

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO BÀI TẬP LỚN MÔN:**

**HỌC MÁY VỚI DỮ LIỆU LỚN**

*Đề tài***:**

**Why does Unsupervised Pre-training Help**

**Deep learning**

*Giảng viên hướng dẫn*: PGS.TS. Thân Quang Khoát

*Sinh viên thực hiện*:

Nguyễn Viết Việt : CA190037

Nguyễn Đức Thắng : CA190043

Nguyễn Tiến Thạo : CA190044

Hoàng Nghĩa Bảo : CB190208

*HÀ NỘI: 04/2020*

# **Mục lục**

[Mục lục 1](#_Toc38116707)

[Danh mục hình ảnh 3](#_Toc38116708)

[1. Giới thiệu 4](#_Toc38116709)

[2. Những thách thức của Deep Learning 6](#_Toc38116710)

[3. Xem xét mô hình đã huấn luyện không giám sát như regularization 7](#_Toc38116711)

[4. Các nghiên cứu liên quan 10](#_Toc38116712)

[4.1. Học bán giám sát 10](#_Toc38116713)

[4.2. Early stopping là tiêu chuẩn của Regularization 11](#_Toc38116714)

[5. Cài đặt và thử nghiệm 12](#_Toc38116715)

[5.1. Mô hình 12](#_Toc38116716)

[5.1.1. Deep Belief Networks 13](#_Toc38116717)

[5.1.2. Stacked Denoising Auto-encoders 13](#_Toc38116718)

[5.2. Mô tả dữ liệu 14](#_Toc38116719)

[6. Ảnh hưởng của unsupervised pre-training 14](#_Toc38116720)

[6.1. Tổng quát hóa tốt hơn 15](#_Toc38116721)

[7. Vai trò của unsupervised pre-training 16](#_Toc38116722)

[7.1. Thí nghiệm 1: Pre-training có giúp cho Supervised Learning có một quá trình điều kiện (condition process) tốt hơn không ? 16](#_Toc38116723)

[7.2. Thí nghiệm 2: Ảnh hưởng của pre-training lên training error 17](#_Toc38116724)

[7.3. Thí nghiệm 3: Ảnh hưởng của số lượng hidden layers 18](#_Toc38116725)

[7.4. Thí nghiệm 4: Kiểm chứng giả thiết pre-training as an optimization scheme 19](#_Toc38116726)

[8. Online Learning 20](#_Toc38116727)

[8.1. Ảnh hưởng của pre-training đối với tập dữ liệu rất lớn 20](#_Toc38116728)

[9. Thực nghiệm đánh giá 21](#_Toc38116729)

[9.1. Giới thiệu bài toán 21](#_Toc38116730)

[9.2. Cài đặt thực nghiệm 26](#_Toc38116731)

[9.3. Kết quả thực nghiệm của nhóm 27](#_Toc38116732)

[9.3.1. Đánh giá kết quả lỗi tổng quan 27](#_Toc38116733)

[10. Tổng kết 30](#_Toc38116734)

[Tài liệu tham khảo 31](#_Toc38116735)

# **Danh mục hình ảnh**

[Hình 1: Early stopping. Đường màu xanh là train error, đường màu đỏ là validation error 9](#_Toc38116660)

[Hình 2: Sử dụng pre-trainning trong Deep Learning task 12](file:////Users/vietnguyen/Documents/Projects/Master/Report_v1.docx#_Toc38116661)

[Hình 3: Phân bố của test classification error của model được unsupervised pre-training (trái) và model được unsupervised pre-training (phải) với 400 bộ tham số trọng số ngẫu nhiên 15](#_Toc38116662)

[Hình 4: Histogram của test errors trên tập data MNIST sử dụng model unsupervised pre-training và model bình thường với số lượng hidden layers là 1 (hình trái) và 4 (hình phải) 16](#_Toc38116663)

[Hình 5: Kết quả của 2 hai loại model với/với không unsupervised pre-training sử dụng 400 bộ trọng số khởi tạo ngẫu nhiên 17](#_Toc38116664)

[Hình 6: Sử ảnh hưởng của unsupervised pre-training khi số lượng hidden layers tăng 18](#_Toc38116665)

[Hình 7: Kết quả của khi thí nghiệm khi không sử dụng early stopping 19](#_Toc38116666)

[Hình 8: So sánh các mô hình có 1 và 3 layers trên bộ dữ liệu InfiniteMNIST 20](#_Toc38116667)

[Hình 9: Minh hoạ bài toán co giãn tài nguyên 22](#_Toc38116668)

[Hình 10: Phương pháp sử dụng chu kỳ 23](#_Toc38116669)

[Hình 11: Co giãn tài nguyên sử dụng ngưỡng 24](#_Toc38116670)

[Hình 12: Co giãn tài nguyên bằng phương pháp dự đoán 25](#_Toc38116671)

[Bảng 1: Ảnh hưởng của từng chiến lược khởi tạo tham số. Giá trị từng ô trong bảng là mean và std của test errors 18](#_Toc38115749)

[Bảng 2: Phân tích lỗi của 2 mô hình 28](#_Toc38115750)

# Giới thiệu

Các mô hình học sâu tập trung vào việc học các features phân cấp với các features cấp cao được cấu trúc bởi các feature cấp thấp hơn.

Chúng bao gồm các phương pháp học cho các kiến trúc sâu: Bao gồm mạng neural với nhiều lớp ẩn, các mô hình đồ thị với nhiều mức độ biến ẩn.

Các kết quả lý thuyết, được đánh giá và thảo luận bởi Bengio và LeCun khuyến nghị rằng để học các hàm phức tạp mà có thể được biểu diễn với mức độ trừu tượng cao có thể cần thiết tới các kiến trúc sâu.

Nhiều nghiên cứu chỉ ra rằng các công việc liên quan tới Computer vision, xử lý ngôn ngữ tự nhiên hay truy xuất thông tin - các mô hình deep learning đã đem lại kết quả tốt.

Bước đột phá trong các chiến lược huấn luyện các mô hình học sâu xuất hiện năm 2006 với sử xuất hiện của Deep belief networks và stacked auto-encoders dựa trên cách tiếp cận giống nhau: Đào tạo trước một mô hình không giám sát và theo sau đó là tinh chỉnh mô hình có giám sát. Mỗi lớp được đào tạo trước với một thuật toán học tập không giám sát, học một phép biến đổi phi tuyến của đầu vào của nó (đầu ra của lớp trước) để nắm bắt các biến thể chính trong đầu vào của nó. Việc đào tạo trước không được giám sát này tạo tiền đề cho giai đoạn đào tạo cuối cùng trong đó kiến ​​trúc sâu được điều chỉnh phù hợp với tiêu chí đào tạo được giám sát với tối ưu hóa dựa trên độ dốc. Trong khi sự cải thiện về hiệu suất của các mô hình sâu được đào tạo được cung cấp bởi chiến lược đào tạo trước là rất ấn tượng, thì ít ai hiểu về các cơ chế làm nên thành công này.

Mục tiêu của nghiên cứu:

* Mục tiêu của bài viết này là khám phá, thông qua thử nghiệm sâu rộng, làm thế nào unsupervised pre-training làm cho việc học kiến ​​trúc sâu hiệu quả hơn và tại sao chúng có vẻ hoạt động tốt hơn nhiều so với các phương pháp đào tạo mạng lưới thần kinh truyền thống. Có một vài giả thuyết hợp lý tại sao unsupervised pre-training có thể hoạt động. Một khả năng là đào tạo trước không được giám sát hoạt động như một loại tiền xử lý mạng, đưa các giá trị tham số vào phạm vi thích hợp để đào tạo được giám sát thêm. Một khả năng khác, được đề xuất bởi Bengio et al. (2007), là unsupervised pre-training sẽ khởi tạo mô hình đến một điểm trong không gian tham số bằng cách nào đó làm cho quá trình tối ưu hóa hiệu quả hơn, theo nghĩa là đạt được mức tối thiểu thấp hơn của hàm chi phí thực nghiệm.
* Bài viết nhóm nghiên cứu lập luận ràng các thử nghiệm của nhóm cung cấp một góc nhìn về việc unsupervised pre-training hoạt động giống như một mô hình regularization: giảm thiểu phương sai và định hướng cấu hình các không gian tham số giúp cho quá trình unsupervised learning. Theo góc nhìn này sẽ đặt unsupervised pre-training vào cùng tập với các phương pháp phát triển bán giám sát gần đây. Tuy nhiên điểm khác biệt duy nhất của unsupervised pre-training so với các phương pháp học bán giám sát khác là nó hoạt động bằng cách xác định một điểm khởi tạo cụ thể cho các mô hình học có giám sát thay vì sửa đổi hàm mục tiêu của mô hình có giám sát (Barron, 1991) hoặc áp đặt rõ ràng các ràng buộc đối với thông số trong suốt quá trình đào tạo (Lasserre et al., 2006). Chiến lược về khởi tạo giống như regularization đã được ưu tiên trong các mạng nơron dưới hình thức của ý tưởng early stopping (Sjoberg and Ljung, 1995; Amari et al., 1997) và trong cộng đồng and in the Hidden Markov Models (HMM) (Bahlet al., 1986; Povey and Woodland, 2002). Trong bài bài viết khuyên rằng trong trường hợp không có sự lồi lõm của quá trình đào tạo một kiến trúc học sâu việc xác định một điểm khởi tạo cụ thể ngầm định trong các ràng buộc, cái mà đặc tả hàm chi phí cực tiểu (trong không gian rất lớn về số lượng hàm cực tiểu có thể). Theo cách nhìn này có thể nghĩ unsupervised pre-training liên quan đến hướng tiếp cận của Lasserre et al.
* Một đặc tính quan trọng và khác biệt nữa của chiến lược unsupervised pre-training là trong tính huống đào tạo tiêu chuẩn sử dụng stochastic gradient descent các hiệu ứng khái quát hoá đào tạo từ trước dường như không giảm đi số lượng các ví dụ được gán nhán tăng lên rất lớn. Bài báo lập luận rằng đâu là kết quả của việc kết hợp giữa tính không lồi(non-convexity) của hàm mục tiêu và sự phụ thuộc của của các phương pháp stochastic gradient descent vào thứ tự ví dụ. Bài báo lập luận rằng những thay đổi sớm trong các tham số có tác động lớn đến các khu vực cuối cùng. Cụ thể unsupervised pre-training giúp đặt các tham số trong một vùng mà từ đó các kết quả tốt hơn có thể đạt được. Do đó unsupervised pre-training hoạt động như một regularizer nó có thể có tác động tích cực đến mục tiêu đào tạo khi số lượng ví dụ đào tạo lớn.
* Như đã đề cập ở trên, bài viết này liên quan đến đánh giá thử nghiệm các giả thuyết khác nhau về vai trò của unsupervised pre-training trong sự thành công gần đây của phương pháp học sâu. Để kết thúc phần này nhóm nghiên cứu trình bày một loạt các thực nghiệm để giải thích các giả thuyết trong nỗ lực giải quyết một số bí ẩn xung quanh hiệu quả của của việc unsupervised pre-training.
* Trong thí nghiệm đầu tiên (Section 6) nhóm nghiên cứu xác định hiệu quả của unsupervised pre-training trong việc cải thiện lỗi khái quát hoá của các kiến trúc học sâu. Trong phần này, nhóm cũng khai thác các kỹ thuật giảm kích thước để minh họa cách unsupervised pre-training ảnh hưởng đến vị trí của cực tiểu trong không gian tham số.
* Trong tập thí nghiệm thứ hai (trong Phần 7), nhóm nghiên cứu so sánh trực tiếp hai giả thuyết thay thế (pre-training như một pre-conditioner và pre-training như một một kế hoạch tối ưu hóa) chống lại giả thuyết rằng unsupervised pre-training sát là một chiến lược chính quy (regularization strategy). Trong tập thí nghiệm cuối cùng, (trong Phần 8), nhóm nghiên cứu khám phá vai trò của unsupervised pre-training trong môi trường học tập trực tuyến(online trainning), nơi số lượng ví dụ đào tạo có sẵn tăng lên rất lớn. Trong các thử nghiệm này, nhóm nghiên cứu kiểm tra các khía cạnh chính của giả thuyết của nhóm nghiên cứu liên quan đến cấu trúc liên kết của hàm chi phí và vai trò của unsupervised pre-training trong việc thao túng vùng không gian tham số mà đào tạo được giám sát khởi xướng.
* Trước khi đi sâu vào các thí nghiệm, nhóm nghiên cứu bắt đầu với một cái nhìn sâu sắc hơn về những thách thức trong việc đào tạo kiến trúc sâu và cách nhóm nghiên cứu tin rằng các công việc unsupervised pre-training giúp để vượt qua những thách thức này.

