hạn của giá trị. Làm thế nào chúng ta quyết định liệu neuron có nên kích hoạt hay không. Chúng ta quyết định thêm hàm kích hoạt sau neuron cho mục đích đó.



Step function (Hàm bước):

Điều đầu tiên chúng ta nghĩa về 1 threshold (ngưỡng) dựa trên hàm kích hoạt. Tức là nếu Y lớn hơn 1 giá trị nào đó, chúng ta sẽ cho neuron kích hoạt, nếu nó nhỏ hơn theshold thì nó sẽ không kichsb= hoạt.

Hàm kích hoạt A="kích hoạt"if Y> theshold else not Viết cách khác : A=1 if Y> Theshold else 0

Điều này tạo nên activation function (Hàm kích hoạt) cho neuron.Tuy nhiên vẫn có nhược điểm cho vấn đề này, để hiểu rõ hơn, chúng ta cùng đi tiếp. Giả sử bài toán của chúng ta là phân loại nhị phân (2 lớp đầu ra)(binary classifier). Neuron cần quyết định là có hay không. Hàm step function có thể làm điều đó, nó sẽ cho đầu ra chính xác là 1 hoặc 0. Bây giờ nghĩ về trường hợp chúng ta muốn phân loại nhiều lớp : lớp 1, lớp 2, lớp 3. Điều gì xảy ra nếu hơn 1 neuron kích hoạt, tất cả đầu ra sẽ là 1 (do qua step function), vậy chúng ta biết chọn lớp nào.

Bạn muốn mạng của chúng ta chỉ kích hoạt 1 neuron và các neuron khác bằng 0. Và khi nhiều hơn 1 neuron kích hoạt, bạn sẽ muốn tìm neuron nào có giá trị hàm kích hoạt cao nhất (từ đó lựa chọn được neural phân loại vào lớp nào).

Từ vấn đề kể trên, người ta đưa ra các loại hàm kích hoạt như sau

sigmoid function, Tanh Fuction, Relu

Vậy chúng ta nên sử dụng hàm nào và dùng khi nào Có phải chúng ta chỉ sử dụng relu cho tất cả ? hoặc sigmod hoặc tanh. Sigmod hoạt động tốt cho các bài toán phân loại bởi vì xấp xỉ 1 hàm phân loại là tổ hợp của nhiều hàm sigmoid là dễ hơn relu. Nó dẫn đến tăng tốc cho quá trinh huấn luyện. Hoặc bạn có thể dùng 1 hàm của riêng bạn,

1.2.6 Gradient Descent

Đến đây, chúng ta đã biết được neural network (mạng neural) được tổ chức dưới các layer (lớp). Mỗi mạng neural (Trừ input layer) là tổng của tất cả các lớp trước nó, nói cách khác nó được tính từ đầu ra của các lớp trước đó nhân với các weights. Và sau đó chúng ta thêm bias vào tổng đó. Sau đó tất cả được thêm hàm kích hoạt

vào sau kết quả.Các tham số (weights, bias) là các giá trị số, chúng ta cố gắng điều chỉnh nó bằng quá trình huấn luyện model với các tập dữ liệu đã được dán nhãn. Kết quả cuối cùng sẽ là 1 model được xây dựng từ dữ liệu đó và dùng để dự đoán các dữ liệu chưa được nhìn thấy bao giờ

Để huấn luyện mạng neuron của chúng ta (sau khi đã chọn được cấu trúc mạng) , việc đầu tiền chúng ta cần phải làm đó là khởi tạo giá trị cho các tham số. Nếu chúng ta tạo model từ ban đầu, không dùng model pre-train, quá trình thường sẽ là:

- Khởi tạo các weight ngẫu nhiên, nó giúp phá vỡ sự đối xứng và cũng ngăn chặn tất các các neuron trong cùng 1 layer học những thứ giống nhau. Các weight cho mỗi lớp (layer) thường được khởi tạo dùng phân phối chuẩn với giá trị trung bình là 0 và phương sai 1/n hoặc 2/n (n là số các điểm dữ liệu). Giá trị phương sai phụ thuộc vào hàm kích hoạt ở sau các neuron (ví dụ phương sai sẽ là 2/n cho trường hợp hàm kích hoạt là Relu)

- Khởi tạo các bias có giá trị 0. Từ đây chúng ta sẽ tiếp tục quá trình bởi các thuật toán tối ưu, đó là cố gằng cực tiểu hóa sự khác biệt giữa giá trị thật của đầu ra và giá trị được tính toán bởi mạng của chúng ta. Thuật toán tối ưu hay được sử dụng để huấn luyện mô hình neuron là gradient descent. Gradient là gì, chúng ta sẽ định nghĩa nó sau, bây giờ, ý tưởng sẽ hiểu là : gradient là 1 tính toán số cho phép chúng ta biết các để điều chỉnh các tham số của mạng neuron của chúng ta sao cho sự chênh lệch giữa đầu ra của mạng và giá trị thực của điểm dữ liệu là nhỏ nhất. Thuật toán có 1 vài phiên bản phụ thuộc vào số lượng các mẫu:

+ Batch gradient descent: Tất cả các dữ liệu hiện có được cung cấp hết 1 lần để tính. Phiên bản này sẽ có rủi ro cao bị nghẽn tính toán, vì gradient sẽ được tính cho tất các các mẫu , và các variations sẽ được cực tiểu sớm hơn hoặc muộn hơn. Nguyên tắc chung là : với 1 mạng neuron , nó luôn tốt khi đầu vào luôn là ngẫu nhiên.

+ Stochasic gradient descent: Với 1 điểm dữ liệu ngẫu nhiên được cho vào 1 vòng lặp. Gradient sẽ được tính chỉ trên điểm này, với cách này yếu tố ngẫu nhiên được đảm bảo và việc tính toán khó bị tắc ngẽn. Vấn đề chính với phiên bản này là sự chậm chạp của nó, vì nó cần nhiều lần lặp. Thêm nữa, nó chưa tận dụng hết được tài nguyên tính toán

+ (Stochasic) Mini batch gradient descent: thay vì cung cấp cho mạng với chỉ 1 mẫu ngẫu nhiên, chúng ta cung cấp cho nó N mẫu ngẫu nhiê. Nó là 1 phiên bản nâng cấp của cách thứ 2, nó sẽ chạy nhanh hơn do dùng được GPU tính toán song song hóa. Chúng ta sẽ sử dụng phiên bản này và chọn giá trị của N sao cho nó có sự cân bằng giữa sự ngẫu nhiên và thời gian huấn luyện (trainning time) (và cũng N cũng không quá lớn để GPU còn tính toán tốt)

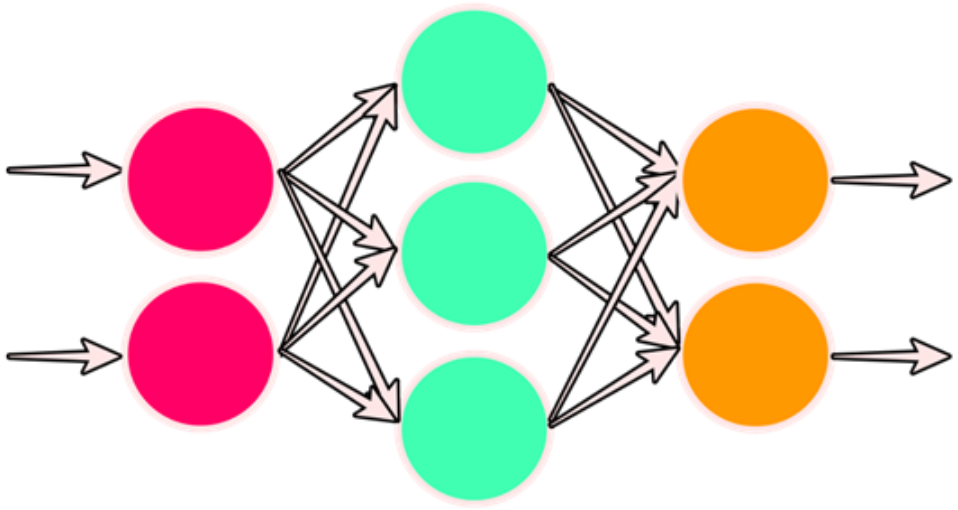
Đây là các vòng lặp của thuật toán Gradient Descent sử dụng phiên bản thứ 3 của nó (Mini batch gradient descent):

1.Đầu tiên chúng ta lấy N mẫu ngẫu nhiên làm input (đã có nhãn)

2.Sau khi tính toán phù hợp trên mỗi lớp của mạng, chúng ta nhận được dự đoán như là đầu ra. bước này có tên là foward propagation (lan truyền)

3.Chúng ta đánh giá hàm mất mát của mini-batch. Hàm mất mát sẽ tính độ sai khác của dự đoán và giá trị thực. Giá trị của hàm mất mát là 1 số mà thuật toán tối ưu cố gắng để làm nó nhỏ nhất

4. Chúng ta tính toán giá trị gradient như 1 đạo hàm hàm nhiều biến của hàm mất mát , các biến chính là các tham số của mạng. Về mặt đồ thị, nó chính là độ



Hình 1.9: Hình ảnh minh họa mạng neuron: 1 input layer, 1 hidden layer và 1 output layer

dốc của đường tiếp tuyến với hàm mất mát tại điểm hiện tại (đánh giá với các tham số của mạng hiện tại). Về mặt toán học, nó là 1 vector cho chúng ta hướng mà hàm mất mát tăng nhanh hơn, và chúng ta có thể di chuyển theo hướng ngược lại hướng đó để tối thiểu hóa hàm mất mát.

Chúng ta sẽ lấy ví dụ trường hợp ít chiều để dễ tưởng tượng và vẽ chúng. Tưởng tượng rằng 1 mạng với chỉ 2 tham số, vì thế nên hàm mất mát có thể được biểu diễn trong không gian 3 chiều (XY là các trục cho 2 tham số, z là trục cho giá trị hàm mất mát). Gradient trong trường hợp này là 1 vector với 2 thành phần (phương x và phương y), và nó chỉ về 1 hướng mà nếu chúng ta dịch tham số x và y theo hướng đó, hàm mất mát sẽ tăng lên. Vậy ý tưởng là cố gắng di chuyển theo hướng ngược lại của đạo hàm để đạt được cực tiểu

Tất nhiên trong mạng thật của chúng ta, hàm mất mát phức tạp hơn nhiều. CHúng ta không chỉ có 2 tham số , con số thực tế có thể lên tới hàng triệu. Vì vậy, trong mạng thực tế việc tìm cực tiểu địa phương tốt hơn là cố gắng tìm cực tiểu cục bộ.

1.3 Dùng Deep learning trong XY 2 chiều tổng quát

Dữ liệu của chúng ta sẽ là 1 sample và 1 label tương ứng. Tất cả những sample được gán nhãn như vậy phù hợp với bài toán học có giám sát.

Chúng ta sẽ xây dựng mô hình , sau đó dùng những sample đã có để huấn luyện mô hình, sau đó dùng mô hình được huấn luyện để dự đoán những mẫu chúng chưa nhìn thấy, chưa biết.

Cách làm của chúng ta để đây là sẽ chia tệp dữ liệu thành 3 phần: tập train, tập validation và tập test.

