Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

A comprehensive survey on regularization strategies in machine learning

Yingjie Tian [a,c,d,*], Yuqi Zhang [b,c,d]

[a] School of Economics and Management, University of the Chinese Academy of Sciences, Beijing 100190, China
[b] School of Mathematical Sciences, University of the Chinese Academy of Sciences, Beijing 100049, China
[c] Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, Beijing 100190, China
[d] Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Keywords:
Overfitting
Generalization
Regularization
Machine learning

ABSTRACT

In machine learning, the model is not as complicated as possible. Good generalization ability means that the model not only performs well on the training data set, but also can make good prediction on new data. Regularization imposes a penalty on model's complexity or smoothness, allowing for good generalization to unseen data even when training on a finite training set or with an inadequate iteration. Deep learning has developed rapidly in recent years. Then the regularization has a broader definition: regularization is a technology aimed at improving the generalization ability of a model. This paper gave a comprehensive study and a state-of-the-art review of the regularization strategies in machine learning. Then the characteristics and comparisons of regularizations were presented. In addition, it discussed how to choose a regularization for the specific task. For specific tasks, it is necessary for regularization technology to have good mathematical characteristics. Meanwhile, new regularization techniques can be constructed by extending and combining existing regularization techniques. Finally, it concluded current opportunities and challenges of regularization technologies, as well as many open concerns and research trends.

1. Introduction

Training a large-scale model is a challenging problem. A model might be selected by maximizing its performance on some set of training data. Its generalization might be determined by its ability to perform well on unseen data. As an example in Fig. 1, the data is fitted into a linear function and a polynomial function. Although the polynomial function fits the data more perfectly, while the linear function may make better predictions for unseen data. The concept of overfitting is that the loss decreases on training data, but the loss increases on unseen data.

Overfitting is a fundamental issue in supervised machine learning because of the presence of data noise, the limited size of the training set, and the complexity of classifiers. Regularization is a key component of machine learning [1], allowing for good generalization to unseen data even when training on a finite training set or with an inadequate iteration.

Regularization adds some constraints to the minimized objective function, which cannot be obtained from the data and represents the prior preference. The first prior preference originated in Occam's razor principle which holds that it is good to explain the phenomenon with the simplest hypothesis. In the early optimization literatures, 'regularization' is the penalty term. In the optimization problem, many models

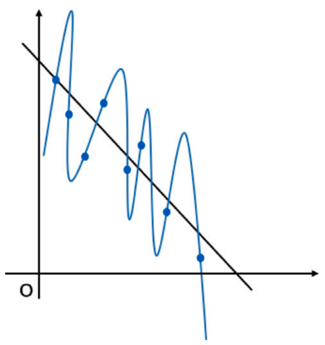


Fig. 1. An example of overfitting.

that can fit data but some models overfit data. The penalty terms usually limit the complexity the models to avoid overfitting. Fig. 2 shows an overfitted model and a regularized model. The green line works best with training data, but it is overly reliant on it and is likely to have a higher error rate with new, unseen data. The black line represents a regularized model, in which the regularization term limits the model's complexity and causes problems with using a simple model to

* School of Economics and Management, University of the Chinese Academy of Sciences, Beijing 100190, China.
E-mail addresses: tyj@ucas.ac.cn (Y. Tian), zhangyuqi201@mails.ucas.ac.cn (Y. Zhang).

https://doi.org/10.1016/j.inffus.2021.11.005
Received 13 May 2021; Received in revised form 23 October 2021; Accepted 2 November 2021
Available online 14 November 2021

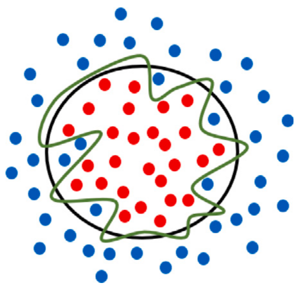


Fig. 2. The green line represents an overfitted model and the black line represents a regularized model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

fit the data. The second prior preference tends to add more convincing regularization to the model. If there are some stochastic/deterministic noise, the objective function will become unsmooth and easy to overfitting. Therefore, the regularization should make target smooth. The third prior preference is target-dependent. Usually, regularization can be constructed according to some properties of target. The fourth prior preference is to make the model easy to solve, such as weight decay.

Deep models can learn complex representational spaces to deal with the difficult learning tasks. They are prone to overfitting, particularly in networks with millions or billions of learnable parameters, such as convolutional neural networks (CNN) and Recurrent Neural Networks (RNN). Hence, for deep learning, regularization need a broader definition: regularization is any supplementary technique that aims to improve the model's generalization, i.e. produce better results on the test set [2]. Explicit regularization, such as dropout and weight decay, may improve generalization performance. When neural networks have far more learnable parameters than training samples, the generalization takes place even in the absence of any explicit regularization. In this case, explicit regularization is useless and unnecessary. A possible phenomenon is that the optimization is introducing some implicit regularization to make that there is no significant overfitting and test error continues decreasing as network size increases past the size required for achieving zero training error. It has been proved that early stopping implicitly regularize some convex learning problems. Batch normalization is an operator that normalizes the layer responses within each mini-batch. It has been widely used in many modern neural networks. Although there is no explicit design for regularization, it is usually found that batch normalization can improve generalization performance. In fact, the algorithm itself is an implicit regularization solution (this paper does not consider the implicit regularization based on algorithm) [3].

This paper provides a comprehensive review of the regularization strategies in both machine learning and deep learning. Different regularizations are classified into four categories in machine learning: sparse vector regularization, sparse matrix regularization, low-rank matrix regularization, and manifold regularization. We discuss the characteristics of each approach and the applied formulation. Moreover, we review the strategies to overcome the overfitting in deep learning such as data augmentation, dropout, early stopping, batch normalization, etc. Finally, we discuss several future directions for regularization.

To the best of our knowledge, there have been some regularization-related surveys conducted up to this point [2,4–6]. The work [2] gave the border definition of regularization and proposed a taxonomy to categorize existing methods but few specific introduction. The work [4] only introduced the low-rank regularizations and their applications. The work [6] only considered the regularization in traditional machine learning but not in deep learning. The work [5] gave a relatively complete introduction, but there are still omissions, and no application and comparison are given. In comparison, the novelties and contributions of this survey are as follows:

- Provide a thorough review of the strategies for overcoming overfitting, and these strategies are regarded as the comprehensive definition of regularization.
- Give the characteristics of regularizations and the comparison.
- Discuss how to choose or design a regularization for specific tasks.
- Introduce the related algorithms of solving the regularization problems.
- Conclude existing regularization opportunities and challenges, and provide several open issues and research trends.

The rest of this paper is organized as follows. Section 2 provides the regularization in machine learning. Subsequently, the regularization strategies of deep learning are introduced in Section 3. Finally, it concludes this paper and provides several open issues and research trends in Section 4.

2. The regularization strategies in the traditional machine learning

In this section, a comprehensive overview of sparse and low-rank regularization is provided, which are empirically categorized into four groups. These regularizations are usually proposed to solve specific applications. Thus, this section will give various practical problems before introducing regularizations.

For brief, note vector and matrix in bold to distinguish from the one-dimension variables.

2.1. Sparse vector-based regularization

Some variables are required to be sparse in many practical problems, such as compressing sensing, feature selection, sparse signal separation, sparse PCA, and sparse signals separation. Sparse regularization, which has attracted much attention in recent years, usually imposes the penalty on the variables. Many examples in different fields can be found where sparse regularization beneficial and favorable. This section will review four applications of sparse regularization-based sparse vector. Meanwhile, some vector-based sparse regularizations will be summarized.

2.1.1. Application scenario

Compressing Sensing In compressing sensing, radar [7], communications [8], medical imaging [9], image processing [10], and speech signal processing [11]. The objective of compressing sensing is to reconstruct a sparse signal $\boldsymbol{x}$:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & P(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{aligned} \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \leq n$ is the sensing matrix (also called measurement matrix), $\boldsymbol{y}$ is the compressed measurement of $\boldsymbol{x}$, and P is a penalty function for the sparsity. In fact, there is often measurement noise $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ so that

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}. \tag{2}$$

To minimize the noise $\boldsymbol{n}$, an unconstrained formulation can be written as

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \mu P(\boldsymbol{x}). \tag{3}$$

with $\mu > 0$.

Feature Selection In many fields today, such as genomics, health sciences, economics, and machine learning, the analysis of data sets with a number of variables comparable to or even much larger than the sample size is required. Most high-dimensional problems is infeasible and impractical because of their expensive computational costs. Feature selection has become a very popular topic in the last decade due to its effectiveness in high-dimensional case [12].

Let $\boldsymbol{A}$ is the design(deterministic) matrix, $\boldsymbol{x}$ is the unknown regression coefficients, and $\boldsymbol{y}$ is the response vector. Naturally, the penalty function uses the l_0 norm, i.e., $P(\boldsymbol{x}) = \|\boldsymbol{x}\|_0$ which counts the number of nonzero components in the vector $\boldsymbol{x}$. The equivalent optimization problem is

$$\min_{\boldsymbol{x}} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \quad \text{s.t. } \|\boldsymbol{x}\|_0 \leq k \tag{4}$$

called the k-sparse approximation problem.

Sparse PCA PCA is a useful tool for dimensionality reduction and feature extraction, which has been applied in virtually all areas of science and engineering, such as signal processing, machine learning, statistics, biology, medicine, finance, neurocomputing, and computer networks, to name just a few.

Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a centered data matrix, and the sparse PCA problem can be formulated as maximizing the variance along a direction represented by vector $\boldsymbol{x} \in \mathbb{R}^n$:

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & \frac{1}{n-1}\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \\ \text{s.t.} & \|\boldsymbol{x}\|_2 = 1 \\ & P(\boldsymbol{x}) \leq k \end{array} . \tag{5}$$

The first constraint states that x is a unit vector. In the second constraint, P is typically $l_0$norm, limiting the number of non-zero components in x that is less than or equal to k. An alternative to the sparsity constrained formulation is the sparsity penalized formulation as follows

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & \|\boldsymbol{A}\boldsymbol{x}\|_2^2 - \mu P(\boldsymbol{x}) \\ \text{s.t.} & \|\boldsymbol{x}\|_2 = 1 \end{array} .$$

Sparse Signals Separation Sparse signals separation has broad applications, such as source separation, super-resolution and inpainting, interference cancellation, saturation and clipping restoration, and robust sparse recovery in impulsive (sparse) noise.

The goal is to recover and demix $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ form the mixed linear measurements

$$\boldsymbol{y} = \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 \tag{6}$$

via the following formulation

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}_1,\boldsymbol{x}_2} & \mu g_1\left(\boldsymbol{x}_1\right) + g_2\left(\boldsymbol{x}_2\right) \\ \text{s.t.} & \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{y} \end{array} \tag{7}$$

where g_1 and g_2 are penalties for sparsity promotion. $\boldsymbol{A}_1 \in \mathbb{R}^{m\times n_1}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{m\times n_2}$ are known (deterministic) dictionaries, on which $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be sparsely (or approximately sparsely) represented.

A popular manner is to use convex relaxation that replaces the l_0 norm by the l_1 norm [13]:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}_1,\boldsymbol{x}_2} & \mu\left\|\boldsymbol{x}_1\right\|_1 + \left\|\boldsymbol{x}_2\right\|_1 \\ \text{s.t.} & \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{y} \end{array} . \tag{8}$$

2.1.2. Regularization

l_1 **norm** The l_1 is defined as

$$\|\boldsymbol{x}\|_1 = \sum_{j=1}^{N}\left|x_j\right|, \tag{9}$$

where x_j is the elements of $\boldsymbol{x}$. For sparsity promotion, the l_1 norm regularization is easy to establish convex optimization problems. However, the l_1 norm regularization cause a bias for large coefficients. In recent years, there has been a trend to study sparse regularizations with less bias.

l_p **norm** With the aim of bridging the gap between the l_0 norm and l_1 norm, there is a significant interest in the use of the nonconvex l_p norm,

$$\|\boldsymbol{x}\|_p = \left(\sum_{j=1}^{N}\left|x_j\right|^p\right)^{\frac{1}{p}}, \tag{10}$$

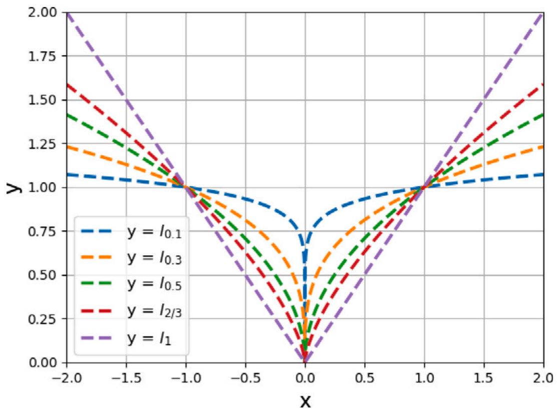


Fig. 3. The curve of $|x|^p$ for various values of p. As p tends to zero, the curve of $|x|^p$ approaches the indicator function.

where x_j are the elements of $\boldsymbol{x}$. For measuring sparsity, $0 < p < 1$ is of most interest. The two special cases of $p = 1/2$ and $p = 2/3$ are in [14]. Fig. 3 presents the curve of the scalar weight function l_p norm, the core of the norm computation, for various values of p, showing that as p goes to zero, this measure becomes a count of the nonzeros in $\boldsymbol{x}$.

Note that among the l_p norms, the choice $p = 1$ gives a convex function, while every choice $0 < p < 1$ yields a concave function. The l_p penalty reduces to the soft-thresholding when $p = 1$, while it tends to the hard-thresholding in the limit as $p \to -\infty$.

Projection Operator The projection operator [15], which can be seen as a regularization, is defined as

$$P(\boldsymbol{x}) = \delta_C(\boldsymbol{x}) \tag{11}$$

where C is a closed and convex set and $\delta_C(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} \in C \\ \infty, & \text{otherwise} \end{cases}$ is the indicator function that makes the solution x project back onto C. The Table 1 shows closed and convex sets and their corresponding (orthogonal) projections:

Smoothly Clipped Absolute Deviation (SCAD) The SCAD was proposed by the work [16]. It has been commonly used in variable selection problems and has shown to be efficient in high-dimensional variable selection problems as compared to other penalties. The SCAD is defined as:

$$P(x_i) = \begin{cases} \lambda|x_i|, & \text{if } |x_i| < \lambda \\ \dfrac{2\gamma\lambda|x_i| - x_i^2 - \lambda^2}{2(\gamma-1)}, & \text{if } \lambda \leq |x_i| < \gamma\lambda \\ (\gamma+1)\lambda^2/2, & \text{if } |x_i| \geq \gamma\lambda \end{cases} \tag{12}$$

The SCAD penalty is continuously differentiable on $\mathbb{R}$ but singular at 0, with its derivatives zero outside the range $[-\gamma\lambda, \gamma\lambda]$. SCAD also produces a sparse set of solutions and this penalty is nonconvex.

Minimax Concave Penalty (MCP) The MCP function [17]

$$P(x_i) = \begin{cases} \lambda\left|x_i\right| - \frac{x_i^2}{2\gamma}, & \text{if } \left|x_i\right| \leq \gamma\lambda \\ \frac{1}{2}\gamma\lambda^2, & \text{if } \left|x_i\right| > \gamma\lambda \end{cases} \tag{13}$$

is another alternative to get less biased in sparse models. The MCP provides the sparse convexity to the broadest extent by minimizing the maximum concavity.

Generalized Minimax Concave (GMC) A new penalty based on MCP is GMC [18] which is non-convex but maintains the convexity property of some cost function such as the least squared loss function $\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$. The GMC penalty is defined in terms of a new multivariate generalization of the Huber function:

$$P(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 - S_B(\boldsymbol{x}) \tag{14}$$

Table 1
The convex sets and their corresponding (orthogonal) projections.