# Những thách thức của Deep Learning

Trong phần này, nhóm nghiên cứu trình bày một viễn cảnh về lý do tại sao việc đào tạo tiêu chuẩn cho các mô hình sâu thông qua gradient backpropagation dường như rất khó khăn. Đầu tiên, điều quan trọng là phải thiết lập những gì chúng ta muốn nói đào tạo đó là khó khăn.

Nhóm nghiên cứu tin rằng thách thức trung tâm trong việc đào tạo kiến trúc sâu là đối phó với sự phụ thuộc mạnh mẽ tồn tại trong quá trình đào tạo giữa các tham số giữa các lớp. Một cách để nhận thức được khó khăn của vấn đề là chúng ta phải đồng thời:

* Điều chỉnh các lớp thấp hơn để cung cấp đầu vào đầy đủ cho cài đặt cuối cùng (kết thúc đào tạo) của các lớp trên
* Điều chỉnh các lớp trên để sử dụng tốt cài đặt cuối cùng (kết thúc đào tạo) của các lớp dưới.

Vấn đề thứ hai là dễ dàng hiện tại (ví dụ, khi cài đặt cuối cùng của các lớp khác được biết). Không rõ mức độ khó đầu tiên như thế nào và nhóm nghiên cứu phỏng đoán rằng một khó khăn đặc biệt phát sinh khi cả hai tập hợp các lớp phải được học cùng nhau, vì gradient của hàm mục tiêu được giới hạn trong một thước đo cục bộ với cài đặt hiện tại của các tham số khác. Hơn nữa, vì với đủ năng lực, hai lớp trên cùng có thể dễ dàng vượt qua tập huấn luyện, lỗi đào tạo không nhất thiết phải bộc lộ khó khăn trong việc tối ưu hóa các lớp thấp hơn. Như thể hiện trong các thí nghiệm của nhóm nghiên cứu ở đây, các sơ đồ đào tạo tiêu chuẩn có xu hướng đặt các tham số trong các vùng của không gian tham số có khả năng khái quát kém.

Một vấn đề riêng biệt nhưng có liên quan xuất hiện nếu chúng ta tập trung xem xét các phương pháp đào tạo truyền thống cho các kiến trúc sâu stochastic gradient descent. Một chuỗi các ví dụ cùng với thủ tục gradient descent tuyến (online) xác định quỹ đạo trong không gian tham số, nó hội tụ theo một nghĩa nào đó (lỗi không cải thiện nữa, có thể do chúng ta ở gần mức tối thiểu cục bộ). Giả thuyết là các nhiễu loạn nhỏ của quỹ đạo đó (bằng cách khởi tạo hoặc bằng các thay đổi trong đó các ví dụ được nhìn thấy khi nào) có tác dụng sớm hơn. Trong quá trình tuân theo gradient descent, các thay đổi về trọng số có xu hướng tăng cường độ của chúng và do đó, lượng phi tuyến của mạng tăng lên. Khi điều này xảy ra, tập hợp các vùng có thể truy cập bằng cách giảm gradient descent trên các mẫu của phân phối đào tạo trở nên nhỏ hơn. Đào tạo sớm trên tập nhỏ và nhiễu loạn cho phép các tham số mô hình chuyển từ khu vực này sang khu vực gần đó, trong khi sau đó (thường là với các giá trị tham số lớn hơn), không có khả năng thoát ra khỏi khu vực thu hút như vậy. Do đó, các ví dụ ban đầu có thể có ảnh hưởng lớn hơn và trong thực tế, bẫy các tham số mô hình trong các vùng cụ thể của không gian tham số tương ứng với thứ tự cụ thể và tùy ý của ví dụ đào tạo. Một hậu quả quan trọng của hiện tượng này là ngay cả khi có một lượng dữ liệu được giám sát rất lớn (vô cùng hiệu quả),stochastic gradient descent gặp phải overfitting với dữ liệu đào tạo. Theo nghĩa đó, đ unsupervised pre-training tương tác mật thiết với quá trình tối ưu hóa và khi số lượng ví dụ đào tạo trở nên lớn, tác động tích cực của nó không chỉ được nhìn thấy đối với lỗi tổng quát mà còn về lỗi trên tập huấn luyện

# Xem xét mô hình đã huấn luyện không giám sát như regularization

Greedy layer-wise unsupervised pre-training vượt qua thách thức của deep learning bằng việc giới thiệu 1 quá trình hữu ích trước quá trình huấn luyện có giám sát. Regularization là kết quả của quy trình trước khi đào tạo thiết lập điểm khởi tạo của quy trình tinh chỉnh bên trong một vùng không gian tham số trong đó các tham số bị hạn chế. Các tham số được giới hạn trong một thể tích không gian tham số tương đối nhỏ, được phân định bằng ranh giới của lưu vực thu hút cục bộ của hàm chi phí tinh chỉnh được giám sát.

Quy trình pre-training làm tăng cường độ của trọng số và trong các mô hình sâu tiêu chuẩn, với tính phi tuyến sigmoidal, điều này có tác dụng làm cho cả hàm phi tuyến và hàm chi phí phức tạp hơn với các đặc điểm tôpô hơn như đỉnh, đáy và cao nguyên. Sự tồn tại của các tính năng tô pô này làm cho không gian tham số cục bộ trở nên khó khăn hơn khi di chuyển khoảng cách đáng kể thông qua thủ tục giảm độ dốc. Đây là cốt lõi của tài sản hạn chế được áp đặt bởi quy trình pre-training và do đó là cơ sở của các thuộc tính chính quy của nó.

UPT hạn chế các tham số vào các khu vực cụ thể: Những cấu trúc tương ứng với phân phối đầu vào P(X). Nói UPT là một chiến lược regularization phần nào làm giảm tính hiệu quả của nó. Không phải tất cả các regularizers được tạo bằng nhau và trong so sánh với các chiến lược regularization tiêu chuẩn như L1, L2, UPT có hiệu quả đáng kể. Thành công này có thể được đóng góp bởi quá trình huấn luyện không giám sát.

Trong mỗi quá trình của chiến lược huấn luyện U, các tầng được huấn luyện để biểu diễn các tham số có ưu thế của sự thay đổi trong dữ liệu. Điều này có tác dụng tận dung tri thức của X để biểu diễn, tại mỗi tầng, một đại diện của X bao gồm các thành phần đáng tin cậy về mặt thống kê của X có thể được sử dụng để dự đoán output Y. Phối cảnh này đặt trước UPT trong nhóm các chiến lược học được gọi là phương pháp bán giám sát. Như các công việc khác hiện nay tập trung tính hiệu quả của các phương pháp bán giám sát trong regularizing các tham số mô hình, hiệu quả của chiến lượng UPT bị giới hạn bởi mức độ học P(x) hữu ích trong quá trình học P(Y|X). Ở đây, nhóm nghiên cứu tìm các biến đổi của X(các feature học) - là có thể dữ đoạn được của các tham số chính của biến thể trong P(X). và khi chiến lược pre-training là hiệu quả, một vài features học được này của X cũng có thể là dự đoán của Y. Trong context của deep learning, chiến lược greedy unsupervised có thể cũng có một hàm đặc biệt. Ở một mức độ nào đó, nó giải quyết được vấn đề học tập đồng thời các tham số của tất cả các tầng bằng việc giới thiệu một tiêu chí proxy. Tiêu chí proxy này khuyến khích các yêu tố quan trọng của biến thể, được thể hiện trong dữ liệu đầu vào và đực biểu diễn trong các tầng trung gian.

Nó chỉ đơn giản là chỉ ra cách người ta có thể nghĩ về khái niệm điểm khởi tạo như là một trình chuẩn hóa và không nên được coi là một cách giải thích theo nghĩa đen về cách đạt được sự chính quy hóa một cách rõ ràng vì chúng ta không có một công thức phân tích để tính toán πk và vk. Thay vào đó, chúng được định nghĩa ngầm bởi toàn bộ quy trình đào tạo trước không được giám sát.

Chúng ra sẽ làm rõ các kỹ thuật regularization trong bài toán học máy.

Một nhược điểm lớn của cross-validation là số lượng training runs tỉ lệ thuận với k. Điều đáng nói là mô hình polynomial như trên chỉ có một tham số cần xác định là bậc của đa thức. Trong các bài toán Machine Learning, lượng tham số cần xác định thường lớn hơn nhiều, và khoảng giá trị của mỗi tham số cũng rộng hơn nhiều, chưa kể đến việc có những tham số có thể là số thực. Như vậy, việc chỉ xây dựng một mô hình thôi cũng là đã rất phức tạp rồi. Có một cách giúp số mô hình cần huấn luyện giảm đi nhiều, thậm chí chỉ một mô hình. Cách này có tên gọi chung là regularization.

Regularization, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách di chuyển nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình ít phức tạp hơn mặc dù giá trị của hàm mất mát có tăng lên một chút.

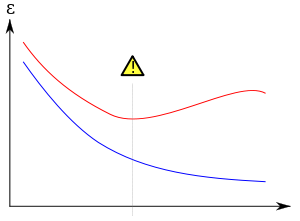
Một kỹ thuật rất đơn giản là early stopping.

* Early stopping

Trong nhiều bài toán Machine Learning, chúng ta cần sử dụng các thuật toán lặp để tìm ra nghiệm, ví dụ như Gradient Descent. Nhìn chung, hàm mất mát giảm dần khi số vòng lặp tăng lên. Early stopping tức dừng thuật toán trước khi hàm mất mát đạt giá trị quá nhỏ, giúp tránh overfitting.

Vậy dừng khi nào là phù hợp?

Một kỹ thuật thường được sử dụng là tách từ training set ra một tập validation set như trên. Sau một (hoặc một số, ví dụ 50) vòng lặp, ta tính cả train error và validation error, đến khi validation error có chiều hướng tăng lên thì dừng lại, và quay lại sử dụng mô hình tương ứng với điểm và validation error đạt giá trị nhỏ.



Hình : Early stopping. Đường màu xanh là train error, đường màu đỏ là validation error

* Thêm số hạng vào hàm mất mát

Kỹ thuật regularization phổ biến nhất là thêm vào hàm mất mát một số hạng nữa. Số hạng này thường dùng để đánh giá độ phức tạp của mô hình. Số hạng này càng lớn, thì mô hình càng phức tạp. Hàm mất mát mới này thường được gọi là regularized loss function, thường được định nghĩa như sau:

Trong đó:

: Tham số của mô hình

: Hàm mất mát của mô hình

: Số hạng *regularization*

: Hệ số regularization

Việc tối thiểu regularized loss function, nói một cách tương đối, đồng nghĩa với việc tối thiểu cả loss function và số hạng regularization.

Chúng ta vẫn mong muốn rằng sự khác nhau này là nhỏ, vì vậy tham số regularization (regularizaton parameter) thường được chọn là một số nhỏ để biểu thức regularization không làm giảm quá nhiều chất lượng của nghiệm.