Dữ liệu được sinh ra và gán nhãn nhờ mô phỏng monte-calo. Do chúng ta có thể sinh ra được số lượng lớn dữ liệu nhờ monte-calo nên việc áp dụng deep learning

vào bài toán này là khả thi.

Chương 2

Mô hình và phương thức

2.1 Mô phỏng mô hình XY sử dụng Monte Carlo

2.1.1 Mô phỏng mô hình Ising sử dụng Monte Carlo

Xét N nguyên tử tồn tại trong từ trường định hướng z có cường độ H. Giả sử rằng mọi nguyên tử đều là hệ spin $\frac{1}{2}$ như nhau. Điều này dẫn đến hoặc $s_i = +1$ (spin hướng lên), hoặc $s_i = -1$ (spin hướng xuống), trong đó s_i là thành phần theo phương z của spin nguyên tử thứ i. Tổng năng lượng của hệ được viết là:

$$\text{E=-J } \sum\langle ij\rangle S_i S_j - \mu H \sum_{i=1}^{n} s_i \tag{2.1}$$

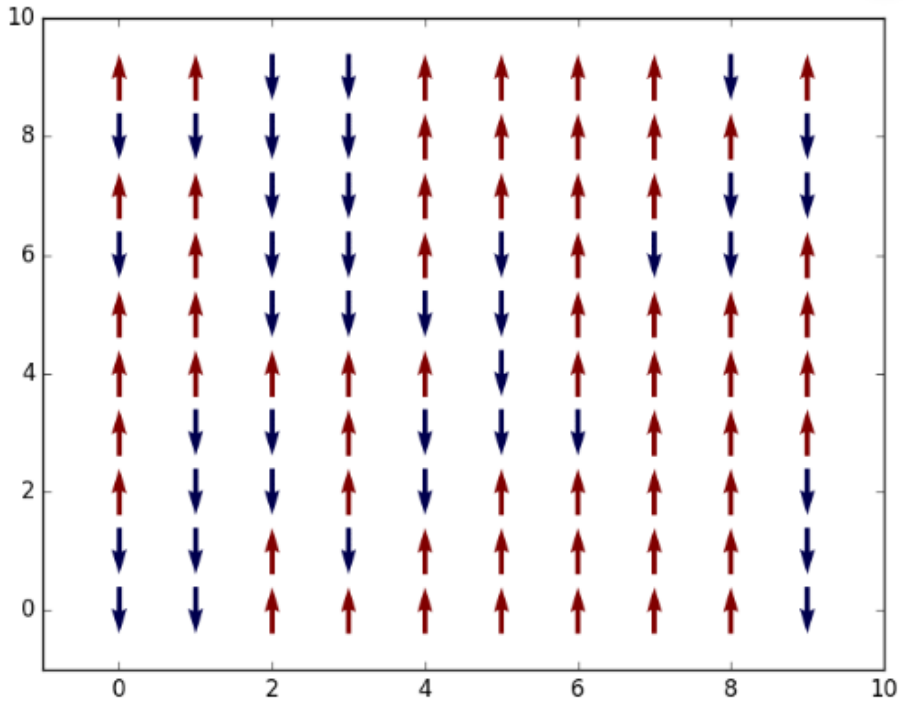
Ở đây, $\langle ij\rangle$ được dùng để chỉ tổng theo các cặp nguyên tử lân cận. Ngoài ra, J được gọi là năng lượng trao đổi, còn μ là mô-men từ nguyên tử. Phương trình trên là cốt lõi của mô hình Ising.

Phần tử thứ nhất ở vế phải của phương trình 2.1.1 cho thấy rằng tổng năng lượng bị giảm xuống khi các spin nguyên tử lân cận được sắp xếp. Hiệu ứng này chủ yếu là do nguyên lý ngoại trừ Pauli

Phương pháp đầu tiên phân tích mô hình Ising là dùng một cách đơn giản hóa được biết với tên gọi xấp xỉ trường trung bình. Tuy nhiên, những tính toán này lại sai ở một số chi tiết trong sự chuyển pha bậc hai. Để tính đúng hơn, ta phải bỏ cách xấp xỉ trường trung bình và dùng cách Monte-Carlo.

Ta hãy xét một mảng vuông hai chiều chứa các nguyên tử. Đặt L là kích thước mảng, và $N = L^2$ là số nguyên tử có trong mảng, như trên hình.Phương pháp Monte-Carlo dùng cho mô hình Ising, hoàn toàn không sử dụng cách xấp xỉ trường trung bình, được dựa trên thuật toán sau:

- Lần lượt đi qua từng nguyên tử trong mảng:
- Với mỗi nguyên tử, hãy tính độ thay đổi năng lượng của hệ, ΔE , khi spin nguyên tử bị đảo ngược.
- Nếu ΔE



Hình 2.1: Mô phỏng Monte Carlo trên mô hình Ising

- Nếu $\Delta E > 0$ thì đảo ngược spin với xác suất $P = exp(-\beta \Delta E)$. Trong đó $\beta = \frac{1}{kT}$, k là hằng số Boltzmann.
- Lặp lại quá trình nhiều lần cho đến khi đạt được cân bằng nhiệt.

Mục đích của thuật toán này là thử nhiều nhất các trạng thái có thể có của hệ thống, và đảm bảo rằng hệ thống chiếm giữ một trạng thái cho trước với xác suất Boltzmann: nghĩa là một xác suất tỉ lệ thuận với $exp(-\beta E)$, trong đó E là năng lượng của trạng thái.

Để biểu diễn tính đúng đắn của thuật toán trên, ta hãy xét việc đảo spin của nguyên tử thứ i. Giả sử rằng hành động này khiến cho hệ thống chuyển từ trạng thái a (năng lượng E_a) sang trạng thái b (năng lượng E_b). Hơn nữa, giả sử rằng $E_a < E_b$. Theo thuật toán trên, xác suất để chuyển từ trạng thái a sang trạng thái b là

$$\mathrm{P}_{a \to b} = \exp[-\beta(E_b - E_a)] \tag{2.2}$$

trong khi xác suất để chuyển từ trạng thái b sang trạng thái a là $P_{b \to a} = 1$

Trong cân bằng nhiệt, nguyên lý cân bằng chi tiết (principle of detailed balance) phát biểu rằng

$$\mathrm{P}_a P_{a \to b} = P_b P_{b \to a}$$

(2.3)

trong đó Pa là xác suất để hệ chiếm giữ trạng thái a, còn Pb là xác suất để hệ chiếm giữ trạng thái b. . Sắp xếp lại phương trình trên ta được

$$\mathrm{P}_b \overline{P_a = \exp[-\beta(E_b - E_a)]}$$
(2.4)

điều này thống nhất với phân bố Boltzmann.

Bây giờ, mỗi nguyên tử trong mảng đang xét, trừ những nguyên tử ở rìa mảng, còn lại đều có bốn nguyên tử lân cận, Ta có thể loại bỏ ngoại lệ phiền phức này bằng cách đưa vào điều kiện biên tuần hoàn: nghĩa là bằng cách đồng nhất các rìa ở hai phía đối diện của mảng

2.1.2 Mô phỏng xy model sử dụng Monte Calo

Các spin là các vector đơn vị có thể xoay trong không gian 2 chiều

$$\mathrm{s}_i = (\cos\theta_i, \sin\theta_i)$$
(2.5)

2.2 Sinh dữ liệu sử dụng Monte Calo

Đầu tiên chúng ta tiến hình mô phỏng Monte Carlo tại $\Delta = 0.2$, $\Delta = 0.34$, $\Delta = 0.7$ và tại $\Delta = 1.0$ để sinh cấu hình spin làm dữ liệu đầu vào. Local spin-flip và cluster spin-flip (thuật toán Wolff) cập nhật được dùng cho mô phỏng Monte Carlo,tương tự công việc trước đó trong bài báo này [3]. Với mỗi Delta , có 76 điểm nhiệt độ phân bố bằng nhau, điểm chuyển pha nằm đâu đó ở giữa khoảng nhiệt độ và $T_{max} - T_{min} = 0.9$. Cho mỗi T và Delta, chúng tôi chạy 20 runs độc lập, tại mỗi run sẽ có 4×10^6 Monte Carlo tập nhật cho đến khi có 300 mẫu. Do đó tổng cộng sẽ có $20 \times 300 \times 76 = 456000$ mẫu dùng để làm đầu vào cho huấn luyện và kiểm thử mô hình học sâu cho mỗi giá trị Δ và 1 kích thước mạng (L)

2.3 Quá trình dùng Deep Learning

2.3.1 Dữ liệu

Chúng ta sẽ nghiên cứu $\Delta = 0.2$, $\Delta = 0.34$, $\Delta = 0.7$ và tại $\Delta = 1.0$

- Với $\Delta = 0.2$ hệ sẽ có 2 chuyển pha, từ F sang N và từ N sang P

- Với $\Delta = 0.34$ hệ sẽ có 1 chuyển pha, từ F sang P và $\Delta = 0.34$ khá gần với điểm ba tại $\Delta = 0.325$
- Với $\Delta = 0.7$ và $\Delta = 1.0$ hệ sẽ có 1 chuyển pha, từ F sang P

Mỗi mỗi Δ chúng ta dùng mô phỏng monte carlo để sinh ra dữ liệu, các chiều dài mô phỏng ở đây là $L = 12$, $L = 16$, $L = 24$,$L = 32$,$L = 40$,$L = 48$,$L = 56$,$L = 64$. Vậy là với mỗi Δ và mỗi L ta sẽ có 1 tệp dữ liệu các cấu hình spin (456000 mẫu). Mỗi cấu hình spin đều được đánh nhãn label từ trước, tức là chúng ta sẽ biết mỗi cấu hình tương ứng với pha nào, F, P hay N. Để cho tiện tính toán, ta quy ước 0 ứng P, 1 ứng với N, 2 ứng với F

Với mỗi Δ khác nhau và mỗi L khác nhau ta sẽ có 1 model deep learning, tương ứng với đó ta chia dữ liệu luôn thành 3 phần

- 20% của dữ liệu (với 1 cặp Δ và 1 L, có tổng cộng 456000 mẫu) sẽ làm tập test
- 20% của 80% dữ liệu còn lại sẽ làm tập validation
- 80% của 80% dữ liệu còn lại sẽ làm tập train

2.3.2 Xử lý dữ liệu

Trước khi cho vào mô hình, ta cần xử lý dữ liệu

Ở đây chúng ta sẽ có 4 cách xử lý dữ liệu đầu vào (Cụ thể về 4 loại dữ liệu đầu vào sẽ được nhắc lại trong phần kết quả)

- Dùng xy, 1 cấu hình đầu vào sẽ có shape là L,L,2. Với mỗi một điểm trên 1 cấu hình, ta lấy hình chiếu spin theo trục x và hình chiếu spin theo trục y làm dữ liệu đâu vào.
- Dùng angle, 1 cấu hình đầu vào sẽ có shape là L,L,1. Với mỗi một điểm trên 1 cấu hình, ta chỉ lấy góc của nó làm dữ liệu đầu vào.
- Dùng vortex, 1 cấu hình đầu vào sẽ có shape là L,L,1. Với 1 cụm 4 điểm trên 1 cấu hình, ta sẽ được 1 vortex, biến đổi toàn bộ cấu hình góc thành vortex , áp dụng thêm điều kiện biên tuần hoàn
- Dùng angle-vortex, 1 cấu hình đầu vào sẽ có shape là L,L,2. Ghép dữ liệu góc và dữ liệu vortex lại với nhau được dữ liệu angle-vortex

