A Survey on Deep Semi-Supervised Learning

Xiangli Yang, Zixing Song, Irwin King, *Fellow, IEEE*, and Zenglin Xu, *Senior Member, IEEE*

Abstract—Deep semi-supervised learning is a fast-growing field with a range of practical applications. This paper provides a comprehensive survey on both fundamentals and recent advances in deep semi-supervised learning methods from perspectives of model design and unsupervised loss functions. We first present a taxonomy for deep semi-supervised learning that categorizes existing methods, including deep generative methods, consistency regularization methods, graph-based methods, pseudo-labeling methods, and hybrid methods. Then we provide a comprehensive review of 60 representative methods and offer a detailed comparison of these methods in terms of the type of losses, architecture differences, and test performance results. In addition to the progress in the past few years, we further discuss some shortcomings of existing methods and provide some tentative heuristic solutions for solving these open problems.

Index Terms—Deep semi-supervised learning, semi-supervised learning, deep learning

1 Introduction

Deep learning has achieved great successes in both theory and practice [1], [2], especially in supervised learning scenarios, by leveraging a large amount of high-quality labeled data. However, labeled samples are often difficult, expensive, or time-consuming to obtain. By contrast, the unlabeled data is usually abundant and can be easily or inexpensively obtained. Consequently, it is desirable to leverage a large number of unlabeled data for improving the learning performance given a small number of labeled samples. For this reason, semi-supervised learning (SSL) has been a hot research topic in machine learning in the last decade [3], [4].

SSL is a learning paradigm associated with constructing models that use both labeled and unlabeled data. SSL methods can improve learning performance by using additional unlabeled instances compared to supervised learning algorithms, which can use only labeled data. It is easy to obtain SSL algorithms by extending supervised learning algorithms or unsupervised learning algorithms. SSL algorithms provide a way to explore the latent patterns from unlabeled examples, alleviating the need for a large number of labels [5]. Depending on the key objective function of the systems, one may have a semi-supervised classification, a semi-supervised clustering, or a semi-supervised regression. We provide the definitions in the Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3220219.

Since the 1970 s when the concept of SSL first came to the fore [6], [7], [8], there have been a wide variety of SSL methods, including generative models [9], [10], semi-supervised support vector machines [11], [12], [13], [14], graph-based methods [15], [16], [17], [18], and co-training [19]. We refer interested readers to [4], [20], which provide a comprehensive overview of traditional SSL methods. Nowadays, deep neural networks have played a dominating role in many research areas. It is important to adopt the classic SSL framework and develop novel SSL methods for deep learning settings, which leads to deep semi-supervised learning (DSSL). DSSL studies how to effectively utilize both labeled and unlabeled data by deep neural networks. A considerable amount of DSSL methods have been proposed. These methods have been applied in different tasks and domains, such as image classification, object detection, semantic segmentation, text classification and sequence learning. In the Appendix E, available in the online supplemental material, we present the most commonly used datasets and their representative works in various fields.

Some representative works of SSL were described in the early survey [4], [20], however, emerging technologies based on deep learning, such as adversarial training which generates new training data for SSL, have not been included. Besides, [5] focuses on unifying the evaluation indices of some consistency regularization methods, and [21] only reviews generative models and teacher-student models of SSL, both without making a comprehensive overview of SSL. Although [22] tries to present a whole picture of SSL, the taxonomy is quite different from ours. A recent review by Ouali et al. [23] gives a similar notion of DSSL as we do. However, it does not compare the presented methods based on their taxonomy and provide perspectives on future trends and

- *Xiangli Yang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610056, China. E-mail: xlyang@std.uestc.edu.cn.*
- *Zixing Song and Irwin King are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China. E-mail: {zxsong, king}@cse.cuhk.edu.hk.*
- *Zenglin Xu is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 150001, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: xuzenglin@hit.edu.cn.*

Manuscript received 18 October 2021; revised 3 July 2022; accepted 3 November 2022. Date of publication 8 November 2022; date of current version 8 August 2023.
This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0100204, in part by the key program of fundamental research from Shenzhen Science and Technology Innovation Commission under Grant JCYJ20200109113403826, and in part by the Major Key Project of PCL under Grant PCL2021A06.
(Corresponding author: Zenglin Xu.)
Recommended for acceptance by L. Chen.
This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2022.3220219, provided by the authors.
Digital Object Identifier no. 10.1109/TKDE.2022.3220219