* L2 regularization

Trong kỹ thuật này:

Trong đó:

: Là norm 2 của hệ số

Norm 2 có 1 số đặc điểm đó là:

* Norm 2 là một hàm số rất mượt, tức có đạo hàm tại mọi , đạo hàm của nó đơn giản là w, vì vậy đạo hàm của regularized loss function cũng rất dễ tính, chúng ta có thể hoàn toàn dùng các phương pháp dựa trên gradient để cập nhật nghiệm.
* Việc tối thiểu norm 2 đồng nghĩa với việc khiến cho các giá trị của hệ số w trở nên nhỏ gần với 0. Với Polynomial Regression, việc các hệ số này nhỏ có thể giúp các hệ số ứng với các số hạng bậc cao là nhỏ, giúp tránh overfitting. Với Multi-layer Pereceptron, việc các hệ số này nhỏ giúp cho nhiều hệ số trong các ma trận trọng số là nhỏ. Điều này tương ứng với việc số lượng các hidden units hoạt động (khác không) là nhỏ, cũng giúp cho MLP tránh được hiện tượng overfitting.

L2 regularization là kỹ thuật được sử dụng nhiều nhất để giúp Neural Networks tránh được overfitting. Nó còn có tên gọi khác là weight decay.

# Các nghiên cứu liên quan

## Học bán giám sát

Theo (Ng and Jordan, 2002) chỉ ra rằng generative model giảm overfitting hơn discriminant model.

Trong bài toán học máy nói chung, đầu vào là biến X và dự đoán đầu ra Y. Mục tiêu của bài toán là tính P(Y/X).

* Generative model

Là một mô hình thống kê dựa trên phân bố xác suất chung của X và Y: P(X,Y). Sau đó, dựa vào định lý Bayes để tính xác suất P(Y/X).

* Discriminant model

Là mô hình thống kê tính trực tiếp xác suất P(Y/X) dựa trên dữ liệu đầu vào.

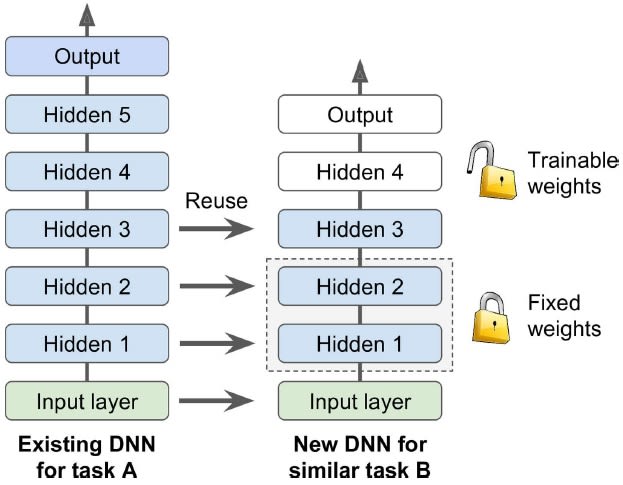
Việc tìm ra phân phối xác suất P(X) cải thiện tính tổng quát hóa cho bài toán phân loại (đây là mấu chốt cơ bản của việc học bán giám sát). Ví dụ, ta có thể xây dựng một mô hình học không giám sát để đại diện X bởi x1 và x2 (quá trình này còn được gọi là embedding) thuộc về cùng một cụm. Một mô hình học có giám sát sử dụng tính khái quát hóa của mô hình không giám sát trên để phân lớp. Cụ thể (Belkin and Niyogi, 2002; Chapelle et al., 2003) đã thực nghiệm sử dụng Principal Components Analysis là quá trình tiền xử lý trước khi áp dụng mô hình phân lớp. Trong các mô hình này, dữ liệu được chuyển đổi đầu tiên trong một biểu diễn mới bằng cách học không có giám sát và một bộ phân loại được giám sát được xếp chồng lên nhau, học cách ánh xạ dữ liệu trong biểu diễn mới này vào các dự đoán của các lớp.

Thay vì tách biệt giữa quá trình học không giám sát và học có giám sát trong việc xây dựng mô hình. Chúng ta có thể quan tâm tới việc sử dụng chúng trong cùng một mô hình với việc chia sẻ bộ tham số. Cụ thể P(X) hay P(X, Y) và P(Y/X) sẽ chia sẻ bộ tham số và việc tối ưu mô hình chính là việc đánh đổi giữa tối ưu hàm mất mát của học có giám sát -log(P/X) và học không có giám sát -log(P(X)) hoặc -log(P(X, Y)) (Lasserre et al., 2006). Mô hình sinh tỏ ra rất hiệu quả trong việc khái quát hóa dữ liệu (Lasserre et al., 2006; Larochelle and Bengio, 2008).

## Early stopping là tiêu chuẩn của Regularization

Sử dụng pre-traning chính là việc hạn chế không gian khởi tạo của tham số tối ưu cho bài toán học có giám sát. Giúp việc khởi tạo tham số gần với nghiệm tối ưu cục bộ của hàm mất mát. Early stopping cũng có hiệu quả tương tự, bằng cách ràng buộc việc tối ưu hóa của một vùng không gian tham số gần với cấu hình của tham số khởi tạo. Việc hạn chế số lượng hạn chế diện tích không gian tham số có thể tiếp cận từ điểm bắt đầu. Trong trường hợp tối ưu hóa mô hình tuyến tính đơn giản (khởi tạo ở gốc) sử dụng hàm mất mát bậc hai và giảm độ dốc đơn giản, early stopping sẽ có tác động tương tự như chính quy truyền thống.

Do đó, cả pre-traning và early stopping, các tham số của hàm mất mát đều bị ràng buộc bởi tham số đầu vào. Hay nói cách khác, tham số khởi tạo của quá trình học có giám sát là kết quả của quá trình pre-trainning.



Hình : Sử dụng pre-trainning trong Deep Learning task

# Cài đặt và thử nghiệm

## Mô hình

Có 2 models chính được sử dụng trong bài báo: Deed Belief Network (DBN) và Stacked Denoising Autoencoder (SDAE). Quá trình training của các mạng này sử dụng hướng tiếp cận unsupervised pre-training được chia làm hai giai đoạn chính:

Giai đoạn 1: Unsupervised pre-training:

Ở giai đoạn này, từng tầng (hidden layer) sẽ được training lần lượt. Bắt đầu với tầng ẩn đầu tiên, dữ liệu input đầu vào là label của chính nó. Việc huấn luyện sẽ nhằm mục đích tìm ra bộ trọng số (weights) giữa tầng input và tâng hidden thứ nhất sao cho giá trị hàm lỗi là nhỏ nhất. Sau khi training xong tầng thứ nhất, tương tự như tầng 1, tầng thứ 2 sẽ được training với input đầu vào là output của tầng 1. Lần lượt training từng tầng với input là output của tầng trước đó và label của input là chính input.

Giai đoạn 2: Fine-tunning:

Sau giai đoạn 1 ta sẽ được một mạng ANN với bộ trọng số đã được pre-trained.

Từ đây, chúng ta sẽ training mạng nhưng một mạng ANN bình thường.

DBV và SDAE sẽ được giới thiệu qua ở phần dưới đây.

### Deep Belief Networks

Mô hình đầu tiên là Deep Belief Net (DBN) của Hinton et al. (2006), thu được bằng cách đào tạo và sắp xếp một số lớp của Boltzmann Machines (RBM) bị hạn chế một cách tham lam. Khi ngăn xếp RBM này được huấn luyện, nó có thể được sử dụng để khởi tạo một mạng lưới thần kinh nhiều lớp để phân loại.

Một RBM với n lớp ẩn là một Markov Random Field (MRF) của một phân phối kết hợp giữa hai biến ngẫu nhiên ẩn là h và x.

RBM có thể được khái quát thành các phân phối có điều kiện khác bên cạnh nhị thức, bao gồm các biến liên tục. Welling et al. (2005) mô tả tổng quát hóa các mô hình RBM cho các phân phối có điều kiện từ gia đình hàm mũ. Các mô hình RBM có thể được đào tạo bằng cách giảm độ dốc ngẫu nhiên gần đúng. Mặc dù P(X) không dễ điều khiển trong RBM. Contrastive Divergence estimator (Hinton, 2002) là xấp xỉ tốt.

DBN là mô hình thế hệ nhiều lớp với các biến lớp h0 (lớp đầu vào hoặc lớp nhìn thấy), h1, h2, v.v… Hai lớp trên cùng có phân phối chung là RBM và P (hk | hk + 1) tham số theo cách tương tự như đối với RBM. Do đó, DBN 2 lớp là một RBM và một nhóm RBM chia sẻ tham số với một DBN tương ứng.

Hướng cập nhật tương phản có thể được sử dụng để khởi tạo mỗi lớp của DBN dưới dạng RBM, như sau. Hãy xem xét lớp đầu tiên của DBN được đào tạo như một RBM P1 với lớp ẩn h1 và lớp có thể nhìn thấy v1. Chúng ta có thể huấn luyện RBM P2 thứ hai mô hình (trong lớp hiển thị của nó) các mẫu h1 từ P1 (h1 | v1) khi v1 được lấy mẫu từ tập dữ liệu huấn luyện. Có thể thấy rằng điều này tối đa hóa giới hạn thấp hơn về khả năng đăng nhập của DBN. Số lượng các lớp có thể được tăng lên một cách tham lam, với lớp trên cùng mới được đào tạo như một RBM để mô hình hóa các mẫu được tạo ra bằng cách xâu chuỗi các lớp sau P (hk | hk1) của các lớp thấp hơn (bắt đầu từ h0 từ tập dữ liệu huấn luyện)

### Stacked Denoising Auto-encoders

Mô hình thứ hai, đề xuất bởi Vincent et al. (2008), được gọi là Stacked Denoising Auto-Encoder (SDAE). Nó mượn nguyên tắc tham lam từ các DBN, nhưng sử dụng bộ mã hóa tự động khử nhiễu làm khối xây dựng cho mô hình không giám sát. Bộ mã hóa tự động học một bộ mã hóa h (·) và bộ giải mã g (·) có thành phần tiếp cận danh tính cho các ví dụ trong tập huấn luyện, nghĩa là g (h (x)) x cho x trong tập huấn luyện.

Giả sử rằng một số ràng buộc ngăn g (h (·)) trở thành nhận dạng cho các đối số tùy ý, bộ mã hóa tự động phải nắm bắt cấu trúc thống kê trong tập huấn luyện để giảm thiểu lỗi tái tạo. Tuy nhiên, với mã dung lượng cao (h (x) có quá nhiều chiều), một bộ mã hóa tự động thông thường có khả năng học được một mã hóa tầm thường. Lưu ý rằng có một mối liên hệ mật thiết giữa việc giảm thiểu lỗi tái cấu trúc cho bộ mã hóa tự động và đào tạo phân kỳ tương phản cho RBM, vì cả hai đều có thể được hiển thị để xấp xỉ độ dốc khả năng log-likelihood (Bengio và Delalleau, 2009).

Bộ mã hóa tự động khử nhiễu (Vincent et al., 2008; Seung, 1998; LeCun, 1987; Gallinari et al., 1987) là một biến thể ngẫu nhiên của bộ mã hóa tự động thông thường với đặc tính đặc biệt ngay cả với một mô khả năng cao, nó không thể học kết nối nhận dạng. Bộ mã hóa tự động khử nhiễu được đào tạo rõ ràng để khử nhiễu phiên bản đầu vào. Tiêu chí đào tạo của nó cũng có thể được xem như là một biến thể thấp hơn ràng buộc về khả năng của một mô hình thế hệ cụ thể. Nó đã được hiển thị trên một loạt các tập dữ liệu để thực hiện tốt hơn đáng kể so với các bộ mã hóa tự động thông thường và tương tự hoặc tốt hơn RBM khi được xếp chồng lên một kiến trúc được giám sát sâu (Vincent et al., 2008). Một cách khác để ngăn chặn các bộ mã hóa tự động thông thường có nhiều đơn vị mã hơn đầu vào để tìm hiểu danh tính là hạn chế khả năng đại diện bằng cách áp dụng độ thưa thớt trên mã (Ranzato et al., 2007, 2008).

## Mô tả dữ liệu

Bài báo thử nghiệm trên 3 bộ dataset là:

MNIST: Bộ dữ liệu ảnh chữ số viết tay. Bao gồm 60000 dữ liệu huấn luyện và 10000 dữ liệu kiểm thử, mỗi ảnh có kích thước là 28x28.