Tiếp theo chúng ta cần chuẩn hóa dữ liệu. Với mỗi trường hợp đầu vào khác nhau, ta có 1 cách biến đổi để chuẩn hóa dữ liệu

- Khi dùng xy: $(x_{data} + 1) \times 127.5 => x_{data} = x_{data}/255 =>$ chuẩn hóa về $[0, 1]$
- Khi dùng angle: góc là góc giữa vector spin và trục x $=> [0, 2\pi] =>$ chia cho $2\pi =>$ chuẩn hóa về [0,1]
- Khi dùng vortex: Chuẩn hóa đầu vào là : 1 là vortex, -1 là anti-vortex, 0 là không có gì(không có vortex tại vị trí đó)
- Khi dùng angle-vortex, tại mỗi vị trí trên mạng, ghép angle đã chuẩn hóa với vortext đã chuẩn hóa thành 1 array

2.3.3 Xây dựng mô hình

Để tiến hành huấn luyện và kiểm thử tập dữ liệu này, chúng tôi xây dựng 1 cấu trúc mạng convolutional neural network(CNN) , trong đó có 2 lớp CNN , tiếp theo là 1 lớp max-pooling và tất cả kết nối với với 1 mạng fully-connected neural network (FCNN) . Chi tiết các mạng được liệt kê như sau:

- Lớp 1: CNN 8 filters, kernel là 3x3, hàm kích hoạt là RELU
- Lớp 2: CNN 16 filters, kernel là 3x3, hàm kích hoạt là RELU
- Lớp 3: Max Pooling (2x2), lớp này sẽ giảm đặc trưng và trừ tượng hóa đặc trưng
- Lớp 4: Dense FCNN, 32 neurons ,activation là relu
- Lớp 5: Lớp Softmax, 3 neuron, tác dụng lớp này dùng để phân loại 3 pha F,P,N
- Chúng tôi chọn loss function là categorical-crossentropy và Optimazation là Adam

Chúng tôi dùng thư viện keras (keras được tích hợp sẵn trong tensorflow package) ([9]) và tiến hành train và test trên google collab và cụm máy tính cục bộ. Chúng tôi dùng cross-entropy loss function làm loss function vì trong bài toán phân loại, cross-entropy function luôn tỏ ra hiệu quả. Bên cạnh đó Adam optimazer cũng được chọn làm optimazer (chúng tôi đã thử dùng các optimazer khác nhưng Adam có vẻ nhanh và ít nhiễu nhất). Hình 2.2 về việc so sánh giữa các việc dùng các hàm tối ưu (optimizer) để lựa chọn ra hàm Adam.

Tương tự sau khi thử các batch size khác nhau, chung tôi quyết định dùng chung batch size = 64 cho tất cả các mô hình. Vì tại batch size bằng 64, đường cong học tập ít nhiễu nhất và hiệu suất mô hình đạt cao nhất. Chi tiết xem tại hình 2.3

Tiếp theo cần xác định giá trị epoch. Epoch là thời gian train, model cần được dừng lại khi độ chính xác (accuracy) trên tập validation không có dấu hiệu tăng nữa, ngược lại có dấu hiệu giảm. Chúng tôi đặt giá trị epoch chung cho tất cả mô hình là 200 và đặt hàm call back trong quá trình train là 'early stopping' với 'patience' là 30 epoch

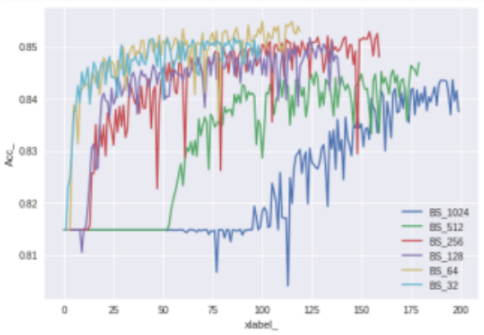
Như vậy đến đây, chúng ta đã xác định được cụ thể cấu trúc mạng và các tham số (hyper-paramater) cần thiết cho model. Đến đây chúng ta đã xong bước khởi tạo model deep learning.

2.4 Dùng SHAP để hiểu mô hình học sâu

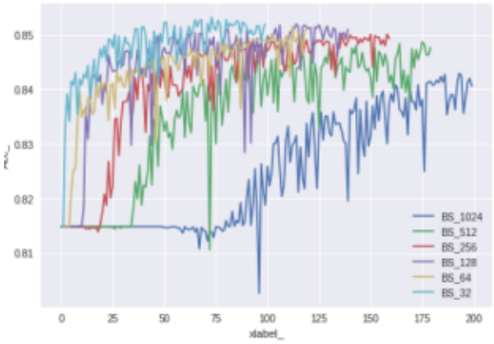
Có 1 công cụ mới tên là SHAP values (Shapley Addtive exPlanation) , cho phép chúng ta xây dựng model phức tap nhưng vẫn cho phép hiểu được nó, diễn gải nó ở mức cá nhân (individual-value interpretation) , diễn giải xem mức độ ảnh hưởng của từng đặc trưng (feature) đến kết quả

Để hiểu tại tại sao điều này lại quan trọng, chúng ta cần nhìn gần hơn về khái niệm của độ chính xác mô hình học sâu (model deep learning) và khả năng có thể diễn giải (model accuracy và interpretability) Cho đến gần đây, chúng ta luôn chọn

(a) opimizer: adam



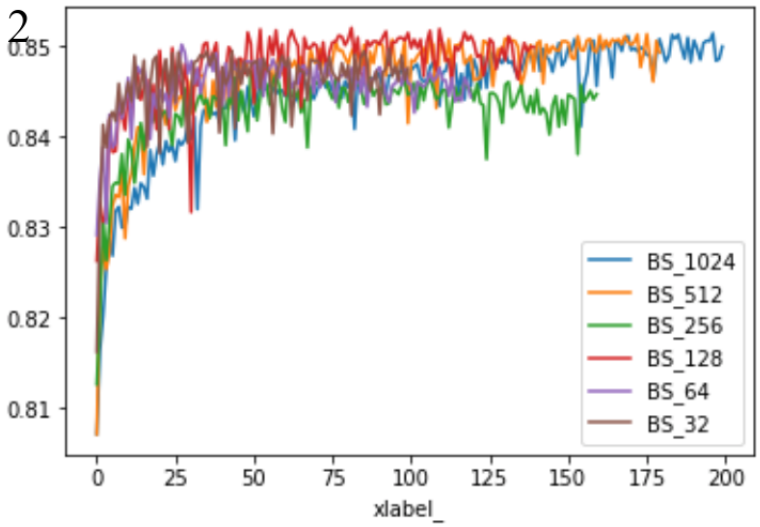
(b) opimizer: NAG



(c) opimizer: SGD

(d) opimizer: ada

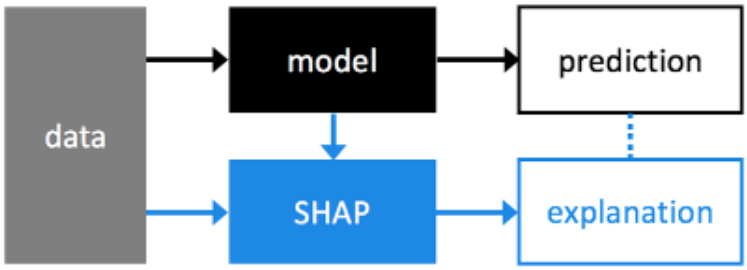
Hình 2.2: Đồ thị các accuray theo epoch của các batch size khác nhau theo 4 opimizer



Hình 2.3: batch size thay đổi từ 32 đến 1024

Tên Lớp	Kích thước đầu ra của mỗi lớp	Số tham số	
conv2d (Conv2D)	(None, 10, 10, 8)	152	
conv2d_1 (Conv2D)	(None, 8, 8, 16)	1168	
max_pooling2d (MaxPooling2D)	(None, 4, 4, 16)	0	
flatten	(None, 256)	0	
dense	21	(None, 32)	8224
dense_1	(None, 3)	99	

Bảng 2.1: Ví dụ cấu trúc mô hình cho L=12, delta=0.2 và type=xy



Hình 2.4: Sơ đồ về mối quan hệ giữa dự đoán và diễn giải mô hình

giữa mô hình chính xác cao nhưng khó giải thích hoặc chọn mô hình đơn giản dễ giải thích nhưng hy sinh độ chính xác. Những mô hình cổ điển như logistic regression hoặc decision tree dễ giải thích. Để gia tăng độ chính xác , chúng ta có thể kết hợp hàng nghìn decision tree và sau đó tổng hợp bằng ensemble rule. Cách tiếp cận khác là dùng các mạng neuron, học sâu sử dụng các mạng neuron với các lớp kết nối bên trong, mỗi lớp (layer) được nhìn ở mức độ trừu tượng cao. Khi cần thêm độ phức tạp cho mô hình, ta thêm sự linh hoạt, nó cho phép chúng ta đạt đến kết quả chính xác cao, những thứ mà chúng ta khó thu được từ mô hình đơn giản, nhưng cái giá phải trả là chúng ta không hiểu nó đã làm gì, nó đã dựa vào nhân tố nào để đưa ra quyết định, tại sao nó lại dự đoán như vậy.

Thậm chí người thiết kế và huấn luyện mô hình (train model) cũng không giải thích được tại sao như vậy. Vì vậy có 1 sự đánh đổi giữa độ chính xác mô hình và khả năng có thể hiểu được. Với SHAP, chúng ta đã phần nào hiểu được tại sao những mô hình phức tạp lại đưa ra quyết định như vậy

SHAP được trình bày trong bài báo của Scott M,Lundberg từ đại học Washington [10]. Nó dựa trên Shapley values, 1 kỹ thuật thường được dùng trong lý thuyết game để xác định mỗi người chơi đã đóng góp bao nhiêu vào kết quả chung trong 1 game mà người chơi cộng tác với nhau. Trong trường hợp của chúng ta, mỗi giá trị SHAP đo độ lớn mỗi đặc trưng trong đóng góp kết quả đầu ra của mô hình, sự đóng góp này là âm hoặc dương. Ý tưởng này giống như ý tưởng tìm ra đặc trưng quan trọng (feature importance) trong logistic regression, nơi chúng ta xác định ảnh hưởng của mỗi đặc trưng bằng cách nhìn vào độ lớn của các hệ số của mỗi đặc trưng

Shap values cho chúng ta 2 lợi ích quan trọng:

- SHAP có thể tính toán cho bất kỳ tree-based model(tree-based model cho phép xây dựng các mối quan hệ không tuyến tính giữa y và x, ví dụ decision trees, random forest, gradient boosting), thay vì hạn chế ở mô hình đơn giản như logistic hoặc mô hình tuyến tính, chúng ta có thể xây nhưng mô hình phức tạp, ko tuyến tính, độ chính xác cao

- Mỗi cá nhân hoặc mỗi điểm dữ liệu sẽ có riêng 1 bộ shap value. Vd mỗi điểm x_{train} có m đặc trưng thì khi tính giá trị SHAP của chúng, ta cũng được 1 bộ số (m số) riêng biệt cho mỗi điểm.