	Convex set	The projections
Nonnegative orthant	$C_1 = \mathbb{R}^n_+$	$[\boldsymbol{x}]_+$
Affine set	$C_2 = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\}$	$\boldsymbol{x} - \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$
Box	$C_3 = \text{Box}\{[b_i, u_i]^n_{i=1}\}$ where $b_i, u_i \in (-\infty, \infty]$	$\left(\min\left\{\max\left\{x_i, b_i\right\}, u_i\right\}\right)^n_{i=1}$
Half-space	$C_4 = \left\{\boldsymbol{x} : \boldsymbol{a}^T\boldsymbol{x} \le \alpha\right\}$	$\boldsymbol{x} - \frac{[\boldsymbol{a}^T\boldsymbol{x}-\alpha]_+}{\|\boldsymbol{a}\|^2}\boldsymbol{a}$

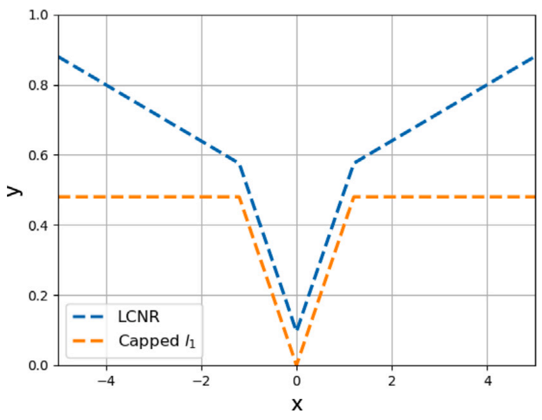


Fig. 4. Geometric view of two kinds of norms.

where

$$S_B(\boldsymbol{x}) = \inf_{\boldsymbol{v}\in\mathbb{R}^n}\left\{\|\boldsymbol{v}\|_1 + \frac{1}{2}\|B(\boldsymbol{x}-\boldsymbol{v})\|_2^2\right\}. \tag{15}$$

Obviously, when the matrix B meets the condition

$$B^TB \le \frac{1}{\lambda}A^TA, \tag{16}$$

the whole cost function

$$\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda(\|\boldsymbol{x}\|_1 - S_B(\|\ \boldsymbol{x})), \quad \lambda > 0 \tag{17}$$

is convex.

Capped l_1 The capped l_1 [19] is defined as

$$P(x_i) = \lambda\min\left(|x_i|, \gamma\right). \tag{18}$$

This formulation only regularizes those weights that are below a certain threshold γ. For those beyond this value, they can grow arbitrarily without experiencing penalties. It is noteworthy that it is a generalization of l_1 norm. Particularly, it becomes l_1 norm as $\gamma \to \infty$. Generally, with a finite γ, it approximates l_0 norm better than l_1 norm, and therefore leads to higher sparsity in some real-world applications. The key feature of capped l_1 norm is the lack of penalty for weights whose magnitudes are greater than γ, this feature, however, may become a drawback under some circumstances.

Leaky Capped l_1 **Norm Regularizer (LCNR)** With the analysis of capped l_1 in mind, the work [20] propose the leaky capped l_1 norm regularizer as an improved variant:

$$P(x) = \alpha\sum_i \min\left(|x_i|, \gamma\right) + \beta\sum_i \max\left(|x_i|, \gamma\right), \tag{19}$$

where $0 < \beta < \alpha$. As shown in Fig. 4, the functions of LCNR and capped l_1 are piecewise linear. The key difference between the proposed formulation and the standard l_1 norm and the capped l_1 is that large weights (i.e. those greater than γ) are still penalized positively but less heavily. To be more specific, this generalizes the capped l_1 norm, with an additional coefficient β to control how much those large weights are regularized. In particular, when $\beta = 0$, it reduces to the regular capped l_1.

Firm Penalty The firm penalty is formulated as [21]

$$P(x) = \begin{cases} \lambda\left[|x| - x^2/(2\gamma)\right], & \text{if } |x| < \gamma \\ \lambda\gamma/2, & \text{if } |x| \ge \gamma \end{cases}. \tag{20}$$

Moreover, the firm thresholding is a continuous and piecewise-linear approximation of the hard-thresholding.

Others Many penalty functions have been developed, such as the log sum penalty (LSP) [22,23]

$$P(x) = \lambda\log\left(1 + |x|/\gamma\right), \tag{21}$$

the rational penalty(Rat) [23,24]

$$P(x) = \frac{\lambda|x|}{1 + |x|/2\gamma}, \tag{22}$$

the arctangent penalty (Atan) [25]

$$P(x) = \lambda\frac{2}{\sqrt{3}}\left(\tan^{-1}\left(\frac{1+\gamma|x|}{\sqrt{3}}\right) - \frac{\pi}{6}\right), \tag{23}$$

and the exponential penalty (Exp) [26]

$$P(x) = \lambda(1 - \mathrm{e}^{-\gamma|x|}). \tag{24}$$

The penalty function usually satisfies the following properties:
Property 1 (P1): P is continuous on $\mathbb{R}$.
Property 2 (P2): P is non-decreasing, and concave on $\mathbb{R}_+$.
Property 3 (P3): P is continuously differentiable on $\mathbb{R}_+$.
Property 4 (P4): P' is convex and nonnegative on $\mathbb{R}_+$.
Property 5 (P5): $P(0) = 0$.
Property 6 (P6): $P(-x) = P(x)$.
Property 7 (P7): $x \to x^2/2\alpha + P(x)$ is convex on $\mathbb{R}$ with $\alpha > 0$.
Property 8 (P8): $\lim_{x\to\infty} P''(x) = 0$.

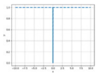
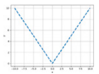
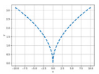
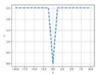
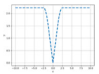
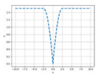
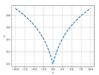
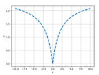
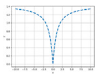
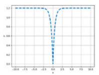
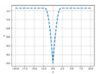
Table 2 lists several penalty functions, and their properties which are important in both theoretical analysis and applications. These properties can guide us to construct the regularizations.

Furthermore, ElasticNet uses a combination of l_1 norm and l_2 norm. These models have a tuning variable that controls the impact of the shrinkage method on the model parameter estimation. The algorithms to solve these regular problems include: (1) greedy strategy approximation; (2) constrained optimization; (3) proximal methods; and (4) homotopy algorithm-based sparse representation.

2.2. Sparse matrix-based regularization

Estimating large covariance or inverse covariance matrices has recently gained prominence due to the prevalence of high-dimensional data in modern applications such as economics and finance, bioinformatics, social networks, smart grid, climate studies, and health sciences [27–29]. It is well known that the sample covariance based on observed data is singular when the dimension is greater than the sample size. The estimation problem is generally difficult when the dimension of the covariance matrix is large [27]. An important assumption in the calculation of covariance and inverse covariance matrices is that the objective matrix is sparse. This section will reviews the large covariance matrix and inverse covariance matrix estimation that are important in modern multivariate analysis. Furthermore, this section will provides sparse matrix-based regularizations.

Table 2
The properties of the regularizations. The horizontal axis of Image is the value of x, and the vertical axis of Image is the value of the corresponding penalty formulate.

Penalty	Formulation	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	Image
l_0 norm	$\begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$	×	√	√	√	√	√	×	√	
l_1 norm	$\lvert x\rvert$	√	√	√	√	√	√	√	√	
$l_{0.5}$ norm	$\lvert x\rvert^p$	√	√	√	√	√	√	×	√	
Capped l_1	$\lambda \min(\lvert x\rvert, \gamma)$	√	√	×	–	√	√	×	√	
SCAD	$\begin{cases} \lambda\lvert x\rvert, & \text{if } \lvert x\rvert < \lambda \\ \frac{2\gamma\lambda\lvert x\rvert - x^2 - \lambda^2}{2(\gamma-1)}, & \text{if } \lambda \le \lvert x\rvert < \gamma\lambda \\ (\gamma+1)\lambda^2/2, & \text{if } \lvert x\rvert \ge \gamma\lambda \end{cases}$	√	√	×	–	√	√	√	√	
MCP	$\begin{cases} \lambda\lvert x\rvert - \frac{x^2}{2\gamma}, & \text{if } \lvert x\rvert \le \gamma\lambda \\ \frac{1}{2}\gamma\lambda^2, & \text{if } \lvert x\rvert > \gamma\lambda \end{cases}$	√	√	×	–	√	√	√	√	
LSP	$\lambda \log(1 + \lvert x\rvert/\gamma)$	√	√	√	√	√	√	√	√	
Rat	$\frac{\lambda\lvert x\rvert}{1+\lvert x\rvert/2\gamma}$	√	√	√	√	√	√	√	√	
Atan	$\lambda\frac{2}{\sqrt{3}}\left(\tan^{-1}\left(\frac{1+\gamma\lvert x\rvert}{\sqrt{3}}\right) - \frac{\pi}{6}\right)$	√	√	√	√	√	√	√	√	
Exp	$\lambda(1 - e^{-\gamma\lvert x\rvert})$	√	√	√	√	√	√	√	√	
Firm	$\begin{cases} \lambda\left[\lvert x\rvert - x^2/(2\gamma)\right], & \text{if } \lvert x\rvert \le \gamma \\ \lambda\gamma/2, & \text{if } \lvert x\rvert \ge \gamma \end{cases}$	√	√	×	–	√	√	√	√	

2.2.1. *Application scenario*

Large Sparse Covariance Matrix Estimation Large inverse covariance matrix estimation is a fundamental problem in a wide range of applications, from economics and finance to genetics, social networks, and health sciences. The estimation problem becomes more difficult when the dimension of the covariance matrix is high. The assumptions for high-dimensional covariance matrix estimation is sparsity. The sparsity means that a majority of the off-diagonal elements are nearly zero and thus decrease the number of free parameters to estimate [30]. Since the diagonal elements of a correlation (also known as covariance) matrix are always positive, the assumption is for the off-diagonal elements.

Let $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^n \in \mathbb{R}^n$ is independent and identically distributed (i.i.d.). The covariance which between each dimensions forms an $n \times n$ matrix called the covariance matrix. Calculate the sample covariance matrix $\boldsymbol{S}$ [31],

$$\boldsymbol{S} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{s}^i - \overline{\boldsymbol{s}}\right)\left(\boldsymbol{s}^i - \overline{\boldsymbol{s}}\right)^\top \tag{25}$$

where $\overline{\boldsymbol{s}}$ is the mean of $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^n$, i.e.

$$\overline{\boldsymbol{s}} = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{s}^i. \tag{26}$$

The generalize thresholding estimator solves the following problem

$$\min_{\boldsymbol{X}} \quad \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{S}\|_F^2 + \sum_{i\neq j} M\left(X_{ij}\right), \tag{27}$$

where X_{ij} are the elements of the matrix $\boldsymbol{X}$ and M is generalized penalty function for sparsity. These notations still apply in the rest of this section.

Large Sparse Inverse Covariance Matrix Estimation Sparse inverse covariance matrix estimation is a fundamental problem in a Gaussian network model and has attracted much attention in the last decade [32]. The estimation of large sparse inverse covariance matrices is a tricky statistical problem in many application areas such as mathematical finance, geology, health, or many others. Hence, sparsity regularized negative log-likelihood minimization has become a popular approach for estimating the sparse inverse covariance matrix. A common method is adding some mechanism such as regularization term in the estimation model to explicitly enforce sparsity in $\boldsymbol{X}$.

$$\min_{\boldsymbol{X}} \quad \operatorname{tr}(\boldsymbol{S}\boldsymbol{X}) - \log|\boldsymbol{X}| + \sum_{i\neq j} M\left(X_{ij}\right), \tag{28}$$

where $tr(\cdot)$ and $|\cdot|$ represent the trace and the determinant of matrix respectively. The sample covariance matrix $\boldsymbol{S}$ is invertible.

2.2.2. *Regularization*

Norm Penalty A convex model based on the $l_{1,\text{off}}$-regularized maximum likelihood problem

$$\min_{\mathbf{X}\succ 0} \operatorname{tr}(\boldsymbol{S}\boldsymbol{X}) - \log|\boldsymbol{X}| + \lambda\|\boldsymbol{X}\|_{1,\text{off}} \tag{29}$$

where

$$\|\boldsymbol{X}\|_{1,\text{off}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left|X_{ij}\right|, \tag{30}$$

which refers to the element-wise 1 norm.

The F norm is defined as

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}^2\right)^{1/2}. \tag{31}$$

Generalizing the above norm, we denote the $l_{p,\text{off}}$ norm of matrix $\boldsymbol{X}$ as follows

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{p,\text{off}} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left|X_{ij}\right|^{p}\right)^{1/p} \tag{32}$$

with $p \geq 1$. These above norm is called the elements-wise matrix norms since they imposes the same penalty to all elements.

Note that $l_{1,\text{off}}$ norm makes a distinction with the 1 norm of matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$,

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{1} = \max_{1\leq j\leq n}\sum_{i=1}^{m}\left|X_{ij}\right|, \tag{33}$$

which is simply the maximum absolute column sum of the matrix. Minimizing the maximum absolute column sum is equivalent to minimizing the upper bound of absolute column sum which enforces the column sum close to zero, i.e. feature selection.

The $l_{2,1}$ norm of a matrix was first introduced in [33] as rotational invariant l_1 norm, that is

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{2,1} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m}X_{ij}^{2}\right)^{1/2}, \tag{34}$$

which is proposed to overcome the difficulty of robustness to outliers [33]. Similarly, the $l_{1,2}$ norm of matrix $\boldsymbol{X}$ is defined as

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{1,2} = \left(\sum_{j=1}^{n}\left(\sum_{i=1}^{m}\left|X_{ij}\right|\right)^{2}\right)^{1/2}. \tag{35}$$

For $p, q \geq 1$, a generalized form of the $l_{1,2}$ norm is the $l_{p,q}$ norm as follows [34]:

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{p,q} = \left(\sum_{j=1}^{n}\left(\sum_{i=1}^{m}\left|X_{ij}\right|^{p}\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}. \tag{36}$$

Moreover, there are many norm penalties based on the ∞ norm

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{\infty} = \max_{1\leq i\leq m}\sum_{j=1}^{n}\left|X_{ij}\right|, \tag{37}$$

which are respectively $l_{\infty,1}$ norm [35]

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{\infty,1} = \sum_{i=1}^{m}\max_{1\leq j\leq n}\left|X_{ij}\right|, \tag{38}$$

and $l_{\infty,\text{off}}$ norm

$$M(\boldsymbol{X}) = \|\boldsymbol{X}\|_{\infty,\text{off}} = \max_{i\neq j}\left|X_{ij}\right|. \tag{39}$$

Non-Norm Penalty The following penalties are not real norms, because they are nonconvex and do not satisfy the triangle inequality of a norm. The capped $l_{p,1}$ is defined as [36]:

$$M(\boldsymbol{X}) = \lambda\sum_{j=1}^{n}\min\left(\left(\sum_{i=1}^{m}\left|X_{ij}\right|^{p}\right)^{\frac{1}{p}},\gamma\right) \tag{40}$$

with the given threshold θ, the capped $l_{p,1}$ penalty focuses on rows with smaller l_p norms than θ, which is more likely to be sparse.

The elements-wise MCP function is defined as,

$$M(X_{ij}) = \begin{cases} \frac{\gamma\lambda^{2}}{2}, & \left|X_{ij}\right| \geq \gamma\lambda \\ \lambda\left|X_{ij}\right| - \frac{X_{ij}^{2}}{2\gamma}, & \text{otherwise} \end{cases} \tag{41}$$

which is considered the natural extension of MCP functions on matrices [37]. The MCP function is an elements-wise matrix penalty.

As shown in Fig. 5, these penalties can be classified into three categories: rows sparsity, columns sparsity, and sparsity of both columns and rows, and it depends on which part of the matrix is punished.

The penalty function of matrix usually satisfies the following properties:

Property 9 (P9): $M(\boldsymbol{X}) \geq 0$.

Property 10 (P10): $M(\boldsymbol{X}) \leq \|\boldsymbol{X}\|_{1,\text{off}}$.

Table 3 lists several penalty functions and their properties.

The algorithms to solve these regular problems include: (1) alternative minimized algorithm; (2) block coordinate descent.

2.3. Low-rank matrix recovery based on low-rank regularization

This section reviews low-rank regularization based on the low-rank recovery problems which conclude two hop topics problems: matrix completion and robust PCA. Matrix completion aims to recover a low-rank matrix from partially observed entries, while robust PCA aims to decompose a low-rank matrix from sparse corruption.

2.3.1. Application scenario

Matrix Completion In many applications, the matrix is excepted to be constructed in the sense that it is low-rank, which can be recovered from incomplete portions of the entries. For instance, vendors provide recommendations to the user based on their preferences. Users and ratings are represented as rows and columns, respectively, in a data matrix. Users can rate movies but they typically rate only very few movies so that very few scattered entries can be observed [38]. Commonly, only a few factors effect to the preference of users so that the data matrix of all users-rating can be regarded as a low-rank matrix. The goal of this problem is to complete the data matrix of all users-rating using the observed data.

