Understanding How Pretraining Regularizes Deep Learning Algorithms

Yu Yao, Baosheng Yu, Chen Gong, *Member, IEEE*, and Tongliang Liu, *Senior Member, IEEE*

***Abstract*— Deep learning algorithms have led to a series of breakthroughs in computer vision, acoustical signal processing, and others. However, they have only been popularized recently due to the groundbreaking techniques developed for training deep architectures. Understanding the training techniques is important if we want to further improve them. Through extensive experimentation, Erhan *et al.* (2010) empirically illustrated that unsupervised pretraining has an effect of regularization for deep learning algorithms. However, theoretical justifications for the observation remain elusive. In this article, we provide theoretical supports by analyzing how unsupervised pretraining regularizes deep learning algorithms. Specifically, we interpret deep learning algorithms as the traditional Tikhonov-regularized batch learning algorithms that simultaneously learn predictors in the input feature spaces and the parameters of the neural networks to produce the Tikhonov matrices. We prove that unsupervised pretraining helps in learning meaningful Tikhonov matrices, which will make the deep learning algorithms uniformly stable and the learned predictor will generalize fast w.r.t. the sample size. Unsupervised pretraining, therefore, can be interpreted as to have the function of regularization.**

***Index Terms*— Classification, deep learning, regularization, representation learning, unsupervised pretrain.**

I. Introduction

DEEP neural networks having deep architectures (or many layers of nonlinearities) are efficient to compactly represent highly varying functions [2]–[5] and thus highly discriminative feature representations. Deep learning algorithms thus usually have small approximation errors and are easy to be over-fitted [6]–[8] or be stuck into bad local solutions [9]. Before the breakthroughs of optimization techniques, e.g., [10], [11], it appeared that deep architectures were not successfully trained due to the aforementioned reasons.

Manuscript received August 2, 2020; revised June 29, 2021; accepted October 26, 2021. The work of Chen Gong was supported in part by the NSF of China under Grant 61973162 and in part by the NSF of Jiangsu Province under Grant BZ2021013. The work of Tongliang Liu was supported in part by ARC under Grant DP180103424 and Grant DE190101473. *(Corresponding author: Tongliang Liu.)*

Yu Yao, Baosheng Yu, and Tongliang Liu are with the School of Computer Science, The University of Sydney, Camperdown, NSW 2006, Australia (e-mail: yyao0814@uni.sydney.edu.au; baosheng.yu.usyd@gmail.com; tongliang.liu@sydney.edu.au).

Chen Gong is with the PCA Laboratory, and the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: chen.gong@njust.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3131377.

Digital Object Identifier 10.1109/TNNLS.2021.3131377

Unsupervised pretraining methods, such as restricted Boltzmann machines (RBMs) [12], [13] and autoencoder [14], [15], had been popular training techniques for deep neural networks, such as multilayer perceptron (MLP) and convolutional neural network (CNN). They had been widely empirically studied and had achieved impressive results in several areas [16]–[23].

Due to some recent advances, such as large-scale labeled datasets [24], [25], rectified linear unit (ReLU) [26], designed initialization [27]–[29], data augmentation [6], and dropout [30], CNNs without unsupervised pretraining offer some state-of-the-art performances on large-scale labeled datasets. As a sequel, unsupervised pretraining seems have been fallen out of favor for supervised networks due to the recent advances.

To answer the question "*is unsupervised pretraining still useful given recent advances*," Paine *et al.* [31] empirically found that unsupervised pretraining, as expected, is helpful to train CNNs when the ratio between unsupervised and supervised sample is high. A review of deep learning in *Nature* [32] expects much for learning with unsupervised observations by concluding with the statement "*unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised learning. Although we have not focused on it in this review, we expect unsupervised learning to become far more important in the longer term.*" Motivated by this expectation, in this article, we will focus on analyzing how unsupervised pretraining regularizes deep learning algorithms.

Theoretical analysis for the success of unsupervised pretraining remains elusive in the literature. However, it is essential for deep understanding and further improving the unsupervised pretraining techniques. Some empirical work has been conducted to understand the function of unsupervised pretraining. Bengio *et al.* [11] empirically concluded that the unsupervised training strategy helps optimization by initializing weights in a region near a good local minimum that brings improved generalization; Erhan *et al.* [1] empirically showed that unsupervised pretraining plays a significant role of regularization in training the deep architectures.

Different from the previous works which analyze the property of unsupervised pretraining based on the empirical results, in this article, we analyze the property of unsupervised pretraining from a theoretical perspective. Specifically, we have compared unsupervised pretraining with manifold regularization which shows that it helps find meaningful representations of instances in lower dimensional subspace. We have also provided the generalization bound when unsupervised pretraining is used, which shows that unsupervised pretraining helps the learned predictor generalize fast. Motivated by this, we have also proposed a computational cost-efficient regularizer that improves all the baselines on both clean and ambiguous datasets.

The rest of the article is organized as follows. In Section II, we set up notation and briefly introduce unsupervised pretraining and fine-tuning. In Section III, we interpret unsupervised pretraining as to have the function of regularization. In Section IV, we interpret deep learning algorithms as the traditional batch learning algorithms with Tikhonov regularizations that simultaneously learn predictors in the input feature space and the parameters of the neural networks to form the Tikhonov matrices. In Section V, we show that unsupervised pretraining can help deep learning algorithms be uniformly hypothesis stable and generalize fast. The experimental evaluation is included in Section VII. The detailed proofs are provided in Section VI. Section VIII contains concluding remarks.

II. Notation and Preliminaries

We use upper case nonitalics to denote sets. We use upper case italics to denote matrices, A_i to denote the ith column of the matrix A, and A_{ij} to denote the ijth entry of A. We use lower case italics to denote vectors and scalers, where a_i means the ith entry of a. The inner product and norm in Euclidean space are denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|_2$.

A. Unsupervised Pretraining

In deep learning algorithms, unsupervised pretraining can be used to initialize the parameters of the neural network. During the unsupervised pretraining procedure, RBM [12], [13] or auto-encoder [11], [19], [33] is usually used as single-layer components to perform initialization. Bengio *et al.* [11], Larochelle *et al.* [34], and Vincent *et al.* [14] empirically showed that RBM and auto-encoder have similar performances. Bengio and Delalleau [35] further showed that the RBMs trained by contrastive divergence and the auto-encoder training criterion are close in the sense that they both minimize approximations of the log-likelihood of a generative model. Specifically, they use the gradient of a stochastic reconstruction error to estimate the gradient of the log-likelihood of RBM. The mean-field approximation of the reconstruction error is the same as the reconstruction error typically used in training auto-encoder. This can be used to explain why reconstruction error has been used to monitor the progress in training RBMs; see, e.g., [11], [36]. In this article, we will theoretically analyze the function of unsupervised pretraining by exploiting its reconstruction property. For simplicity and clarity, we will focus on exploiting the auto-encoder training criterion for unsupervised pretraining.

Auto-encoder combines both encoder and decoder mappings in the hope that encoder mapping loses little information. The following deterministic mapping that transforms an input vector x into a hidden representation x' is called the encoder:

$$x' = \phi(Wx + b)$$

where W is an affine matrix, b is an offset vector, and $\phi : \mathbb{R}^D \to \mathbb{R}^D$ represents a nonlinear mapping that $\phi_i(x) = \psi(x_i)$ and $\psi : \mathbb{R} \to \mathbb{R}$. ReLU $\psi(\cdot) : x_i \mapsto (x_i)_+$ is currently widely used for deep learning algorithms.

A deterministic mapping that transforms the hidden representation x' back to a vector $\hat{x}$, which forces a small reconstruction error $\|x - \hat{x}\|_2^2$, is called the decoder. $\hat{x}$ can be defined as

$$\hat{x} = \phi\left(W^\top x' + b'\right) \quad \text{or } \hat{x} = W^\top x' + b'$$

where W' is an affine matrix and b' is an offset vector. In this article, for simplicity, we discuss the linear case $\hat{x} = W^T x'$ and do not consider the offset.[1]

Let U be the training sample for unsupervised pretraining. It contains $x_{1,p}, \ldots, x_{m,p} \in \mathbb{R}^c$ which are the independent and identically distributed (i.i.d.) observations, where m is the sample size, and p indicates that the observations are used for pretraining. Let $x_{1,p}^{(l)}, \ldots, x_{m,p}^{(l)}$ be the new representations of the observations in the lth layer (note that $x_{1,p}^{(0)}, \ldots, x_{m,p}^{(0)} = x_{1,p}, \ldots, x_{m,p}$), where $l \in \{1, \ldots, L\}$ and L is the number of the layers used in the neural network.

In unsupervised pretraining, we consider that the parameters $W^{(l)} \in \mathbb{R}^{\mathcal{D}(l-1)\times\mathcal{D}(l)}$ of the lth layer are learned by an auto-encoder, where $\mathcal{D}(l)$ is the dimensionality of the output of the lth layer, and $\mathcal{D}(0) = c$ and $\mathcal{D}(L) = d$. That is

$$W_U^{(l)} = \arg \min_{W^{(l)} \in \mathbb{R}^{\mathcal{D}(l-1)\times\mathcal{D}(l)}} \frac{1}{m} \sum_{i=1}^{m} \left\| x_{i,p}^{(l-1)} - W^{(l)\top} \phi\left(W^{(l)} x_{i,p}^{(l-1)}\right) \right\|_2^2.$$

Then, auto-encoder encourages the residual

$$\varepsilon\left(x_{i,p}, W_U^{(1)}, \ldots, W_U^{(L)}\right) = x_{i,p} - \left(W_U^{(1)}\right)^\top, \ldots, \left(W_U^{(L)}\right)^\top \phi\left(W_U^{(L)}, \ldots, \phi\left(W_U^{(1)} x_{i,p}\right)\right) \tag{1}$$

to be small.

B. Fine-Tuning

In deep learning algorithms, a supervised fine-tuning step uses the labeled sample to learn a predictor v' and tune the pretrained parameters of the neural network, simultaneously.