  Thuật toán đặc trưng quan trọng (feature important) truyền thống sẽ nói cho chúng ta đâu là đặc trưng quan trọng nhất trong toàn bộ tập dữ liệu, nhưng nhiều khi nó không có ý nghĩa cho từng mẫu cá nhân. (có những đặc trưng

khi xem xét trên toàn bộ tập dữ liệu thì nó quan trọng nhưng cũng đặc trưng đó khi xét trên phương diện cá nhân thì nó lại không quan trọng)

Bên cạnh đó 1 feature là quan trọng với sample này nhưng với sample khác là chưa chắc quan trọng. Với giá trị shap cho từng điểm dữ liệu (individual-level Shap values), chúng ta có thể xác định chính xác ảnh hưởng của từng đặc trưng (feature) cho từng điểm dữ liệu(data point)

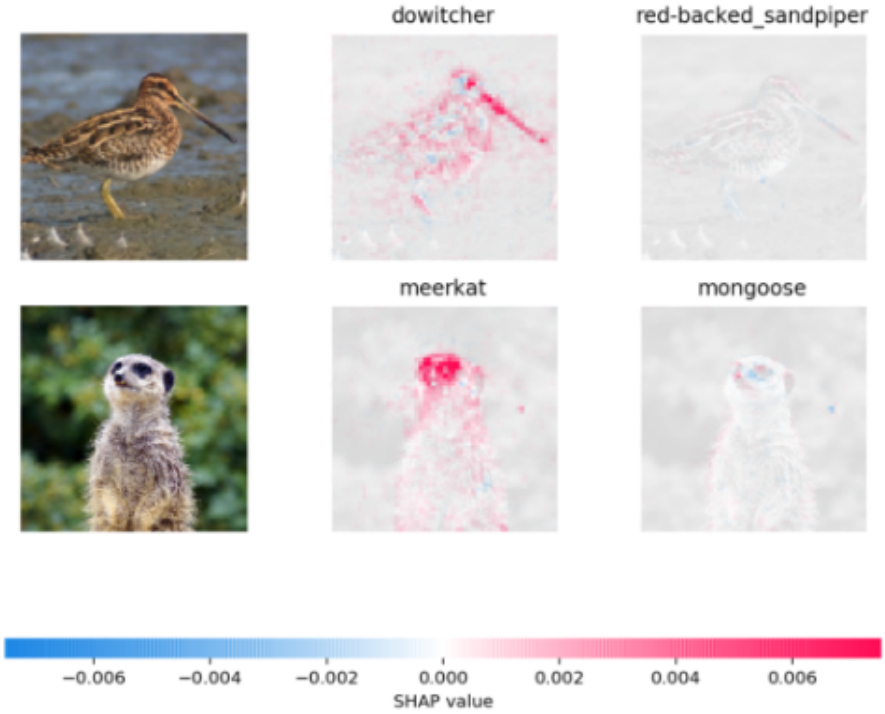
Tất nhiên SHAP vẫn có nhiều hạn chế

- SHAP nhạy cảm với những đặc trưng tương quan lẫn nhau
- SHAP chỉ dùng 1 mô hình diễn giải (explanation model) để xấp xỉ mô hình chính

Để hiểu hơn về SHAP, chúng ta cùng xem khi áp dụng SHAP cho một mô hình phân loại vật thể với 1000 lớp đầu ra. Đó là mô hình VGG16. Tập dữ liệu gồm 14 triệu ảnh thuộc 1000 lớp khác nhau. Đây là 1 mô hình nổi tiếng từ 2014. Nó được là một mô hình được phát triển từ mạng AlexNet. Chi tiết về cấu trúc mạng và tập dữ liệu có thể xem tại đây [11]

Sau khi có được model của vgg 16 (ở đây chúng ta không huấn luyện lại mô hình, mà chỉ load lại mô hình vgg16 có sẵn rồi dùng), ta thử áp dụng SHAP cho 2 trường hợp nhận dạng 1 con dowitcher(1 loài chim) và meerkat (1 thú ăn thịt nhỏ thuộc loài cầy). Trong hình 2.5 ta vẽ SHAP plot cho 2 điểm dữ liệu này.

Trong hình 2.5 ta thấy giữa 2 lớp dowicher và real-backed sandpiper, các giá trị piexel đỏ cho dowicher nhiều hơn và các pixel này tập trung tại vùng đầu và mỏ của con chim. Từ đó mô hình vgg16 đã phân biệt dowither với các lớp khác dựa vào đặc trưng là khu vực mỏ và thân. Tương tự với trường hợp meerkat



Hình 2.5: Ví dụ 2 bộ shape value cho 2 bức ảnh con vật. Những điểm đỏ là những feature mà model cho là quan trọng (ảnh hưởng dương đến output), những chấm xanh là những feature mà model cho là ảnh hướng âm đến output (Ở đây, mỗi pixel được coi là 1 feature)

Chương 3

Kết quả

3.1 Hiệu suất mô hình

Đầu tiên, chúng tôi xem xét hiệu suất mô hình mạng neural bằng cách xem xét đầu ra lớp softmax, nó là đánh giá dễ nhất có được từ quá trình training-testing. Tuy nhiên nó vẫn bao gồm nhiều thông tin giúp chúng ta hiểu mô hình. Chúng tôi không nghiên cứu hiệu suất mô hình của toàn dải Δ, nó đã được chứng minh ở bài báo khác [4]. Thay vào đó, chúng tôi tiến hành huấn luyện với các Δ được chọn như sau (0.2, 0.34, 0.7 và 1.0) để thử xem kết quả sẽ phụ thuộc như thế nào khi Δ gần hoặc xa điểm ba. Kích thước mạng L được dùng là 12,16, 24, 32, 40, 48, 56, 64, cho phép chúng tôi xem xét hiệu ứng kích thước (finite-size) ảnh hưởng như thế nào đến kết quả.

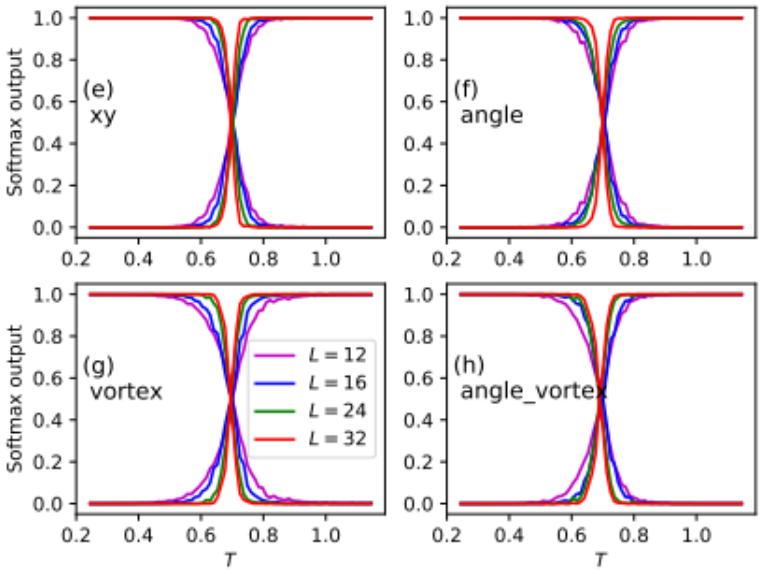
Hiện tại với mỗi Δ và mỗi L khác nhau chúng ta có model riêng biệt với 4 đầu vào khác nhau. Đó là xy, angle, vortex và angle-vortex. Cụ thể các đầu vào được giải thích bên dưới.

- Thứ nhất là đầu vào XY. Mỗi cấu hình spin (mỗi điểm dữ liệu) chúng ta có kích thước Lx L. Tại mỗi điểm trên mạng là 1 spin. Ta lấy hình chiếu của mỗi spin lên 2 phương x và y (phương ngang và phương dọc) để làm đầu vào cho model. Từ đó mỗi điểm dữ liệu nếu biến đổi đầu vào theo cách này sẽ có kích thước là LxLx2

- Thứ 2 là đầu vào angle. Mỗi cấu hình spin có kích thước Lx L. Tại mỗi spin trên 1 mạng, chúng ta lấy góc tạo bởi spin đó với trục x để làm đầu vào. Từ đó mỗi điểm dự liệu theo các này sẽ có kích thước là LxL (với cách này , hiệu suất mô hình tốt hơn cả)

- Thứ 3 là đầu vào vortex. Theo như cách bên trên, với mỗi cấu hình spin ta có 1 mảng 2 chiều LxL, mỗi điểm trên mảng là góc tạo bởi spin với trục x. Sau đó ta áp dụng cách tính vortex như sau, với 1 cụm 4 spin, ta sẽ tính ra được vortex Như vậy từ 1 mảng các góc LxL ta áp dụng quy tắc tính vortex này cộng với việc áp dụng thêm quy luật điều kiện biên tuần hoàn với mỗi điểm dữ liệu ta sẽ có 1 mảng LxL vortex

- Thứ 4 là đầu vào anglep-vortex: Với 1 điểm dữ liệu, ta biến đổi theo cách 2 được 1 mảng LxL, ta tiếp tục biến đổi theo cách 3 ta được 1 mảng khác cũng LxL.Sau đó ta ghép 2 mảng Lx L này lại với nhau ta được kiểu đầu vào là angle-vortex và dữ liệu đầu vào có kích thước là LxLx2



Hình 3.1: Giá trị trung bình của output softmax dọc theo các điểm có nhiệt độ khác nhau. Hình (d) (e) (f) (g) đều cùng là $\Delta = 0.34$. Hình (d) (e) (f) (g) tương ứng với các đầu vào là xy, angle, vortex, angle-vortex

Bây giờ ta cùng so sách 4 cách biến đổi đầu vào này xem cách nào cho hiệu suất tốt nhất. Để xem xét hiệu suất model với các cách biến đổi đầu vào khác nhau, ta vẽ đầu ra của lớp softmax (giá trị trung bình softmax của những điểm cùng nhiệt độ)với nhiệt độ. Chúng ta đánh giá hiệu suất model trên trên 2 khía cạnh đó là độ nhiễu và độc dốc của đường đầu ra softmax với nhiệt độ. Cùng quan sát hình 3.1.

Bằng mắt thường ta có thể quan sát thấy hiệu suất model cho 2 trường hợp xy và angle là đầu vào là tương đương nhau, trường hợp vortex và angle-vortex nhiều nhiễu hơn và đỡ nhiễu khi L lớn dần. Bên cạnh đó angle được suy ra từ xy, với khối lượng tính toán ít đi mà hiệu suất cũng tương đương nên chúng ta sẽ chọn angle là đầu vào chính cho những kết quả ben dưới. Ngoài ra các giá trị chuyển pha khi dùng angle làm đầu vào khá tương tự như kết quả của các phương pháp khác.