Given the incomplete observations A_{ij}, the goal is to recover a $m\times n$ matrix X_{ij}, that is,

$$\begin{aligned} \min_{\mathbf{X}} \quad & \text{rank}(\boldsymbol{X}) \\ \text{s.t.} \quad & X_{ij} = A_{ij} \quad (i,j) \in \Omega \end{aligned} \tag{42}$$

where $\Omega \subset [1,\dots,m]\times[1,\dots,n]$ is the index set in which X_{ij} is observed. The $\text{rank}(\boldsymbol{X})$ is equal to the rank of the matrix $\boldsymbol{X}$. It usually transforms to an unconstrained formulation which is written as [39]

$$\min_{\boldsymbol{X}} \frac{1}{2}\left\|\mathcal{P}_{\Omega}(\boldsymbol{X}) - \mathcal{P}_{\Omega}(\boldsymbol{M})\right\|_{\text{F}}^{2} + F(\boldsymbol{X}) \tag{43}$$

where $\mathcal{P}_{\Omega}(\mathbf{X}) = \begin{cases} X_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$.

If $F(\boldsymbol{X}) = \text{rank}(\boldsymbol{X})$, the objective function is nonconvex. A popular convex relaxation method is to approximate the rank function using the nuclear norm, i.e. $\|\boldsymbol{X}\|_{*}$ which is the sum of singular value of $\boldsymbol{X}$ [40].

Robust PCA Robust PCA is applied in many important applications such as foreground detection of video [41], image processing [42], fault detection and diagnosis [43] and so on. The robust PCA problem is commonly thought of as a low-rank matrix recovery problem with incorporates sparse corruption. The goal of robust PCA is to enhance the robustness of PCA against outliers or corrupted observations of PCA. In fact, the data matrix $\boldsymbol{A}$ of this problem is a composite matrix of sparse and low-rank recovery which needs to be decomposed into two components such that $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S}$, where $\boldsymbol{L}$ is a low-rank matrix and $\boldsymbol{S}$ is a sparse matrix. The matrix $\boldsymbol{A}$ is observed and the problem can be represented as:

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{S}} \quad & F(\boldsymbol{L}) + \lambda M(\boldsymbol{S}) \\ \text{s.t.} \quad & \boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S} \end{aligned} \tag{44}$$

where F and M are penalties for low-rank and sparsity promotion, respectively. A straightforward formulation is

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{S}} \quad & rank(\boldsymbol{L}) + \lambda\|\boldsymbol{S}\|_{0} \\ \text{s.t.} \quad & \boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S} \end{aligned} \tag{45}$$

Since the problem is NP-hard, the nuclear norm $\|\boldsymbol{L}\|_{*}$ and $l_{1,\text{off}}$ norm $\|\boldsymbol{S}\|_{1,\text{off}}$ are used to instead of the $rank(\boldsymbol{L})$ and $\|\boldsymbol{S}\|_{0}$, respectively.

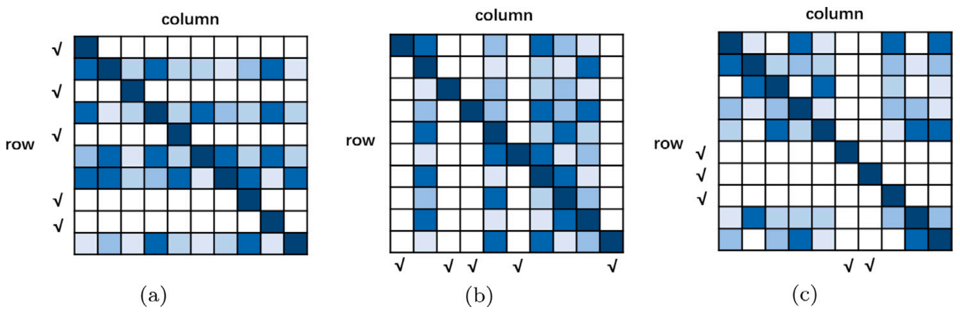


Fig. 5. Left: rows sparsity *(Row 1, 3, 5, 8, 9)*. Middle: columns sparsity *(Column 1, 3, 4, 6, 10)*. Right: sparsity of both columns and rows *(Row 6, 7, 8 and Column 6,7)*.

Table 3
The sparse matrix regularizations.

	Penalty	Formulation	Rows sparsity	Columns sparsity	Rows & Columns sparsity	P9	P10
Elements-wise	$\|\boldsymbol{X}\|_{1,\text{off}}$	$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|X_{ij}\right\|$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{F}$	$\left(\sum_{i=1}^{n}\sum_{j=1}^{m}X_{ij}^{2}\right)^{1/2}$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{p,\text{off}}$	$\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left\|X_{ij}\right\|^{p}\right)^{1/p}$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{\infty,\text{off}}$	$\max_{i\neq j}\left\|X_{ij}\right\|$	×	×	√	√	√
	MCP	$\begin{cases}\frac{\gamma\lambda^{2}}{2}, & \text{if } \left\|X_{ij}\right\| \geq \gamma\lambda\\ \lambda\left\|X_{ij}\right\| - \frac{X_{ij}^{2}}{2\gamma}, & \text{otherwise}\end{cases}$	×	×	√	√	√
Non-elements-wise	$\|\boldsymbol{X}\|_{2,1}$	$\sum_{j=1}^{n}\left(\sum_{i=1}^{m}X_{ij}^{2}\right)^{1/2}$	×	√	×	√	√
	$\|\boldsymbol{X}\|_{2,1}$	$\sum_{i=1}^{m}\left(\sum_{j=1}^{n}X_{ij}^{2}\right)^{1/2}$	√	×	×	√	√
	$\|\boldsymbol{X}\|_{p,q}$	$\left(\sum_{j=1}^{n}\left(\sum_{i=1}^{m}\left\|X_{ij}\right\|^{p}\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{\infty}$	$\max_{1\leq i\leq m}\sum_{j=1}^{n}\left\|X_{ij}\right\|$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{1}$	$\max_{1\leq j\leq n}\sum_{i=1}^{m}\left\|X_{ij}\right\|$	×	×	√	√	√
	$\|\boldsymbol{X}\|_{\infty,1}$	$\sum_{i=1}^{m}\max_{1\leq j\leq n}\left\|X_{ij}\right\|$	×	×	√	√	√
	Capped $l_{p,1}$	$\lambda\sum_{j=1}^{n}\min\left(\left(\sum_{i=1}^{m}\left\|X_{ij}\right\|^{p}\right)^{\frac{1}{p}},\gamma\right)$	×	×	√	√	√

Considering the unconstrained problem is more tractable, the formulations can be alternated by

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_{*} + \|\boldsymbol{S}\|_{1} + \frac{1}{2\mu}\|\boldsymbol{A}-\boldsymbol{L}-\boldsymbol{S}\|_{\boldsymbol{F}}^{2} \tag{46}$$

where $\mu > 0$ is the penalty parameter.

2.3.2. Regularization

In recent years, the rank-norm is replaced by some alternative regularizers, which can be divided into two groups: convex relaxations and non-convex relaxations [4]. The main idea of these regularizers is to make singular values sparse which is equal to make the matrix low-rank.

The work [44] proposed the elastic-net regularization for singular values

$$F(\boldsymbol{X}) = \sum_{i=1}^{k}\left(\sigma_i + \lambda\sigma_i^2\right), \tag{47}$$

where σ_i is the singular values of $\boldsymbol{X}$ and $k = \min\{m, n\}$. It is widely applicable to robust subspace learning problems with heavy corruptions such as outliers and missing entries.

Schatten p-Norm (Sp-Norm) [45] of matrix $\mathbf{X}$ is defined as

$$F(\boldsymbol{X}) = \|\boldsymbol{X}\|_{S_p} = \left(\sum_{i=1}^{k}\sigma_i^p\right)^{\frac{1}{p}} = \left(\mathrm{Tr}\left(\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{\frac{p}{2}}\right)\right)^{\frac{1}{p}} \tag{48}$$

which is equivalent to l_p norm of the singular values. The Sp-norm minimization method provides a better approximation to the original NP-hard problem, resulting in better theoretical and practical results [46].

The weighted nuclear norm is to improve the flexibility of nuclear norm [47], which is defined as:

$$F(\boldsymbol{X}) = \|\boldsymbol{X}\|_{w} = \sum_{i=1}^{k} w_i\sigma_i, \tag{49}$$

where the singular values are assigned different weights. Moreover, this idea can be generalized to the Schatten p-norm minimization [48].

The capped trace norm is defined as

$$F(\boldsymbol{X}) = \|\boldsymbol{X}\|_{C_*^{\varepsilon}} = \sum_{i=1}^{k}\min\left(\sigma_i, \varepsilon\right). \tag{50}$$

This regularization is used to improve the robustness to outliers by closely approximating the rank minimization.

Moreover, numerous non-convex relaxations derived from sparse learning of the vectors have been proposed [49], such as Log Nuclear Norm (LNN) [50]

$$F(\boldsymbol{X}) = \lambda\log\det\left(\boldsymbol{I} + \boldsymbol{X}^T\boldsymbol{X}\right) = \sum_{i=1}^{k}\lambda\log\left(\sigma_i + 1\right), \tag{51}$$

MCP

$$F(\boldsymbol{X}) = \sum_{i=1}^{k}\begin{cases}\lambda\sigma_i - \dfrac{\sigma_i}{2\gamma}, & \text{if } \sigma_i < \gamma\lambda\\ \dfrac{\gamma\lambda^2}{2}, & \text{if } \sigma_i \geq \gamma\lambda\end{cases}, \tag{52}$$

SCAD

$$F(\boldsymbol{X})=\sum_{i=1}^{k}\begin{cases}\lambda\sigma_i, & \text{if } \sigma_i\le\lambda\\ \dfrac{-\sigma_i^2+2\gamma\lambda\sigma_i-\lambda^2}{2(\gamma-1)}, & \text{if } \lambda<\sigma_i\le\lambda\gamma\\ \dfrac{\lambda^2(\gamma+1)}{2}, & \text{if } \sigma_i>\lambda\gamma\end{cases}, \tag{53}$$

ETP [51]

$$F(\boldsymbol{X})=\sum_{i=1}^{k}\frac{\lambda\left(1-\exp\left(-\gamma\sigma_i\right)\right)}{1-\exp(-\gamma)}, \tag{54}$$

Logarithm [52]

$$F(\boldsymbol{X})=\sum_{i=1}^{k}\frac{\log\left(\gamma\sigma_i+1\right)}{\log(\gamma+1)}, \tag{55}$$

Geman [53]

$$F(\boldsymbol{X})=\sum_{i=1}^{k}\frac{\lambda\sigma_i}{\sigma_i+\gamma}, \tag{56}$$

Laplace [54]

$$F(\boldsymbol{X})=\sum_{i=1}^{k}\lambda\left(1-\exp\left(-\frac{\sigma_i}{\gamma}\right)\right), \tag{57}$$

and so on. Note that Laplace, ETP, and Exp are similar in formulation, and they can be transformed into each other by adjusting parameters. Similarly, this relationship exists between LNN, Logarithm and LSP.

For the problems with low-rank regularization, a popular method is proximal method. It requires computing the proximal operator

$$\mathrm{Prox}_F^{\lambda}(\tilde{\boldsymbol{X}})=\arg\min_{\boldsymbol{X}}\frac{1}{2}\|\boldsymbol{X}-\tilde{\boldsymbol{X}}\|_F^2+F(\boldsymbol{X}) \tag{58}$$

where usually $\tilde{\boldsymbol{X}}$ is known. However, there may not exist a general explicit solution of the proximal operator. Hence, the proximity operator is calculated by minimizing the problem

$$\min_{\boldsymbol{X}}\frac{1}{2}\|\boldsymbol{X}-\tilde{\boldsymbol{X}}\|_F^2+F(\boldsymbol{X}), \tag{59}$$

whose results are solved by the optimization solver here. In order to explain the shrinkage effect more intuitively, these results are regarded as approximate to explicit solutions.

Table 4 shows some low-rank regularizations. Among it, the shrinkage image shows the results $\mathrm{Prox}_F^{\lambda}(b)$ where b is a variable of the singular value. And it can be seen that when b takes a small value the shrinkage effect of different regularizers are similar. Nevertheless, when b takes a large value the difference of $\mathrm{Prox}_F^{\lambda}(b)$ between non-convex regularizers and nuclear norm is significant. In particular, the shrink of non-convex relaxations on larger values are very small, which is in contrast with the nuclear norm taking serious shrinks on larger singular values. Hence, using non-convex regularizers can preserve main information of b[55].

The algorithms to solve these regular problems include: (1) proximal methods; (2) block coordinate methods; (3) alternating linearization algorithms; (4) greedy strategy approximation.

2.4. Manifold regularization

This section will introduce the manifold regularization, a regularization technique for manifold learning, based on the assumption that the data has some inherent structure.

2.4.1. Application scenario

3D point cloud is a discrete collection of triples marking exterior object surface locations in 3D space [56,57]. 3D point cloud has been widely adopted in indoor navigation, autopilot, and augmented reality, etc. In the conventional imperfect acquisition processes of 3D point cloud, noise data is non-negligible. An idea is the manifold learning. To learn something from data, it is assumed that the data has some inherent structure. In manifold learning, this assumption is explicit: it assumes that the observed data lies on a low-dimensional manifold embedded in a higher-dimensional space which can be seen in Fig. 6. Intuitively, this assumption states that the shape of data is relatively simple. The low-dimensional manifold model can be applied to surface patches in the point cloud, which uses he patch manifold prior to seek self-similar patches and remove noise [58].

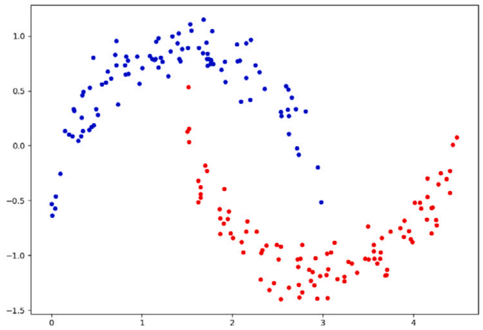


Fig. 6. The geometry structure of the data distribution in low-dimension space.

In semi-supervised learning, there are two sets of samples $x\in\mathbb{R}^n$ which consists l labeled samples $Q=\left\{\left(x_i,y_i\right)\right\}_{i=1}^{l}$ and u unlabeled samples $U=\left\{\left(x_i,y_i\right)\right\}_{i=1+l}^{u+l}$. The optimization can be written as

$$f^*=\underset{f\in\mathcal{H}_K}{\mathrm{argmin}}\frac{1}{l}\sum_{i=1}^{l}V\left(x_i,y_i,f\right)+\gamma\|f\|_K^2. \tag{60}$$

where $\mathcal{H}_K$ represents reproducing kernel Hilbert spaces. The first term is the loss function and the regularization $\|f\|_K$ is used to control the complexity of the classification model. γ is a trade-off parameters. Manifold learning adds a regularization $\|f\|_I$ to the loss function that penalizes functions which are more complex with respect to the intrinsic geometry of the data manifold:

$$f^*=\underset{f\in\mathcal{H}_K}{\mathrm{argmin}}\frac{1}{l}\sum_{i=1}^{l}V\left(x_i,y_i,f\right)+\gamma_A\|f\|_K^2+\gamma_B\|f\|_I^2 \tag{61}$$

where γ_A and γ_B are trade-off parameters and $\mathcal{H}_K$ represents reproducing kernel Hilbert spaces. The regularization should bias learning toward smoother functions f. Since $\|\nabla f(x)\|$ describes the smoothness of a function f at x, a notion of the smoothness of f on the entire manifold M is given as:

$$\int_M\|\nabla f(x)\|^2dx. \tag{62}$$

This regularization expresses the idea that the functions should be smooth on the manifold, not just smooth in the extrinsic space.

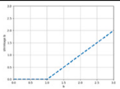
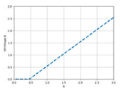
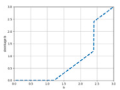
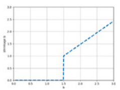
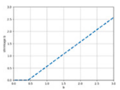
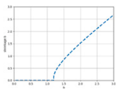
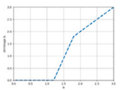
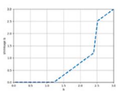
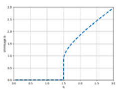
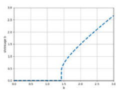
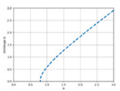
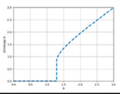
To smooth f in the intrinsic space, there are many manifold regularizations including the Laplacian regularization (LapR), Hessian regularization (HesR) and p-Laplacian regularization (pLapR) [59].