InfiniteMNIST: một dữ liệu được cung cấp bởi Loosli et al. (2007), là phần mở rộng của MNIST mà từ đó người ta có thể thu được số lượng ví dụ gần như vô hạn. Các mẫu thu được bằng cách thực hiện các biến thể ngẫu nhiên của các chữ số MNIST ban đầu. Trong tập dữ liệu này, chỉ có một tập hợp các ví dụ và các mô hình sẽ được so sánh bởi hiệu suất (trực tuyến) của chúng trên đó.

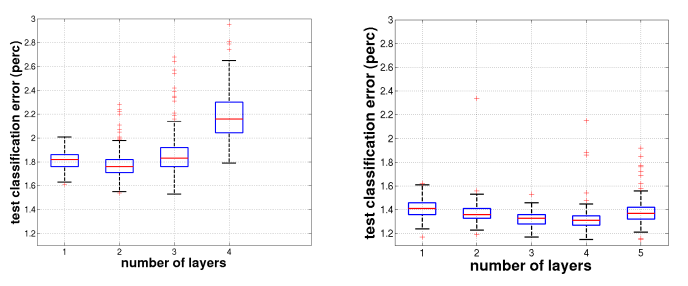
Shapeset: là một tập dữ liệu tổng hợp với một phạm vi kiểm soát các bất biến hình học. Kích thước mỗi ảnh là 10 × 10 của hình tam giác và hình vuông. Các ví dụ cho thấy hình ảnh của các hình dạng với nhiều biến thể, chẳng hạn như kích thước, định hướng và mức độ xám. Tập dữ liệu bao gồm 50000 dữ liệu huấn luyện, 10000 dữ liệu xác nhận và 10000 hình ảnh kiểm thử.

# Ảnh hưởng của unsupervised pre-training

Mục đích của phần này là chứng minh sự ảnh hưởng tích cực của unsupervised pre-training trong vấn đề tối ưu hóa generalization error của các mô hình deep learning. PhẦn này cũng trực quan hóa sự ảnh hưởng của unsupervised pre-training đối với vị trí của minima trong không gian tham số bằng các kĩ thuật giảm chiều dữ liệu.

## Tổng quát hóa tốt hơn

Các nghiên cứu đã có thường sử dụng một hoặc một số bố trọng số ngẫu nhiên tốt để train. Trong nghiên cứu này, tác giả dùng 40-500 bộ trọng số ngẫu nhiên khác nhau để khám giá sự ảnh hưởng của các bộ trọng số ngẫu nhiên lên hiệu năng của các model bình thường và model được unsupervised pre-training.

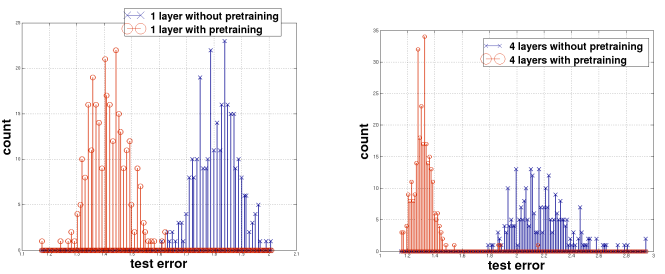


Hình : Phân bố của test classification error của model được unsupervised pre-training (trái) và model được unsupervised pre-training (phải) với 400 bộ tham số trọng số ngẫu nhiên

Hình 3.1 mô tả phân bố test classification error của model được unsupervised pre-training và model bình thường khi số layers tăng lên. Có thể dễ thấy, model được unsupervised pre-training cho kết quả test error giảm xuống dần khi số layers tăng lên. Trong khi đó, với model không được unsupervised pre-training, test error chỉ giảm khi số layers tăng từ 1 lên 2 và sau đó test error tăng rất nhanh khi số layer tăng dần. Một chú ý khác là với model không được unsupervised, chúng ta không thể train hiệu quả khi số layers bằng 5 (test error quá lớn).

Hình 3.2 cho ta thấy số lượng outliers của model không được unsupervised tăng lên đáng kể khi số layers tăng lên. Với một layers kết quả test errors của cả hai model có phần bố khá ổn định. Tuy nhiên khi số lượng layers tăng lên 4, số lượng outliers của model không được pre-training tăng lên đột biến trong khi model với unsupervised pre-training tăng lên không đáng kể. Điều này minh chứng rằng, khi tăng số lượng layers, xác suất tìm được local minima sẽ giảm đi.

Từ hình 3.1 và 3.2, có thể suy luận rằng unsupervised pre-training mà một kĩ thuật mạnh khi bộ trọng số được khởi tạo ngẫu nhiên.



Hình : Histogram của test errors trên tập data MNIST sử dụng model unsupervised pre-training và model bình thường với số lượng hidden layers là 1 (hình trái) và 4 (hình phải)

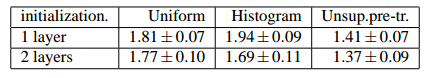
# Vai trò của unsupervised pre-training

Mục đích của phần này là so sánh các giả thiết lý giải tại sao unsupervised pre-training lại supervised learning tổng quát hóa dữ liệu tốt hơn. Ở đây, tác giả so sánh hai giả thiết đã có là: pre-training as a pre-conditioner và pre-training as an optimization scheme với giả thiết tác giả đưa ra là unsupervised pre-training is a regularization strategy. Việc so sánh các giải thiết này được thông qua các thí nghiệm của tác giả.

## Thí nghiệm 1: Pre-training có giúp cho Supervised Learning có một quá trình điều kiện (condition process) tốt hơn không ?

Mục đích của thí nghiệm này là để kiểm chứng sự đúng đắn của giả thiết đầu tiên: pre-training as a pre-conditioner. Giả thiết này cho rằng unsupervised pre-training đơn giản chỉ là việc làm các trọng số khởi tạo ở giá trị ban đầu lớn hơn so với việc khởi tạo ngẫu nhiên với các giá trị gần bằng 0. Từ đó, điều kiện ban đầu (khoảng giá trị) khi khởi tạo tham số sẽ tốt hơn giúp cho quá trình tối ưu tốt hơn. Một cách khác nếu giải thiết đúng, chúng ta có thể đạt hiệu năng tương đương như unsupervised pre-training nếu ta khởi tạo các bộ trọng số từ 1 khoảng thích hợp hơn thay vì khởi tạo theo phân phối đều từ khoảng [-1/k, 1/k ]. Để kiểm chứng điều này, nhóm tác giả đã tính toán phân bố của trọng số và bias của từng layer của model với unsupervised pre-training. Sau đó train mô hình không sử dụng unsupervised pre-training với các bộ trọng số khởi tạo ngẫu nhiên trong phân bố trên (phân bố của unsupervised pre-training).

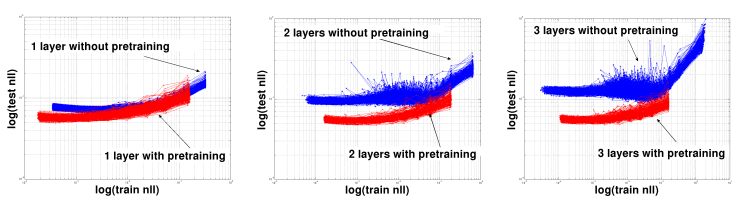
Bảng : Ảnh hưởng của từng chiến lược khởi tạo tham số. Giá trị từng ô trong bảng là mean và std của test errors



Kết quả của thí nghiệm này được thể hiện ở bảng trên. Với các mô hình có 1 layer, kết quả của chiến lược khởi tạo các bộ trọng số từ phân bố giống với unsupervised learning thấp hơn cả chiến lược khởi tạo ngẫu nhiên truyền thống và kém hơn rất nhiều so với chiến lược unsupervised learning. Với mô hình có 2 layers, kết quả của chiến lược của giả thiết 1 chỉ tốt hơn rất ít so với việc khởi tạo trọng số truyền thống. Từ các kết quả này, chúng ta có thể kết luận giả thiết pre-training as a pre-conditioner không chính xác.

## Thí nghiệm 2: Ảnh hưởng của pre-training lên training error

Mục đích của phần này là so sánh sự ảnh hưởng của pre-training lên training error. Thí nghiệm được tiến hành với 400 bộ trọng số ngẫu nhiên với hai loại model: model sử dụng unsupervised pre-training và không sử dụng unsupervised pre-training. Mỗi loại model sẽ được tăng số layers từ 1 đến 3. Kết quả thí nghiệm được hiển thị ở hình 8.



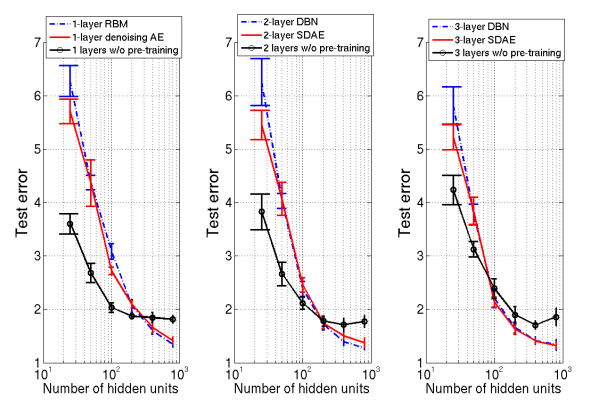
Hình : Kết quả của 2 hai loại model với/với không unsupervised pre-training sử dụng 400 bộ trọng số khởi tạo ngẫu nhiên

Từ hình 5 có thể thấy, với các model có 1 layer, sử dụng unsupervised pre-training giúp giảm training error so với model không sử dụng. Một khía cạnh khác có thể thấy là, ở cùng 1 mức độ training error, pre-trained model cho kết quả test tốt hơn các model bình thường. Ở đây ta có thể thấy, unsupervised pre-training giúp tổng quát hóa tốt hơn thay vì là giúp tối ưu hóa hàm lỗi tốt hơn.

## Thí nghiệm 3: Ảnh hưởng của số lượng hidden layers

Mục đích của thí nghiệm này là thể hiện sự ảnh hưởng của unsupervised pre-training khi số lượng layers đổi. Một đặc tính của regularization là sự ảnh hưởng của regularization tăng lên khi độ phức tạp của mô hình tăng. Thông qua thí nghiệm này tác giả muốn củng cố giả thiết của mình : unsupervised pre-training is a regularization strategy và kỳ vọng rằng khi số lượng hidden unit tăng thì sự ảnh hưởng của unsupervised pre-training sẽ tăng theo, điều này sẽ củng cố giả thiết unsupervised pre-training là một regularizer.

Trong thí nghiệm này, tác giả train model trên bộ dữ liệu MNIST sử dụng/không sử dụng chiến lược unsupervised pre-training. Mỗi thí nghiệm ứng với một layer size khác nhau. Số lượng layer tác giả tăng từ 25, 50, 100, 400, 800. Kết quả của thí nghiệm được hiển thị ở hình 6.



Hình : Sử ảnh hưởng của unsupervised pre-training khi số lượng hidden layers tăng

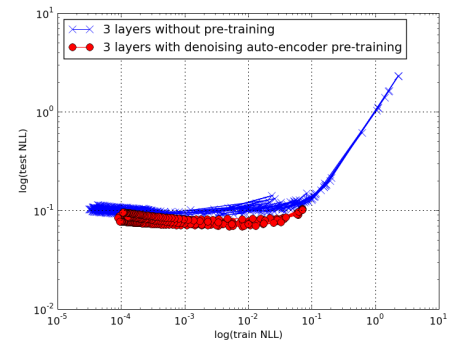
Ở hình trên, ta có thể thấy unsupervised pre-training giúp cho các model lớn (số lượng hidden unit lớn) có test error ổn định nhưng lại cho hiệu suất kém hơn so các model không sử dụng pre-training khi số lượng hidden unit nhỏ.

## Thí nghiệm 4: Kiểm chứng giả thiết pre-training as an optimization scheme

Mục đích phần này là so sánh hai giả thiết pre-training as an optimization scheme và giả thiết tác giả đưa ra: unsupervised pre-training is a regularization strategy. Giả thiết đầu tiên cho rằng : unsupervised pre-training giúp giảm training error. Trong khi đó, giả thiết của tác giả cho rằng: unsupervised pre-training không giúp giảm test error vì nó là một regularizer.