Specifically, let S be the training sample for fine-tuning. It contains $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^c \times \mathbb{R}$ which are the i.i.d. observations, where n is the sample size. Fine-tuning can be modeled as

$$\left\{v'_S, W_S^{(1)}, \ldots, W_S^{(L)}\right\} = \arg \min_{v' \in \mathbb{R}^d, W^{(l)} \in \mathcal{B}\left(W_U^{(l)}, r\right), l \in \{1,\ldots,L\}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \left\langle v', \phi\left(W^{(L)}, \ldots, \phi\left(W^{(1)} x_i\right)\right)\right\rangle\right) + \lambda \|v'\|_2^2 \tag{2}$$

[1]Note that our analysis can be easily extended to the case of nonlinear decoding or $W' \neq W$ or having the offset.

where ℓ is a convex surrogate loss function, λ is a tradeoff parameter, and $\mathcal{B}(W_U^{(l)}, r)$ denotes the closed ball of radius r centered at $W_U^{(l)}$.

Without the unsupervised pretraining step, there will be no specific restriction placed on the parameters of the neural network. However, it is reasonable to assume that unsupervised pretraining makes r to be small, i.e., the parameters learned by fine-tuning are dependent on the parameters initialized by unsupervised pretraining. Thus, the residual

$$\begin{aligned}\varepsilon\Big(x_i, W_S^{(1)}, \ldots, W_S^{(L)}\Big) \\ = x_i - \Big(W_S^{(1)}\Big)^\top, \ldots, \Big(W_S^{(L)}\Big)^\top \phi\Big(W_S^{(L)}, \ldots, \phi\Big(W_S^{(1)} x_i\Big)\Big)\end{aligned} \tag{3}$$

will also be small.

III. Unsupervised Pretraining Regularizes the Predictor

To clearly see the regularization property of unsupervised pretraining, we start with the case that fine-tuning does not tune the pretrained parameters.

Let $\upsilon \in \mathbb{R}^c$ represent a vector in the dual space of the input space, and υ' be a vector in the dual space of the latent space $\mathbb{R}^d$. If we do not update $W_U = \{W_U^{(1)}, \ldots, W_U^{(L)}\}$ in the fine-tuning step, the pretrained deep learning (2) can be modeled as

$$\begin{aligned}\upsilon'_{S,W_U} = \arg\min_{\upsilon' \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell\Big(y_i, \Big\langle \upsilon', \phi\Big(W_U^{(L)}, \ldots, \phi\Big(W_U^{(1)} x_i\Big)\Big)\Big\rangle\Big) \\ + \lambda \big\|\upsilon'\big\|_2^2.\end{aligned} \tag{4}$$

Lemma 1: Assuming that $x_{j,p}$, $j \in \{1, \ldots, m\}$ and x_i, $i \in \{1, \ldots, n\}$ have the same marginal distribution. If $W_U^{(1)}, \ldots, W_U^{(L)}$ are fixed in the fine-tuning step, then unsupervised pretraining regularizes the predictor υ as follows:

$$\begin{aligned}\upsilon_{S,W_U} &= \arg\min_{\upsilon \in \mathbb{R}^c} \frac{1}{n} \sum_{i=1}^n \ell\Big(y_i, \Big\langle \upsilon, x_i - \varepsilon\Big(x_i, W_U^{(1)}, \ldots, W_U^{(L)}\Big)\Big\rangle\Big) \\ &\quad + \lambda \Big\| W_U^{(L)}, \ldots, W_U^{(1)} \upsilon \Big\|_2^2 \\ &\approx \arg\min_{\upsilon \in \mathbb{R}^c} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \langle \upsilon, x_i \rangle) + \lambda \Big\| W_U^{(L)}, \ldots, W_U^{(1)} \upsilon \Big\|_2^2\end{aligned} \tag{5}$$

where $W_U^{(L)}, \ldots, W_U^{(1)} \upsilon = \upsilon'$, and y_i is the label of x_i.

In (5), we used $\sum_{i=1}^n \ell(y_i, \langle \upsilon, x_i \rangle)$ to approximate $\sum_{i=1}^n \ell(y_i, \langle \upsilon, x_i - \varepsilon(x_i, W_U^{(1)}, \ldots, W_U^{(L)}) \rangle)$. We would like to know how the approximation affects the minimizers. For simplicity, we use $\upsilon_1 = \arg\min_\upsilon f(\upsilon)$ and $\upsilon_2 = \arg\min_\upsilon (f(\upsilon) + \epsilon(\upsilon))$ to represent the problems in (4) and (5), respectively, where $f(\upsilon)$ is a strongly convex function w.r.t. υ, and $\epsilon(\upsilon)$ is a small value caused by the residual $\varepsilon(X, W_U^{(1)}, \ldots, W_U^{(L)})$. Note that $\epsilon(\upsilon)$ will converge to zero when the residuals are small enough and that a differentiable function $f(\upsilon)$ is c-strongly convex if for any two inputs υ_1 and υ_2 it holds that $\|\upsilon_1 - \upsilon_2\|^2 \le (\nabla f(\upsilon_1) - \nabla f(\upsilon_2))^\top (\upsilon_1 - \upsilon_2)/s$. Through some algebra and using Cauchy–Schwarz inequality, we have $\|\upsilon_1 - \upsilon_2\| \le \|\epsilon(\upsilon_2)\|/s$, which means that υ_1 and υ_2 will be close to each other when $\|\epsilon(\upsilon_2)\|$ is encouraged to be small.

The analysis is mainly based on the reconstruction property of unsupervised pretraining. If there are non-linear mappings in decoder, we could build the predictor by finding υ such that

$$\begin{aligned}&\Big\langle \upsilon', \phi\Big(W_U^{(L)}, \ldots, \phi\Big(W_U^{(1)} x_i\Big)\Big)\Big\rangle \\ &= \Big\langle \varphi_{W_U^{(L)}, \ldots, W_U^{(1)}}(\upsilon), \phi\Big(W_U^{(L)}, \ldots, \phi\Big(W_U^{(1)} x_i\Big)\Big)\Big\rangle \\ &= \Big\langle \upsilon, \phi\Big(\Big(W_U^{(1)}\Big)^\top, \ldots, \phi\Big(\Big(W_U^{(L)}\Big)^\top \\ &\qquad\qquad \times \phi\Big(W_U^{(L)}, \ldots, \phi\Big(W_U^{(1)} x_i\Big)\Big)\Big)\Big)\Big\rangle.\end{aligned}$$

Then, Tikhonov regularization has a nonlinear form $\|\varphi_{W_U^{(L)}, \ldots, W_U^{(1)}}(\upsilon)\|_2^2$. Note that for a given υ', the predictor υ may not exist because the matrix $W_U^{(L)}, \ldots, W_U^{(1)}$ may not be invertible. However, we do not really obtain such predictor υ. With the help of υ, we could analyze the property of υ' learned in the newly represented space from the perspective of the input feature space, which is nontrivial when the network is nonlinear.

Equation (5) clearly shows that unsupervised pretraining regularizes the algorithms for learning predictors when all the parameters of a pretrained neural network are fixed. Learning algorithm (5) is the well-known Tikhonov-regularized batch learning algorithm, where the matrix $W_U^{(L)}, \ldots, W_U^{(1)}$ is called the Tikhonov matrix. When $W_U^{(L)}, \ldots, W_U^{(1)}$ degenerates to the identity matrix I, the learning algorithm (5) degenerates to the widely used ℓ_2-regularized batch learning algorithm. Generally, it would be more flexible to let algorithm exploit suitable values for the regularization matrix $W_U^{(L)}, \ldots, W_U^{(1)}$ rather than simply fixing $W_U^{(L)}, \ldots, W_U^{(1)} = I$. Unsupervised pretraining provides a way to find such a suitable Tikhonov matrix, and its function as regularization has been empirically justified [1].

We further show that unsupervised pretraining regularizes the deep neural network when both the predictor and the parameters of the pretrained neural network are tuned in the fine-tuning step. Specifically, let $\mathcal{B}(W_U^{(l)}, r)$ denote the closed ball of radius r centered at $W_U^{(l)}$. As mentioned in Section II-B, it is reasonable to assume that r will be small when there is unsupervised pretraining, and we then modify (5) to

$$\begin{aligned}&\Big\{\upsilon_S, W_S^{(1)}, \ldots, W_S^{(L)}\Big\} \\ &= \arg \min_{\upsilon \in \mathbb{R}^c, W^{(l)} \in \mathcal{B}\left(W_U^{(l)}, r\right), l \in \{1, \ldots, L\}} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \hat{y}_i) \\ &\quad + \lambda \big\| W^{(L)}, \ldots, W^{(1)} \upsilon \big\|_2^2.\end{aligned} \tag{6}$$

From (6), we can see that the unsupervised pretrained deep learning algorithms minimize not only the "distance" between true labels and predictions but also a term $\|W_U^{(L)}, \ldots, W_U^{(1)} \upsilon\|_2^2$. We will show that it also exploits the meaningful data representations for learning predictors, which can be seen by comparing it with manifold regularization [37]–[40].

Manifold regularization is formulated as $\lambda f^\top M f$, where $f = [v^\top x_1, \ldots, v^\top x_n]^\top$ and M is the generalized graph Laplacian matrix defined by

$$M_{ij} = \begin{cases} d_i, & \text{if } i = j \\ -\Omega_{ij}, & \text{if } i \neq j \end{cases}$$

where $d_i = \sum_{j \neq i} \Omega_{ij}$ is the degree of the ith vertex, Ω is the adjacency matrix usually defined by $\Omega_{ij} = \exp(-\|x_i - x_j\|_2^2 / 2\sigma^2)$, and σ is an adjustable kernel width. It can be verified that

$$f^\top M f = \frac{1}{2} \sum_{i=1}^{n} \Omega_{ij} \big(v^\top x_i - v^\top x_j\big)^2. \tag{7}$$