1041-4347 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

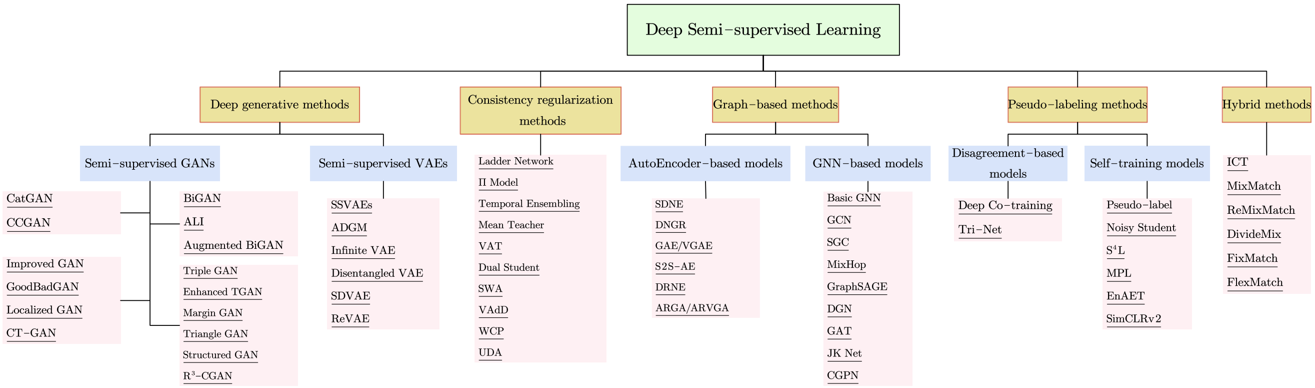


Fig. 1. The taxonomy of major deep semi-supervised learning methods based on loss function and model design.

existing issues. Summarizing both previous and the latest research on SSL, we survey the fundamental theories and compare the DSSL methods. In summary, highlights of our contributions are as follows.

- *New taxonomy.* We introduce a new generalized taxonomy for major DSSL methods, and outline these methods into five major categories. We review the variants on each category and give standardized descriptions and unified sketch maps.
- *Comprehensive overview.* We present a detailed description of each approach, often with key equations, explain the development context beneath the methods, and report the necessary performance comparisons.
- *Abundant resources.* A collection of DSSL resources is compiled, including the open-source codes for some of the reviewed methods, the popular benchmark datasets, and the performance comparisons with different label rates on the different benchmark datasets.[1]
- *Future trends.* We identify six open problems and discuss these directions for future research based on the latest representative studies in this filed.

This survey is organized as follows. In Section 2, we introduce the SSL preliminaries, including problem formulation, the overview of taxonomy, and the evaluations. Sections 3, 4, 5, 6, and 7 introduce the primary deep semi-supervised techniques, i.e., deep generative methods in Section 3, consistency regularization methods in Section 4, graph-based methods in Section 5, pseudo-labeling methods in Section 6 and hybrid methods in Section 7. In Section 8, we discuss the challenges in semi-supervised learning and provide some heuristic solutions and future directions for these open problems.

2 Preliminaries

In this section, we first introduce the preliminaries for DSSL before presenting an overview of the techniques of DSSL.

Problem Formulation. To illustrate the DSSL framework efficiently, we limit our focus on the single-label classification tasks which are simple to describe and implement. We refer interested readers to [24], [25] for multi-label classification tasks. Let $X = \{X_L, X_U\}$ denote the entire data set, including a small labeled subset $X_L = \{x_i, y_i\}_{i=1}^{L}$ and a large scale unlabeled subset $X_U = \{(x_i)\}_{i=1}^{U}$, and generally we assume $L \ll U$. We assume that the dataset contains K classes and $\{y_i\}_{i=1}^{L} \in (y_i^1, y_i^2, \ldots, y_i^K)$, where $y_i^k = 1$ if the sample is labeled by kth class, otherwise $y_i^k = 0$. Formally, SSL aims to solve the following optimization problem,

$$\min_{\theta} \underbrace{\sum_{(x,y)\in X_L} \mathcal{L}_s(x,y,\theta)}_{\text{supervised loss}} + \alpha \underbrace{\sum_{x\in X_U} \mathcal{L}_u(x,\theta)}_{\text{unsupervised loss}} + \beta \underbrace{\sum_{x\in X} \mathcal{R}(x,\theta)}_{\text{regularization}}, \tag{1}$$

where $\mathcal{L}_s$ denotes the per-example supervised loss, e.g., cross-entropy for classification, $\mathcal{L}_u$ denotes the per-example unsupervised loss, and $\mathcal{R}$ denotes the per-example regularization, e.g., consistency loss or a designed regularization term. Note that unsupervised loss terms are often not strictly distinguished from regularization terms, as regularization terms are normally not guided by label information. Lastly, θ denotes the model parameters and $\alpha, \beta \in \mathbb{R}_{>0}$ denotes the trade-off.

Taxonomy Overview. Different choices of architectures and the unsupervised loss functions or regularization terms lead to different semi-supervised methods. As shown in Fig. 1, we will review them from several different perspectives and frameworks. The methods for DSSL fall into the following five groups of research.