2.4.2. Regularization

Laplacian Regularization The manifold regularization cannot be calculated exactly with only finite data. The main idea behind manifold regularization is to approximate this concept by substituting a graph approximation for the manifold. Laplacian regularization is one of the most popular technologies which uses the graph Laplacian to approximate the manifold regularization.

Let G be a connected, undirected graph with edges and vertices. When discussing weighted graphs, the weight on the edge between nodes i and j is denote by W_{ij}. A function on a graph is smooth if its value at a node is similar to its value at each of the node's neighbors, intuitively. Thus, the regularization $\int_M\|\nabla f(x)\|^2dx$ is analogous to

Table 4
The low-rank regularizations. The horizontal axis of shrinkage image is the value of σ_i, and the vertical axis of shrinkage image is corresponding shrinkage value.

Regularization	Convexity	Formulation	Shrinkage image
$\|\mathbf{X}\|_*$	Convex	$\sum_{i=1}^{k} \sigma_i$	
$\|\mathbf{X}\|_w$	Convex	$\sum_{i=1}^{k} w_i \sigma_i$	
$\|\mathbf{X}\|_{C_*^\varepsilon}$	Non-Convex	$\sum_{i=1}^{k} \min\left(\sigma_i, \varepsilon\right)$	
$\|\mathbf{X}\|_{S_p}$	Non-Convex	$\left(\sum_{i=1}^{k} \sigma_i^p\right)^{\frac{1}{p}}$	
Elastic-net	Convex	$\sum_{i=1}^{k} \left(\sigma_i + \lambda \sigma_i^2\right)$	
LNN	Non-Convex	$\sum_{i=1}^{k} \lambda \log\left(\sigma_i + 1\right)$	
MCP	Non-Convex	$\sum_{i=1}^{k} \begin{cases} \lambda\sigma_i - \frac{\sigma_i}{2\gamma}, & \text{if } \sigma_i < \gamma\lambda \\ \frac{\gamma\lambda^2}{2}, & \text{if } \sigma_i \geq \gamma\lambda \end{cases}$	
SCAD	Non-Convex	$\sum_{i=1}^{k} \begin{cases} \lambda\sigma_i, & \text{if } \sigma_i \leq \lambda \\ \frac{-\sigma_i^2+2\gamma\lambda\sigma_i-\lambda^2}{2(\gamma-1)}, & \text{if } \lambda < \sigma_i \leq \lambda\gamma \\ \frac{\lambda^2(\gamma+1)}{2}, & \text{if } \sigma_i > \lambda\gamma \end{cases}$	
ETP	Non-Convex	$\sum_{i=1}^{k} \frac{\lambda(1-\exp(-\gamma\sigma_i))}{1-\exp(-\gamma)}$	
Logarithm	Non-Convex	$\sum_{i=1}^{k} \frac{\log(\gamma\sigma_i+1)}{\log(\gamma+1)}$	
Geman	Non-Convex	$\sum_{i=1}^{k} \frac{\lambda\sigma_i}{\sigma_i+\gamma}$	
Laplace	Non-Convex	$\sum_{i=1}^{k} \lambda\left(1 - \exp\left(-\frac{\sigma_i}{\gamma}\right)\right)$	

the squared difference between a node's value and the values of its neighbors in the graph case. That is, $\|f\|_I^2$ is approximated as

$$\frac{1}{n^2} \sum_{i,j} W_{ij} \left(f\left(x_i\right) - f\left(x_j\right)\right)^2. \tag{63}$$

LapR constructs a nearest neighbor graph on the feature space which utilizes a graph similarity (affinity) matrix $\boldsymbol{W} = [W_{ij}]$ which measures the similarity between samples x_i and x_j:

$$W_{ij} = \begin{cases} \exp\left\{-\frac{\left\|x_i - x_j\right\|_2^2}{2\sigma^2}\right\}, & \text{if } x_i \in N_q\left(x_j\right) \text{ or } x_i \in N_q\left(x_i\right) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

where $N_p(\boldsymbol{x})$ is the set of p-nearest neighbors of sample $\boldsymbol{x}$ and $\boldsymbol{W}$ is usually symmetrized by $\boldsymbol{W} = (\boldsymbol{W}^T + \boldsymbol{W})/2$. It assumes that if two point x_1 and x_2 are close in the manifold geometry, their labels should be similar.

The geometry structure is exploited as

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \left(f\left(x_i\right) - f\left(x_j\right)\right)^2 = \operatorname{Tr}\left(\boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f}\right) \tag{65}$$

where $\boldsymbol{f} = \left[f\left(x_1\right), f\left(x_2\right), \ldots, f\left(x_n\right)\right]^T$ is prediction of all of the data(labeled and unlabeled), $\boldsymbol{D}$ is a diagonal matrix with $W_{ii} = \sum_{j=1}^{n} W_{ij}$, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is called the Laplacian of the graph G, which quantifies the smoothness of functions. Notice that, in Eq. (61), $\{f_i\}_{i=1}^{l}$ should satisfy the constraint $f\left(x_i\right) = y_i, i = 1, \ldots, l$, which is reflected in the term $\frac{1}{l} \sum_{i=1}^{l} V\left(x_i, y_i, f\right)$.

Considering the V as the hinge loss, the framework of LapR is written as follow:

$$f^* = \arg \min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} \left(1 - y_i f\left(x_i\right)\right)_+ + \gamma_A \|f\|_K^2 + \frac{\gamma_B}{n^2} \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} \tag{66}$$

where $n = l + u$ and $\mathcal{H}_K$ represents reproducing kernel Hilbert spaces. In this way, the data geometric structure existed in high dimensional space is preserved and the data distribution information is well explored.

In the image annotation, the manifold regularization is used to ensure that local geometric structures are consistent between different view feature spaces and the latent semantics matrix among different views [60,61]. The work [62] further learns different the graph Laplacian from the different views and adjusts the combination coefficients. An alternative approach assumes that the intrinsic manifold is the

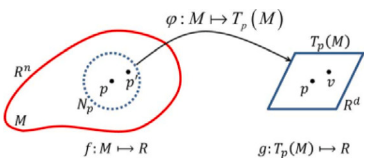


Fig. 7. The process of defining the tangent Hessian [149].

convex combination of the pre-learned manifold candidates, that is

$$\boldsymbol{L} = \sum_{k=1}^{m} \mu_k \boldsymbol{L}_k, \quad \text{s.t.} \sum_{k=1}^{m} \mu_k = 1, \mu_k \geq 0, \text{ for } k = 1, \ldots, m \tag{67}$$

where m represents m views and $\boldsymbol{C} = \{\boldsymbol{L}_1, \boldsymbol{L}_2, \ldots, \boldsymbol{L}_m\}$ is the set of the candidates graph Laplacian. Thus the $L \in \mathbf{conv}\boldsymbol{C}$ is also a graph Laplacian [63]. The work [64] proposed pairwise constraints where the graph Laplacian matrix is composed of the must-link Laplacian matrix and cannot-link Laplacian matrix.

Hessian Regularization The geodesic function in the null space of Laplacian is no other than a const, which implicates that LapR biases the solution towards a constant function and then leads to poor extrapolation capability [65]. A theory of the Hessian is a substitute for the Laplacian. Given a smooth manifold $M \subset \mathbb{R}^n$ and the tangent space $T_p(M) \subset \mathbb{R}^n$ for each point $p \in M$, thus the Hessian regularization can be defined as

$$H(f) = \int_M \|\nabla_a \nabla_b f\|^2_{T_p(M) \otimes T_p(M)} \, dV(p) \tag{68}$$

where $\nabla_a \nabla_b f$ is the second covariant derivative of f and $dV(p)$ is the natural volume element.

The construction of the Hessian matrix of a point depends on the choice of the coordinate system. In the normal coordinates, hessian matrix can be evaluated quite easily. The local normal coordinates is used to define the Hessian of a function f. Let $N_k(p)$ as a neighborhood of the k nearest of a sample p. Considering the tangent space as a subspace of $\mathbb{R}^n$, the orthonormal coordinate system of $T_p(M)$ an be approximated by the eigenspace of neighborhood $N_k(p)$, which is spanned by the largest d eigenvalues. For each point $p' \in N_k(p)$, there is a unique closest point v on $T_p(M)$. The details are shown in Fig. 7. Then the rule $g(x) = f(p')$ defines a function $g : U \to \mathbb{R}$, where U is a neighborhood of 0 in $\mathbb{R}^n$. Thus, the Hessian of f at p in tangent coordinates is defined as the ordinary Hessian of g,

$$\left(H_f^{\tan}(p)\right)_{i,j} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} g(x)|_{x=0}. \tag{69}$$

Meanwhile, the Frobenius norm of a Hessian matrix is invariant to the coordinate changes, and therefore

$$H(f) = \int_{p \in M} \left\| H_f^{\tan}(p) \right\|_F^2 dp \tag{70}$$

measures the average curviness of f over the manifold M. Kim et al. [66] introduce a sparse matrix approximation H by fitting a quadratic function to the data points, which yields an objective function almost identical to that of Laplacian-based manifold regularization:

$$H(f) \approx \sum_{r,s=1}^{m} \left(\sum_{\alpha=1}^{k} H_{rs\alpha}^{(i)} f_\alpha \right)^2 = \sum_{i=1}^{n} \sum_{\alpha \in N_k(p_i)} \sum_{\beta \in N_k(p_i)} f_\alpha f_\beta H_{\alpha\beta}^{(i)} = \boldsymbol{f}^T \boldsymbol{H} \boldsymbol{f} \tag{71}$$

where $H_{rs\alpha}^{(i)}$ is local Hessian operator with normal coordinate x_r and x_s, and $\boldsymbol{H}$ is the accumulated matrix summing up all the matrices $H_{rs\alpha}^{(i)}$.

A learning framework mHLR [67] was proposed which integrates multiple kernel learning and ensemble graph Hessian learning, and combines multiple Hessian regularizations to boost the exploring of local geometry. The work [68] applied the Hessian regularization to remote sensing image recognition. The work [69] applied hessian-based norm regularization for image restoration with biomedical applications. The multiview Hessian regularization (mHR) optimally combined multiple Hessian regularizations which are obtained from the particular view of instances and direct the classification function which varies linearly along the data manifold [70].

p**-Laplacian Regularization** As a nonlinear generalization of the standard graph Laplacian, p-Laplacian has attracted attention from machine learning fields.

Similar to the graph Laplacian [71], the unnormalized p-Laplacian (for $p > 1$) can be defined by

$$\boldsymbol{f}^T \boldsymbol{L}_p \boldsymbol{f} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} |f_i - f_j|^p. \tag{72}$$

And if $p = 2$, the p-Laplacian becomes the standard graph Laplacian.

The objective function of p-Laplacian can be computed as follows [72]:

$$f^* = \operatorname{argmin}_{f \in \mathcal{H}(V)} \left\{ S_p(f) + \mu \|f - y\|^2 \right\}, \tag{73}$$

where $S_p(f) := \frac{1}{2} \sum_{v \in V} \|\nabla_v f\|^p$ and μ is the trade-off parameters. The objective function is convex so that it has a unique solution. However, it is still strenuous work to approximate graph p-Laplacian so that extremely limits the applications of p-Laplacian regularization. Luo et al. [73] used the p-Laplacian for multi-class clustering. Inspired by Luo et al. [73], Liu *et al.* [74] proposed an efficient approximation method of graph p-Laplacian and built p-Laplacian regularization framework

$$\begin{array}{ll} \min\limits_{f} & J(\boldsymbol{f}) = \sum_k \frac{\sum_{ij} w_{ij} \left| f_i - f_j \right|^p}{\|f\|_p^p} \\ \text{s.t.} & \boldsymbol{f}^T \boldsymbol{f} = I \end{array} \tag{74}$$

where $\boldsymbol{f} = \left[f(x_1), f(x_2), \ldots, f(x_n) \right]^T$. The results show the pLapR can fit the data exactly and extrapolates smoothly to unseen data with the geodesic distance. Especially, they proposed an efficient approximation method of graph p-Laplacian and built p-Laplacian regularization framework.

The work [75] introduced a new class of non-local p-Laplacian operators that interpolate between non-local Laplacian and infinity Laplacian. Graph-based p-Laplacian regularization has found further applications in semi-supervised learning and image processing [76]. Liu et al. [77] proposed p-Laplacian regularized sparse coding for human activity recognition. Ma *et al.* [78] presented an efficient and effective approximation algorithm of hyper-graph p-Laplacian and then proposed hypergraph p-Laplacian regularization (HpLapR) provided more potential to exploiting the local structure preserving. Ma et al. [79] developed an ensemble p-Laplacian regularization (EpLapR) to fully approximate the intrinsic manifold of the data distribution. According to the manifold regularization framework, the proposed EpLapR can be written as the following optimization problem:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \boldsymbol{f}^T \boldsymbol{L}_p \boldsymbol{f} \tag{75}$$

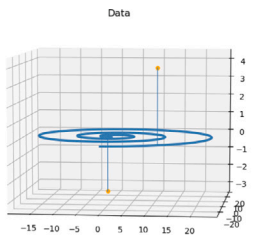
where $\boldsymbol{L}_p = \sum_{k=1}^{m} \mu_k \boldsymbol{L}_k^p$, and s.t. $\sum_{k=1}^{m} \mu_k = 1, \mu_k \geq 0$, for $k = 1, \ldots, m$.

Fig. 8 illustrates the differences of semi-supervised regression by using LapR, HesR, and pLapR for fitting two points on the 1-D spiral. Differences of semi-supervised regression by using LapR, HesR, and pLapR for fitting two points on the 1-D spiral. The LapR prefers to fit data as the constant function which seems improper. The HesR fits the data as linear function along the spiral thus it is better than LapR. The pLapR can fit the data exactly and more smooth [80].

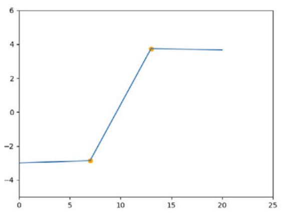
2.5. Discussion

This part will conclude how to choose a regularization for traditional machine learning.

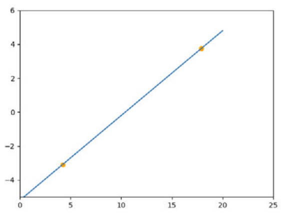
In the first subsection, the vector-based sparse regularizations are summarized. Sparse regularizations are important since many fields



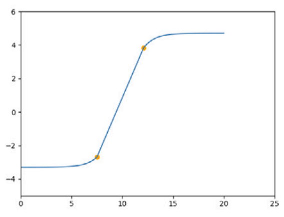
(a) data



(b) Laplacian



(c) Hessian



(d) p-Laplacian

Fig. 8. The differences of semi-supervised regression by using LapR, HesR, and pLapR for fitting two points on the 1-D spiral.

need to impose sparse constraints on variables, such as compressing sensing, feature selection, sparse signals separation, sparse PCA, and sparse signals separation. l_0 norm constrains the value of the weight to close to zero. Except for no penalty to zero, l_0 norm penalizes nonzero variables in the same way. As seen from images of regularizations, penalties function decrease rapidly as variables close to zero. Thus, the nonconvex penalties are closer to l_0 norm. Moreover, the common properties of these regularizations are as well proposed. It is worthwhile to discuss some good mathematical properties since having these properties may result in some positive characteristics, such as being more conducive to solving or being easier to obtain convergence. These images and ideas can be used to construct vector-based regularizations.

The second subsection summarized the matrix-based sparse regularizations which are important in large covariance matrix and inverse covariance matrix estimation. These regularizations are classified into three types: row sparsity, column sparsity, and column and row sparsity. Furthermore, two properties of these regularizations are proposed, which give the regularizations' binds. All of the above can make us choose or construct a regularization for specific tasks.

Some matrix-based low-rank regularizations are given in the third subsection, which can be applied in matrix completion and robust PCA. The low-rank regularization adds penalty to singular value, which is motivated by the vector-based sparse regularizations. Thus, a straightforward way to construct regularization can be proposed by extending the vector-based regularizations.

The last subsection reviewed the manifold regularizations. The manifold assumes the data has some inherent structure, which is widely used in computer vision and multimedia. The manifold learning prefers to find a more smooth model. This regularization captures the intuition that our functions should be smooth on the manifold, not just smooth in the extrinsic space. A natural way is to construct a regularization to measure the smoothness of f, such as $\int_M \|\nabla f(x)\|^2 dx$.