Next, we reformulate $\|W^{(L)}, \ldots, W^{(1)} v\|_2^2$ as follows. Let $X = [x_1, \ldots, x_n] \in \mathbb{R}^{c \times n}$, and $y \in \mathbb{R}^n$ be the corresponding labels of the instances. Furthermore, let

$$\begin{aligned} &\mathbf{\Phi}\big(X, W^{(1)}, \ldots, W^{(L)}\big) \\ &\quad = \big[\phi\big(W^{(L)}, \ldots, \phi\big(W^{(1)} x_1\big)\big), \ldots, \phi\big(W^{(L)}, \ldots, \phi\big(W^{(1)} x_n\big)\big)\big] \\ &\quad \in \mathbb{R}^{d \times n} \end{aligned}$$

and

$$\begin{aligned} &\varepsilon(X, W^{(1)}, \ldots, W^{(L)}) \\ &\quad = \big[\varepsilon\big(x_1, W^{(1)}, \ldots, W^{(L)}\big), \ldots, \varepsilon\big(x_n, W^{(1)}, \ldots, W^{(L)}\big)\big]^\top \end{aligned}$$

be the newly represented feature matrices and the residuals, respectively. For simplicity, we will use $\mathbf{\Phi}$ and $\varepsilon(X)$ to denote $\mathbf{\Phi}\big(X, W^{(1)}, \ldots, W^{(L)}\big)$ and $\varepsilon(X, W^{(1)}, \ldots, W^{(L)})$, respectively. Then, we have

$$\begin{aligned} &\big\|W^{(L)}, \ldots, W^{(1)} v\big\|_2^2 \\ &\quad = v^\top \big(W^{(1)}\big)^\top, \ldots, \big(W^{(L)}\big)^\top W^{(L)}, \ldots, W^{(1)} v \\ &\quad = v^\top \big(W^{(1)}\big)^\top, \ldots, \big(W^{(L)}\big)^\top \mathbf{\Phi}\big(\mathbf{\Phi}^\top \mathbf{\Phi}\big)^+ \mathbf{\Phi}^\top \\ &W^{(L)}, \ldots, W^{(1)} v \\ &\quad = v^\top (X - \varepsilon(X)) \big(\mathbf{\Phi}^\top \mathbf{\Phi}\big)^+ (X - \varepsilon(X))^\top v \\ &\quad = \hat{Y}^\top \big(\mathbf{\Phi}^\top \mathbf{\Phi}\big)^+ \hat{Y} \\ &\quad = \operatorname{tr}\Big(\hat{Y}^\top \big(\mathbf{\Phi}^\top \mathbf{\Phi}\big)^+ \hat{Y}\Big) \end{aligned} \tag{8}$$

where A^+ represents the pseudo inverse of matrix A, $\operatorname{tr}(\cdot)$ denotes the trace operator, and $\hat{Y} = (X - \varepsilon(X))^\top v$ are the predicted labels using the predictor v.

Note that the Laplacian matrix M in (7) measures the data structure and the corresponding manifold regularization exploits the pairwise (distance) similarities of the observations to guide predictions.

The matrix $(\mathbf{\Phi}^\top \mathbf{\Phi})^+$ in (8) is an inverse covariance matrix, also known as the concentration or precision matrix. It is helpful to find the pairwise correlation structure among multiple data points [41], which enables exploiting structure information across examples and thus leading to a good predictive performance. Specifically, the inverse covariance matrix measures the partial correlations[2] [43] and represents the statistical dependence relationships. Let $E(v) = F^\top M F$ be the quadratic energy function, Zhu [37] interpreted the Laplacian manifold learning as to have a model of Gaussian–Markov random field: $p_\beta(v) = e^{-\beta E(v)} / Z_\beta$, where $p_\beta(v)$ is the probability distribution on v, β is an "inverse temperature" parameter, and Z_β is the partition function. They also used an inverse covariance matrix by considering that the independence is expressed over data features instead of over data points and explained that the edges in the graph they created correspond to nonzeros in the inverse covariance matrix.

Note that to be positive semidefinite, zero row sums, positive diagonal, and negative otherwise is a necessary and sufficient condition for Laplacian matrix. Although it has been well-defined and frequently studied (see; e.g., [**?**], [37], [44]) that when graph signals are analyzed as random vectors with a Gaussian–Markov random field distribution, its inverse covariance matrix is a Laplacian matrix, it is interesting to see under what conditions the inverse covariance matrix in (8) is a Laplacian matrix. A future work is to study and design a deep network such that the corresponding matrix in (8) is (or is encouraged to be) a Laplacian matrix.[3]

Before proving that unsupervised pretraining helps predictors learned by deep learning algorithms generalize fast, in Section V, we interpret deep learning algorithms as the traditional batch learning algorithms with Tikhonov regularizations that simultaneously learn predictors in the input feature space and the parameters of the neural networks to form the Tikhonov matrices. Tikhonov regularization also encourages the neural networks to learn representations having pairwise correlations which are close to that of the labels.

IV. Unsupervised Pretraining Regularizes Deep Learning Algorithm

In Section III, to illustrate the regularization property of unsupervised pretraining, we have assumed that fine-tuning does not tune the parameters of the neural networks. However, in practice, the fine-tuning step tunes both the predictor and the parameters of the neural network. In this section, we show how unsupervised pretraining works in this scenario.

For a deep learning algorithm, we have written the model as

$$\begin{aligned} &\Big\{v'_S, W_S^{(1)}, \ldots, W_S^{(L)}\Big\} \\ &\quad = \arg \min_{v' \in \mathbb{R}^d, W^{(l)} \in \mathcal{B}\left(W_U^{(l)}, r\right), l \in \{1, \ldots, L\}} \\ &\qquad \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, \big\langle v', \phi\big(W^{(L)}, \ldots, \phi\big(W^{(1)} x_i\big)\big)\big\rangle\big) + \lambda \|v'\|_2^2 \end{aligned}$$

[2]If an entry of the inverse of a covariance matrix is equal to 0, it corresponds to a pair of variables that have no partial correlation. In other words, pairs of variables that are conditionally independent given all the other features in the data. We have interpreted $\mathbf{\Phi}^\top \mathbf{\Phi}$ as a covariance matrix by considering that the independence is being expressed over data features instead of over data points. The credibility of the consideration has been discussed in [42, Sec. 2.4].

[3]Note that Gaussian process has been exploited to incorporate the inverse of the regularized Laplacian as a prior [37] to discover structural relationships from data. Similar methods may be introduced to deep Gaussian processes [46] or deep probabilistic models [47].

where $\mathcal{B}(W_U^{(l)}, r)$ denotes the closed ball of radius r centered at $W_U^{(l)}$.

There is no doubt that the high performance of deep learning algorithms is due to its ability to learn meaningful new representations. We now show that deep learning algorithms can be interpreted as the traditional batch learning algorithms with Tikhonov regularization that simultaneously learn predictors in the input feature space and the parameters of the neural networks to form the Tikhonov matrices for regularization. Let $\mathbf{\Phi}$ and $\varepsilon(X)$ denote $\mathbf{\Phi}(X, W^{(1)}, \ldots, W^{(L)})$ and $\varepsilon(X, W^{(1)}, \ldots, W^{(L)})$, respectively, and $\hat{Y} = (X - \varepsilon(X))^\top v$. Similar to those of (5) and (8), let $v' = W^{(L)}, \ldots, W^{(1)} v$, we can rewrite the model of a deep learning algorithm as

$$\begin{aligned}
&\left\{v_S, W_S^{(1)}, \ldots, W_S^{(L)}\right\} \\
&\quad = \arg \min_{v \in \mathbb{R}^c, W^{(l)} \in \mathcal{B}\left(W_U^{(l)}, r\right), l \in \{1, \ldots, L\}} \\
&\qquad \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \lambda \left\| W^{(L)}, \ldots, W^{(1)} v \right\|_2^2 \\
&\quad = \arg \min_{v \in \mathbb{R}^c, W^{(l)} \in \mathcal{B}\left(W_U^{(l)}, r\right), l \in \{1, \ldots, L\}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) \\
&\qquad + \lambda \operatorname{tr}\left(\hat{Y}^\top \left(\mathbf{\Phi}^\top \mathbf{\Phi}\right)^+ \hat{Y}\right).
\end{aligned} \tag{9}$$

Equation (9) motivates us to explore a regularization $\lambda \operatorname{tr}(\hat{Y}^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^+ \hat{Y})$ for deep learning algorithms (such as CNNs).

From (9), we can see that deep learning algorithms not only minimize the empirical risks for predictions but also learn meaningful data representations and structures to help learn predictors. The empirical risk minimization algorithms usually minimize the distance between true label and predicted label by learning a predictor v (or v') minimizing $\sum_{i=1}^n \ell(y_i, \hat{y}_i)$. Equation (9) shows that the neural network also provides a penalty minimizing the difference between the pairwise similarities of the new representations and that of the predictions. Specifically, since $\hat{Y}^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^+ \hat{Y}$ is a semipositive definite matrix,[4] the trace regularization encourages the condition number $\lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of $\hat{Y}^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^+ \hat{Y}$, to be small. A small condition number also means that the difference between $\mathbf{\Phi}^\top \mathbf{\Phi}$ and $\hat{Y}\hat{Y}^\top$ is small. Moreover, the matrix $\mathbf{\Phi}^\top \mathbf{\Phi} \in \mathbb{R}^{n \times n}$ can be regarded as representing the pairwise similarities of new representations and $\hat{Y}\hat{Y}^\top \in \mathbb{R}^{n \times n}$ represent the pairwise similarities of the predictions.

The residuals $\varepsilon(x_i, W^{(1)}, \ldots, W^{(L)}), i \in \{1, \ldots, n\}$ are uncertain noise for generalization. Since $\hat{Y}$ contains the residual terms, large residuals imply that small values of $\sum_{i=1}^n \ell(y_i, \hat{y}_i)$ and $\lambda \operatorname{tr}(\hat{Y}^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^+ \hat{Y})$ w.r.t. the learned arguments $\{v_S, W_S^{(1)}, \ldots, W_S^{(L)}\}$ cannot generalize well on unseen data. Unsupervised pretraining which aims to make the residuals small on unseen data is therefore important for deep learning algorithms.

[4]Because every positive definite matrix is invertible and its inverse is also positive definite, and if a matrix M is positive semidefinite, for any given matrix Q, $Q \top M Q$ is also positive semidefinite.

In Section V, we prove that unsupervised pretraining is essential. Because by making the residuals small, the predictor learned by deep learning algorithms generalizes fast w.r.t. the sample size.

V. Unsupervised Pretraining Helps the Learned Predictor Generalize Fast

In this section, we show that the generalization ability of learning algorithms can be improved using unsupervised pretraining. In summary, Theorem 1 shows that when only the predictor and are tuned in fine-tuning step, the unsupervised pre-training encourages deep learning algorithms to be end-to-end stable, and therefore will generalize fast [48]; Lemma 2 shows that when both the predictor and the parameters of the pretrained neural network are tuned in the fine-tuning step, unsupervised pretraining also can encourage deep learning algorithms more stable than the random initialization seed when the change in pretrained parameters is not large during the fine-tuning step.