Tiếp theo sau khi đã cố định được đầu vào là góc, chúng ta xem xét hiệu suất model thay đổi như thế nào cho các trường hợp delta khác nhau. Có 4 trường hợp delta được so sánh ở đây : $\Delta = 0.2$,$\Delta = 0.34$,$\Delta = 0.7$,$\Delta = 1.0$. Quan sát hình 3.2

Như chúng ta đã biết tại $\Delta = 0.2$, khi chúng ta tăng nhiệt độ lên sẽ xảy ra 2 chuyển pha. Khi ở nhiệt độ thấp, cấu hình của chúng ta ở pha F (các spin sắp xếp cùng hướng). Khi nhiệt độ cao lên, sẽ xảy ra chuyển pha lần 1, chuyển cấu hình từ pha F sang pha N (các spin cùng phương), đây là chuyển pha Ising. Khi nhiệt độ tiếp tục tăng sẽ xảy ra chuyển từ pha N sang pha P (các spin hỗn loạn), đây là chuyển pha BKT

Với các $\Delta = 0.34$,$\Delta = 0.7$,$\Delta = 1.0$, sẽ chỉ có 1 chuyển pha đó là chuyển pha BKT từ pha F(các spin cùng chiều) sang pha P (các spin hỗn loạn).Do đó trên hình 3.2 tại $\Delta = 0.2$ ta thấy có 3 đường còn tại các Δ khác sẽ chỉ có 2 đường (vì 1 đường xác suất rơi vào pha N sẽ luôn bằng 0)

Cho các trường hợp $\Delta = 0.34$,$\Delta = 0.7$,$\Delta = 1.0$, ta thấy có 2 đường , 2 đường đó tuong ứng với xác xuất rơi vào pha F và pha P. Khi nhiệt độ thấp, xác xuất model dự đoán cấu hình là pha F cao, nên đường cho pha F sẽ đi ngang tại tung độ là 1.0,

biểu thị model khi đó sẽ dự đoán chắc chắn là cấu hình rơi vào pha F. Ngược lại đường cho pha P sẽ luôn là 0. Khi đi qua điểm chuyên pha, vị trí 2 đường này sẽ được hoán đổi. Nếu model của chúng ta đủ tốt, thì độ rộng của vùng hoán đổi này sẽ hẹp.

Ta có thể nhận xét về hình vẽ 3.2 như sau:

- Ta có thể thấy với $\Delta = 1.0$ đồ thị của chúng ta nhiều nhiễu nhất và nhiễu sẽ giảm dần khi giảm Δ dần từ 1.0 về 0.7 và 0.34.
- Khu vực chuyển pha sẽ hẹp hơn khi giảm Δ dần từ 1.0 về 0.7 và 0.34.

Từ đây ta đưa ra kết luận thứ nhất: khi cùng là chuyển BKT, thì Δ nhỏ hơn sẽ có hiệu suất mô hình tốt hơn Δ lớn

Tiếp theo khi quan sát cả 4 trường hợp Δ với các chiều dài kích thước mạng thay đổi , với tiêu chí so sánh độ nhiễu, ta có thể đưa ra kết luận thứ hai: khi cùng 1 model, L lớn hơn sẽ có hiệu suất mô hình tốt hơn. Điều này có thể được coi là hiệu ứng khi kích thước thay đổi. (finite size)

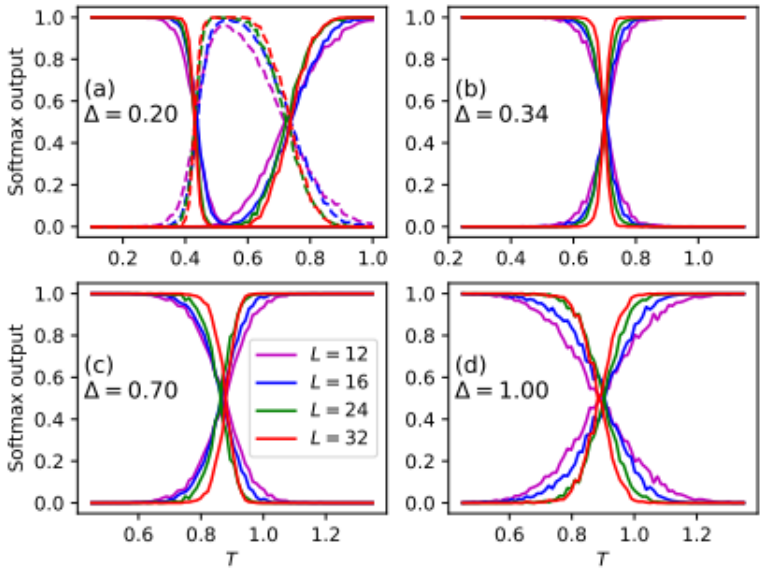
Tiếp theo quan sát hình vẽ 3.2 ,đó là trường hợp chuyển pha cho $\Delta = 0.2$. Ở điểm giao cắt thứ nhất (giao cắt bên trái) là giao giữa xác xuất nhận biết pha F với pha N. Ở điểm giao cắt thứ hai (giao cắt bên phải) là giao cắt giữa 2 xác xuất nhận biết pha N và pha F. So sánh 2 chuyển pha này, ta có thể thấy vùng giao cắt chuyển pha thứ nhất hẹp hơn vùng giao cắt chuyển pha thứ 2 và nhiễu ở vùng 1 cũng tốt hơn. Từ đó ta thấy hiệu suất cho xác định chuyển pha thứ nhất tốt hơn thứ 2, nói cách khác thì model của chúng ta nhận biết chuyển pha Ising tốt hơn chuyển pha BKT cho trường hợp 0.2. Vậy kết luận thứ 3 sẽ là mạng neural của chúng ta phân biệt chuyển pha Ising tốt hơn chuyển pha BKT

Từ góc nhìn của con người, việc phân loại chuyển pha Ising tốt hơn BKT có vẻ hiểu được. Chuyển pha từ F sang N sẽ quan sát hơn vì một người có thể nhìn sự thay đổi của spin nếu chỉ có sự đổi hướng mà chiều vẫn giữ nguyên. Chuyển pha P sang F sẽ khó quan sát hơn vì nó đi từ hệ spin hỗn loạn không xắp xếp chuyển sang hệ spin song song. Do đó việc không trích xuất đặc trưng ,cụ thể qua việc lấy góc làm dữ liệu đầu vào thì model sẽ có xu hướng phân loại giống cách con người quan sát, tất nhiên máy thì hiệu quả hơn và chính xác hơn.

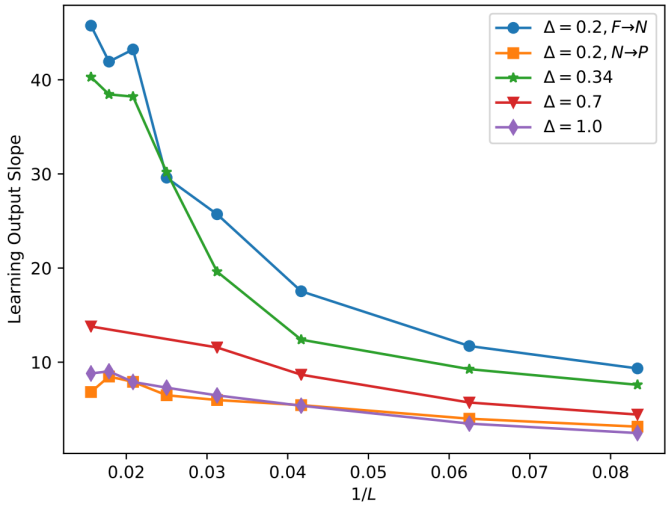
3.2 Độ dốc của hiệu suất máy học

Ta đã có được các đường trung bình đầu ra softmax theo nhiệt độ với các trường hợp Δ và các L khác nhau. Như chúng ta đã nhận xét, hiệu suất mô hình học sâu tốt biểu hiện ở độ dốc vùng chuyển pha. Độ dốc càng cao nghĩa là hiệu suất model càng tốt, lý tưởng nhất là góc thẳng đứng, độ dốc là 90 độ. Ta vẽ đồ thị của độ dốc các mô hình với các Δ khác nhau theo $\frac{1}{L}$, quan sát hình 3.3

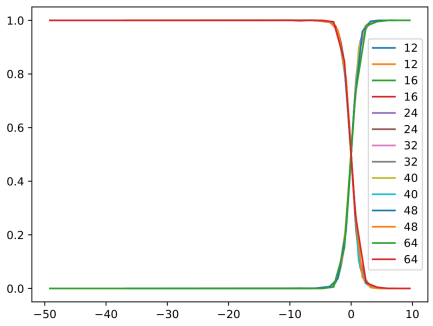
Từ hình vẽ 3.3 với các trường hợp delta khác nhau cho các L, ta thấy 3 điều: Thứ nhất, khi so sánh cùng 1 Δ,khi L càng lớn thì độ dốc càng lớn. Nói cách khác, khi L càng lớn thì hiệu suất các mô hình tốt hơn. Thứ hai, so sánh các Δ khác nhau với cùng 1 độ lớn L, chuyển pha tại $\Delta = 0.2$ từ F sang N có độ dốc lớn nhất. Vậy có vẻ hiệu suất mô hình cho nhận biết chuyển pha ISING tốt nhất.Thứ ba, hiệu suất của mô hình cho trường hợp $\Delta = 0.2$ từ N sang P xấp xỉ với hiệu suất của mô hình khi $\Delta = 1.0$



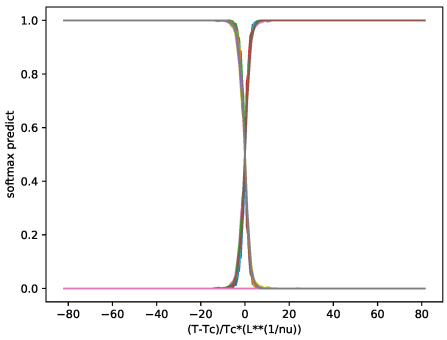
Hình 3.2: Giá trị trung bình của output softmax dọc theo các điểm có nhiệt độ khác nhau. Cả 4 hình đều dùng góc là đầu vào. Hình (a) (b) (c) (d) tương ứng với các $\Delta = 0.2, 0.34, 0.7$ và 1.0.



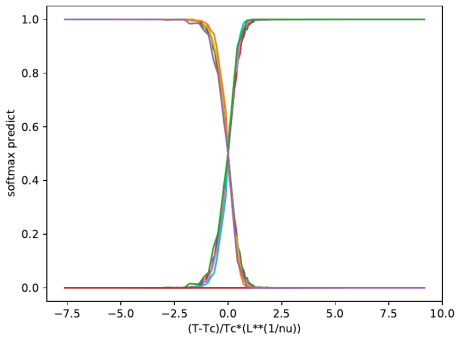
Hình 3.3: Độ dốc vùng chuyển pha của các model với các $\Delta = 0.2, 0.34, 0.7$ và 1.0 và các L khác nhau



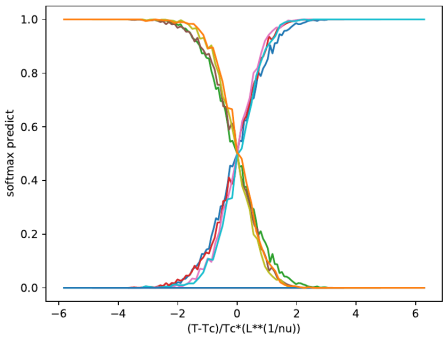
(a) $\Delta = 0.2$, $\eta = 1.0$



(b) $\Delta = 0.34$, $\eta = 0.8$



(c) $\Delta = 0.7$, $\eta = 1.5$



(d) $\Delta = 1.0$, $\eta = 1.4$

Hình 3.4: Hình vẽ trung bình đầu ra của lớp softmax theo $\frac{(T-Tc)}{Tc} \times L^{\frac{1}{\eta}}$ với 4 trường hợp Δ khác nhau.