3. The regularization strategies in deep learning

There are many strategies for deep learning. Aiming at the reason which consists of noise data, the limited size of the training set, and the complexity of classifiers, this section main introduced the data augmentation, dropout, early stopping, batch normalization and several common methods to avoid the overfitting.

3.1. Data augmentation

One powerful way to improve model generalization performance is to train it on more data. In practice, the lack of a sufficient amount of training data or uneven class balance within the datasets are common problems. Data augmentation is based on an assumption that more information can be obtained from the new data after augmentations [81]. Dataset augmentation has been a particularly effective technique for image classification. For some image classification tasks, it is reasonable to create new fake data to add to the data set. In computer vision fields, there are some basic augmentation operations, which can be seen as a kind of oversampling [82–84]. This section also contains black-box approaches focused on deep neural networks, in addition to traditional white-box methods.

3.1.1. Traditional augmentation

The augmentation methods [85,86] include roughly: flipping, cropping, resizing, rotating, transposing, inverting, brightness, sharpness, equalize, auto contrast, convert and color balance. These methods have experimented on the bee image of hymenoptera data. The image pixels are 500 × 464. The result of these methods is shown in Fig. 9. Table 5 lists the methods and their own descriptions which are used to augment data.

Affine transformation is essentially a linear transformation of the image coordinate vector space, it could be formulated as:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{76}$$

in which $\boldsymbol{x}$ and $\boldsymbol{y}$ are 2-D image pixel coordinate vector before and after affine transformation, $\boldsymbol{W}$ and $\boldsymbol{b}$ are linear transformation matrix and translation vector respectively. The common affine-transformation-based image data augmentation methods include translation, rotation, and flipping, as illustrated in Fig. 9(c), 9(d), 9(e) respectively.

They have been proven to be easy, fast, repeatable, and dependable. However, some techniques result in the loss of image data. The disadvantages of color space transformations are increased memory, transformation costs, and training time. Moreover, color transformations may discard essential color details and thus are not always a label-preserving transformation. For example, when the pixel value of an image is reduced to simulate a darker environment, the objects in the image may be invisible.

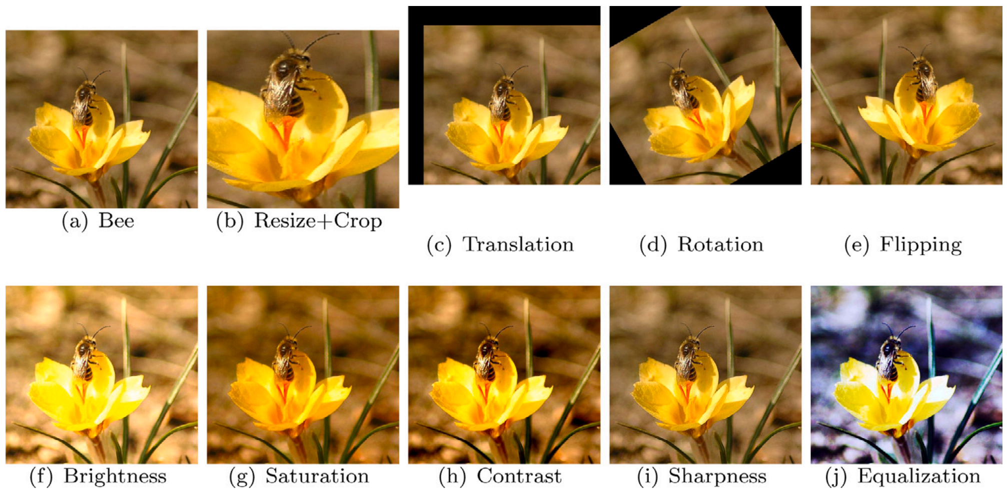


Fig. 9. Examples of image data augmentation methods.

Table 5
The methods used to augment data and their own descriptions.

Translate	Description
Crop	Crop the image by a box and it will reduce the size of input.
Resize	Zoom in or Zoom out the image and select the center of the scaled image.
Rotate	Rotate the image some degrees.
Transpose	Converted the image horizontally or vertically.
Invert	Invert the pixels of the image.
Brightness	Brightness enhancement and magnitude is proportional to brightness.
Sharpness	Sharpness enhancement and magnitude is proportional to sharpness.
Equalize	Equalize the image histogram.
Auto contrast	Maximize the contrast of the image.
Convert	Convert the mode of image.
Color balance	Adjust the color balance of the image. A $magnitude = 0$ gives a black and white image, while $magnitude = 1$ gives the original image.

It is worth noting that data augmentation approaches should take into account preserving the label's consistency post-transformation. Rotations and cropping, for example, are usually secure on ImageNet challenges like cat versus dog, but not on digit recognition tasks like 6 versus 9 [87].

3.1.2. Feature space augmentation

Data augmentation techniques have traditionally been applied to input images only. However, for a computer, there is no real difference between an input image and an intermediate representation. Feature space augmentation performs the transformation, not in input space, but in a learned feature space [88].

Shake-Shake regularization, which applies data augmentation techniques to internal representations, was created as an attempt to produce this sort of effect by stochastically 'blending' 2 viable tensors [89]. Shake-Shake can be applied to ResNeXt only. The work [90] proposed the ShakeDrop regularization which can be applied to more neural network such as ResNet, Wide ResNet, and PyramidNet. The work [91] dynamically adjusted the regularization strength in the training procedure, thereby balancing the underfitting and overfitting of CNNs.

Adding the difference between two examples to a new example is a simple yet effective data augmentation method [92]. SMOTE is a popular augmentation utilizing the k nearest neighbors to form new instances [93].

The use of auto-encoders is particularly useful for performing feature space augmentations on data [94]. An auto-encoder works by mapping one half of the network (encoder) to a low-dimensional vector representation, which the other half of the network (decoder) may reconstruct back to the original picture. For feature space augmentations, this encoded representation is used. The auto-encoder network can be used to recover new vectors and convert them into images as a solution to this problem. For deep CNNs, the proceeding is very difficult and time-consuming to train because it needs to copy the entire encoding part of the CNN.

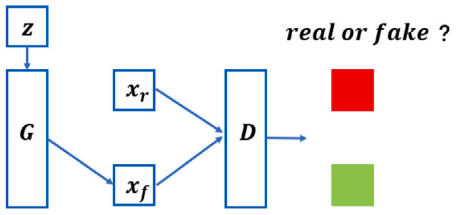


Fig. 10. GAN: G represents the generator, $\boldsymbol{z}$ represents the latent variable(random noise), $\boldsymbol{x}_r$ represents the real sample, $\boldsymbol{x}_f$ represents the fake sample which is generated by G, and D represents the discriminator which distinguishes the real or fake of $\boldsymbol{x}_f$ and $\boldsymbol{x}_r$.

3.1.3. Generative Adversarial Networks

Generative Adversarial Networks (GAN) gained considerable attention in the field of unsupervised learning. An example of GAN is shown in Fig. 10. The idea of GANs is to use two adversarial networks, $G(\boldsymbol{z})$ and $D(\boldsymbol{x})$, where one generates a photo-realistic image to fool the generator $G(\boldsymbol{z})$ to better distinguish fake images which are created by the discriminator $D(\boldsymbol{z})$.

Table 6 shows the recent progress of GAN.

Image data generation is one of the most frequently used fields of GAN. GAN generates an approximate real data distribution. When there are few real data, the performance of GAN is not ideal. Sample generation based on a poor data distribution may not be effective. GAN needs

Table 6
The GAN-variants.

Model	Description	Ref
CGAN	Conditional GAN: Constructed by merely extra auxiliary information (e.g., class label).	[150]
DCGAN	Deep Convolutional GAN: First structure of de-convolutional neural networks (de-CNN).	[151]
LapGAN	Laplacian GAN: Combine the CGAN with the framework of the Laplacian pyramid.	[152]
InfoGAN	Information Maximizing GAN: Learn disentangled design in a wholly unsupervised way.	[153]
WGAN	Wasserstein GAN: Its loss function derived through Earth-Mover or Wasserstein distance.	[154]
BEGAN	Boundary Equilibrium GAN: Keep-up an equilibrium between variety and superiority.	[155]
PGGAN	Progressive-Growing GAN: A multi-scale based GAN architecture.	[156]
BigGAN	BigGAN: Train bigger neural networks.	[157]
StyleGAN	Style-Based Generator Architecture for GAN: Control the image synthesis process.	[158]
EBGAN	Energy-Based GAN: Its discriminator works as an energy function.	[159]
RPDGAN	Realistic Painting Drawing GAN: An unsupervised cross-domain image translation framework.	[160]

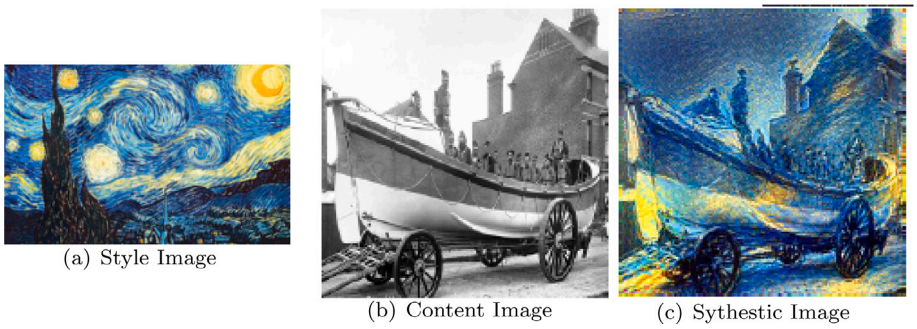


Fig. 11. Style Transfer: The style of van Gogh is applied to the content of Black-and-White image to generate an image which preserves the original content of (b) and has the style of (a) [99].

enough data to support the convergence of network training. When training data are not sufficient, GAN is difficult to achieve a satisfactory equilibrium, and it is easy to fall into mode collapse. Although the generated sample size has been expanded, it is not obviously helpful for the diversity of samples, which is similar to the simple replication of samples. For the downstream neural network, the model will be biased. At the same time, when dividing training, verification and test sets, data leakage may occur. For large scale data, there is generally no need to expand the sample or use simple data enhancement methods to expand the sample size. When the amount of data is medium, it seems that using GAN to generate samples and expand data sets is helpful to the training convergence of downstream neural networks. GAN's data augmentation may be more effective in some tasks with low resolution and acceptable definition. The work [95] proposed a training scheme that first uses classical data augmentation to enlarge the training set and then used GAN to synthetic data augmentation that enlarged the size and diversity of datasets. The work [96] used GAN to generate synthetic data with the purpose of training classifiers without using the original data set and oversampling a few class in unbalanced classification scenario.

3.1.4. *Neural style transfer*

Neural Style Transfer algorithms [97] can apply the artistic style of one image to another image while preserving its original content, which is shown in Fig. 11. It serves as a great tool for Data Augmentation which is somewhat analogous to color space lighting transformations for images. The style transfer algorithms consist of a descriptive approach and a generative approach. The descriptive approach refers to changing the pixels of a noise image in an iteration, and the generative approach uses a pre-trained model of the desired style to achieve the same effect in a single forward pass [98].

A model is trained in advance for each style image in the generative approach. Johnson et al. [99] proposed a two-component architecture-generator and loss networks. They also introduce a novel loss function based on a perceptual differences between the content and target. In [100], an approach was proposed to move the computational burden to a learning stage to improve the work of [99].

Some data augmentation strategies recently proposed methods follow a generative approach. In the work [101], Wang et al. explored Generative Adversarial Nets to generate images of different styles to augment the dataset. Zheng et al. generated two stylized images for each input image, then merged the stylized images and original images to compose the final training dataset [102]. A disadvantage of Neural Style Transfer Data Augmentation is the effort required to select styles to transfer images into. If the style set is too small, further biases could be introduced into the dataset.

3.1.5. *Meta-learning*

Meta-learning, also known as learning to learn, is a scientific approach to observe how different machine learning methods perform in a wide range of learning tasks, and then learning from this experience or meta data to learn new tasks faster than other possible methods [103]. In contrast to conventional machine learning, which solves tasks from scratch using a fixed learning algorithm, meta-learning attempts to refine the learning algorithm itself based on the experience of multiple learning episodes, aiming to learn on the basis of tasks rather than samples, that is, learning task-agnostic learning systems rather than task-specific models. Successful applications have been demonstrated in areas spanning few-shot image recognition [104], unsupervised learning [105], data efficient [106,107] and self-directed [108] reinforcement learning (RL), hyper-parameter optimization [109], and neural architecture search (NAS) [110].

Smart Augmentation is the process of learning suitable augmentations when training deep neural networks. Its goal is to learn the best augmentation strategy for a given class of input data. The work [111] used two networks, Network A and Network B. Network A is an augmentation network to generate new samples to train Network B. Network B can perform some specific tasks. The change in the error rate in Network B is then backpropagated to update Network A.

Auto-Augment is a Reinforcement Learning algorithm that searches for an optimal augmentation policy among a constrained set of geometric transformations with miscellaneous levels of distortions. The work [112] took the labeled images and predefined preprocessing

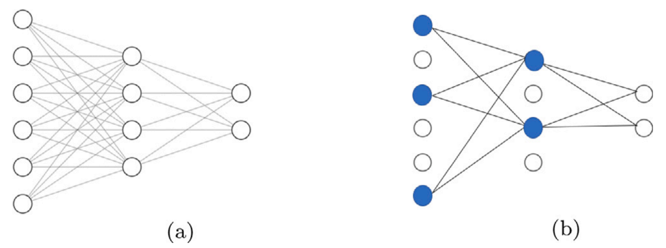


Fig. 12. An example of standard dropout. The left network is fully connected, and the right has had neurons dropped with probability 0.5.

transformations as the input set. It jointly learned the classifier and the optimal preprocessing transformations for individual images.

A disadvantage to meta-learning is that it is a relatively new concept and has not been heavily tested. Additionally, meta-learning schemes can be difficult and time-consuming to implement.

3.2. Dropout

Training a deep neural net with a large number of learnable parameters is very costly. Overfitting and long training time will become two fundamental challenges. Model compression can simplify model size or running time but maintain accuracy, that is, training fewer parameters without losing model accuracy. Dropout zeros out the activation values of randomly selected neurons during training in order to improve sparsity in neural network weights. This property means that dropout methods can be applied in compressing neural network models by reducing the number of parameters needed to perform effectively [113].

3.2.1. Standard dropout

Hinton et al. proposed the dropout method for the first time [114] and was subsequently applied to large scale visual recognition problems [115]. The main idea is that individual nodes are either kept with a probability of p or omitted from the network with a probability of $1-p$ in each training iteration. Dropout is not applied to the output layer. As shown in Fig. 12, on each presentation of each training case, each hidden unit is randomly omitted from the network with a probability of 0.5.

Mathematically, the behavior of standard dropout during training for a neural network layer is given by:

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x})\circ\boldsymbol{m}, \quad m_i \sim \text{Bernoulli}(p), \tag{77}$$

where $\boldsymbol{y}$ is the layer output, $f(\cdot)$ is the activation function, $\boldsymbol{W}$ is the layer weight matrix, $\boldsymbol{x}$ is the layer input, and $\boldsymbol{m}$ is the layer dropout mask, with each element m_i being 0 with probability $1-p$. Once trained, the layer output is given by:

$$\boldsymbol{y} = (1-p)f(\boldsymbol{W}\boldsymbol{x}). \tag{78}$$

Standard dropout is equivalent to adding layer after a layer of neurons that simply sets values to zero with some probability during training, and multiplies them by $1-p$ during testing.

The standard dropout promotes sparsity in the weights of neural networks, causing more weights to be near zero [115]. Several variants have been produced since the standard dropout was proposed, as shown in Table 7.

3.2.2. Dropconnect

One of the variations on standard dropout was Dropconnect [116]. Therefore, in training, the output of a network layer is given by:

$$\boldsymbol{y} = f((\boldsymbol{W}\circ\boldsymbol{M})\boldsymbol{x}), \quad M_{ij} \sim \text{Bernoulli}(p), \tag{79}$$

where $\boldsymbol{M}$ is a binary matrix encoding the connection information and $M_{ij} \sim \text{Bernoulli}(p)$. Dropconnect is only applicable to fully connected layers and randomly drop the weights rather than the activations.