Các công trình trước đó mà ủng hộ giả thiết pre-training as an optimization scheme đã cho ra kết quả là pre-training giúp cho training error và test error đều thấp hơn việc không sử dụng. Tuy nhiên, trong các thí nghiệm này, tác giả đã sử dụng early stopping như 1 regularizer.

Điều này cần được kiểm chứng lại. Trong nghiên cứu này, tác giả đã thực hiện lại thí nghiệm mà không sử dụng early stopping. Kết quả được thể hiện qua hình 7.



Hình : Kết quả của khi thí nghiệm khi không sử dụng early stopping

Từ hình 7, ta có thể thấy khi không sử dụng early stopping, training error của các mô hình pre-trained cao hơn các mô hình không sử dụng pre-training mặc dù test error thấp hơn. Kết quả này đã chỉ ra rằng giả thiết pre-training as an optimization scheme không chính xác.

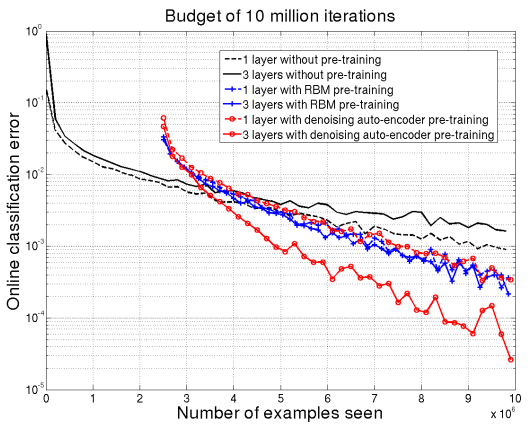
# Online Learning

Mục đích của phần này là khám phá vai trò của unsupervised pre-training trong online learning (lúc số lượng dữ liệu training tăng rất nhanh).

## Ảnh hưởng của pre-training đối với tập dữ liệu rất lớn

Thí nghiệm ở phần này được thực hiện với 6 mô hình trên bộ dữ liệu InfiniteMNIST bao gồm :

DBNs với 1 và 3 layers, SDAE với 1 và 3 layers, ANN với 1 và 3 layers. Kết quả được thể hiện ở hình 8.



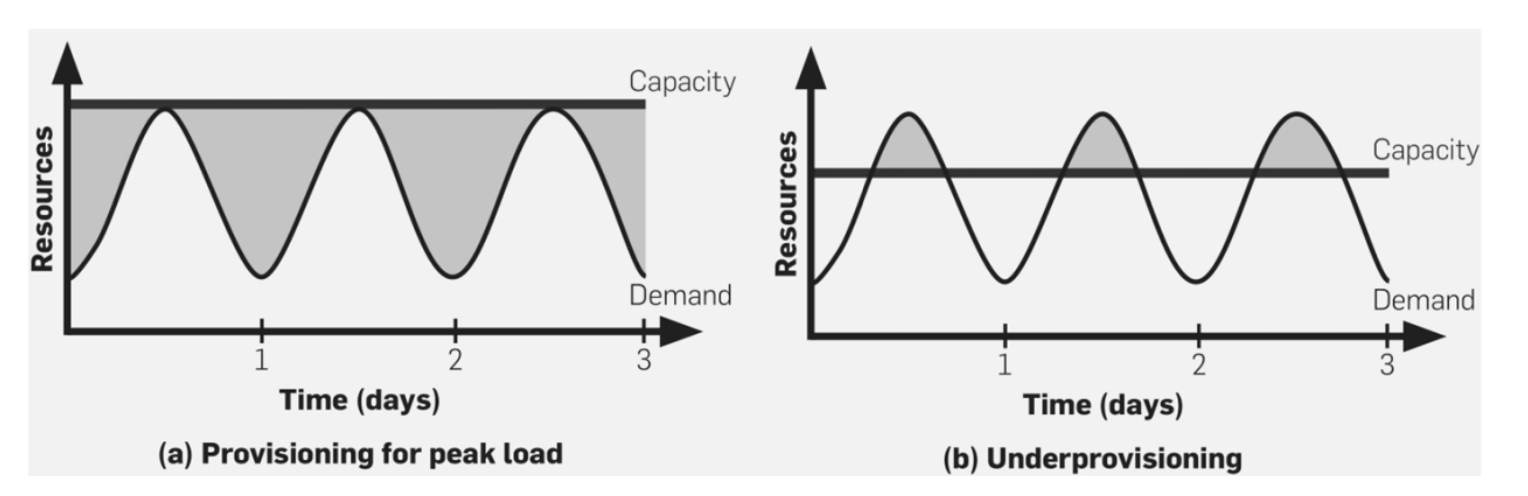
Hình : So sánh các mô hình có 1 và 3 layers trên bộ dữ liệu InfiniteMNIST

Từ hình 8 ta có thể thấy, mô hình 3 layer không sử dụng pre-training cho kết quả test error kém nhất trong các mô hình kể cả với mô hình 1 layer không pre-training. Điều này chứng minh giả thiết rằng, kể cả với online learning, tối ưu hóa các deep model khó hơn rất nhiều với các shallow model. Một nhận xét khác đó là, SDAE 3 layers cho kết quả tốt hơn DBN 3 layers. Cuối cùng nhận xét quan trọng nhất, lợi ích của pre-training không biến mất khi số lượng training data tăng. Từ đây có thể kết luận là pre-training vẫn rất hữu chính cho online learning.

# Thực nghiệm đánh giá

## Giới thiệu bài toán

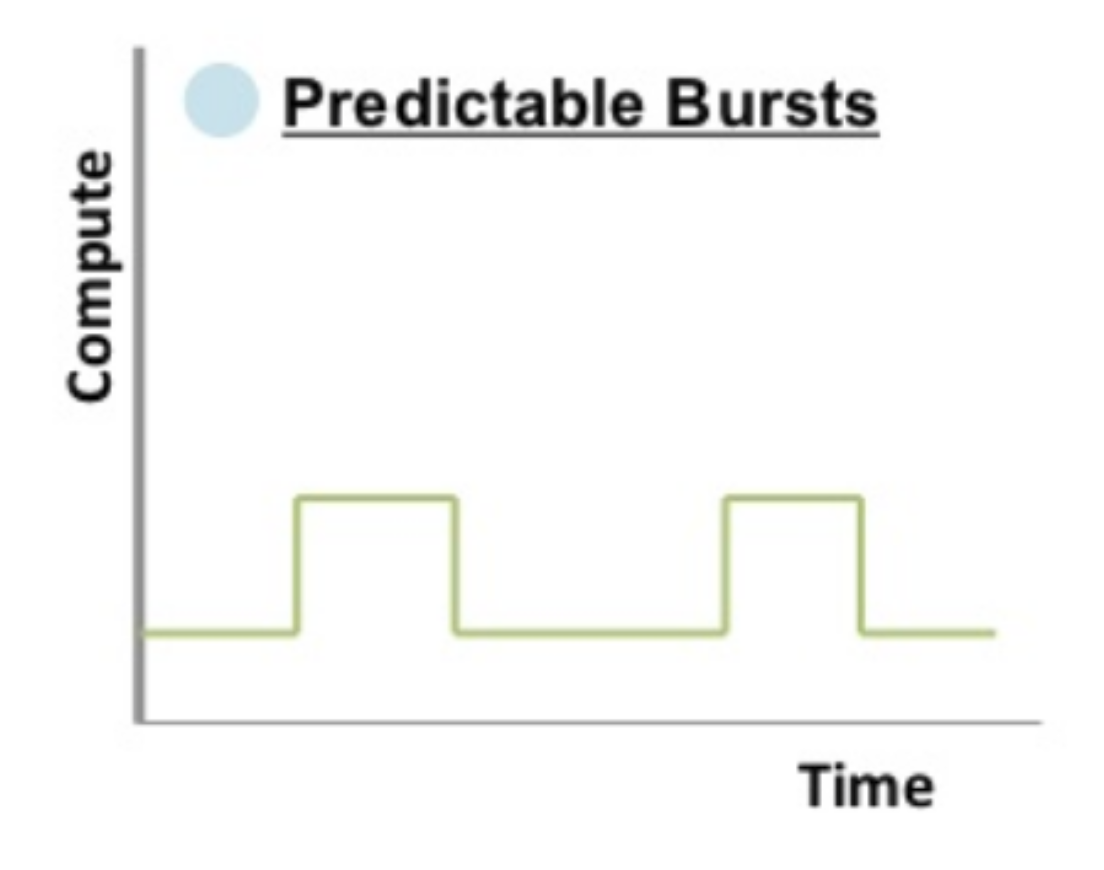
Sự phát triển của các mô hình điện toán đám mây trong những năm qua được cho là nhờ lợi ích mà các mô hình trả phí theo nhu cầu sử dụng (pay-as-you-go model) mang lại. Mô hình pay-as-you-go là mô hình mà những người dùng điện toán đám mây trả phí cho các nhà cung cấp dịch vụ dựa trên lượng tài nguyên tính toán mà họ đã sử dụng. Lợi ích chính của mô hình điện toán đám mây là tránh lãng phí tài nguyên. Người dùng sẽ chỉ cần trả phí cho những tài nguyên mà họ sử dụng thay vì cung cấp một lượng tài nguyên cố định mà có thể không dùng đến. Với mô hình pay-as-you-go, các nhà cung cấp dịch vụ đám mây cần bảo đảm chất lượng cung cấp dịch vụ đám mây của mình với người dùng dựa trên các độ đo chất lượng dịch vụ khác nhau (Quality of Service). Nhờ đó, người dùng dịch vụ điện toán đám mây có thể tập trung vào việc cung cấp các dịch vụ của mình tới khách hàng mà không cần phải có kiến thức, kinh nghiệm hay các cơ sở hạ tầng phục vụ các hệ thống đám mây. Các mô hình điện toán đám mây khắc phục được 2 yếu điểm quan trọng của mô hình truyền thống về “khả năng mở rộng (scalability)” và “độ linh hoạt (flexibility)”. Theo đó, các hệ thống điện toán có khả năng co giãn (mở rộng hoặc thu hồi) tài nguyên tính toán một cách tự động dựa vào nhu cầu sử dụng của người dùng mà không cần họ phải có các kiến thức cho quá trình này. Với khả năng mở rộng và tính linh hoạt, các hệ thống điện toán đám mây có thể ngăn ngừa tình trạng vượt quá khả năng cung cấp vào giờ cao điểm hoặc lãng phí tài nguyên vào giờ thấp điểm.