3.3 Tỉ đối kích thước

Chúng tôi cũng xem xét hiệu ứng tỷ đối kích thước (finite-size scaling) với các $\Delta = 0.2, 0.34, 0.7$ và 1.0 Từ các hình trung bình đầu ra lớp soft max theo nhiệt độ bên trên, trục x là T, ta biến đổi T thành $\frac{(T-Tc)}{Tc} \times L^{\frac{1}{\eta}}$, còn trục tung giữ nguyên vẫn là trung bình đầu ra tại lớp softmax.

Ta sẽ thay đổi η cho đến khi nào các đường ứng với L khác nhau gần chồng chập vào nhau nhất. Trên hình vẽ 3.4, ta vẽ tất cả các kích thước trên một hình tại η mà các đừng ứng với các L gần trùng vào nhau nhất

Ta thay đổi η từ 0.1 đến 2.0, sau đó quan sát tại η bằng bao nhiêu thì các đường chập lại làm 1. Với $\Delta = 0.2$ ứng với $\eta = 1$ thì các đường gần chập vào nhau. $\Delta = 0.34$ thì $\eta = 0.8$ là các đường gần nhau nhất. $\Delta = 0.7$ thì $\eta = 1.5$ là các đường gần nhau nhất. $\Delta = 1.0$ thì $\eta = 1.4$ là các đường gần nhau nhất

Trong quá trình điều chỉnh η, tôi nhận thấy, với các trường hợp $\Delta = 0.34$,$\Delta = 0.7$,$\Delta = 1.0$, các trường hợp này khó tìm được η để cho các đường trung bình lớp softmax theo các L gần nhau nhất. Có lẽ trong các trường hợp Δ mà chuyển pha là chuyển pha BKT thì cần 1 công thức finite size scaling khác thay vì công thức $\frac{(T-Tc)}{Tc} \times L^{\frac{1}{\eta}}$ cho trường hợp chuyển pha là chuyển pha ISING.

Những kết quả của kết quả đầu ra của mô hình học sâu là thú vị khi so sánh với mô phỏng Monte Carlo , cách chúng ta đã dùng để tạo ra dữ liệu đầu vào. Khi sử dụng mô phỏng Monte Carlo, các tính toán tại 2 đầu $\Delta = 0$ và $\Delta = 1$ dễ dàng

hội tụ, trong khi các tính toán xung quanh khu vực điểm ba cần nhiều thời gian mô phỏng để có được kết quả phù hợp do bị cạnh tranh giữa 2 tương tác. Phương pháp học sâu cho một xu hướng ngược lại: $\Delta = 1.0$ đầu ra lớp softmax nhiều nhiễu và độ rộng vùng chuyển pha lớn, khi gần vùng điểm ba, các đường cong mượt hơn và độ rộng chuyển pha hẹp hơn. Chúng tôi diễn giải điều này từ cách mỗi phương pháp hoạt động: được cho dữ liệu đã hội tụ, học sâu chỉ quan tâm đến việc dữ liệu ảnh đầu vào , trong khi mô phỏng Monte Carlo , việc cập nhật cho cấu hình spin khó xảy ra hơn khi Δ gần điểm ba

3.4 Nghiên cứu 32 neuron layer gần cuối

Chúng ta đã khảo sát đầu ra của lớp cuối của mạng, lớp softmax, bằng cách vẽ trung bình đầu ra theo nhiệt độ. Bây giờ chúng ta dịch về phía trước 1 lớp nữa, đó là lớp 32 neuron gần cuối. Chúng ta phân tích đầu ra của hàm kích hoạt của mỗi neuron ở lớp này như là cách đầu tiên trong nỗ lực hiểu mô hình của chúng ta đã làm gì. Với mục đích này, chúng tôi áp dụng mô hình đã huấn luyện cho tất cả các điểm trong tập dữ liệu test và tính hàm kích hoạt cho mỗi neuron. Cụ thể chúng ta tính trung bình tất cả các đầu ra cho từng neuron với, lấy trung bình trên toàn tập dữ liệu test (cách tiếp giống với cách vẽ trung bình softmax theo nhiệt độ thì chúng ta sẽ lấy trung bình đầu ra mỗi neuron lớp gần cuối theo nhiệt độ). Hình 3.5 chỉ ra trung bình hàm kích hoạt mỗi neuron lớp này theo nhiệt độ, với các trường hợp Δ khác nhau và các L khác nhau. Dựa trên những hàm kích hoạt này, chúng ta chia các neuron này thành 2 loại là "lazy"và "active".

Các lazy neuron sẽ không hoạt động, đầu ra của nó luôn bằng không trên toàn dải nhiệt độ. Các active neuron thì sẽ hoạt động như 1 chiếc công tắc tắt bật, trước nhiệt độ $T < Tc$ nó tắt, qua nhiệt độ Tc nó được bật lên hoặc ngược lại. Tuy nhiên có 1 độ trễ là có thể active neuron không tắt bật ngay đúng tại nhiệt độ chuyển pha. Chúng ta cũng thấy rằng có một vài neuron không biến mất hoàn toàn khi chuyển trạng thái.

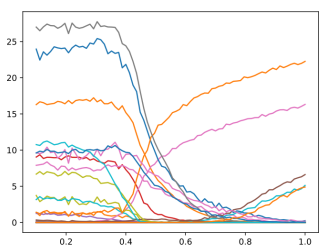
Chúng tôi cũng mong muốn rằng số lượng lazy neuron có hiệu ứng kích thước thay đổi. Từ đó chúng tôi đếm số lượng lazy neuron cho các trường hợp Δ và các L khác nhau , ta được bảng 3.1 ,nhưng số lượng lazy neuron phụ thuộc vào L lại theo 1 quy luật không rõ ràng. Nó có thể rằng cho mỗi lần chúng tôi huấn luyện mô hình với 1 giá trị L nhất định, tối ưu hóa cho hàm mất mát dẫn đến kết quả sự khác loại của cực tiểu hóa địa phương.

Chúng tôi cũng chú ý rằng trong bảng 3.1 rằng cho tất cả các kích thước mạng L được tính, $\Delta = 0.34$ thường xuyên có số lượng lazy neuron nhỏ nhất (ngoại trừ trường hợp $L = 56$). Chúng tôi biết rằng tại $\Delta = 0.34$ là gần vùng điểm ba, nơi cạnh tranh giữa tương tác pha F và pha N, do đó nó có thể nhiều active neurons để mô tả các tính chất vật lý nào đó. Ở mặt đối lập khi $\Delta \to 1$ hoặc $\Delta \to 0$, ở đó chỉ có 1 tương tác thống trị, do đó chỉ cần 1 số lượng nhỏ của các active neuron là đủ.

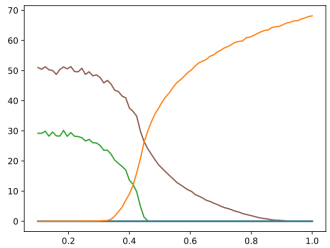
Chúng tôi cũng đã khảo sát các neurons trong lớp CNN, tuy nhiên các mẫu của 16 filters không đủ rõ ràng để có 1 kết luận chặt chẽ về cách neuron học từ ảnh dữ liệu.

L	$\Delta = 0.2$	$\Delta = 0.34$	$\Delta = 0.7$	$\Delta = 1.0$
12	8	6	5	12
16	7	11	13	14
24	11	14	16	19
32	11	9	14	8
40	18	12	NA	19
48	21	18	NA	19
56	17	19	NA	20
64	29	16	23	28

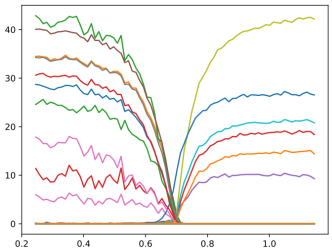
Bảng 3.1: Bảng số lượng lazy neuron theo Δ và L



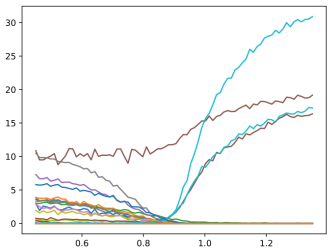
(a) $\Delta = 0.2, L = 32$



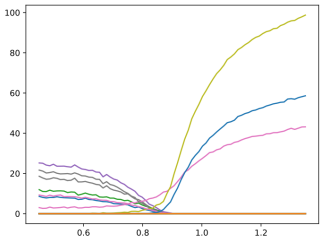
(b) $\Delta = 0.2, L = 64$



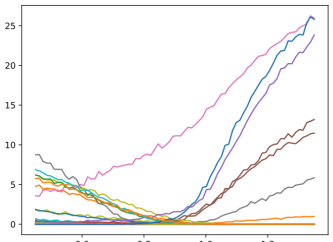
(c) $\Delta = 0.34, L = 64$



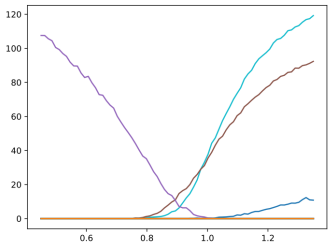
(d) $\Delta = 0.7, L = 32$



(e) $\Delta = 0.7, L = 64$



(f) $\Delta = 1.0, L = 32$



(g) $\Delta = 1.0, L = 64$

Hình 3.5: Trung bình 32 output neuron theo nhiệt độ. Trục x là nhiệt độ, trục y là giá trị trung bình tại tất cả các điểm có cùng 1 nhiệt độ ứng với trục x

3.5 Tính giá trị SHAP cho mô hình học sâu

Từ ví dụ trên, chúng ta thấy SHAP khá trực quan, chúng ta hy vọng nó cũng có ý nghĩa cho tập dữ liệu của chúng ta. Để tiến hành dùng SHAP, chúng ta dùng thư viện SHAP framework và chạy trên google colab (kết quả của SHAP plot có dùng đến javascipt nên chúng ta tiến hành chạy trên trình duyệt)