3.2.3. Standout

To ensure unconfident units more frequently dropout than confident units, Standout overlays a binary belief network onto a neural network which controls the dropout properties of individual neurons [117]. The belief network is interpreted as tuning the architecture of the neural network. For each weight in the original neural network, Standout adds a corresponding weight parameter in the binary belief network. A layer's output during training is given by:

$$y = f(\boldsymbol{W}\boldsymbol{x})\circ\boldsymbol{m}, \quad m_i \sim \text{Bernoulli}\left(g\left(\boldsymbol{W}_s\boldsymbol{x}\right)\right), \tag{80}$$

where $\mathbf{W}_s$ representing the belief network's weights for that layer and $g(\cdot)$ representing the belief network's activation function.

An effective approach to determine belief network weights is setting them as:

$$\boldsymbol{W}_s = \alpha\boldsymbol{W} + \beta \tag{81}$$

at each training iteration for some constants α and β. The output of each layer during testing is given by:

$$y = f(\boldsymbol{W}\boldsymbol{x})\circ g\left(\boldsymbol{W}_s\boldsymbol{x}\right). \tag{82}$$

3.2.4. Curriculum dropout

In a traditional machine learning algorithm, all training examples are presented to the model in an unorganized fashion, with frequent random shuffling. The level of complexity of the concepts to learn in curriculum learning is proportional to the age of the people, i.e. handling easier knowledge when babies and harder knowledge when adults. Inspired by this, training examples can be subdivided based on their difficulty. Then, the learning is configured so that easier examples come first, eventually complicating them and processing the hardest ones at the end of training.

Curriculum Dropout was proposed to using a time schedule for the probability of retaining neurons in the network [118]. This results in an adaptive regularization scheme that dynamically increases the expected number of suppressed units to boost the generalization of the model while smoothly increasing the difficulty of the optimization problem.

Using a fixed dropout probability during training is proven to be a suboptimal choice. At the beginning of Curriculum Dropout, no entry of z_0 is set to zero. This clearly corresponds to the easiest available example and considers all possible available visual information. As learning time grows, a greater number of entries are set to zero. This complicates the challenge and necessitates a greater effort on the part of the model to capitalize on the limited amount of uncorrupted data available at that point in the training phase.

3.2.5. DropMaps

Moradi et al. proposed DropMaps, where for a training batch, each feature is kept with a probability of p and is dropped with the probability of $1-p$[119]. At test time, all feature maps are kept, and everyone is multiplied by p. Like Dropout, DropMaps has a regularization effect that causes the coincidence of feature maps to be avoided. DropMaps can handle the problem of overfitting large models for tiny images.

Table 7 shows some proposed methods and theoretical advances in dropout.

3.3. Early stopping

Actually, noisy labels are very common in real-world training data. Since model is overfitting to noisy labels in the training, the test data may have weak generalization. As shown in Fig. 13, the training error decreases steadily over time, but validation set error begins to rise again. A copy of the model parameters is saved every time the error on the validation set improves. When the training algorithm finishes, instead of using the most recent parameters, use these parameters as the result. Usually the model is stopped when the validation set error has not been improved for a while, rather than the model reaching

Table 7
Some proposed methods and theoretical advances in dropout.

Year	Methods
2012	Standard dropout
2013	Standout; Fast dropout; Dropconnect; Maxout
2014	Annealed dropout; DropAll
2015	Variational dropout; RNNdrop; MaxTpooling dropout; Spatial dropout
2016	Evolutionary dropout; Variational RNN dropout; Monte Carlo dropout; Selective CNN dropout; Swapout
2017	Cutout; Concrete dropout; Curriculum dropout; Tuneout.
2018	Adversarial dropout methods; Fraternal dropout; Information dropout; Targeted dropout; DropBlock; Jumpout
2019	Spectral dropout; Ising dropout; Weighted Channel Dropout; DropMaps
2020	Gradient Dropout; Surrogate Dropout
2021	Ranked dropout; LocalDrop

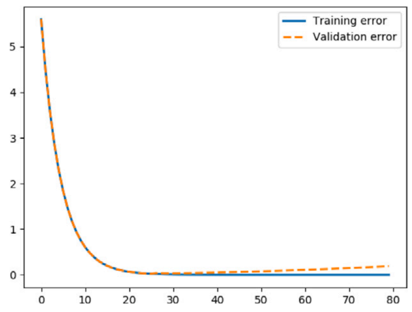


Fig. 13. The training and validation error curves.

the (local) minimum of the validation error. This strategy is known as early stopping. Early stopping is widely used because it is simple to understand and implement and has been reported to be superior to regularization methods in many cases. Early stopping has proven to be a quite effective method for avoiding overfitting [120].

However, if the Early Stopping Rule (ESR) stops too late, the model will miss out faster training and eventually lose generality. If it stops too early, the model may miss out on the accuracy of the predictions.

There are two strategies for the ESR, primary rules and secondary rules. Primary rules are known to become true at some point in time during training and cannot evaluate prediction clearly, such as the Maximum Number of Epochs. Secondary rules are that cannot be guaranteed to fire sometimes but carry the potential to improve average accuracy and training speed.

To formally describe these rules, some definitions are introduced first. Let E be the objective function (error function) of the training algorithm, for example the mean squared error (MSE) [121]. Then $E_{tr}(t)$ is the average error per example over the training set, measured after epoch t. $E_{va}(t)$ is the corresponding error on the validation set and is used by the stopping criterion. $E_{te}(t)$ is the corresponding error on the test set, which characterizes the quality of the model resulting from training. E_{opt} is the optimal error.

3.3.1. Loss of Generality (GL)

Loss of Generality is an automatic rule adapted from manual stopping procedure. Stop training when the error of the validation dataset exceeds a certain threshold, which is the relatively minimal error observed so far [122].

The optimal error is defined as:

$$E_{opt}(t) = \min_{t' \le t} E_{\text{va}}\left(t'\right). \tag{83}$$

Then the loss of generality is defined as:

$$GL(t) = 100 \cdot \left(\frac{E_{\text{va}}(t)}{E_{\text{opt}}\ (t)} - 1\right). \tag{84}$$

A high generalization loss is one obvious candidate reason to stop training since it directly indicates overfitting. This leads the rule GL_α as:

$$GL_\alpha \ : \text{stop after first epoch } t \text{ with } GL(t) > \alpha. \tag{85}$$

The tested α was 1, 2, 3, 4 and 5.

3.3.2. Low Progress (LP)

In general, it is impossible to judge that the global minimum has already been reached or not from the start of the curve, i.e., whether a rise in the generalization error implies real overfitting or is only temporary. Furthermore, since the generalization loss may still be running, the stopping may be suppressed if the training error is still rapidly decreasing. It is assumed that overfitting will only be triggered when the error decreases slowly. The low progress rule (LP) fires when the improvements on the training error stall. Using a k length training strip to measure the training progress, the formulate of P_k is

$$P_k = 1000 \cdot \left(\frac{\sum_{t'=t-k+1}^{t} E_{tr}\left(t'\right)}{k \cdot \min_{t'=t-k+1}^{t} E_{tr}\left(t'\right)} - 1\right), \tag{86}$$

where n is divisible by k. It calculates that how much was the average training error during the strip larger than the minimum training error during the strip.

This simply means to consider how much has changed in the past k steps where k is proposed to set to 5. Low Progress is then defined as:

$$LP_\alpha \ : P_5(t) < \alpha. \tag{87}$$

The tested α was 1, 2, 3, 4 and 5.

3.3.3. Generality to Progress Ratio (PQ)

Accepting higher generalization loss when there is more development on the training set in the hopes of later changes on the validation set seems to be often preferable. The following formulate uses the quotient of generalization loss and progress:

$$PQ_\alpha \ : \frac{GL(t)}{P_5(t)} > \alpha \tag{88}$$

with setting α to 1, 2, 3, 4 and 5, and measure the validation error only at the end of each strip.

3.3.4. Consecutive loss in generality (UP)

While GL is a global method that takes the minimum of all of the observed samples, UP is a local version of GL. UP is defined as:

$$UP_k \ : \underbrace{E_{va}(t-k) < E_{va}(t-k+1) < \cdots < E_{va}(t)}_{k}. \tag{89}$$

The main idea of this definition is that when the validation error increases more than once during s consecutive strip (independent of how large the increases), it is considered that this indicates the beginning of the final overfitting. The advantage of UP criteria is that changes can be measured locally. During long training periods, the error must be allowed to remain at a much higher level than the previous minimum. Only any of these criteria cannot guarantee termination, so the maximum number of iterations is often set at the same time.

3.3.5. Steady-State Stop Training Trigger (SSSTT)

The basic concept of Steady-State Stop Training Trigger (SSSTT) is to stop training when the change in prediction is invisible in the residuals measured by the training data subset. Usually, the training data set is randomly sampled from 20% to 30% as a small subset. When the noise exceeds the learning effect then stop training [123].

3.3.6. High Noise Ratio (HNR)

However, it is observed that in some cases the training curves E_{tr} get noisier at the end indicating the attempt of the network to overfit. Motivated by the SSSTT rule, a novel rule is introduced to stop training before noise grows too high which is called the High Noise Ratio rule (HNR). The formulate is

$$HNR_k(t) = \frac{\sum_{i=1}^{k} E_{tr}(t-i) - 2\cdot E_{tr}(t-i-1) + E_{tr}(t-i-2)}{\sum_{j=1}^{k} E_{tr}(t-j)}. \tag{90}$$

The k value was set at 20, which is the best value from experience. Early stopping occurs when

$$HNR_\alpha : HNR_{20}(t) > \alpha \tag{91}$$

with setting α to 5, 10, 15, 20, and 25.

3.3.7. Others

A cross-validated stop is one technique, in which a network is registered and trained for a long time before being cross-validated at recorded epochs [124]. A novel early stopping criteria based on quickly calculated local statistics of computed gradients eliminates the need for a held-out validation collection entirely [125]. Song et al. resume training the early stopped network using a maximal safe set which maintains a collection of almost certainly true-labeled samples at each epoch at the beginning of the early stopping [126].

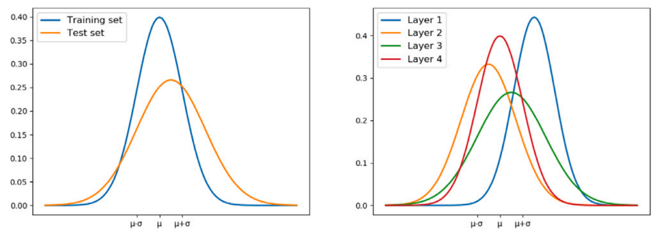
Early stopping is advantageous since it reduces the computational cost of training procedure. In addition to the obvious cost savings by minimizing the number of training iterations, it provides regularization without needing the addition of penalty terms to the cost function or the estimation of the gradients of those additional terms. Different ESR would have individual strengths and disadvantages. Early stopping may be used individually or in combination with other regularization techniques because computing the ESR is not very costly. Even when using regularization strategies that modify the objective function to encourage better generalization, it is rare for the best generalization to occur at a local minimum of the training objective.

3.4. Batch normalization

During the training period, the network parameters are continually changed. In addition to the data of the input layer, the update of the training parameters of the previous layer will cause the change of the input data distribution of the latter layer. The change of data distribution in the hidden layers is called 'Internal Covariate Shift' (Fig. 14). Ideally, each layer should be transformed into space where they have the same distribution but the functional relationship stays the same. Batch Normalization (BN) incorporates data standardization into the network architecture, avoiding this issue during the training process. This is achieved by adding the layers that set the first two moments (mean and variance) of the distribution of each activation to be zero and one respectively. Moreover, BN improves both convergence and generalization in training neural networks. BN is most widely used for visual fields, while it is limit in nature language processing (NLP). In NLP, the length of input sentences is different which makes it difficult to batch process.

During the forward pass, the batch normalization process as the follows:

$$\mu_B = \frac{1}{n}\sum_{i=1}^{n} x_i^{(\ell-1)}, \tag{92}$$



(a) Covariate shift (b) Internal covariate shift

Fig. 14. Covariate shift vs. Internal covariate shift.

and

$$\sigma_B^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^{(\ell-1)} - \mu_B\right)^2 \tag{93}$$

are the mean and variance, respectively. Then,

$$\hat{x}_i^{(\ell-1)} = \frac{x_i^{(\ell-1)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{94}$$

$$x_i^{(\ell)} = \gamma^{(\ell)}\hat{x}_i^{(\ell-1)} + \beta^{(\ell)}. \tag{95}$$

where μ_B is the batch mean and σ_B^2 the batch variance. The learned scale and shift parameters are denoted by γ and β, respectively. Any layer that previously received x as the input, now receives $f_{BN}(x)$. The difference between the approaches are how to calculate the mean μ_i and variance σ_i^2. The work [127] considered a mini-batch batch normalization which replaces n by m in Eqs. (92) and (93).

Another problem is the covariate shift which frequently occurs in real-world problems (Fig. 14). A common covariate shift problem is the difference in the distribution of the training and test set which can lead to suboptimal generalization performance. At test time, μ and σ of each set of mini-batch training data in each layer of the network are retained. The batch mean and variance are replaced by the statistics of all samples which, specifically, uses unbiased estimates of mean and variance:

$$\mu_{\text{test}} = \mathbb{E}\left(\mu_{\text{batch}}\right), \quad \sigma_{\text{test}}^2 = \frac{m}{m-1}\mathbb{E}\left(\sigma_{\text{batch}}^2\right). \tag{96}$$

where $\mathbb{E}$ is expectation.

One drawback of BN is its crucial reliance on batch size since a single batch influence the calculation of the approximation of the mean and variance. Some works suggest that this restriction has not been completely eliminated, but it has been alleviated, which will be further discussed below.

3.4.1. Layer Normalization

Layer normalization (LN) [128] normalizes the inputs crossing the features, while batch normalization normalizes the input features crossing the batch dimension. In batch normalization, the statistics are computed across the batch and are the same for each example in the batch. BN is for the output value of the same dimension of multiple samples. In contrast, layer normalization is for the output value of each dimension of a sample where the statistics are computed across each feature and independent of other samples.

Compared with BN, LN is that all hidden units in a layer share same normalization terms μ and σ, but different training cases have different normalization terms. LN has no restrictions on the size of a mini-batch and can be used in the pure online regime of batch size 1.

In BN, there will be a problem that some channels do not have certain sample data, even if specific character padding is used, it will cause uneven distribution of certain channels. The forward normalization is the only decisive factor to LN which makes the input distribution more stable, thus brings better convergence. The result of [129] shows that forward normalization has little to do with the effectiveness and the derivatives of the mean and variance play a significant role in LN.

Table 8
For each normalization, N is the batch axis, C is the channel axis, and (H, W) is the spatial axes. The pixel in blue are normalized by the sam mean and variance, computed by aggregating the values of these pixel.

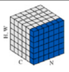
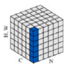
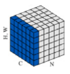
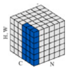
Item	Content	Mean and variant	Image
BN	BN normalizes the data in each batch.	$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}$ $\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_c(x)\right)^2 + \epsilon}$	
IN	IN is a separate normalization operation for each channel in a sample.	$\mu_{nc}(x) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}$ $\sigma_{nc}(x) = \sqrt{\frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_{nc}(x)\right)^2 + \epsilon}$	
LN	LN normalizes all data in a sample.	$\mu_n(x) = \frac{1}{CHW} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}$ $\sigma_n(x) = \sqrt{\frac{1}{CHW} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_n(x)\right)^2 + \epsilon}$	
GN	GN divides the channel of a sample into multiple groups, then normalized each group.	$\mu_{ng}(x) = \frac{1}{(C/G)HW} \sum_{c=gC/G}^{(g+1)C/G} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw}$ $\sigma_{ng}(x) = \sqrt{\frac{1}{(C/G)HW} \sum_{c=gC/G}^{W} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(x_{nchw} - \mu_{ng}(x)\right)^2 + \epsilon}$	

Table 9
Some normalization methods.

Model	Description	Ref
Weight normalization	It normalizes the weights of the layer.	[161]
Batch-Instance Normalization	It tries to learn how much style information should be used for each channel.	[162]
Decorrelated Batch Normalization	It centers, scales and whitens activations of each layer.	[163]
Iterative Normalization	It employs Newton's iterations for much more efficient whitening and avoids the eigen-decomposition.	[164]
Batch Renormalization	The model outputs are dependent only on the individual examples during both training and inference.	[165]
Switchable Normalization	It selects different normalizers for different normalization layers of a deep neural network.	[166]
Differentiable Dynamic Normalization	It learns arbitrary normalization forms in data- and task-driven way for deep neural networks.	[167]

3.4.2. Instance Normalization

Instance Normalization (IN) is a separate normalization operation for each channel in a sample [130]. LN and IN are very similar to each other but the difference between them is that IN normalizes across each channel in each training example instead of normalizing across input features in a training example. Unlike BN, the instance normalization layer is applied at test time as well because of the non-dependency of mini-batch [131].