Hình : Minh hoạ bài toán co giãn tài nguyên

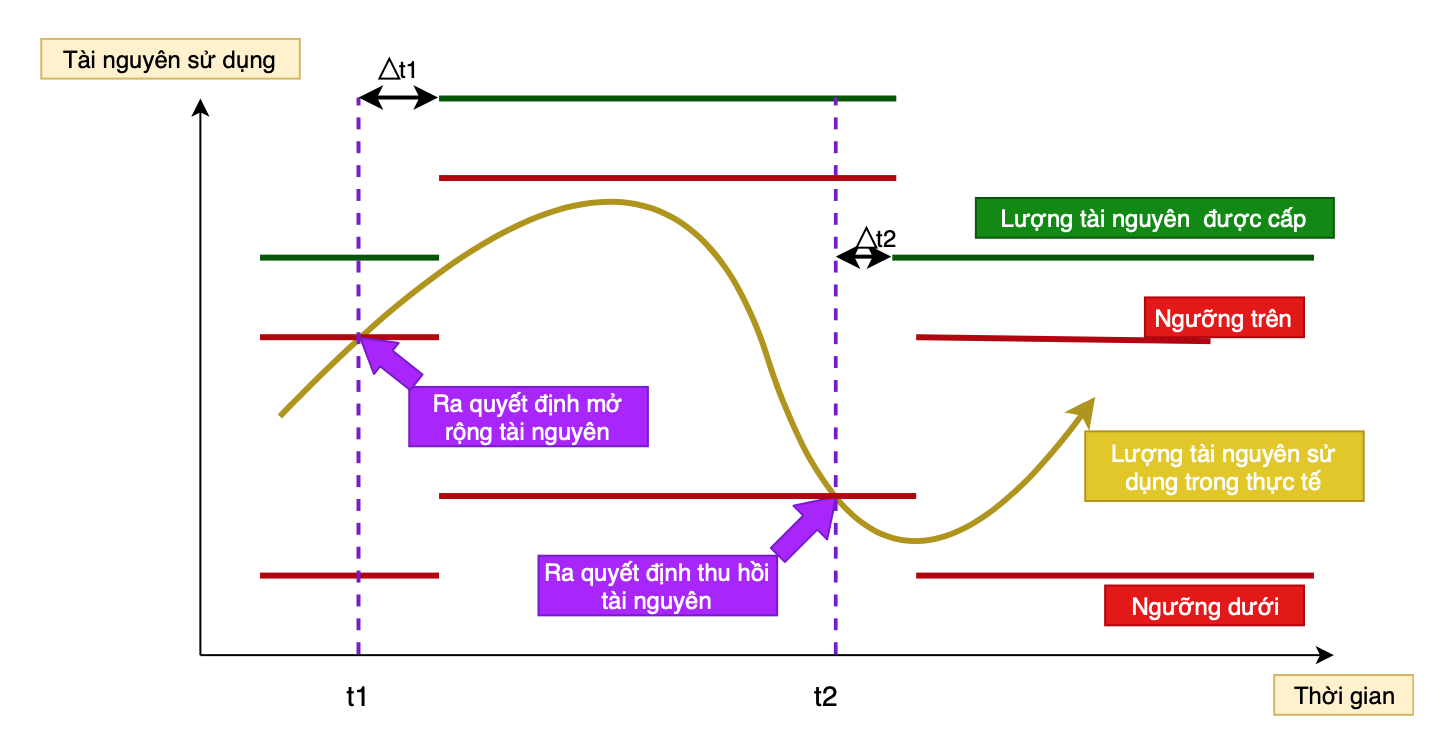
Hình 09 minh họa quá trình cung cấp tài nguyên một cách cố định tới người dùng các dịch vụ điện toán đám mây với sự biến thiên của người dùng hệ thống theo thời gian. Theo đó, hình bên trái thể hiện việc cung cấp tài nguyên quá hào phóng có thể dẫn đến tình trạng lãng phí tài nguyên (phần màu xám) mặc dù đảm bảo rằng tất cả người dùng sẽ được đáp ứng yêu cầu của họ. Trong khi với hình bên phải, việc cung cấp lượng tài nguyên thấp hơn sẽ giúp tiết kiệm tài nguyên hơn, tuy nhiên, phần yêu cầu màu xám là phần không đáp ứng đủ của hệ thống đối với yêu cầu người dùng. Vì vậy, giải pháp đặt ra là sẽ không cố định lượng tài nguyên cung cấp cho người dùng mà thay vào đó cung cấp một cách linh hoạt cho họ, nhằm đảm bảo việc tránh cung cấp thừa gây lãng phí tài nguyên và đảm bảo chất lượng dịch vụ (QoS) tới khách hàng.

Có hai cách tiếp cận cho việc mở rộng tài nguyên đối với các hệ thống điện toán đám mây. Đó là mở rộng tài nguyên theo chiều dọc và mở rộng tài nguyên theo chiều ngang. Đối với mở rộng tài nguyên theo chiều dọc, việc mở rộng tài nguyên sẽ tiếp cận theo hướng bổ sung tài nguyên vào máy ảo đã được cung cấp cho người dùng. Ví dụ khi người dùng đang được cung cấp một máy ảo có 4GB bộ nhớ và 2 lõi CPU. Khi người dùng muốn mở rộng tài nguyên, hệ thống sẽ bổ sung thêm 4GB bộ nhớ và thêm 2 lõi CPU vào máy ảo hiện tại của người dùng. Tuy nhiên, cách tiếp cận này ít phổ biến hơn vì việc tăng tài nguyên như thế sẽ gặp giới hạn nếu máy ảo cần lượng tài nguyên lớn hơn tài nguyên của máy chủ vật lý. Cách tiếp cận mở rộng theo chiều ngang được sử dụng phổ biến hơn. Với cách tiếp cận này, khi người dùng cần thêm tài nguyên, hệ thống sẽ bổ sung một máy ảo mới để phục vụ nhu cầu của người dùng. Đối với bài toán tự động mở rộng tài nguyên sử dụng đối với các dịch vụ đám mây, có 3 phương pháp chính bao gồm: Phương pháp chu kỳ (Periodicity), Phương pháp mở rộng dựa trên ngưỡng (Threshold) và Phương pháp dự đoán trước tài nguyên sử dụng. Trước tiên, ta xét tới phương pháp chu kỳ. Với phương pháp chu kì này, trong suốt quá trình sử dụng mô hình SaaS có chu kỳ tiêu dùng tài nguyên theo thời gian, minh hoạ ở hình ?? (ví dụ phút, giờ, ngày hay tháng). Sử dụng những đặc điểm trên, nhà cung cấp có thể căn cứ vào các thời điểm phù hợp để có thể ra quyết định tăng giảm cho tài nguyên cung cấp, cũng chính cách làm trên dẫn đến nhược điểm lớn nhất của phương pháp này là hệ thống không thể đáp ứng các yêu cầu cần tài nguyên ngay tức thì từ ứng dụng.



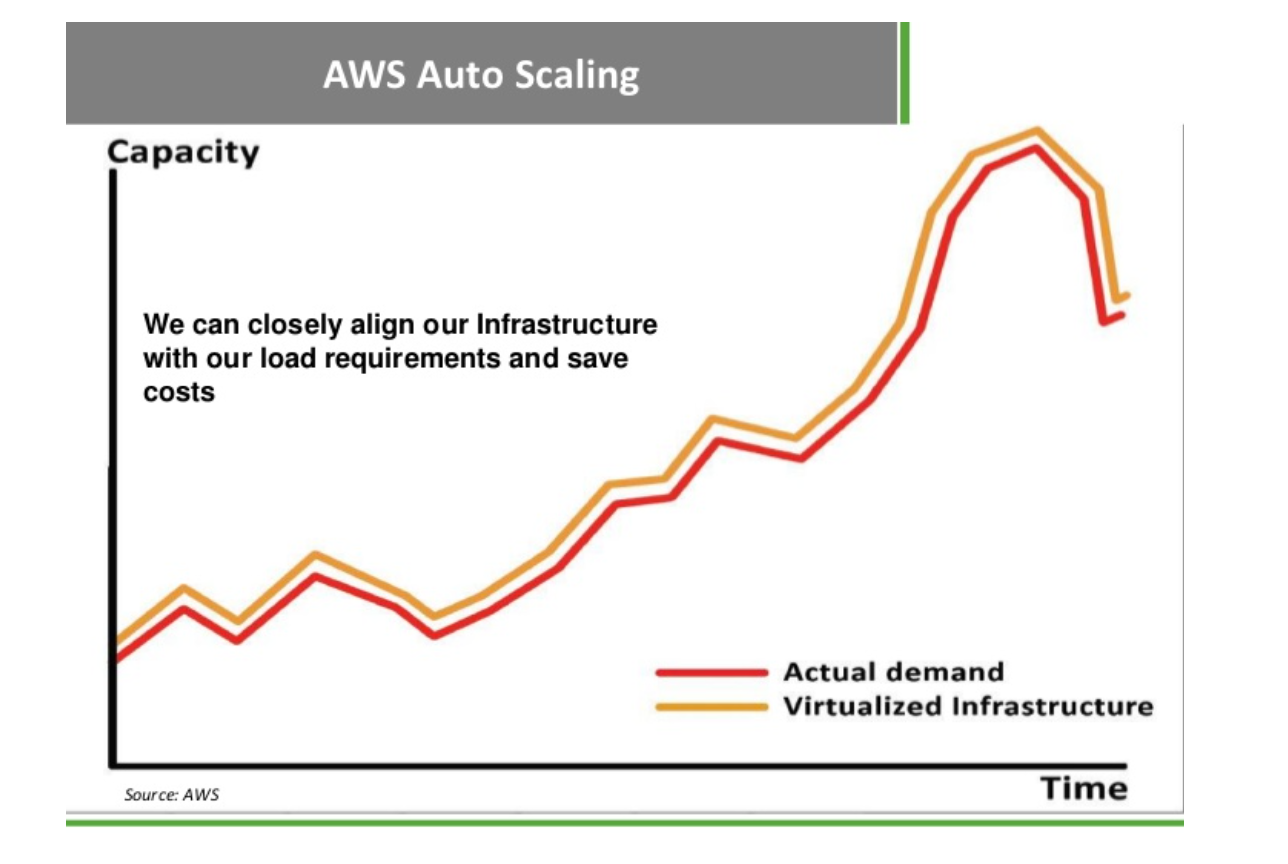
Hình : Phương pháp sử dụng chu kỳ

Đối với phương pháp sử dụng ngưỡng, người dùng sẽ định nghĩa một giá trị ngưỡng đối với các thông số sử dụng của hệ thống (CPU, Memory, ...) để quyết định xem khi nào hệ thống sẽ tăng hay giảm lượng tài nguyên cung cấp cho ứng dụng.



Hình : Co giãn tài nguyên sử dụng ngưỡng

Hình 11 minh họa việc sử dụng ngưỡng để mở rộng tài nguyên cho hệ thống. Tại thời điểm t, hệ thống được cung cấp một lượng tài nguyên sử dụng. Người dùng cũng sẽ định nghĩa giá trị ngưỡng trên và giá trị ngưỡng dưới. Khi lượng tài nguyên tiêu thụ của hệ thống chạm tới giá trị ngưỡng trên, hệ thống sẽ tự động mở rộng thêm máy ảo để đáp ứng nhu cầu của khách hàng và làm ngược lại nếu lượng tài nguyên tiêu thụ của hệ thống chạm tới giá trị ngưỡng dưới nhằm tiết kiệm tài nguyên của hệ thống. Tuy nhiên, cách tiếp cận này có một vài nhược điểm. Đầu tiên, đó là việc rất khó để xác định được giá trị ngưỡng phù hợp để đảm bảo việc tránh lãng phí tài nguyên cũng như đảm bảo chất lượng người dùng. Hơn nữa, việc ra quyết định mở rộng thêm máy ảo sẽ mất một khoảng thời gian (∆t) để tiến hành. Sau khoảng thời gian này, lượng tài nguyên tiêu thụ của hệ thống đã thay đổi, có thể đã vượt khả năng cung cấp của hệ thống (ảnh hưởng tới chất lượng dịch vụ) hoặc đã giảm xuống tới mức không cần mở rộng nữa (lãng phí tài nguyên).



Hình : Co giãn tài nguyên bằng phương pháp dự đoán

Một cách tiếp cận khác cho bài toán tự động mở rộng đó là dự đoán trước lượng tài nguyên sử dụng của hệ thống trong tương lai. Kỹ thuật cốt lõi được sử dụng trong cách tiếp cận này là phân tích chuỗi thời gian. Theo đó, các giá trị chuỗi thời gian thường có các tính chất: Tính chu kì, tính xu hướng, tình ngẫu nhiên và tính có thể dự đoán được. Công việc của cách tiếp cận này là thu thập lượng tài nguyên sử dụng trong quá khứ, sau đó xây dựng các mô hình nhằm học được các tính chất của chuỗi thời gian đó và mô hình hóa chuỗi thời gian nhằm đưa ra dự đoán về giá trị tài nguyên sử dụng trong tương lai. Hình ?? mô tả tài nguyên sử dụng trong thực tế và tài nguyên sử dụng được dự đoán. Từ đó, việc mở rộng hệ thống sẽ được quyết định dựa vào lượng tài nguyên dự đoán. Hiện nay cũng đã có rất nhiều các nghiên cứu được công bố nhằm giải quyết bài toán này. Tuy nhiên, cách tiếp cận này vẫn chưa được áp dụng phổ biến ở thời điểm hiện tại. Lí do là các nhà cung cấp dịch vụ đám mây vẫn còn nghi ngờ về độ chính xác của các mô hình dự đoán. Ngoài ra, các nghiên cứu này mới chỉ đánh giá các độ đo đơn lẻ trong khi việc ra quyết định mở rộng hệ thống lại phụ thuộc vào nhiều độ đo khác nhau về tài nguyên sử dụng của hệ thống.