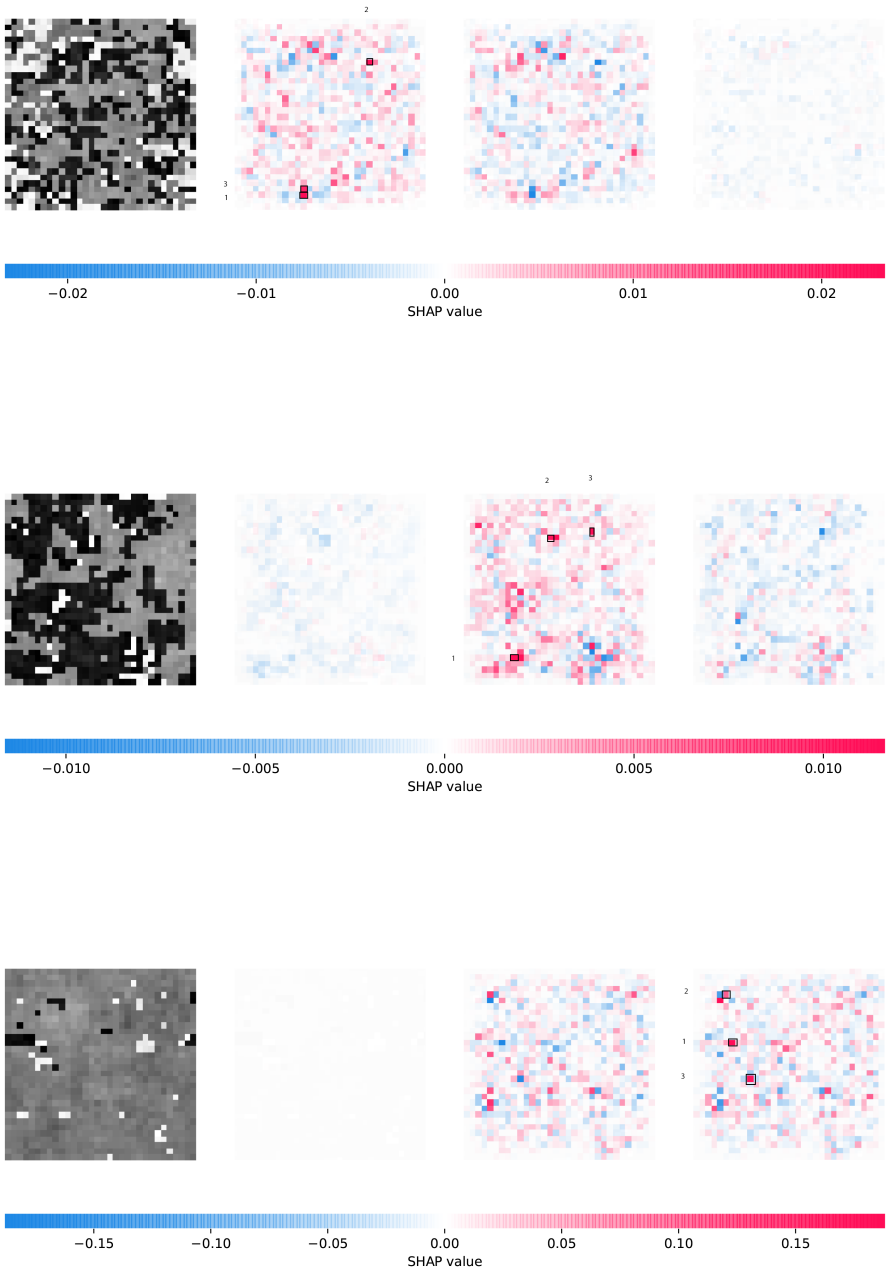
Phân tích bên trên dựa trên hiệu suất của mô hình và kết quả đầu ra của mỗi neuron, những phân tích bên trên chỉ ra đằng sau kết quả của mô hình học sâu cũng bao gồm các tính chất vật lý của hệ như là hiệu ứng kích thước thay đổi, chuyển pha và sự khác biệt giữa các loại chuyển pha. Tuy nhiên những phân tích ở trên là chưa đủ cho hiểu học sâu đã hoạt động như thế nào. Trong thực tế, bởi vì số lượng lớn của dữ liệu đầu vào, diễn dịch mô hình dự trên mô hình dựa trên kết quả đầu ra của mỗi neural có vẻ không là cách tiếp cận tốt, chúng cần một cách tiếp cận nhiều tính thống kê hơn để phân tích model của chúng ta. Bằng việc sử dụng SHAP, chúng ta thu được một model gải thích (explanation model) , model giải thích là model xấp xỉ của model học sâu ban đầu. Model xấp xỉ có kết quả đầu ra là tổ hợp tuyến tính của các đặc trưng của điểm dữ liệu (data point). Trong trường hợp của chúng ta, đặc trưng là góc cho spin trong cấu hình. Chúng tôi vẽ SHAP cho $\Delta = 0.2$ và $L = 32$ làm ví dụ (3.6), chúng tôi vẽ SHAP cho 3 trường hợp tương ứng với 3 pha : sắt từ, nematic, và paramagnetic (các đặc trưng là các góc của các spin) . Với hình vẽ SHAP cho mỗi pha, điểm ảnh (pixel, từ đây ta gọi điểm ảnh này là pixel để phân biệt với điểm dữ liệu (1 cấu hình spin)) với màu đỏ (giá trị dương) hoặc pixel màu xanh (giá trị âm) tương ứng với việc đặc trưng tương ứng đóng góp dương hay âm vào explaination model.

Trong hình 3.6, chúng ta chọn ra 3 điểm để vẽ SHAP, điểm đầu tiên (bên dưới cùng của hình) là điểm có nhiệt độ gần nhiệt độ chuyển pha Tc1, T<Tc1 (Tc1 là nhiệt độ chuyển pha giữa pha F và pha N); điểm thứ 2 (nằm giữa hình) là điểm có nhiệt độ nằm giữa Tc1 và Tc2 (Tc1 là nhiệt độ chuyển pha giữa pha N và pha P); điểm thứ 3 là điểm có có nhiệt độ nằm sau nhiệt độ chuyển pha Tc2.

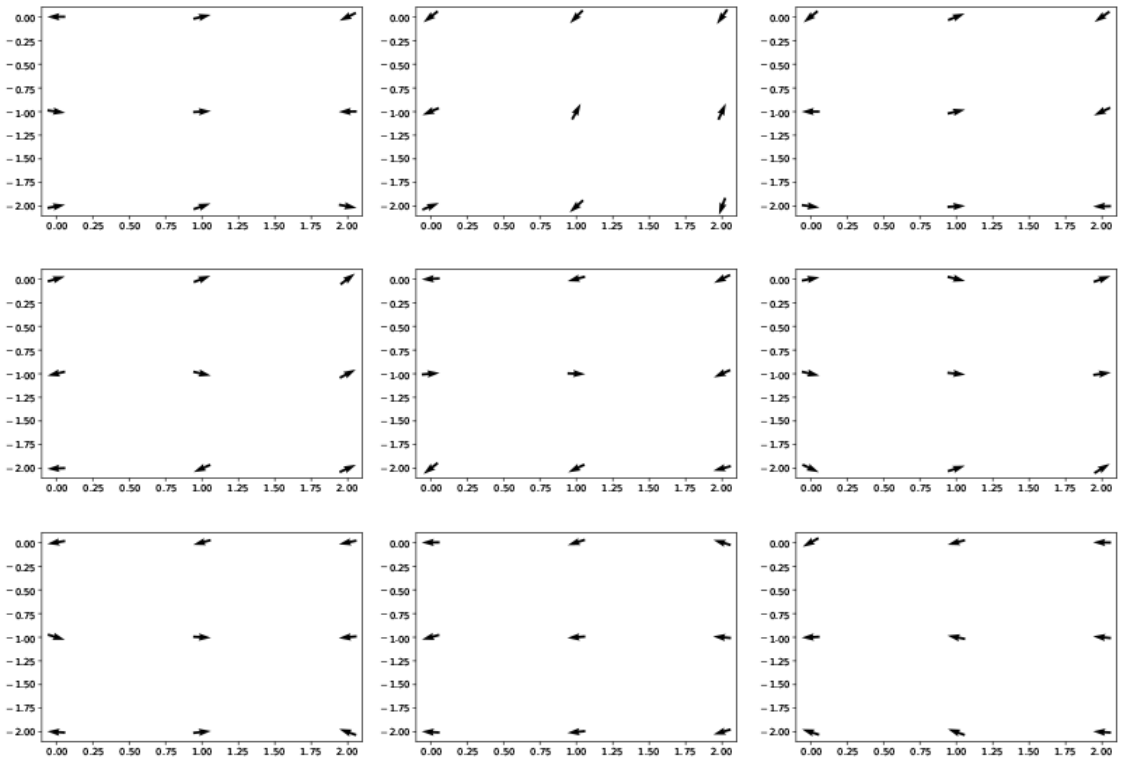
Ở điểm dữ liệu thứ nhất, điểm có T<Tc1 và gần với Tc1, ta thấy hình SHAP cho 2 pha F và pha N nổi bật lên, trong trường hợp này học sâu dự đoán điểm dữ liệu này thuộc về pha F. Ta thử tập trung vào những pixel màu đỏ cho pha F, để xem tại sao mô hình của chúng ta lại đánh trọng số cho nó cao như vậy, khi model muốn phân biệt nó là pha F so với các pha còn lại. Tuy nhiên việc tập trung vào 1 pixel , việc đó lại không mang lại ý nghĩa gì vì tương ứng mỗi pixel chỉ là 1 góc của spin, vì vậy chúng tôi sẽ quan sát pixel có màu đỏ và những pixel xung quanh điểm màu đỏ ấy (quan sát ma trận 3x3 với tâm ma trận là pixel màu đỏ đó). Ta chọn 3 pixel màu đỏ có độ sáng lớn nhất để xem khu vực đó có gì (Hình vẽ 3.7)

Ở điểm dữ liệu thứ 2, khi mô hình của chúng ta dự đoán điểm dữ liệu này thuộc pha N, tương tự như điểm dữ liệu thứ nhất, ta cũng khoanh vùng những vùng màu đỏ của SHAP với pha N.Ở điểm dữ liệu thứ 3, khi mô hình của chúng ta dự đoán điểm dữ liệu này thuộc pha P, tương tự như điểm dữ liệu thứ nhất, ta cũng khoanh vùng những vùng màu đỏ của SHAP với pha P

Sau khi đã khoanh vùng các điểm nổ bật, ta vẽ các cấu hình spin tại các khu vực đó để cùng xem cách mà mô hình học sâu của chúng ta đã học (3.7). Ở đây chúng ta chọn khu vực quan sát là ma trận 3x3 bởi vì kích thức của kernel CNN của chúng ta có kích thước là 3x3 nên ta kỳ vọng mô hình của chúng ta sẽ nhìn cấu



Hình 3.6: Hình vẽ SHAP cho trường hợp $\Delta = 0.2$ và $L = 32$. Từ dưới lên trên : điểm tại T<Tc1 (pha F), điểm tại Tc1<T<Tc2 (pha N), điểm tại Tc2<T (pha P). Cột đầu tiên là hình vẽ spin với đầu vào là góc, các cột tiếp theo từ trái qua phải tương ứng với SHAP của các pha P,N và F



Hình 3.7: Hình vẽ các cấu hình spin quanh các khu vực được bôi đỏ trong các hình SHAP cho trường hợp $\Delta = 0.2$ và $L = 32$. Mỗi hàng có 3 cột đại diện cho 3 điểm màu đỏ nhất. Từ dưới lên trên, hàng cuối tương ứng 3 điểm đỏ nhất của SHAP pha F cho điểm dữ liệu T<Tc1; hàng giữa là 3 điểm đỏ nhất của SHAP cho pha N cho điểm dữ liệu Tc1<T<Tc2; hàng đầu là 3 điểm đỏ nhất cho SHAP pha P cho điểm dữ liệu T>Tc2

hình spin theo từng cụm 3x3.

Cùng quan sát hình 3.7, quan sát hàng cuối cùng, đó là trường hợp T<Tc1, model đã dự đoán điểm dữ liệu này thuộc pha F (các spin cùng chiều), ta thấy 3 spin được vẽ có xu hướng các spin cùng chiều. Vậy cách model dự đoán pha F đó là đánh trọng số cho các cụm spin mà ở đó chúng cùng chiều. Tiếp theo ta quan sát hàng ở giữa của hình 3.7, model đã dự đoán điểm dữ liệu này là pha N (các spin cùng phương), có vẻ model sẽ dự đoán pha N bằng các cụm spin cùng phương. Cuối cùng là hàng trên cùng, trường hợp này model của chúng ta kết luận điểm dữ liệu đó thuộc pha P (các spin hỗn loạn), ta thấy có vẻ chúng hỗn loạn , có lẽ mô hình của chúng ta quan sát các cụm mà spin hỗn loạn để phân biệt pha P. Tuy nhiên ở hàng 1, cột 1, ta thấy các spin có vẻ là cùng phương , có vẻ ở đây nó đã đánh trọng số không tốt.