3.4.3. Group normalization

Group Normalization (GN) is a simple and effective alternative to batch normalization. Since inaccurate batch statistics estimation, the BN error grows exponentially as the batch size decreases. This problem limits the application of BN to computer vision tasks including video, detection, and segmentation which require small batches constrained by memory consumption. Unlike BN, the accuracy of GN is stable over a wide range of batch sizes, and its computation is independent of batch size.

GN normalizes over the group of channels for each training example, that is Group Norm is in between IN and LN [132]. GN divides the channels into groups and computes the mean and variance for normalization within each group. The computation of GN is independent of batch sizes, and its precision is stable in a wide variety of batch sizes. When each channels is seen as a single group, GN is transformed into LN. When each channel is separated into separate classes, GN is transformed into IN.

Table 8 shows the comparison of four normalization methods which mentioned above. Table 9 shows many other existing methods of normalization.

3.5. Other

3.5.1. Pretraining

This course is so long in deep networks that the gradient signal becomes very small at the network's front. On the other hand, updates in later layers are done based on the weights of former layers that are going to be changed. These effects slow the convergence process. An excellent way to avoid this hassle is to pre-train the network in an unsupervised manner, layer by layer. The identification feature is thought to each layer from bottom to top in this technique. Weights find a favorable point in parameter space in this manner, resulting in regularizing effects during the fine-tuning process.

Table 10
The main applications of the regularization technologies in deep learning.

Technology	Main application
Data augmentation	Computer vision
Dropout	Model compress
Early stopping	Label noise
Batch normalization	Computer vision
Pre-training	Multi-task learning
Weight share	Model compress
Multi-task learning	Multi-task learning
Adding noise	Label noise

The work proposed a pre-training parameter method based on unsupervised feature learning to solve the problem of gradient instability [133]. The original data is abstracted into highly conceptual and independent features. After pre-training of network parameters, the gradient descent algorithm is used to fine-tuning the parameters for specific classification tasks. Common pre-training methods to initialize parameters are restricted Boltzmann machine (RBN) [134], autoencoder [135] and sparse encoding symmetric machine (SESM) [136].

3.5.2. Weight share

Reusing trainable parameters in multiple parts of the network is called weight sharing, which makes the model less complex [137]. Using less weights makes model simple if the weights have high probability density under the mixture model. Clustering the weights into subsets, even if only one, to reduce the number of sets with weights that have very close values in each cluster. Since the effective mean or variance of the clustering is usually unknown in advance, it is permissible to change the parameters of the mixed model concurrently during the training stage.

A famous example is CNN. Weight sharing does not merely reduce the number of weights, but also encodes the prior knowledge about the locality of feature extraction.

Table 11
The advantages and disadvantages of the regularization technologies in deep learning.

Technology	Advantages	Disadvantages
Data augmentation	Obtains a smaller robust error	Induces the data distribution bias
Dropout	Prevents units from co-adapting	Increases the training time
Early stopping	No change to model/algorithm	Need validation data
Batch normalization	Removes dropout and increased accuracy	Difficult to estimate μ and σ in the test.

Table 12
Model accuracy, number of epochs the model needs to be learned and the number of operation per input sample during the training phase [5].

Method	None	l_2 norm	Dropout	Data augmentation	Batch normalization	Adding noise
Accuracy	82.1	88.3	90.6	88.1	90.8	84.5
Epochs	12	12	28	17	6	16
Training Ops/Sam (M)	230	232	118	232	420	231

3.5.3. Multi-task learning

A special type of regularization technology is multi-task learning (MTL). Most multi-task data can be collected from diverse domains or various tasks. MTL helps the relationship among tasks. MTL improves the performance of all tasks by learning multiple tasks simultaneously [138].

MTL consists feature learning approach, task clustering approach and task relation learning approach. The feature learning approach assumes that related tasks are in a common feature subspace [139]. The task clustering approach assumes that the tasks form several clusters where the model is trained by task groups [140]. Task relation learning approach assumes more complex relationships among tasks which usually learns pairwise relations directly among tasks, such as task similarity [141], task correlations [142], and task covariance [143].

3.5.4. Adding noise

Adding noise to a model or weight may be used to make structures more generalizable and avoid overfitting [144,145]. In deep learning, noise can be added either to the input data or to the weights of the network. Adding noise to the input data has a relatively long history in data pre-processing and in some literature, it is referred to as dithering. In deep learning, adding noise to the input data hinders memorizing but preserves learnability. The idea of adding noise has similar effects on different problems in machine learning.

3.6. Discussion

This part will compare the mentioned regularization technologies and conclude existing opportunities and challenges of the regularizations in deep learning.

While simple to state, these comparisons might have profound implications:

- The main applications of these regularization technologies are shown in Table 10.
- The advantages and disadvantages of the regularization technologies in deep learning are discussed in Table 11.

Regularization technologies have rapidly become an essential part of the deep learning toolkit. However, a fundamental problem that has accompanied this rapid progress is a lack of theoretical proof. In particular, it is unclear how these technologies improve generalization, even unknown whether they achieved better generalization.

The lack of these theories greatly limits the development of regularization technologies. However, a recent work thinks BN makes the optimization landscape significantly smoother instead of avoiding covariate shift [146]. Specifically, the Lipschitzness of both the loss and the gradients induce a more predictive and stable behavior of the gradients, allowing for faster training. Regularizations with good properties should be preferred.

The combination of two or more regularization techniques may lead to a great effect, but others might make '1 + 1 < 1'. For example, Li et al. [147] reconciles dropout and batch normalization, which reduces the error rate in those applications. It recommends that apply dropout after batch normalization with a small dropout rate. While the work [148] shows that l_2 regularization has no regularizing effect when combined with normalization. Therefore, the combination of regularization technologies plays a fundamental role in regularization research.

Computational cost is a factor in the regularization selection. Table 12 showed the model accuracy, the number of convergent epochs and the number of operation per input sample of some regularization technologies [5]. As can be seen, weight decay and data augmentation have little computational side effects. Therefore, they can be used in most applications. In the case of enough computational resources, the related methods of Dropout are reasonable to be used. Moreover, in the case of abundant computing resources, batch normalization family methods are reasonable strategies used as regularization in the network.

4. Conclusion

In this paper, we reviewed favorite regularization techniques in recent years. In machine learning, this paper introduced sparse regularization, low-rank regularization, and manifold regularization. In deep learning, data augmentation, dropout, early stopping, batch normalization are seen as the common regularization strategies. This paper summarized how to choose the regularization technologies as follows:

- For sparse regularization, it is meaningful to construct a regularization that has good mathematical properties for specific tasks.
- The low-rank regularization can be constructed by extending the sparse regularizations.
- Choose the regularization technologies that lead to the optimization properties.
- Manifold regularization should guarantee the smoothness of the learning function.
- Computational cost can be considered as a factor when choosing regularizations.

It seems the trend of employing more effective regularization techniques will continue and we will experience better and smarter methods in the near future.

Several findings of this article, as well as potential future research, are summarized below: (1) Developing more effective regularization strategies has been the subject of significant research efforts. (2) The advancement of regularization technologies is severely hampered by the lack of theoretical clarification. In particular, it is unclear how these regularization technologies improved generalization, and it is also unclear whether they achieved better generalization. (3) The combination of two or more regularization techniques may lead to a great effect, but others might make '1 + 1 < 1'. Therefore, taking into account the combination of regularization technologies is important in regularization research.

CRediT authorship contribution statement

Yingjie Tian: Funding acquisition, Resources, Writing – reviewing and editing, Conceptualization, Supervision. **Yuqi Zhang:** Writing-original draft, Visualization, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been partially supported by grants from: National Natural Science Foundation of China (No. 12071458, 71731009).

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[2] J. Kukavcka, V. Golkov, D. Cremers, Regularization for deep learning: A taxonomy, 2017, arXiv preprint arXiv:1710.10686.
[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Commun. ACM 64 (3) (2021) 107–115.
[4] Z. Hu, F. Nie, R. Wang, X. Li, Low rank regularization: a review, Neural Netw. (2020).
[5] R. Moradi, R. Berangi, B. Minaei, A survey of regularization strategies for deep models, Artif. Intell. Rev. 53 (6) (2020) 3947–3986.
[6] F. Wen, L. Chu, P. Liu, R.C. Qiu, A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning, IEEE Access 6 (2018) 69883–69906.
[7] L.C. Potter, E. Ertin, J.T. Parker, M. Cetin, Sparsity and compressed sensing in radar imaging.
[8] C.R. Berger, Z. Wang, J. Huang, S. Zhou, Application of compressive sensing to sparse channel estimation, IEEE Commun. Mag. 48 (11) (2010) 164–174.
[9] M. Lustig, D. Donoho, J.M. Pauly, Sparse MRI: The application of compressed sensing for rapid mr imaging, Magn. Reson. Med. 58 (6) (2007) 1182–1195.
[10] J. Yang, J. Wright, T.S. Huang, Y. Ma, Image super-resolution via sparse representation, IEEE Trans. Image Process. 19 (11) (2010) 2861–2873.
[11] X. Jiang, R. Ying, F. Wen, S. Jiang, P. Liu, An improved sparse reconstruction algorithm for speech compressive sensing using structured priors, in: 2016 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2016, pp. 1–6.
[12] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 267–288.
[13] M.B. McCoy, V. Cevher, Q.T. Dinh, A. Asaei, L. Baldassarre, Convexity in source separation: Models, geometry, and algorithms, IEEE Signal Process. Mag. 31 (3) (2014) 87–95.
[14] Z. Xu, X. Chang, F. Xu, H. Zhang, l1/2 regularization: A thresholding representation theory and a fast solver, IEEE Trans. Neural Netw. Learn. Syst. 23 (7) (2012) 1013–1027.
[15] N. Parikh, S. Boyd, et al., Proximal algorithms, Found. Trends R Optimiz. 1 (2014).
[16] J. Fan, R. Li, Variable selection via nonconcave penalized likelihood and its oracle properties, J. Amer. Statist. Assoc. 96 (456) (2001) 1348–1360.
[17] C.-H. Zhang, Nearly unbiased variable selection under minimax concave penalty, Ann. Statist. 38 (2) (2010) 894–942.
[18] N. Anantrasirichai, R. Zheng, I. Selesnick, A. Achim, Image fusion via sparse regularization with non-convex penalties, Pattern Recognit. Lett. 131 (2020) 355–360.
[19] T. Zhang, Analysis of multi-stage convex relaxation for sparse regularization., J. Mach. Learn. Res. 11 (3) (2010).
[20] J. Wangni, D. Lin, Learning sparse visual representations with leaky capped norm regularizers, 2017, arXiv preprint arXiv:1711.02857.
[21] H.-Y. Gao, A.G. Bruce, Waveshrink with firm shrinkage, Statist. Sinica (1997) 855–874.
[22] I. Selesnick, M. Farshchian, Sparse signal approximation via nonseparable regularization, IEEE Trans. Signal Process. 65 (10) (2017) 2561–2575.
[23] M. Nikolova, Energy Minimization Methods, Springer, 2011.
[24] D. Geman, G. Reynolds, Constrained restoration and the recovery of discontinuities, IEEE Trans. Pattern Anal. Mach. Intell. 14 (3) (1992) 367–383.
[25] I.W. Selesnick, I. Bayram, Sparse signal estimation by maximally sparse convex optimization, IEEE Trans. Signal Process. 62 (5) (2014) 1078–1092.
[26] M. Malek-Mohammadi, C.R. Rojas, B. Wahlberg, A class of nonconvex penalties preserving overall convexity in optimization-based mean filtering, IEEE Trans. Signal Process. 64 (24) (2016) 6650–6664.
[27] J. Fan, Y. Liao, H. Liu, An overview of the estimation of large covariance and precision matrices, Econom. J. 19 (1) (2016) C1–C32.
[28] J. Fan, F. Han, H. Liu, Challenges of big data analysis, Natl. Sci. Rev. 1 (2) (2014) 293–314.
[29] R.C. Qiu, P. Antonik, Smart Grid using Big Data Analytics: A Random Matrix Theory Approach, John Wiley and Sons, 2017.
[30] H. Liu, L. Wang, T. Zhao, Sparse covariance matrix estimation with eigenvalue constraints, J. Comput. Graph. Statist. 23 (2) (2014) 439–459.
[31] D. Belomestny, M. Trabs, A.B. Tsybakov, Sparse covariance matrix estimation in high-dimensional deconvolution, Bernoulli 25 (3) (2019) 1901–1938.
[32] X. Liu, N. Zhang, Sparse inverse covariance matrix estimation via the-norm with tikhonov regularization, Inverse Problems 35 (11) (2019) 115010.
[33] C. Ding, D. Zhou, X. He, H. Zha, R 1-pca: rotational invariant l 1-norm principal component analysis for robust subspace factorization, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 281–288.
[34] N. Wang, Y. Xue, Q. Lin, P. Zhong, Structured sparse multi-view feature selection based on weighted hinge loss, Multimedia Tools Appl. 78 (11) (2019) 15455–15481.
[35] H. Liu, M. Palatucci, J. Zhang, Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 649–656.
[36] P. Gong, J. Ye, C. Zhang, Multi-stage multi-task feature learning, J. Mach. Learn. Res. 14 (1) (2013) 2979–3010.
[37] S. Wang, D. Liu, Z. Zhang, Nonconvex relaxation approaches to robust matrix recovery, in: Twenty-Third International Joint Conference on Artificial Intelligence, 2013.
[38] E.J. Candes, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (6) (2009) 717–772.
[39] R. Mazumder, T. Hastie, R. Tibshirani, Spectral regularization algorithms for learning large incomplete matrices, J. Mach. Learn. Res. 11 (2010) 2287–2322.
[40] T. Hastie, R. Mazumder, J.D. Lee, R. Zadeh, Matrix completion and low-rank SVD via fast alternating least squares, J. Mach. Learn. Res. 16 (1) (2015) 3367–3402.
[41] T. Bouwmans, E.H. Zahzah, Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance, Comput. Vis. Image Underst. 122 (2014) 22–34.
[42] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, R. Otazo, On the applications of robust PCA in image and video processing, Proc. IEEE 106 (8) (2018) 1427–1457.
[43] L. Luo, S. Bao, C. Tong, Sparse robust principal component analysis with applications to fault detection and diagnosis, Ind. Eng. Chem. Res. 58 (3) (2019) 1300–1309.
[44] E. Kim, M. Lee, S. Oh, Elastic-net regularization of singular values for robust subspace learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 915–923.
[45] F. Nie, H. Huang, C. Ding, Low-rank matrix recovery via efficient schatten p-norm minimization, in: Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012.
[46] L. Liu, W. Huang, D.-R. Chen, Exact minimum rank approximation via schatten p-norm minimization, J. Comput. Appl. Math. 267 (2014) 218–227.
[47] S. Gu, L. Zhang, W. Zuo, X. Feng, Weighted nuclear norm minimization with application to image denoising, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2862–2869.
[48] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, L. Zhang, Weighted schatten p-norm minimization for image denoising and background subtraction, IEEE Trans. Image Process. 25 (10) (2016) 4842–4857.
[49] C. Lu, J. Tang, S. Yan, Z. Lin, Generalized nonconvex nonsmooth low-rank minimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 4130–4137.
[50] C. Peng, Z. Kang, H. Li, Q. Cheng, Subspace clustering using log-determinant rank approximation, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 925–934.
[51] C. Gao, N. Wang, Q. Yu, Z. Zhang, A feasible nonconvex relaxation approach to feature selection, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2011, 25(1).
[52] J.H. Friedman, Fast sparse regression and classification, Int. J. Forecast. 28 (3) (2012) 722–738.
[53] D. Geman, C. Yang, Nonlinear image recovery with half-quadratic regularization, IEEE Trans. Image Process. 4 (7) (1995) 932–946.
[54] J. Trzasko, A. Manduca, Highly undersampled magnetic resonance image reconstruction via homotopic l0-minimization, IEEE Trans. Med. Imaging 28 (1) (2008) 106–121.