First, we introduce the stability framework proposed by Liu *et al.* [48], which plays a central role in proving the generalization bound with a fast convergence rate.

Definition 1 (Uniform hypothesis stability): An deep learning algorithm is $\alpha(n)$-uniformly hypothesis stable w.r.t. a specific domain $\mathcal{Z} = (\mathcal{X} \times \mathcal{Y}) \subset (\mathbb{R}^c \times \mathbb{R})$, if for any training sample $S \in \mathcal{Z}^n$, any $i \in \{1, \ldots, n\}$, and any $z = (x, y) \in \mathcal{Z}$, there exist an $\alpha(n)$, such that the following holds: $\|v_S - v_{S^i}\|_2 \leq \alpha(n)$, where v_S represents the arguments learned by the algorithm using sample S, and S^i denotes the training sample S with the ith example z_i being replaced by an i.i.d. example z_i', and $\alpha(n) \in \mathbb{R}_+$.

Intuitively, the above definition defines the stability of a deep learning algorithm by the maximum difference between two hypotheses v_S and v_{S^i} by minimizing the empirical risk on two training sets S and S^i, respectively, where S and S^i only differ in exactly one example. If the ℓ_2 difference between the two hypotheses v_S and v_{S^i} can be bounded by α which is a function on the sample size n of the training sets, then the algorithm is defined to be $\alpha(n)$-uniformly hypothesis stable.

The following Lipschitz-like definition on a loss function [49] is useful to derive upper bound for stability.

Definition 2 (L-admissible): A loss function ℓ is L-admissible with respect to the hypothesis class H if there exists a function $L \in \mathbb{R}^+$ such that for any $v, v' \in H$ and any example $z = (x, y) \in \mathcal{Z}$ such that

$$|\ell(v, z) - \ell(v', z)| \leq L |\langle v - v', x \rangle|.$$

Now we show that a linear predictor optimizing the objective in (5), namely

$$\min_{v \in \mathbb{R}^c} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle v, x_i \rangle) + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)} v \right\|_2^2 \tag{10}$$

is uniformly hypothesis stable as follows.

Theorem 1: Let the loss function ℓ be convex w.r.t. the predictor, L-admissible, and upper bounded by M, that is for any $z = (x, y) \in \mathcal{Z}$ and $v \in H$, $\ell(y, \langle v, x \rangle) \leq M$. Assume the new representations $\phi(X, W_U^{(1)}, \ldots, W_U^{(L)})$ and the residual $\varepsilon(X, W_U^{(1)}, \ldots, W_U^{(L)})$ are upper bounded by $\wedge_\phi$ and $\wedge_\varepsilon$, respectively. Algorithms that minimize the objective function in (10) are uniformly hypothesis stable with respect to the domain of training sample S. That is, for any independently distributed training sample $S \in \mathcal{Z}^n$, any $i \in \{1, \ldots, n\}$, and any $Z = (X, Y) \in \mathcal{Z}$, the following holds:

$$\left\| W_U^{(L)}, \ldots, W_U^{(1)}(\upsilon_{S,W_U} - \upsilon_{S^i,W_U}) \right\|_2 \le \frac{L\wedge_\phi}{\lambda n} + O\left(\sqrt{\frac{\wedge_\varepsilon}{n}}\right)$$

where υ_{S,W_U} denotes the predictor learned by optimizing (10) using sample S.

It has been proven that support vector machine (SVM) is uniformly hypothesis stable, e.g., [49], which implies that the learning algorithm optimizing

$$\begin{aligned}
&\min_{\upsilon\in\mathbb{R}^c} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle \upsilon', \left(W_U^{(1)}\right)^\top, \ldots, \left(W_U^{(L)}\right)^\top \right.\right.\\
&\qquad\qquad \left.\left. \times\ \phi\left(W_U^{(L)}, \ldots, \phi\left(W_U^{(1)} x_i\right)\right)\right\rangle\right)\\
&\qquad\qquad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2\\
&= \min_{\upsilon\in\mathbb{R}^c} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle W_U^{(L)}, \ldots, W_U^{(1)}\upsilon\ , \right.\right.\\
&\qquad\qquad \left.\left. \phi\left(W_U^{(L)}, \ldots, \phi\left(W_U^{(1)} x_i\right)\right)\right\rangle\right)\\
&\qquad\qquad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2\\
&= \min_{\upsilon\in\mathbb{R}^c} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle \upsilon, x_i - \varepsilon\left(x_i, W_U^{(1)}, \ldots, W_U^{(L)}\right)\right\rangle\right)\\
&\quad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2 \qquad (11)
\end{aligned}$$

is also uniformly hypothesis stable. The stability properties of algorithms that optimize objects (10) and (11) are different by that the algorithm optimizing object (11) is uniformly hypothesis stable w.r.t. the size of newly represented examples; while the algorithm optimizing object (10) is uniformly hypothesis stable w.r.t. the size of the input examples. By comparison, we can conclude that unsupervised pretraining encourages deep learning algorithms to be end-to-end stable by minimizing the residual $\varepsilon(X, W_U^{(1)}, \ldots, W_U^{(L)})$.

Liu *et al.* [48] have proved that uniformly hypothesis stable algorithms will generalize fast. Using the same proof method, it is easy to show that if the residuals are zero, the predictor learned by optimizing the object in (10) will generalize fast w.r.t. the domain of the input examples. This partially explains why unsupervised pretraining is helpful to improve the test performance of deep learning algorithms.

For deep learning algorithms when both the predictor and the parameters of the pretrained neural network are tuned in the fine-tuning step, it is reasonable to assume that the unsupervised pretraining step constraints the fine-tuning step to pick up the parameter $W^{(l)}$ from a closed ball $\mathcal{B}(W_U^{(l)}, r)$, $l \in \{1, \ldots, L\}$, where r is not large. Let $\{\upsilon_S, W_S^{(1)}, \ldots, W_S^{(L)}\}$ and $\{\upsilon_{S^i}, W_{S^i}^{(1)}, \ldots, W_{S^i}^{(L)}\}$ be the arguments learned by algorithm (9) using samples S and S^i, respectively. Let us write $W_S^{(L)}, \ldots, W_S^{(1)} = W_U^{(L)}, \ldots, W_U^{(1)} + \Delta W_S$ and $W_{S^i}^{(L)}, \ldots, W_{S^i}^{(1)} = W_U^{(L)}, \ldots, W_U^{(1)} + \Delta W_{S^i}$. Then, $\|\Delta W_S\| \le r$ and $\|\Delta W_{S^i}\|_2 \le r$. If we further assume that the upper bound $\wedge_\varepsilon$ of the residual $\|\varepsilon(x, W^{(1)}, \ldots, W^{(L)})\|_2$ is also small, we could expect that learning algorithms optimizing the object in (9) are some kind of stable, so unsupervised pretraining will make its upper bound small.

Lemma 2: Let the loss function ℓ be convex w.r.t. the predictor, L-admissible, and upper bounded by M. For any independently distributed training sample $S \in \mathcal{Z}^n$, any $i \in \{1, \ldots, n\}$, and any $z = (x, y) \in \mathcal{Z}$, the following holds:

$$\begin{aligned}
&\left\| \left(W_U^{(L)}, \ldots, W_U^{(1)}\right)(\upsilon_S - \upsilon_{S^i}) \right\|_2\\
&\qquad \le \frac{L\wedge_\phi}{\lambda n} + \sqrt{O(\wedge_\varepsilon) + O(r) + O(r^2)}
\end{aligned}$$

where υ_S is the output of the learning algorithm that minimizes the objective function in (9) using the training sample S.

The proof method of Lemma 2 is the same as that of Theorem 1. We know that unsupervised pretraining makes $\wedge_\varepsilon$ and ΔW (or r) small. If the two terms approach zero, the upper bound in Lemma 2 will degenerate to that in Theorem 1. Thus, Lemma 2 shows that unsupervised pretraining makes deep learning algorithms more stable than the random initialization seed. Unsupervised pretraining therefore regularizes the predictor and helps it generalize fast (according to the results in [48]). This is in accordance with the empirical findings of Bengio *et al.* [11] that the unsupervised training strategy helps optimization by initializing weights in a region near a good local minimum that brings better generalization than without using cosine similarity minimized (CSM) regularizer.

VI. Proof of Things

A. Proof of Lemma 1

Let $\upsilon' = W_U^{(L)}, \ldots, W_U^{(1)}\upsilon$, the objective in (4) becomes

$$\begin{aligned}
\upsilon_{S,W_U} &= \arg\min_{\upsilon\in\mathbb{R}^c} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle W_U^{(L)}, \ldots, W_U^{(1)}\upsilon, \phi\left(W_U^{(L)}, \ldots, \right.\right.\right.\\
&\qquad\qquad \left.\left.\left. \phi\left(W_U^{(1)} x_i\right)\right)\right\rangle\right)\\
&\qquad\qquad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2\\
&= \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle \upsilon, \left(W_U^{(1)}\right)^\top, \ldots, \left(W_U^{(L)}\right)^\top \right.\right.\\
&\qquad\qquad \left.\left. \times\ \phi\left(W_U^{(L)}, \ldots, \phi\left(W_U^{(1)} x_i\right)\right)\right\rangle\right)\\
&\qquad\qquad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2\\
&= \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \left\langle \upsilon, x_i - \varepsilon\left(x_i, W_U^{(1)}, \ldots, W_U^{(L)}\right)\right\rangle\right)\\
&\quad + \lambda \left\| W_U^{(L)}, \ldots, W_U^{(1)}\upsilon \right\|_2^2 \qquad (12)
\end{aligned}$$

where $\varepsilon\left(x_i, W_U^{(1)}, \ldots, W_U^{(L)}\right)$ is the reconstruction error of the neural network pretrained by the auto-encoder and is defined by

$$\begin{aligned}
&\varepsilon\left(x_i, W_U^{(1)}, \ldots, W_U^{(L)}\right)\\
&\quad = x_i - \left(W_U^{(1)}\right)^\top, \ldots, \left(W_U^{(L)}\right)^\top \phi\left(W_U^{(L)}, \ldots, \phi\left(W_U^{(1)} x_i\right)\right). \qquad (13)
\end{aligned}$$

We have assumed that $X_{j,p}$, $j \in \{1,\ldots,m\}$ and x_i, $i \in \{1,\ldots,n\}$ have the same marginal distribution. Thus, unsupervised pretraining encourages the residual $\varepsilon(x_i, W_U^{(1)},\ldots,W_U^{(L)})$ to be zero and thus encourages

$$v_{S,W_U} = \arg\min_{v\in\mathbb{R}^c} \frac{1}{n}\sum_{i=1}^{n}\ell(y_i,\langle v,x_i\rangle) + \lambda\left\| W_U^{(L)},\ldots,W_U^{(1)}v\right\|_2^2. \tag{14}$$

The proof ends. ■

B. Proof of Theorem 1

First, we introduce Bregman divergence [50], which is helpful to upper bound the uniform hypothesis stability.