Vậy model của chúng ta học từ việc quan sát các cụm spin nhỏ trong 1 cấu hình spin lớn, sau đó phân loại cụm đó, đánh trọng số cho nó, rồi cộng tất cả các trọng số lại rồi cộng với giá trị kỳ vọng của toàn bộ tập dữ liệu, từ đó ta được kết quả chính là xác suất đầu ra softmax đã tính ở bên trên.

$$sofmax\_output\_P = explain\_model.expected\_value[0]+np.sum(SHAP\_values[0])$$

$$sofmax\_output\_N = explain\_model.expected\_value[0]+np.sum(SHAP\_values[1])$$

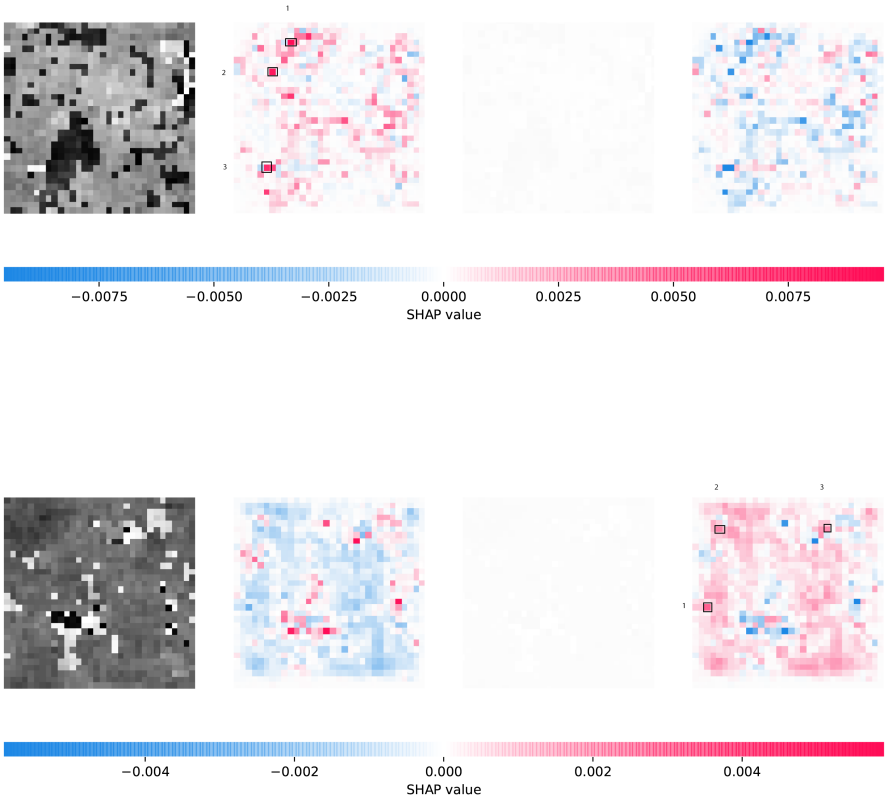
$$sofmax\_output\_F = explain\_model.expected\_value[2]+np.sum(SHAP\_values[2])$$

Ta tiếp tục quan sát cho trường hợp $\Delta = 0.34$ và $L = 32$. Khi $Delta = 0.34$ lúc này chỉ có cạnh tranh của 2 pha nên hình ảnh SHAP cho pha N luôn không có gì. Ta tiếp tục chọn điểm dữ liệu trước và sau chuyển pha để quan sát 3.8. Hàng trên đầu là điểm dữ liệu sau chuyển pha, model dự đoán điểm này có pha F và SHAP cho pha F được nổi bật hẳn lên. Hàng cuối là điểm dữ liệu trước chuyển pha, model dự đoán điểm này có pha P và SHAP cho pha P được nổi lên. Tương tự cách tiếp cận ở trên, ta tiếp tục khoanh vùng các màu đỏ và vẽ các cụm spin (hình 3.9)

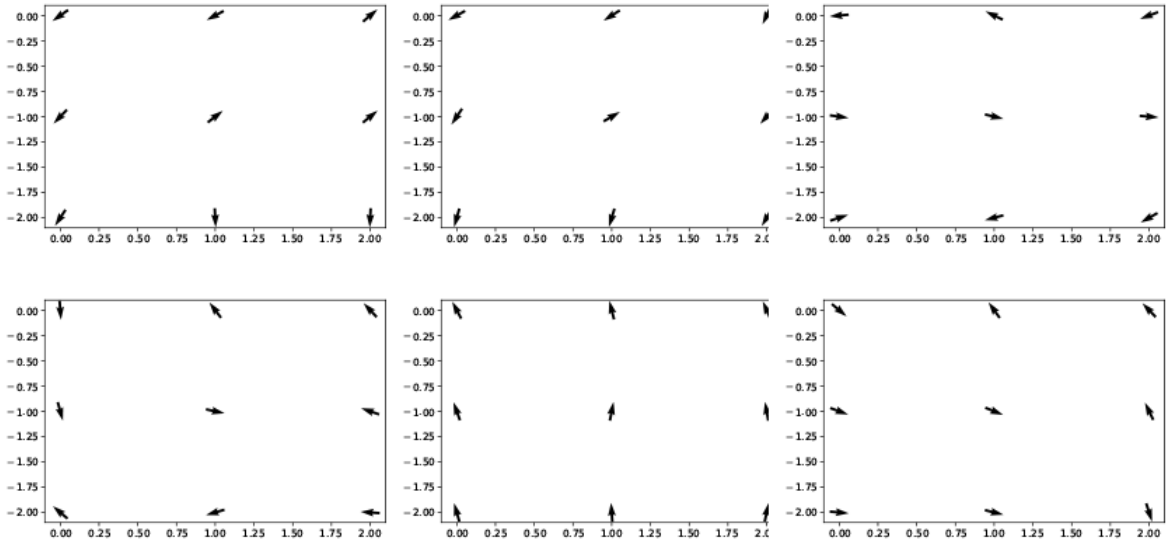
Ở trường hợp $\Delta = 0.34$ do chỉ có cạnh tranh giữa 2 pha nên SHAP chỉ nổi bật cho F hoặc P, còn SHAP cho N luôn bằng 0. Nếu 1 pixel có màu đỏ trong SHAP cho pha F thì cũng tại vị trí pixel đó thì nó sẽ có màu xanh cho SHAP P và ngược lại.

Trong hình 3.9, hàng 1 là 3 cụm đỏ của điểm có pha P, hàng 2 là 3 cụm này đỏ của điểm có pha F.

Cùng quan sát hình 3.9, quan sát hàng cuối cùng, đó là trường hợp T<Tc, model đã dự đoán điểm dữ liệu này thuộc pha F (các spin cùng chiều), ta thấy 3 spin được vẽ có xu hướng các spin cùng chiều. Vậy cách model dự đoán pha F đó là đánh trọng số cho các cụm spin mà ở đó chúng cùng chiều. Nhưng hình đầu tiên hàng 2 các spin lại khá hỗn loạn. . Cuối cùng là hàng trên cùng, trường hợp này model của chúng ta kết luận điểm dữ liệu đó thuộc pha P (các spin hỗn loạn), ta thấy có vẻ chúng hỗn loạn , có lẽ mô hình của chúng ta quan sát các cụm mà spin hỗn loạn để phân biệt pha P.



Hình 3.8: Hình vẽ SHAP cho trường hợp $\Delta = 0.34$ và $L = 32$. Từ dưới lên trên : điểm tại T<Tc (pha F), điểm tại Tc<T (pha P). Cột đầu tiên là hình vẽ spin với đầu vào là góc, các cột tiếp theo từ trái qua phải tương ứng với SHAP của các pha P và F



Hình 3.9: Hình vẽ các cấu hình spin quanh các khu vực được bôi đỏ trong các hình SHAP cho trường hợp $\Delta = 0.34$ và $L = 32$. Mỗi hàng có 3 cột đại diện cho 3 điểm màu đỏ nhất. Từ dưới lên trên, hàng cuối tương ứng 3 điểm đỏ nhất của SHAP pha F cho điểm dữ liệu T<Tc; hàng đầu là 3 điểm đỏ nhất cho SHAP pha P cho điểm dữ liệu T>Tc

Chương 4

Kết luận

Chúng ta đã cùng đi qua các bước để tiếp cận để khảo sát 1 bài toán vật lý sử dụng học sâu. Sau khi qua các bước xây dựng mô hình học sâu, chúng ta đã nhận xét và thử tìm cách diễn giải mô hình học sâu của chúng ta.

Lợi thế của học sâu là sử dụng dữ liệu thay cho các thuật toán phức tạp, chưa kể những thuật toán này cần phải thay đổi khi bài toán thay đổi hoặc dữ liệu khác đi. Với cách tiếp cận học sâu , cụ thể là các mạng neural, dữ liệu chính là các phương tiện sử dụng thay thế thuật toán, tuy nhiên phải cần dữ liệu với số lượng lớn (big data). Hy vọng trong tương lai chúng ta có thể áp dụng học sâu hiệu quả vào nhiều bài toán vật lý hơn nữa , thúc đẩy quá trình nghiên cứu, như cái cách mà học sâu đang thay đổi thế giới.

Chúng ta đã khảo sát việc áp dụng deep learning khảo sát xy model tổng quát. Có chỗ chúng ta chưa lý giải được, nhưng về cơ bản, đây là 1 cách tiếp cận mạnh mẽ và nhiều hứa hẹn

Tài liệu tham khảo

[1] The Nobel Prize in Physics 2016 https://www.nobelprize.org/prizes/physics/2016/press-release/.

[2] Kosterlitz Thouless transition https://en.wikipedia.org/wiki/Kosterlitz\T5\textendashThouless_transition.

[3] Duong Xuan Nui, Le Tuan, Nguyen Duc Trung Kien, Pham Thanh Huy, Hung T. Dang, and Dao Xuan Viet. Correlation length in a generalized two-dimensional xy model. *Phys. Rev. B*, 98:144421, Oct 2018.

[4] Wanzhou Zhang, Jiayu Liu, and Tzu-Chieh Wei. Machine learning of phase transitions in the percolation and xy models. *arXiv preprint arXiv:1804.02709*, 2018.

[5] Li Huang and Lei Wang. Accelerated monte carlo simulations with restricted boltzmann machines. *Phys. Rev. B*, 95:035105, Jan 2017.

[6] Accelerate ab initio molecular dynamics https://www.researchgate.net/publication/269936179_Adaptive_machine_learning_framework_to_accelerate_ab_initio_molecular_dynamics.

[7] Juan Carrasquilla and Roger G Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431, 2017.

[8] Matthew J. S. Beach, Anna Golubeva, and Roger G. Melko. Machine learning vortices at the kosterlitz-thouless transition. *Phys. Rev. B*, 97:045207, Jan 2018.

[9] Tensorflow Framework https://www.tensorflow.org.

[10] Shap Framework Github https://github.com/slundberg/shap.

[11] Vgg16 Network https://neurohive.io/en/popular-networks/vgg16/.