[55] C. Lu, C. Zhu, C. Xu, S. Yan, Z. Lin, Generalized singular value thresholding, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2015, vol. 29(1).
[56] Y. Zhang, M. Rabbat, A graph-cnn for 3d point cloud classification, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 6279–6283.
[57] D. Yang, W. Gao, Pointmanifold: Using manifold learning for point cloud classification, 2020, arXiv preprint arXiv:2010.07215.
[58] J. Zeng, G. Cheung, M. Ng, J. Pang, C. Yang, 3D Point cloud denoising using graph Laplacian regularization of a low dimensional manifold model, IEEE Trans. Image Process. 29 (2019) 3474–3489.
[59] X. Ma, W. Liu, Recent advances of manifold regularization, in: Manifolds II-Theory and Applications, IntechOpen, 2018.
[60] Y. Zhang, J. Wu, Z. Cai, S.Y. Philip, Multi-view multi-label learning with sparse feature selection for image annotation, IEEE Trans. Multimed. 22 (11) (2020) 2844–2857.
[61] C. Shi, Q. Ruan, G. An, C. Ge, Semi-supervised sparse feature selection based on multi-view Laplacian regularization, Image Vis. Comput. 41 (2015) 1–10.
[62] Y. Li, X. Shi, C. Du, Y. Liu, Y. Wen, Manifold regularized multi-view feature selection for social image annotation, Neurocomputing 204 (2016) 135–141.
[63] B. Geng, D. Tao, C. Xu, L. Yang, X.-S. Hua, Ensemble manifold regularization, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1227–1233.
[64] X. Ma, D. Tao, W. Liu, Effective human action recognition by combining manifold regularization and pairwise constraints, Multimedia Tools Appl. 78 (10) (2019) 13313–13329.
[65] W. Liu, D. Tao, J. Cheng, Y. Tang, Multiview Hessian discriminative sparse coding for image annotation, Comput. Vis. Image Underst. 118 (2014) 50–60.
[149] D. Tao, L. Jin, W. Liu, X. Li, Hessian regularized support vector machines for mobile image annotation on the cloud, IEEE Trans. Multimed. 15 (4) (2013) 833–844.
[66] K.I. Kim, F. Steinke, M. Hein, Semi-supervised regression using hessian energy with an application to semi-supervised dimensionality reduction, MPI for Biological Cybernetics, 2010.
[67] W. Liu, H. Liu, D. Tao, Y. Wang, K. Lu, Multiview hessian regularized logistic regression for action recognition, Signal Process. 110 (2015) 101–107.
[68] G. Feng, W. Liu, S. Li, D. Tao, Y. Zhou, Hessian-regularized multitask dictionary learning for remote sensing image recognition, IEEE Geosci. Remote Sens. Lett. 16 (5) (2018) 821–825.
[69] S. Lefkimmiatis, A. Bourquard, M. Unser, Hessian-based norm regularization for image restoration with biomedical applications, IEEE Trans. Image Process. 21 (3) (2011) 983–995.
[70] W. Liu, D. Tao, Multiview hessian regularization for image annotation, IEEE Trans. Image Process. 22 (7) (2013) 2676–2687.
[71] T. Buhler, M. Hein, Spectral clustering based on the graph p-Laplacian, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 81–88.
[72] D. Zhou, B. Scholkopf, Regularization on discrete spaces, in: Joint Pattern Recognition Symposium, Springer, 2005, pp. 361–368.
[73] D. Luo, H. Huang, C. Ding, F. Nie, On the eigenvectors of p-Laplacian, Mach. Learn. 81 (1) (2010) 37–51.
[74] W. Liu, X. Ma, Y. Zhou, D. Tao, J. Cheng, p-Laplacian regularization for scene recognition, IEEE Trans. Cybern. 49 (8) (2018) 2927–2940.
[75] A. Elmoataz, X. Desquesnes, O. Lezoray, Non-local morphological PDEs and p-Laplacian equation on graphs with applications in image processing and machine learning, IEEE J. Sel. Top. Sign. Proces. 6 (7) (2012) 764–779.
[76] A. Elmoataz, F. Lozes, M. Toutain, Nonlocal pdes on graphs: From tug-of-war games to unified interpolation on images and point clouds, J. Math. Imaging Vision 57 (3) (2017) 381–401.
[77] W. Liu, Z.-J. Zha, Y. Wang, K. Lu, D. Tao, p-Laplacian regularized sparse coding for human activity recognition, IEEE Trans. Ind. Electron. 63 (8) (2016) 5120–5129.
[78] X. Ma, W. Liu, S. Li, D. Tao, Y. Zhou, Hypergraph p-Laplacian regularization for remotely sensed image recognition, IEEE Trans. Geosci. Remote Sens. 57 (3) (2018) 1585–1595.
[79] X. Ma, W. Liu, D. Tao, Y. Zhou, Ensemble p-laplacian regularization for scene image recognition, Cogn. Comput. 11 (6) (2019) 841–854.
[80] D. Slepcev, M. Thorpe, Analysis of p-laplacian regularization in semisupervised learning, SIAM J. Math. Anal. 51 (3) (2019) 2085–2120.
[81] C. Sun, A. Shrivastava, S. Singh, A. Gupta, Revisiting unreasonable effectiveness of data in deep learning era, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 843–852.
[82] J. Lu, P. Gong, J. Ye, C. Zhang, Learning from very few samples: A survey, 2020, arXiv preprint arXiv:2009.02653.
[83] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, 2014, arXiv preprint arXiv:1405.3531.
[84] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, Springer, 2014, pp. 818–833.
[85] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q.V. Le, Autoaugment: Learning augmentation strategies from data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 113–123.
[86] C. Lei, B. Hu, D. Wang, S. Zhang, Z. Chen, A preliminary study on data augmentation of deep learning for image classification, in: Proceedings of the 11th Asia-Pacific Symposium on Internetware, 2019, pp. 1–6.
[87] H. Bagherinezhad, M. Horton, M. Rastegari, A. Farhadi, Label refinery: Improving imagenet classification through label progression, 2018, arXiv preprint arXiv:1805.02641.
[88] T. DeVries, G.W. Taylor, Dataset augmentation in feature space, 2017, arXiv preprint arXiv:1702.05538.
[89] X. Gastaldi, Shake-shake regularization, 2017, arXiv preprint arXiv:1705.07485.
[90] Y. Yamada, M. Iwamura, T. Akiba, K. Kise, Shakedrop regularization for deep residual learning, IEEE Access 7 (2019) 186126–186136.
[91] Y. Wang, Z.-P. Bian, J. Hou, L.-P. Chau, Convolutional neural networks with dynamic regularization, IEEE Trans. Neural Netw. Learn. Syst. 32 (5) (2020) 2299–2304.
[92] V. Kumar, H. Glaude, C. de Lichy, W. Campbell, A closer look at feature space data augmentation for few-shot intent classification, 2019, arXiv preprint arXiv:1910.04176.
[93] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[94] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. Big Data 6 (1) (2019) 1–48.
[150] M. Mirza, S. Osindero, Conditional generative adversarial nets, 2014, arXiv preprint arXiv:1411.1784.
[151] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.
[152] E. Denton, S. Chintala, A. Szlam, R. Fergus, Deep generative image models using a laplacian pyramid of adversarial networks, 2015, arXiv preprint arXiv:1506.05751.
[153] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 2180–2188.
[154] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.
[155] D. Berthelot, T. Schumm, L. Metz, Began: Boundary equilibrium generative adversarial networks, 2017, arXiv preprint arXiv:1703.10717.
[156] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, 2017, arXiv preprint arXiv:1710.10196.
[157] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, 2018, arXiv preprint arXiv:1809.11096.
[158] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.
[159] J. Zhao, M. Mathieu, Y. LeCun, Energy-based generative adversarial network, 2016, arXiv preprint arXiv:1609.03126.
[160] X. Gao, Y. Tian, Z. Qi, Rpd-gan: Learning to draw realistic paintings with generative adversarial network, IEEE Trans. Image Process. 29 (2020) 8706–8720.
[95] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, H. Greenspan, Synthetic data augmentation using GAN for improved liver lesion classification, in: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), IEEE, 2018, pp. 289–293.
[96] F.H.K.d.S. Tanaka, C. Aranha, Data augmentation using GANs, 2019, arXiv preprint arXiv:1904.09135.
[97] L.A. Gatys, A.S. Ecker, M. Bethge, A neural algorithm of artistic style, 2015, arXiv preprint arXiv:1508.06576.
[98] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, M. Song, Neural style transfer: A review, IEEE Trans. Vis. Comput. Graphics 26 (11) (2019) 3365–3385.
[99] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European Conference on Computer Vision, Springer, 2016, pp. 694–711.
[100] D. Ulyanov, V. Lebedev, A. Vedaldi, V.S. Lempitsky, Texture networks: Feed-forward synthesis of textures and stylized images, in: ICML, vol. 1(2), 2016, pp. 4.
[101] L. Perez, J. Wang, The effectiveness of data augmentation in image classification using deep learning, 2017, arXiv preprint arXiv:1712.04621.
[102] X. Zheng, T. Chalasani, K. Ghosal, S. Lutz, A. Smolic, Stada: Style transfer as data augmentation, 2019, arXiv preprint arXiv:1909.01056.
[103] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: A survey, 2020, arXiv preprint arXiv:2004.05439.
[104] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.
[105] L. Metz, N. Maheswaranathan, B. Cheung, J. Sohl-Dickstein, Meta-learning update rules for unsupervised representation learning, 2018, arXiv preprint arXiv:1804.00222.

[106] Y. Duan, J. Schulman, X. Chen, P.L. Bartlett, I. Sutskever, P. Abbeel, Rl2: Fast reinforcement learning via slow reinforcement learning, 2016, arXiv preprint arXiv:1611.02779.
[107] R. Houthooft, R.Y. Chen, P. Isola, B.C. Stadie, F. Wolski, J. Ho, P. Abbeel, Evolved policy gradients, 2018, arXiv preprint arXiv:1802.04821.
[108] F. Alet, M.F. Schneider, T. Lozano-Perez, L.P. Kaelbling, Meta-learning curiosity algorithms, 2020, arXiv preprint arXiv:2003.05325.
[109] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, M. Pontil, Bilevel programming for hyperparameter optimization and meta-learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 1568–1577.
[110] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, 2018, arXiv preprint arXiv:1806.09055.
[111] J. Lemley, S. Bazrafkan, P. Corcoran, Smart augmentation learning an optimal data augmentation strategy, Ieee Access 5 (2017) 5858–5869.
[112] T.N. Minh, M. Sinn, H.T. Lam, M. Wistuba, Automated image data preprocessing with deep reinforcement learning, 2018, arXiv preprint arXiv:1806.05886.
[113] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[114] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.
[115] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.
[116] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International Conference on Machine Learning, PMLR, 2013, pp. 1058–1066.
[117] J. Ba, B. Frey, Adaptive dropout for training deep neural networks, Adv. Neural Inf. Process. Syst. 26 (2013) 3084–3092.
[118] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, Curriculum dropout.
[119] R. Moradi, R. Berangi, B. Minaei, Sparsemaps: convolutional networks with sparse feature maps for tiny image classification, Expert Syst. Appl. 119 (2019) 142–154.
[120] A. Lodwich, Y. Rangoni, T. Breuel, Evaluation of robustness and performance of early stopping rules with multi layer perceptrons, in: 2009 International Joint Conference on Neural Networks, IEEE, 2009, pp. 1877–1884.
[121] R. Ganguli, S. Bandopadhyay, Neural network performance versus network architecture for a quick stop training application, in: APCOM 2003: 31st International Symposium on Application of Computers and Operations Research in the Minerals Industries, South African Institute of Mining and Metallurgy, 2003, p. 39.
[122] L. Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Netw. 11 (4) (1998) 761–767.
[123] M.S. Iyer, R.R. Rhinehart, A novel method to stop neural network training, in: Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334), vol. 2, IEEE, 2000, pp. 929–933.
[124] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 161–168.
[125] M. Mahsereci, L. Balles, C. Lassner, P. Hennig, Early stopping without a validation set, 2017, arXiv preprint arXiv:1703.09580.
[126] H. Song, M. Kim, D. Park, J.-G. Lee, How does early stopping help generalization against label noise? 2019, arXiv preprint arXiv:1911.08059.
[127] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[128] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.
[129] J. Xu, X. Sun, Z. Zhang, G. Zhao, J. Lin, Understanding and improving layer normalization, 2019, arXiv preprint arXiv:1911.07013.
[130] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, 2016, arXiv preprint arXiv:1607.08022.
[131] Z. Xu, X. Yang, X. Li, X. Sun, The effectiveness of instance normalization: a strong baseline for single image dehazing, 2018, arXiv preprint arXiv:1805.03305.
[132] Y. Wu, K. He, Group normalization, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3–19.
[161] T. Salimans, D.P. Kingma, Weight normalization: A simple reparameterization to accelerate training of deep neural networks, Adv. Neural Inf. Process. Syst. 29 (2016) 901–909.
[162] H. Nam, H.-E. Kim, Batch-instance normalization for adaptively style-invariant neural networks, 2018, arXiv preprint arXiv:1805.07925.
[163] L. Huang, D. Yang, B. Lang, J. Deng, Decorrelated batch normalization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 791–800.
[164] L. Huang, Y. Zhou, F. Zhu, L. Liu, L. Shao, Iterative normalization: Beyond standardization towards efficient whitening, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4874–4883.
[165] S. Ioffe, Batch renormalization: Towards reducing minibatch dependence in batch-normalized models, 2017, arXiv preprint arXiv:1702.03275.
[166] P. Luo, J. Ren, Z. Peng, R. Zhang, J. Li, Differentiable learning-to-normalize via switchable normalization, 2018, arXiv preprint arXiv:1806.10779.
[167] P. Luo, P. Zhanglin, S. Wenqi, Z. Ruimao, R. Jiamin, W. Lingyun, Differentiable dynamic normalization for learning deep representation, in: International Conference on Machine Learning, PMLR, 2019, pp. 4203–4211.
[133] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets.
[134] C. Xie, J. Lv, X. Li, Finding a good initial configuration of parameters for restricted Boltzmann machine pre-training, Soft Comput. 21 (21) (2017) 6471–6479.
[135] S. Kokalj-Filipovic, R. Miller, N. Chang, C.L. Lau, Mitigation of adversarial examples in rf deep classifiers utilizing autoencoder pre-training, in: 2019 International Conference on Military Communications and Information Systems, ICMCIS, IEEE, 2019, pp. 1–6.
[136] C. Plahl, T.N. Sainath, B. Ramabhadran, D. Nahamoo, Improved pre-training of deep belief networks using sparse encoding symmetric machines, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2012, pp. 4165–4168.
[137] S.J. Nowlan, G.E. Hinton, Simplifying neural networks by soft weight-sharing, Neural Comput. 4 (4) (1992) 473–493.
[138] J. Zhang, J. Miao, K. Zhao, Y. Tian, Multi-task feature selection with sparse regularization to extract common and task-specific features, Neurocomputing 340 (2019) 76–89.
[139] A. Maurer, M. Pontil, B. Romera-Paredes, Sparse coding for multitask and transfer learning, in: International Conference on Machine Learning, PMLR, 2013, pp. 343–351.
[140] Y. Zhang, Q. Yang, A survey on multi-task learning, 2017, arXiv preprint arXiv:1707.08114.
[141] C. Williams, E.V. Bonilla, K.M. Chai, Multi-task Gaussian process prediction, Adv. Neural Inf. Process. Syst. (2007) 153–160.
[142] Y. Zhang, D.Y. Yeung, Multilabel relationship learning, ACM Trans. Knowl. Discov. Data 7 (2) (2013) 1–30.
[143] Y. Zhang, D.-Y. Yeung, A regularization approach to learning task relationships in multitask learning, ACM Trans. Knowl. Discov. Data 8 (3) (2014) 1–31.
[144] B. Poole, J. Sohl-Dickstein, S. Ganguli, Analyzing noise in autoencoders and deep networks, 2014, arXiv preprint arXiv:1406.1831.
[145] S. Hochreiter, J. Schmidhuber, Simplifying neural nets by discovering flat minima, in: Advances in Neural Information Processing Systems, 1995, pp. 529–536.
[146] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 2488–2498.
[147] X. Li, S. Chen, X. Hu, J. Yang, Understanding the disharmony between dropout and batch normalization by variance shift, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2682–2690.
[148] T. Van Laarhoven, L2 regularization versus batch and weight normalization, 2017, arXiv preprint arXiv:1706.05350.