Definition 3 (Bregman Divergence): Let $f:\mathcal{X}\to\mathbb{R}$ be a convex function. For all $s,t\in\mathcal{X}$, we have

$$B_f(s\|t) = f(s) - f(t) - \langle s-t, \nabla f(t)\rangle$$

where $\nabla f(t)$ denotes the gradient of function $f(t)$.

Bregman divergence has the following properties. Detailed discussions can be found in [50].

Lemma 3: Bregman divergence is additive and nonnegative. If $f = f_1 + f_2$, and both f_1 and f_2 are convex, for any $s,t\in\mathcal{X}$, we have

$$B_f(s\|t) = B_{f_1}(s\|t) + B_{f_2}(s\|t) \quad \text{and } B_f(s\|t)\geq 0.$$

Now, we are ready to prove Theorem 1. To prove the algorithms that optimize (5), namely

$$v_{S,W_U} = \arg\min_{v\in\mathbb{R}^D} \frac{1}{n}\sum_{i=1}^{n}\ell(y_i,\langle v,x_i\rangle) + \lambda\left\| W_U^{(L)},\ldots,W_U^{(1)}v\right\|_2^2$$

is uniformly stable, let

$$f_{e,S}(v) = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i,\langle v,x_i\rangle)$$
$$f_r(v) = \lambda\left\| W_U^{(L)},\ldots,W_U^{(1)}v\right\|_2^2$$

and

$$f_S(v) = f_{e,S}(v) + f_r(v).$$

Let v_{S,W_U} be the solution to optimization problem (5) when the input training sample is S, and v_{S^i,W_U} be the solution when the input training sample is S^i. Using Bregman divergence, we have

$$\begin{aligned} &B_{f_S}\left(v_{S^i,W_U}\|v_{S,W_U}\right) + B_{f_{S^i}}\left(v_{S,W_U}\|v_{S^i,W_U}\right) \\ &\quad\geq B_{f_r}\left(v_{S^i,W_U}\|v_{S,W_U}\right) + B_{f_r}\left(v_{S,W_U}\|v_{S^i,W_U}\right). \end{aligned} \tag{15}$$

The right-hand side of inequality (15) can be written as

$$\begin{aligned} &B_{f_r}\left(v_{S^i,W_U}\|v_{S,W_U}\right) + B_{f_r}\left(v_{S,W_U}\|v_{S^i,W_U}\right) \\ &= -\Big\langle v_{S^i,W_U} - v_{S,W_U}, \\ &\qquad \times 2\lambda\left(W_U^{(L)},\ldots,W_U^{(1)}\right)^\top W_U^{(L)},\ldots,W_U^{(1)}v_{S,W_U}\Big\rangle \\ &\quad -\Big\langle v_{S,W_U} - v_{S^i,W_U}, \\ &\qquad \times 2\lambda\left(W_U^{(L)},\ldots,W_U^{(1)}\right)^\top W_U^{(L)},\ldots,W_U^{(1)}v_{S^i,W_U}\Big\rangle \\ &= 2\lambda\left\| W_U^{(L)},\ldots,W_U^{(1)}(v_{S,W_U} - v_{S^i,W_U})\right\|_2^2. \end{aligned} \tag{16}$$

The left-hand side of inequality (15) can be upper bounded as

$$\begin{aligned} &B_{f_S}\left(v_{S^i,W_U}\|v_{S,W_U}\right) + B_{f_{S^i}}\left(v_{S,W_U}\|v_{S^i,W_U}\right) \\ &= f_S(v_{S^i,W_U}) - f_S(v_{S,W_U}) \\ &\quad -\left\langle v_{S^i,W_U} - v_{S,W_U}, \nabla f_S(v_{S,W_U})\right\rangle \\ &\quad + f_{S^i}(v_{S,W_U}) - f_{S^i}(v_{S^i,W_U}) \\ &\quad -\left\langle v_{S,W_U} - v_{S^i,W_U}, \nabla f_{S^i}(v_{S^i,W_U})\right\rangle \\ &= f_S(v_{S^i,W_U}) - f_S(v_{S,W_U}) + f_{S^i}(v_{S,W_U}) - f_{S^i}(v_{S^i,W_U}) \\ &= \frac{1}{n}\Big(\ell\left(y_i,\langle v_{S^i,W_U},x_i\rangle\right) - \ell\left(y_i,\langle v_{S,W_U},x_i\rangle\right) \\ &\quad + \ell\left(y'_i,\langle v_{S,W_U},x'_i\rangle\right) - \ell\left(y'_i,\langle v_{S^i,W_U},x'_i\rangle\right)\Big) \\ &\leq \frac{L}{n}\left(\left|\langle v_{S,W_U} - v_{S^i,W_U},x_i\rangle\right| + \left|\langle v_{S,W_U} - v_{S^i,W_U},x'_i\rangle\right|\right) \\ &= \frac{L}{n}\Bigg(\Bigg|\Big\langle v_{S,W_U} - v_{S^i,W_U}, \left(W_U^{(1)}\right)^\top,\ldots,\left(W_U^{(L)}\right)^\top \\ &\qquad \times \phi\left(W_U^{(L)},\ldots,\phi\left(W_U^{(1)}x_i\right)\right) \\ &\qquad + \varepsilon\left(x_i, W_U^{(1)},\ldots,W_U^{(L)}\right)\Big\rangle\Bigg| \\ &\qquad + \Bigg|\Big\langle v_{S,W_U} - v_{S^i,W_U}, \left(W_U^{(1)}\right)^\top,\ldots,\left(W_U^{(L)}\right)^\top \\ &\qquad \times \phi\left(W_U^{(L)},\ldots,\phi\left(W_U^{(1)}x'_i\right)\right) \\ &\qquad + \varepsilon\left(x'_i, W_U^{(1)},\ldots,W_U^{(L)}\right)\Big\rangle\Bigg|\Bigg) \\ &\leq \frac{2L\wedge_\phi\left\| W_U^{(L)},\ldots,W_U^{(1)}(v_{S,W_U} - v_{S^i,W_U})\right\|_2 + O(\wedge_\varepsilon)}{n} \end{aligned} \tag{17}$$

where the second equality holds since v_{S^i,W_U} and v_{S,W_U} are optimal solutions, and we have $\nabla f_S(v_{S,W_U}) = \nabla f_{S^i}(v_{S^i,W_U}) = 0$; the first inequality holds since the loss function is L-admissible. Combining (15)–(17), we get

$$\left\| W_U^{(L)},\ldots,W_U^{(1)}(v_{S,W_U} - v_{S^i,W_U})\right\|_2 \leq \frac{L\wedge_\phi}{\lambda n} + O\left(\sqrt{\frac{\triangle_\varepsilon}{n}}\right)$$

which completes the proof. ■

VII. Experiments

To demonstrate the regularization property of unsupervised pretraining, (9) motivates us to explore a regularizer similar to $\mathrm{tr}(\hat{Y}\hat{Y}^\top(\mathbf{\Phi}^\top\mathbf{\Phi})^+)$ as follows:

$$\lambda\|YY^\top - \mathbf{\Phi}^\top\mathbf{\Phi}\|_2^2 \tag{18}$$

where $\lambda \geq 0$ is the weight of the regularizer. The proposed regularizer (18) encourages the representations of the examples with the same label to be similar by punishing the difference between $\mathbf{\Phi}^\top\mathbf{\Phi}$ and $YY^\top$. Specifically, according to (8), if the true label y is well-learned by a learning algorithm, the residual $\varepsilon(X)$ will be small, then $\mathrm{tr}(\hat{Y}\hat{Y}^\top(\mathbf{\Phi}^\top\mathbf{\Phi})^+)$ can be approximated by $\mathrm{tr}(YY^T(\mathbf{\Phi}^\top\mathbf{\Phi})^+)$ which punishes the difference between $\mathbf{\Phi}^\top\mathbf{\Phi}$ and $YY^\top$. A good property of the proposed regularizer (18) is that the inverse matrix operation

TABLE I
SUMMARY OF DATASETS USED IN THE EXPERIMENTS

	# of training	# of validation	# of testing	# of class
F-MNIST	48,000	12,000	10,000	10
CIFAR-10	40,000	10,000	10,000	10
CIFAR-100	40,000	10,000	10,000	100

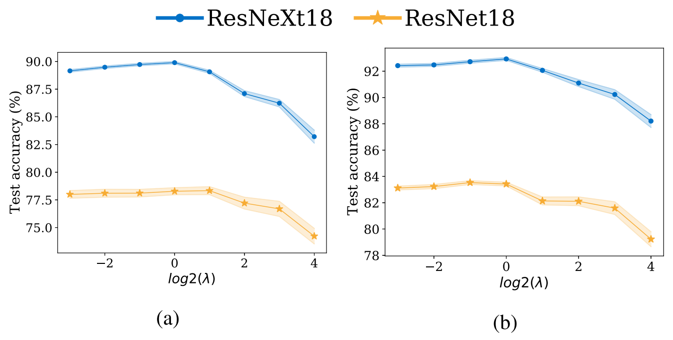


Fig. 1. Average and standard deviation (shaded area) of test classification accuracy with different choices of hyperparameter λ on CIFAR10. (a) Models are trained with 50% training data. (b) Models are trained with 100% training data.

can be avoided, which is computationally more efficient. Additionally, for any pair of instances, the proposed regularizer minimizes the difference between the cosine similarity of their latent representations and the cosine similarity of their labels. Motivated by this property, we name the proposed regularizer "CSM regularizer."