Mục đích chính của kĩ thuật phân phối tài nguyên chủ động là dự đoán một cách chính xác lượng tài nguyên sử dụng. Có một số lượng lớn nghiên cứu về các mô hình dự đoán chuỗi thời gian trong điện toán đám mây. Trong [13], các phương pháp bao gồm sự tự tương quan (AR), trung bình động (MA), lỗi theo xu hướng và chu kì (ETS), sự tự tương quan kết hợp trung bình động (ARIMA) và mô hình mạng nơ-ron lan truyền ngược được đánh giá và so sánh trong việc dự đoán tải của hệ thống đám mây. Gần đây, các mô hình học sâu xuất hiện như một giải pháp hiệu quả cho các vấn đề dự đoán. Trong số đó, mạng nơ ron hồi quy (RNN) được cho là hữu ích đối với việc phân tích dữ liệu chuỗi thời gian vì có sự liên quan giữa các điểm dữ liệu trong loại dữ liệu này và RNNS được thiết kế để mô hình phụ thuộc thời gian và đà tạo bằng cách lan truyền ngược thông qua thời gian. Mạng RNN khác biệt với mạng nơ ron truyền thẳng ở chính sự hoạt động của nó. Vì vậy, luồng thông tin được truyền ngược lại thông qua các vòng lặp phản hồi. Để cải thiện khả năng ghi nhớ của RNN, Hochreiter và các cộng sự [15] đã đề xuất mạng bộ nhớ ngắn hạn dài (LSTM), là khối phổ biến nhất trong các tầng mạng RNN. Với sự cải thiện này, RNN có khả năng nhận diện và học các phụ thuộc trong khoảng thời gian dài. Trong các nghiên cứu áp dụng vào trong hệ thống đám mây, Jitendra và các cộng sự [6] đã đề xuất mô hình LSTM-RNN cho việc dự đoán tải trên các trung tâm dữ liệu đám mây sử dụng bộ dữ liệu NASA. Trong [11], các tác giả đã xây dựng một hệ thống đám mây tự mở rộng khai thác các tài nguyên đa biến, giải thuật di truyền, giải thuật lan truyền ngược và mạng nơ ron lan truyền ngược để dự đoán tải sử dụng. Mặc dù các nỗ lực học sâu đã được đề xuất để sử dụng cho các mô hình đám mây nhưng những mô hình này mới chỉ tập trung vào các chuỗi thời gian một chiều.

Gần đây, một loạt các nghiên cứu cho việc áp dụng các mô hình autoencoder cho các bài toán liên quan tới dữ liệu chuỗi thời gian như: Dự đoán, ước lượng giá trị không chắc chắn, Dự báo các điểm ngoại lai, … đã được công bố. Uber đã xây dựng mô hình Bayesian neural network với 2 thành phần chính bao gồm: Encoder-decoder được xây dựng dựa trên mô hình LSTM inference được xây dựng với mô hình mạng ANN, LSTM nhằm dự đoán giá trị tương lai và xác định độ không chắc chắn của mỗi giá trị dự đoán bằng cách áp dụng các tiếp cận Bayesian với phương pháp Montecarlo dropout.

Trong phạm vi nghiên cứu bài tập lớn môn học, nhận thấy tiềm năng của việc phát triển các mô hình dự đoán sử dụng autoencoder pretraining, nhóm chúng em tiến hành xây dựng và thực nghiệm kiến trúc dự đoán sử dụng autoencoder pretraining cho bài toán co giãn tài nguyên trong điện toán đám mây. Kết quả thực nghiệm được so sánh với mạng LSTM có cấu trúc tương tự. Nhóm chúng em tiến hành thực nghiệm với bộ dữ liệu google cluster trace để tiến hành đánh giá độ hiệu quả của cả 2 phương pháp.

## Cài đặt thực nghiệm

Chúng tôi sử dụng bộ dữ liệu Google cluster trace để tiến hành đánh giá hệ thống của chúng tôi. Bô dữ liệu là sự thu thập của nhiều job và lượng tài nguyên sử dụng của chúng được báo cáo bởi Google năm 2011. Mỗi job chứa nhiều task khác nhau và chạy đồng thời trên nhiều máy khác nhau. Bộ dữ liệu này bao gồm có 20 thông số khác nhau được đo trên một vài thông số chính đó là CPU, bộ nhớ, không gian lưu trữ cục bộ,... Dữ liệu được đo trên 12,5 nghìn máy khác nhau trong vòng 29 ngày với 1232799308 bản ghi dữ liệu về tài nguyên sử dụng. Chúng tôi đánh giá và dự đoán lượng tài nguyên sử dụng cho toàn bộ các job tham gia vào hệ thống. Chúng tôi sử dụng dữ liệu trong 24 ngày đầu để xây dựng mô hình học và 5 ngày cuối cùng để đánh giá hiệu năng của mô hình.

## Kết quả thực nghiệm của nhóm

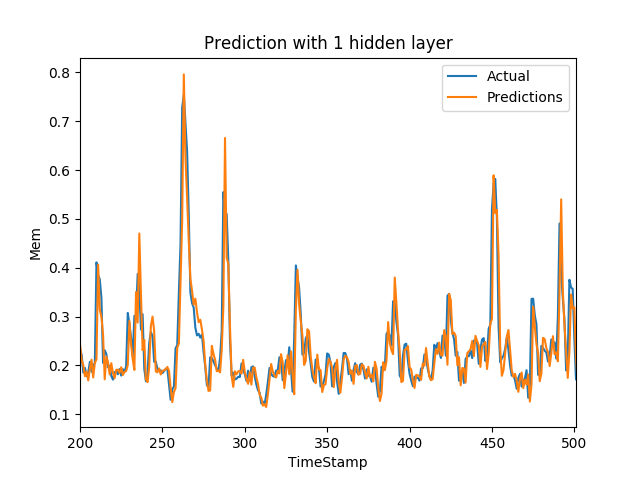
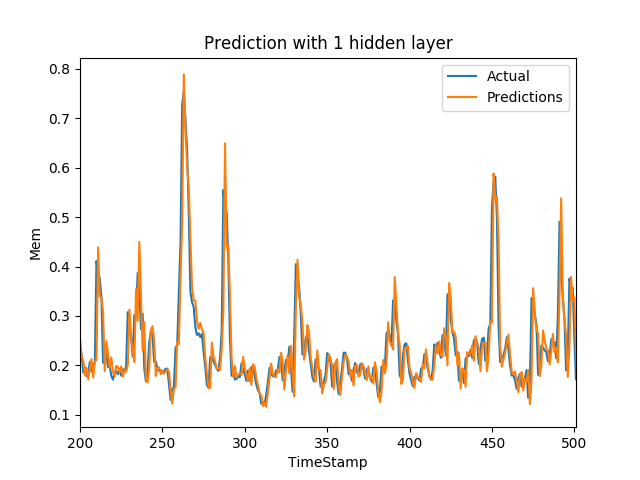
### Đánh giá kết quả lỗi tổng quan

Mục đích của thí nghiệm này là để chỉ ra kết quả tổng quan của cả 2 mô hình LSTM và Autoencoder pretraining model.

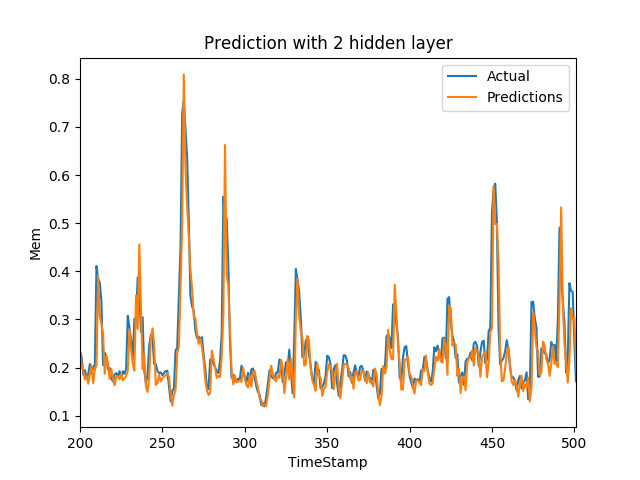
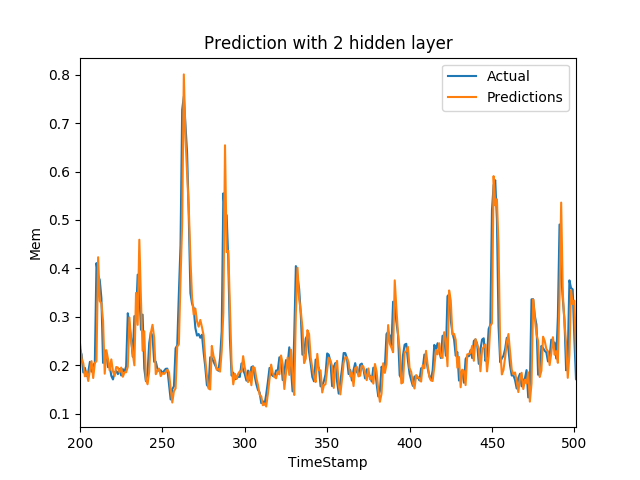
Bảng : Phân tích lỗi của 2 mô hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | LSTM | | Autoencoder pretraining | |
| MAE | RMSE | MAE | RMSE |
| 1 hidden layer | **0.0214** | **0.0333** | 0.0215 | **0.0333** |
| 2 hidden layer | **0.0213** | **0.0332** | 0.0229 | 0.0352 |
| 3 hidden layer | 0.0252 | 0.0375 | **0.0229** | **0.0354** |
| 4 hidden layer | **0.0201** | **0.0326** | 0.0214 | 0.0339 |

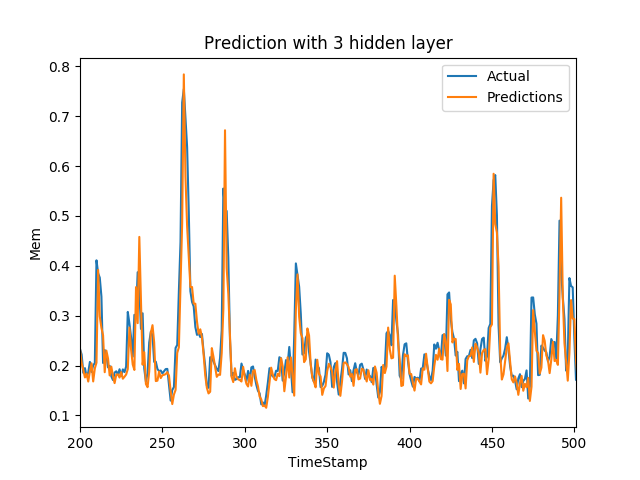
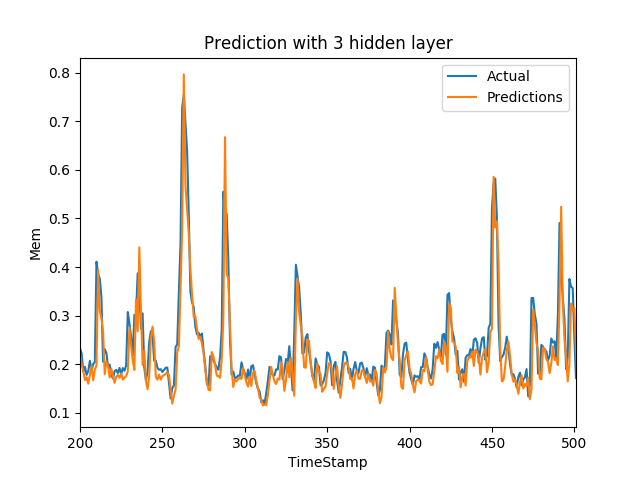
Bảng trên chỉ ra giá trị lỗi tổng quan trên tập dữ liệu thử nghiệm với 2 mô hình LSTM truyền thống và mô hình với autoencoder pretraining với số lượng hidden layer khác nhau. Kết quả tốt nhất thuộc về trường hợp có 4 hidden layer sử dụng mạng LSTM truyền thống với giá trị MAE đạt 0.0201 và RMSE đạt 0.0326. Mô hình Autoencoder pretraining cũng có một điểm sáng khi có kết quả tốt nhất trong trường hợp mạng có 3 tầng ẩn.