We then evaluate the effectiveness of the proposed regularizer (18) on three popular image classification datasets, CIFAR10, CIFAR100 [6], and Fashion-MNIST (or F-MNIST) [51]. Because the datasets originally do not provide validation set, therefore, for all the experiments, we leave out 20% examples from the original training set to be our validation set, and the rest of the 80% examples become our new training set. The sizes of the training set, validation set, and test set are summarized in Table I. Two different popular neural network structures are used in our experiments, which are 18-layer-ResNet (ResNet18) [52] and 18-layer preactivation ResNet (ResNeXt18) [52]. Stochastic gradient descent (SGD) optimizer is used with the batch size 128. The initial learning rate is 0.1 and is decayed by a factor 10 for every 100 epochs. The network is trained for 350 epochs in total. To clearly illustrate the effect of the proposed regularizer, we do not use the weight decay.[5]

The rest of this section is organized as follows. In Section VII-A, we investigate the selection of the weight parameter of CSM regularizer. In Section VII-B, we compare the test accuracy of ResNet18 and ResNeXt18 when CSM regularizer is deployed or not on different datasets and sample sizes. In Section VII-D, we visualize the latent representations of ResNet18 and ResNeXt18 when CSM regularizer is deployed or not.

[5]The code will be published on GitHub upon acceptance.

A. Hyperparameter Selection

In this section, we investigate the selection of the hyperparameter λ. Specifically, we compare the test classification accuracy under different choices of $\lambda = \{0.0125, 0.025, 0.5, 1, 2, 4, 8, 16\}$. The experiments are conducted on both ResNet18 and ResNeXt18. The classification accuracy and the standard deviation generated by investigated methods under different λ are illustrated in Fig. 1. The curves in the figures represent the average classification accuracy over five repeated trials. The shaded area around the accuracy curve indicates the standard deviation of classification accuracy over the five repeated trials. The figures show that for different sample sizes and different neural network structures, the average classification accuracy keeps increasing when the value of λ increases from 0 to 1, and the average classification accuracy keeps decreasing when the value of λ increases from 1 to 16. By observing the shaded area, the standard deviation of classification accuracy does not change significantly when the value of λ is in the range of $[0, 1]$, and the standard deviation of classification accuracy increases when the value of λ increases from 1 to 16. Overall, for both neural network structures and different sample sizes, the best average classification accuracy is achieved at $\lambda = 1$, and the standard deviation is also small at $\lambda = 1$. This means that for different random experiments, setting $\lambda = 1$ is most likely to produce the optimal results. Therefore, we select the hyperparameter $\lambda = 1$ for the following experiments.

B. Classification Accuracy on Clean Data

To verify that the proposed regularizer helps improve the classification accuracy, we evaluate the classification accuracy of ResNet18 and ResNeXt18 on CIFAR10, and F-MNIST with or without adding our regularizer. Different training sample sizes, i.e., 50% and 100% of the original examples in the datasets, are used to train both ResNet18 and ResNeXt18. Additionally, to investigate how the depth of networks affects the regularization effect, we also use ResNets and ResNeXts with different hidden layers on CIFAR100. All the experiments are independently implemented for five trials.

In Figs. 2–4, we illustrate both the training and test accuracy with the increase in training epoch on CIFAR10, CIFAR100, and Fashion-MNIST (or F-MNIST) [51]. The red lines indicate the average accuracy of the neural networks with using CSM regularizer over repeated experiments, and the blue line indicates the average accuracy of the neural networks without using CSM regularizer. By observing the average test accuracy with the increase in training epoch in these figures, we conclude that for most of the experiments, the red lines are continuously aligned above the blue lines. This means that using CSM regularizer, the test accuracy consistently outperforms baselines with the same amount of training epoch. This illustrates the effectiveness of the proposed regularizer. By observing the average training accuracy with the increase in training epoch in these figures, we can observe that for all the experiments, the training accuracy is similar at the same training epoch when CSM regularizer is deployed or not. This reflects that our regularizer has an encouraging generalization

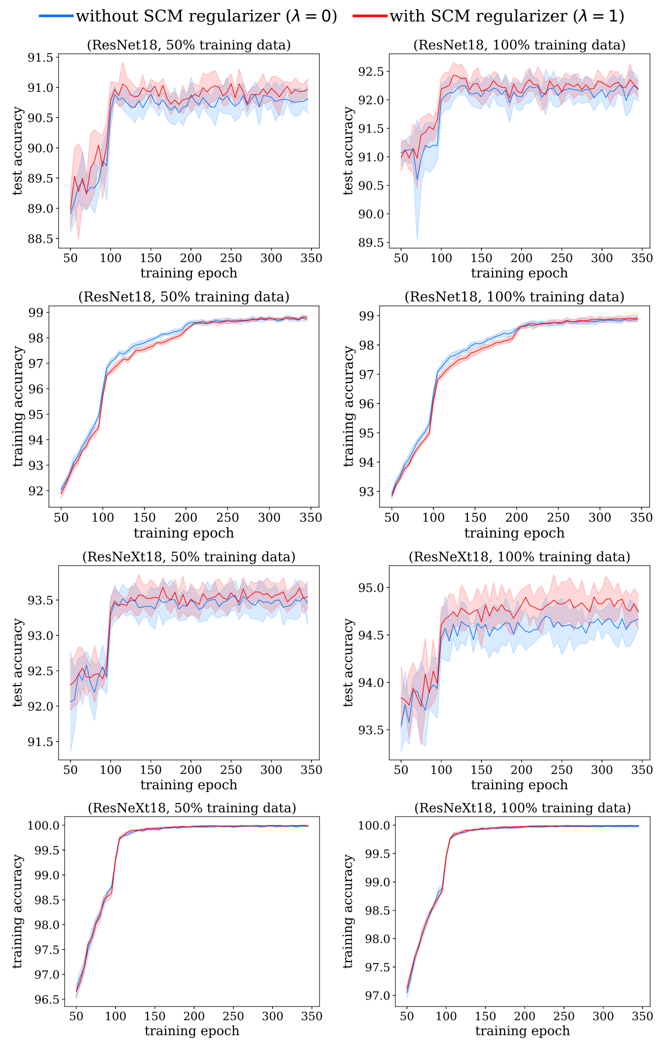


Fig. 2. Average and standard deviation (shaded area) of the training and test accuracy with the increase in training epoch on F-MNIST.

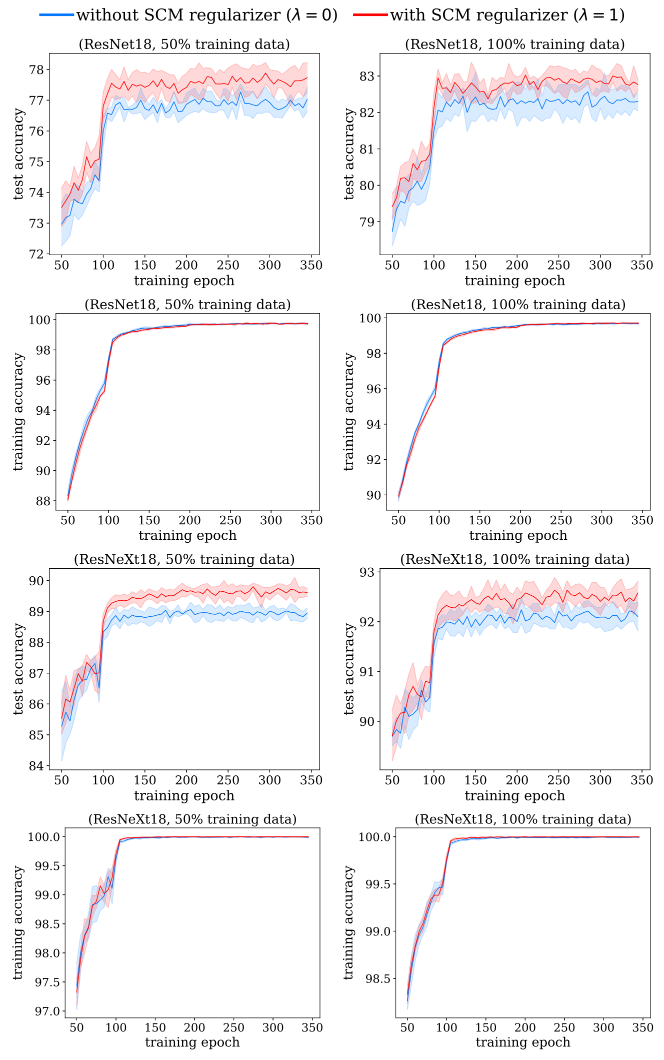


Fig. 3. Average and standard deviation (shaded area) of the training and test accuracy with the increase in training epoch on CIFAR10.

TABLE II
AVERAGE TEST ACCURACY ± STANDARD DEVIATION OVER 350 TRAINING EPOCHS ON CIFAR10 AND F-MNIST WITH 50% TRAINING DATA

Dataset	ResNet18 ($\lambda = 0$)	ResNet18 ($\lambda = 1$)	ResNeXt18 ($\lambda = 0$)	ResNeXt18 ($\lambda = 1$)
F-MNIST	90.944% ±0.221%	**91.152**% ±0.258%	91.532% ±0.102%	**92.201**% ±0.221%
CIFAR10	77.862% ±0.325%	**78.28**% ±0.313%	89.120% ±0.096%	**89.89**% ±0.161%

TABLE III
AVERAGE TEST ACCURACY ± STANDARD DEVIATION OVER 350 TRAINING EPOCHS ON CIFAR10 AND F-MNIST WITH 100% TRAINING DATA

Dataset	ResNet18 ($\lambda = 0$)	ResNet18 ($\lambda = 1$)	ResNeXt18 ($\lambda = 0$)	ResNeXt18 ($\lambda = 1$)
F-MNIST	92.26% ±0.085%	**92.568**% ±0.127%	94.764% ±0.115%	**94.01**% ±0.201%
CIFAR10	83.042% ±0.253%	**83.422**% ±0.196%	92.438% ±0.133%	**92.934**% ±0.105%

property with respect to the sample size. That is to say, with the same training accuracy, the test accuracy is higher than the baseline.

We report the average test accuracy and the standard deviation obtained by the models with the best validation accuracy in Tables II and III. Both the tables show that using CSM regularizer (18), the classification accuracy for different network structures is improved, It indicates that the proposed CSM regularizer can be integrated into different neural network structures to improve their accuracy. The tables also show that SCM regularizer helps improve the classification accuracy with not only the small training sample size but also the large training sample size. The improvement of the average accuracy further validates that our regularizer helps the learned predictor generalize fast with respect to the sample size.

In Tables IV and V, we report the average test accuracy and the standard deviation using ResNet and ResNeXt with 18, 34, and 50 layers on CIFAR100. For most of the experiments, our regularizer improves the classification accuracy. However, the generalization ability of ResNeXt is much better than ResNet on CIAR100 which has much smaller sample size per class, when compared with CIFAR10 and F-MNIST. It is also worth

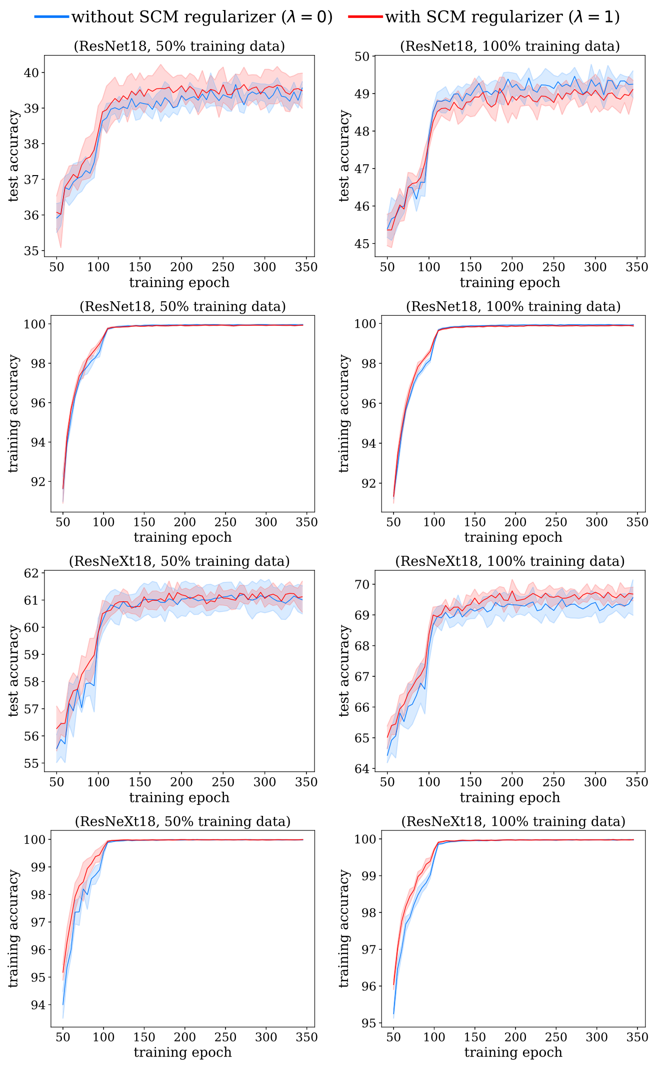


Fig. 4. Average and standard deviation (shaded area) of the training and test accuracy with the increase in training epoch on CIFAR100.

TABLE IV

Average Test Accuracy ± Standard Deviation Over 350 Training Epochs on CIFAR100 Using ResNet With Different Depths and Different Training Sample Sizes

# layers	50% training data ($\lambda = 0$)	50% training data ($\lambda = 1$)	100% training data ($\lambda = 0$)	100% training data ($\lambda = 1$)
18	40.658% ±0.270%	**41.050%** ±0.294%	**50.114%** ±0.348%	49.690% ±0.204%
34	40.12% ±0.35%	**40.96%** ±0.52%	49.78% ±0.78%	**50.21%** ±0.37%
50	31.18% ±0.47%	**31.90%** ±0.25%	40.79% ±0.63%	**41.72%** ±0.33%

to mention that for ResNeXt, the number of layers does not influence the classification accuracy too much. In contrast, the classification accuracy of ResNet drops with the increase in the number of layers, especially the accuracy, is decreased by approximately 10% from 34 to 50 layers. The results show that ResNeXt is earlier to be optimized compared with ResNet when the number of layers of network is large.

TABLE V

Average Test Accuracy ± Standard Deviation Over 350 Training Epochs on CIFAR100 Using ResNeXt With Different Depths and Different Training Sample Sizes

# layers	50% training data ($\lambda = 0$)	50% training data ($\lambda = 1$)	100% training data ($\lambda = 0$)	100% training data ($\lambda = 1$)
18	61.868% ±0.363%	**61.888%** ±0.205%	69.932% ±0.235%	**70.480%** ±0.199%
34	62.46% ±0.67%	**63.67%** ±0.66%	70.86% ±0.27%	**71.66%** ±0.12%
50	62.08% ±0.58%	**63.11%** ±0.56%	71.24% ±0.60%	**72.27%** ±0.53%

TABLE VI

Average Test Accuracy ± Standard Deviation of ResNet Over 350 Training Epochs on CIFAR10 and CIFAR100 Containing 50% Symmetry Noise and 45% Pair-Flip Noise

	$\lambda = 0$		$\lambda = 1$	
Dataset	sym-50%	pair-45%	sym-50%	pair-45%
CIFAR10	55.92% ±0.44%	58.37% ±2.66%	**59.34%** ±0.26%	**63.20%** ±1.53%
CIFAR100	22.65% ±0.37%	21.62% ±0.58%	**27.19%** ±0.43%	**30.08%** ±0.70%

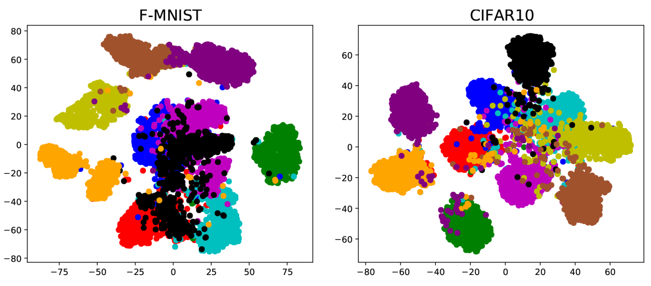
C. Classification Accuracy on Noisy Data

To further validate the regularization property, we use CSM regularizer on the data containing label noise [53], [54]. Because labels $\tilde{Y}$ in training sample contains noise and cannot be fully trusted, we implicitly reduce the importance of noisy labels using predicted labels of models in our regularizer, i.e., $\lambda\|\hat{Y}\hat{Y}^\top - \mathbf{\Phi}^\top\mathbf{\Phi}\|_2^2$.

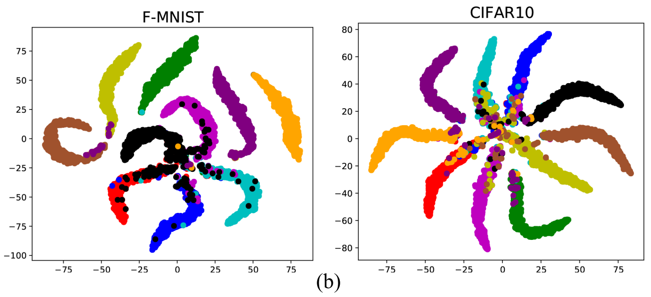
Two types of popular label noise [55], [56] are used in experiments, which are 50% symmetry noise [54] and 45% pair-flip noise [55]. To generate noisy datasets, we manually corrupt the training and validation sets of each dataset according to noise types. We use 18-layer and 34-layer ResNets for CIFAR10 and CIFAR100, respectively. The results are summarized in Table VI, which shows that using SCM regularizer, the classification accuracy on both CIFAR10 and CIFAR100 has been improved in a large margin. This further validates that our method helps improve the robustness of deep learning models and can prevent outfitting.

D. Latent Representations Visualization on Clean Data

To investigate the influence of latent representations using our regularizer, we visualize the latent representations of the test data using t-distributed stochastic neighbor embedding (t-SNE) visualization [57]. The t-SNE algorithm is a popular nonlinear technique for dimensionality reduction and is well-suited for visualization of high-dimensional datasets, which is commonly used in many literatures [58]–[64]. Intuitively, it calculates the similarities among instances in the high-dimensional space and the similarities among instances in the corresponding low-dimensional space, and it iteratively minimizes the difference between these similarities in higher dimensional and lower dimensional spaces to find meaningful representations of instances in lower dimensional space. The

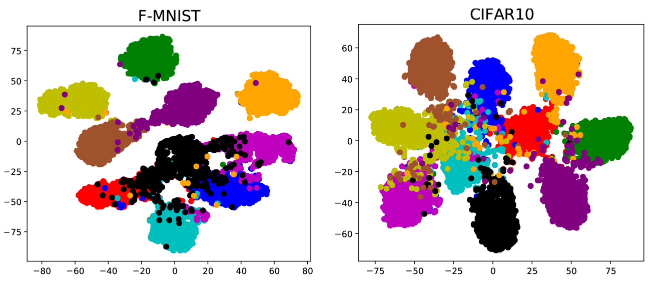


(a)

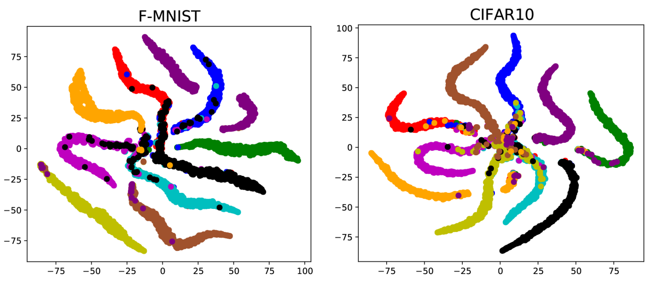


(b)

Fig. 5. Latent representation of ResNet18 with or without using SCM Regularizer on F-MNIST and CIFAR10. (a) Without SCM regularizer ($\lambda = 0$). (b) With SCM regularizer ($\lambda = 1$).



(a)



(b)

Fig. 6. Latent representation of ResNeXt18 with or without using SCM Regularizer on F-MNIST and CIFAR10. (a) Without SCM regularizer ($\lambda = 0$). (b) With SCM regularizer ($\lambda = 1$).

distance between a pair of instances indicates the similarity of their latent representations. The closer the distance, the more similar the latent representations.

In Figs. 5 and 6, we illustrate the latent representations on test set learned by neutral networks on CIFAR10 and CIFAR100. Both ResNet18 and ResNeXt18 are used for visualization. The models with the best validation accuracy are used for visualization. Each point in the figures represents the latent representation of an instance in 2-D space. Different colors represent different labels. The visualization shows that CSM regularizer helps learn meaningful data representations.