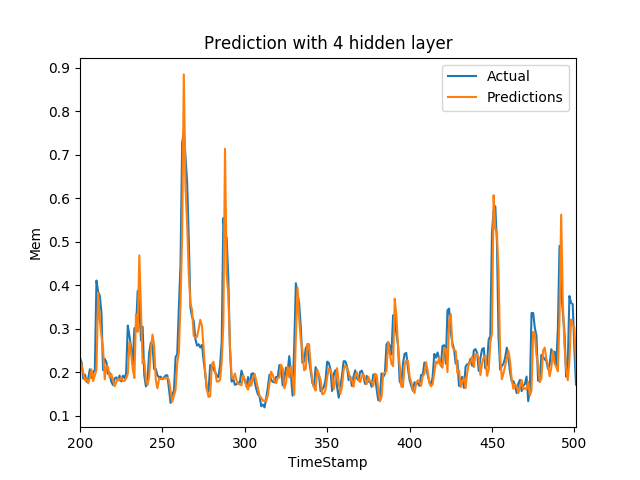
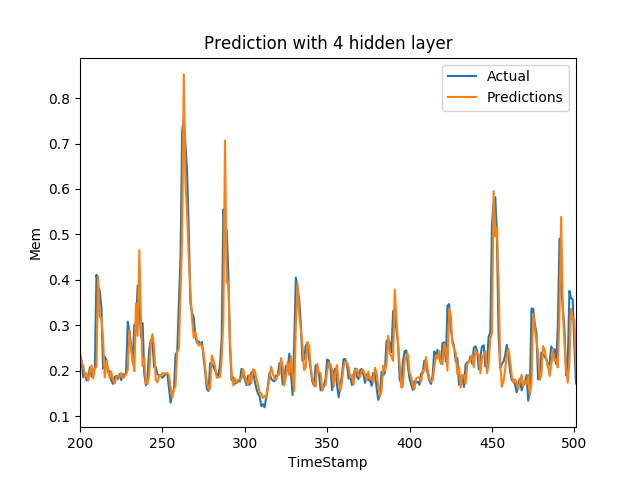
|  |  |
| --- | --- |
| Hình 1a. Dự đoán với 1 tầng ẩn LSTM | Hình 1b. Dự đoán với autoencoder pretraining có 1 tầng ẩn LSTM |



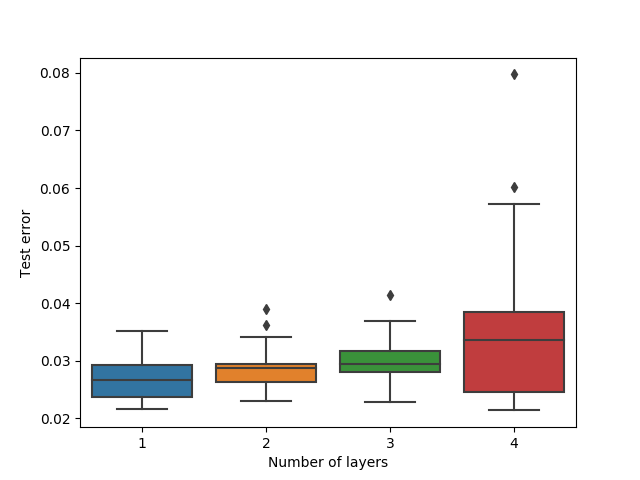
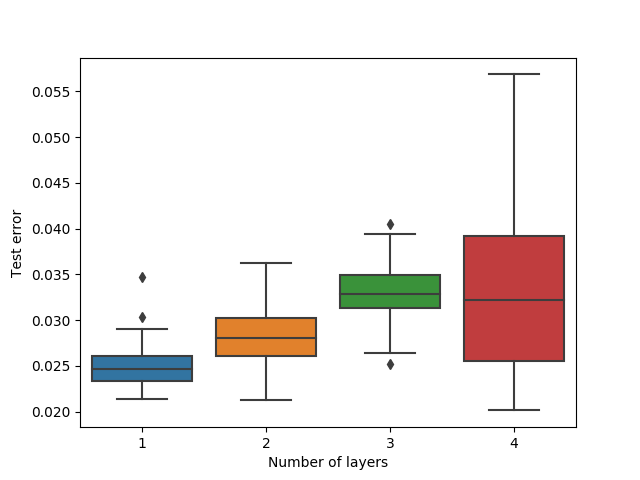
|  |  |
| --- | --- |
| Hình 2a. Dự đoán với 2 tầng ẩn LSTM | Hình 2b. Dự đoán với autoencoder pretraining có 2 tầng ẩn LSTM |



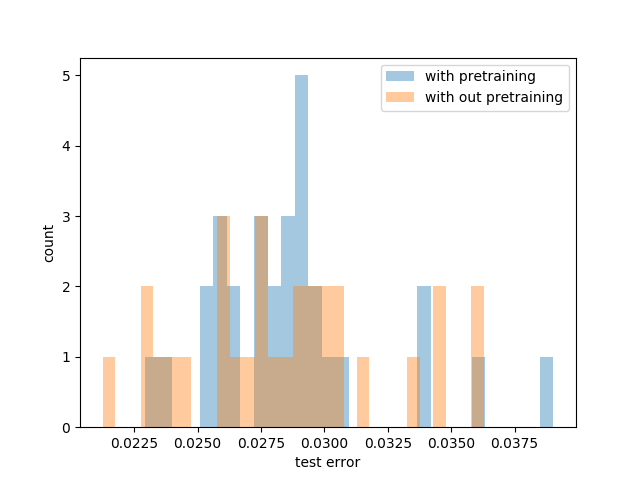
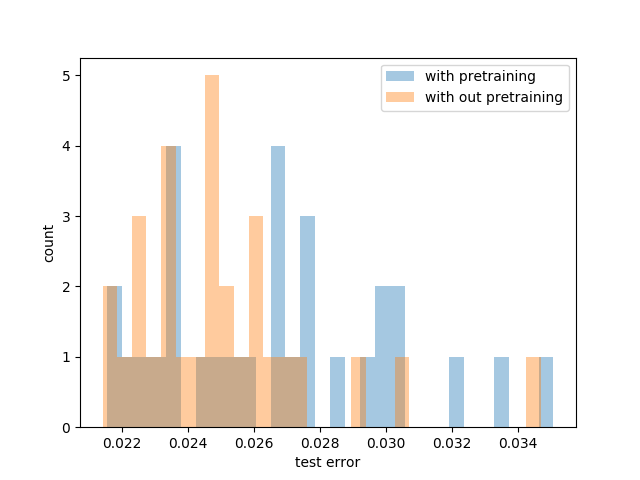
|  |  |
| --- | --- |
| Hình 3a. Dự đoán với 3 tầng ẩn LSTM | Hình 3b. Dự đoán với autoencoder pretraining có 3 tầng ẩn LSTM |



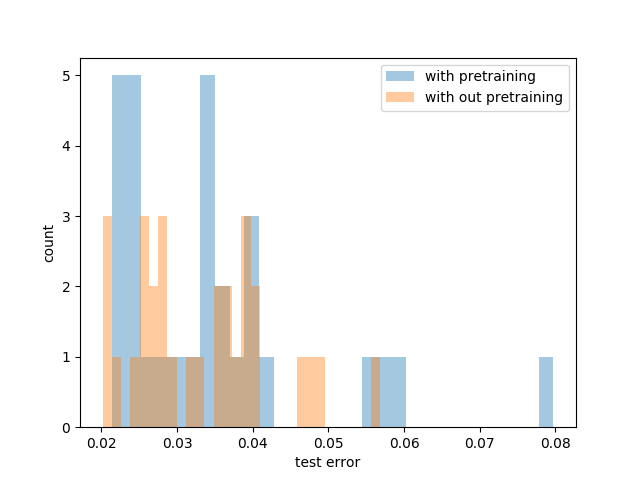
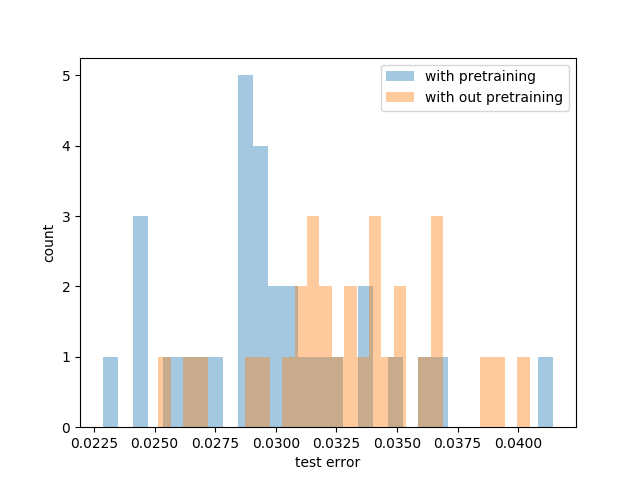
|  |  |
| --- | --- |
| Hình 4a. Dự đoán với 4 tầng ẩn LSTM | Hình 4b. Dự đoán với autoencoder pretraining có 4 tầng ẩn LSTM |



Hình 5. Phân phối tỉ lệ lỗi của 30 seeds



|  |  |
| --- | --- |
| Biểu đồ thể hiện phân bố giá trị lỗi với 1 hidden layer | Biểu đồ thể hiện phân bố giá trị lỗi với 2 hidden layer |



|  |  |
| --- | --- |
| Biểu đồ thể hiện phân bố giá trị lỗi với 3 hidden layer | Biểu đồ thể hiện phân bố giá trị lỗi với 4 hidden layer |

# Tổng kết

Trong nội dung nghiên cứu của môn học, nhóm chúng em đã tiến hành phân tích, đánh giá tổng thể nghiên cứu: “Why Does Unsupervised Pre-training Help Deep Learning?”. Nhóm đã trình bày các vấn đề bài toán, các phân tích và đánh giá của nhóm tác giả đã thực hiện trong nghiên cứu này. Dựa trên các phân tích và đánh giá đó, cùng với quá trình tìm hiểu bài toán co giãn tài nguyên trong các hệ thống điện toán đám mây, cùng với tìm hiểu các phương pháp hiện tại sử dụng các mô hình học không giám sát được huấn luyện trước hiện nay, nhóm đã triển khai và đánh giá tính hiệu quả của mô hình autoencoder pretraining trong bài toán này. Kết quả chỉ ra chưa thực sự khả quan trên các khía cạnh mà nhóm đánh giá và phân tích. Trong tương lai, nhóm mong muốn tiếp tục tiến hành triển khai thực nghiệm sâu hơn nữa với các bộ dữ liệu truy vết hệ thống điện toán đám mây khác.

# Tài liệu tham khảo

[1] *Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.*

[2] *Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, November 1997*.

[3] *Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. Procedia Computer Science, 125:676 – 682, 2018. The 6th International Conference on Smart Computing and Communications.*

[4] *Charles Reiss, Alexey Tumanov, Ganger, and et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In Proceedings of the Third ACM Symposium on Cloud Computing, SoCC ’12, pages 7:1–7:13, 2012.*

[5] *Carlos Vazquez, Ram Krishnan, and Eugene John. Time series forecasting of cloud data center workloads for dynamic resource provisioning. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 6(3):87–110, 2015.*

[6] *https://eng.uber.com/neural-networks/*

[7]*https://github.com/danielgy/Paper-List-of-Time-Series-Forecasting-with-Deep-Learning*