Both the figures show that without using cosine similarity minimized (CSM) regularizer, the classification models prefer to learn ball-shaped clusters. However, the latent representations for instances in different classes are not well-separated, i.e., the interclass distances are small, and the boundary between the classes is not clear. This means that the latent representations of the instances in different classes are close to each other, which obviously has a negative effect on the classification accuracy of the classification models.

The proposed regularizer helps the classification models learn line-shaped clusters which are better than ball-shaped clusters. Because the interclass distances for both the neural network structures are significantly increased, there exists clear boundaries between different classes. It is worth to mention that there still exist some instances with different classes aligned closely in line-shaped clusters. The reason may be that these instances are similar and have some common features, and it is hard for the classification models to completely eliminate all the common features.

VIII. Conclusion

In this article, we analyze why unsupervised pretraining regularizes deep learning algorithms by exploiting its reconstruction property. Specifically, we provide a theoretical justification for observations that unsupervised pretraining works as regularization and helps predictors learned by deep learning algorithms generalize fast. We have interpreted deep learning algorithms as the traditional batch learning algorithms with Tikhonov regularizations that simultaneously learn predictors in the input feature space and the parameters of neural networks to form the Tikhonov matrices. Tikhonov regularization also encourages the neural networks to learn similar representations for examples with the same label.

References

[1] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Feb. 2010.
[2] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," 2015, *arXiv:1505.03654*.
[3] J. W. Siegel and J. Xu, "On the approximation properties of neural networks," 2019, *arXiv:1904.02311*.
[4] U. Shaham, A. Cloninger, and R. R. Coifman, "Provable approximation properties for deep neural networks," *Appl. Comput. Harmon. Anal.*, vol. 44, pp. 537–557, May 2018.
[5] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, "Deep neural network approximation theory," 2019, *arXiv:1901.02220*.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[7] H. Maennel *et al.*, "What do neural networks learn when trained with random labels?" 2020, *arXiv:2006.10455*.
[8] P. Kaul and B. Lall, "Riemannian curvature of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1410–1416, Apr. 2020.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 153–160.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] D. Chen, J. Lv, and Z. Yi, "Graph regularized restricted Boltzmann machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2651–2659, Jun. 2017.

[14] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[15] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[16] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[17] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[18] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, "Unsupervised pre-training of image features on non-curated data," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 2959–2968.

[19] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional sparse autoencoders for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3289–3294, Jul. 2017.

[20] X. Li, H. Zhang, R. Zhang, Y. Liu, and F. Nie, "Generalized uncorrelated regression with adaptive graph for unsupervised feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1587–1595, May 2018.

[21] Z. Dai, B. Cai, Y. Lin, and J. Chen, "UP-DETR: Unsupervised pre-training for object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 1601–1610.

[22] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "Wav2vec: Unsupervised pre-training for speech recognition," 2019, *arXiv:1904.05862*.

[23] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Point-contrast: Unsupervised pre-training for 3D point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 574–591.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[25] L. Jiang, R. Li, W. Wu, C. Qian, and C. C. Loy, "DeeperForensics-1.0: A large-scale dataset for real-world face forgery detection," 2020, *arXiv:2001.03024*.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1–8.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," 2015, *arXiv:1502.01852*.

[29] K. D. Humbird, J. L. Peterson, and R. G. McClarren, "Deep neural network initialization with decision trees," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1286–1295, May 2018.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang, "An analysis of unsupervised pre-training in light of recent advances," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–10.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[33] P. Ge, C.-X. Ren, D.-Q. Dai, J. Feng, and S. Yan, "Dual adversarial autoencoders for clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1417–1424, Apr. 2019.

[34] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 473–480.

[35] Y. Bengio and O. Delalleau, "Justifying and generalizing contrastive divergence," *Neural Comput.*, vol. 21, no. 6, pp. 1601–1621, May 2009.

[36] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1345–1352.

[37] X. Zhu, J. Lafferty, and Z. Ghahramani, "Semi-supervised learning: From Gaussian fields to Gaussian processes," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2003.

[38] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.

[39] H. Liu, F. Shang, S. Yang, M. Gong, T. Zhu, and L. Jiao, "Sparse manifold-regularized neural networks for polarimetric SAR terrain classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3007–3016, Aug. 2020.

[40] X. Chen, J. Weng, W. Lu, J. Xu, and J. Weng, "Deep manifold learning combined with convolutional neural networks for action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 3938–3952, Sep. 2017.

[41] J. D. Storey and R. Tibshirani, "Statistical significance for genomewide studies," *Proc. Nat. Acad. Sci. USA*, vol. 100, no. 6, pp. 9440–9445, 2003.

[42] N. D. Lawrence, "A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 1609–1638, Jan. 2012.

[43] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*. Hoboken, NJ, USA: Wiley, 2009.

[44] C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing—A probabilistic framework," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31, 2015.

[45] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 3257–3261.

[46] A. C. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2013, pp. 207–215.

[47] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," 2015, *arXiv:1504.00641*.

[48] T. Liu, G. Lugosi, G. Neu, and D. Tao, "Algorithmic stability and hypothesis complexity," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2159–2167.

[49] O. Bousquet and A. Elisseeff, "Stability and generalization," *J. Mach. Learn. Res.*, vol. 2, pp. 499–526, Mar. 2002.

[50] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.

[51] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[53] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 447–461, Mar. 2015.

[54] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1944–1952.

[55] B. Han *et al.*, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. NIPS*, 2018, pp. 8527–8537.

[56] X. Xia *et al.*, "Are anchor points really indispensable in label-noise learning?" in *Proc. NIPS*, 2019, pp. 6835–6846.

[57] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[58] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 11313–11320.

[59] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 322–330.

[60] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8050–8058.

[61] K. Lee and K. T. Carlberg, "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," *J. Comput. Phys.*, vol. 404, Mar. 2020, Art. no. 108973.

[62] L. Shao, D. Wu, and X. Li, "Learning deep and wide: A spectral method for learning deep networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2303–2308, Dec. 2014.

[63] T. de Bruin, K. Verbert, and R. Babuska, "Railway track circuit fault diagnosis using recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 523–533, Mar. 2016.

[64] N. Han *et al.*, "Transferable linear discriminant analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5630–5638, Dec. 2020.

Yu Yao received the B.E. degree in software engineering from the University of New South Wales, Sydney, NSW, Australia, in 2018. He is currently pursuing the Ph.D. degree in computer science with The University of Sydney (USYD), Camperdown, NSW, Australia.

His research interests include semisupervised learning, label noise learning, and causal inference.

Dr. Yao was a recipient of the USYD Research Students Excellence Prize at 2019. He served as International Joint Conference on Artificial Intelligence (IJCAI) workshop workflow chair in 2021. He was a reviewer for many Artificial Intelligence (AI) conferences.

Baosheng Yu received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2014, and the Ph.D. degree from The University of Sydney, Camperdown, NSW, Australia, in 2019.

He is currently a Research Fellow with the School of Computer Science and the Faculty of Engineering, The University of Sydney. His research interests include machine learning, computer vision, and deep learning.

Chen Gong (Member, IEEE) received the dual doctoral degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, and University of Technology Sydney (UTS), Ultimo, NSW, Australia, in 2016 and 2017, respectively.

He is currently a Full Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. He has published more than 100 technical articles at prominent journals and conferences such as *Journal of Machine Learning Research (JMLR)*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (T-PAMI), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (T-NNLS), IEEE TRANSACTIONS ON IMAGE PROCESSING (T-IP), International Conference on Machine Learning (ICML), Conference on Neural Information Processing Systems (NeurIPS), International Conference on Learning Representations (ICLR), Conference on Computer Vision and Pattern Recognition (CVPR), Association for the Advancement of Artificial Intelligence (AAAI), and International Joint Conference on Artificial Intelligence (IJCAI). His research interests mainly include machine learning, data mining, and learning-based vision problems.

Mr. Gong was a recipient of the "Excellent Doctoral Dissertation" Awarded by Shanghai Jiao Tong University (SJTU) and Chinese Association for Artificial Intelligence (CAAI), and the "Wu Wen-Jun AI Excellent Youth Scholar Award." He serves as the Reviewer for more than 30 international journals such as *Artificial Intelligence (AIJ)*, *International Journal of Computer Vision (IJCV)*, *Journal of Machine Learning Research (JMLR)*, and IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (T-PAMI), and also the SPC/PC member of several top-tier conferences such as ICML, NeurIPS, ICLR, CVPR, International Conference on Computer Vision (ICCV), AAAI, IJCAI, and ICDM. He was enrolled by the "Young Elite Scientists Sponsorship Program" of Jiangsu Province and China Association for Science and Technology.

Tongliang Liu (Senior Member, IEEE) is currently a Lecturer with the School of Computer Science, The University of Sydney, Camperdown, NSW, Australia. He is heading the Trustworthy Machine Learning Laboratory and is also a Visiting Scientist with RIKEN AIP, Wako, Japan. He has authored or coauthored more than 100 research articles including International Conference on Machine Learning (ICML), Conference on Neural Information Processing Systems (NeurIPS), International Conference on Learning Representations (ICLR), Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Computer Vision (ICCV), European Conference on Computer Vision (ECCV), Association for the Advancement of Artificial Intelligence (AAAI), International Joint Conference on Artificial Intelligence (IJCAI), KDD, Integrated Computational Materials Engineering (ICME), IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (T-PAMI), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (T-NNLS), and IEEE TRANSACTIONS ON IMAGE PROCESSING (T-IP), with best paper awards, e.g., the 2019 ICME Best Paper Award and the PacificVis 2021 Best VisNotes Paper Award. He is/was a (Senior-) Meta Reviewer for many conferences, such as NeurIPS, ICLR, Conference on Uncertainty in Artificial Intelligence (UAI), AAAI, and IJCAI. He is broadly interested in the fields of trustworthy machine learning and its interdisciplinary applications, with a particular emphasis on learning with noisy labels, adversarial learning, transfer learning, unsupervised learning, and statistical deep learning theory.

Mr. Liu was a recipient of the Discovery Early Career Researcher Award (DECRA) from Australian Research Council (ARC); the Cardiovascular Initiative Catalyst Award by the Cardiovascular Initiative; and was named in the Early Achievers Leaderboard of Engineering and Computer Science by the Australian in 2020.