- *Deep Generative Methods.* Generative models, such as Variational Auto-Encoders (VAEs), Generative Adversarial Networks (GANs), and their variants have emerged to explore the distribution of the training data set and then generate new examples. Semi-supervised generative methods have been studied and formalized on the basis of these architectures. We will review these models in Section 3.
- *Consistency Regularization Methods.* Based on the manifold assumption or the smoothness assumption, consistency regularization methods apply consistency constraints to the final loss functions. There are three perspectives for constructing constraints, namely input perturbation, weights perturbation, and layer perturbations of the network. The most common architecture of consistency regularization SSL methods is called the Teacher-Student model. In

[1] A list of resources can be found at https://github.com/cd-xlyang/Deep-Semi-supervised-Learning.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Comparison of Deep Semi-Supervised Learning Methods

Methods	Description	Representative objective functions	Advantage	Disadvantage
Deep generative methods	Using a generative model such as GAN, VAE,and their variants	$\frac{\mathbb{E}_{x\sim p(x)}\log p_D(y\leq K\|x)+\mathbb{E}_{x\sim p_G}\log p_D(y=K+1\|x)}{p_\theta(x,y,z_1,z_2)=p(y)p(z_2)p_\theta(z_1\|y,z_2)p_\theta(x\|z_1)}$	Generating new training examples	Hard to achieve optimal results for both generative task and downstream task
Consistency regularization methods	Constructing consistency constraints	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),\mathcal{T}_x)$	Assumptions are clear and reasonable	Relying on data augmentation and perturbation methods
Graph-based methods	Constructing a graph by training dataset,and using graph method to solve downstream tasks	$\frac{\sum_{u\in\mathcal{V}}\\|\mathrm{Dec}(\mathbf{z}_u)-\mathbf{s}_u\\|_2^2}{\sum_{u\in\mathcal{V}}\\|\mathrm{Dec}(\mathbf{z}_u)-A_u\\|_2^2}$	Learning more information with graphs	Relying on how to make the relationship between training samples well represented
Pseudo-labeling methods	Pseudo-labeling unlabeled examples using labeled examples	$\alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{K}\mathcal{R}(y_i'^m,f_i'^m)$	Producing pseudo-labels	These pseudo labels are noisy
Hybrid methods	Integrating various learning strategies,such as consistency regularization and pseudo-labeling methods	$X'_L,X'_U=\mathrm{MixMatch}(X_L,X_U,T,K,\alpha)$ $\mathcal{L}_L=\mathbb{E}_{x,p\in X'_L}\mathcal{R}(p,f(\theta,x))$ $\mathcal{L}_U=\mathbb{E}_{u,q\in X'_U}\mathcal{R}(q,f(\theta,u))$	More effient and robust	Larger model size

Section 4, different ways of perturbations lead to a diverse of learning models.

- *Graph-Based Methods.* The basic setting of graph-based SSL is that a similarity graph can be constructed from the raw data set, where each node represents a training example, and each weighted edge denotes similarity of node pair. The label information of unlabeled examples can be inferred from the constructed graph based on the manifold assumption. In Section 5, we focus on reviewing graph embedding SSL methods of label inference. For graph construction, we refer the interested readers to [26].
- *Pseudo-Labeling Methods.* The most prominent common approach of the pseudo-labeling methods is to produce pseudo labels for unlabeled instances according to the high confidence model's prediction, and then uses them to regularize the training of the model, so these methods can be considered as bootstrapping algorithms. According to the number of learners, we discuss two kinds of pseudo-labeling methods in Section 6.
- *Hybrid Methods.* Hybrid methods are composed of the combination of various methodologies, such as consistency regularization, pseudo-labeling, data augmentation, entropy estimation, and other components used to improve performance. In Section 7, we will review various genres of hybrid methods.

Generative methods and graph-based methods are the easiest to distinguish, based on whether new examples are generated and whether training instances and labels can construct a graph. The most confusing part is the consistency regularization methods and pseudo-labeling methods, where the difference is that pseudo-labeling methods always label unlabeled examples with pseudo-labels, then we can use these pseudo labeled data for supervised learning, and consistency regularization methods focus on consistency constraints rather than pseudo labels. Hybrid methods are effective combinations of these ideas, for example, the most common combination is the combination of consistency regularization and pseudo-labeling. The primary components of these methods are summarized as Table 1.

Regarding whether test data are wholly available in the training process, SSL can be classified into two settings: the transductive setting and the inductive learning setting. Transductive learning assumes that the unlabeled samples in the training process are exactly the data to be predicted, and the purpose of the transductive learning is to generalize over these unlabeled samples, while inductive learning supposes that the learned semi-supervised classifier will be still applicable to new unseen data.

Evaluations. DSSL methods are often evaluated based on their test performances. There are many factors that affect the final test results. [5] shows that DSSL methods differ in their sensitivity to the amount of labeled and unlabeled samples, and implementations along with their training strategies can also greatly impact the results. For example, in [27], different models with different parameters, but the same architecture, make different test performance results. With different ways of permutation invariant settings and data augmentation, the experimental results also differ greatly under the same conditions of other settings [28]. Different works [29], [30], [31] set different averaged run-times, also resulting in different results. These differences prevent direct comparison between approaches.Therefore, as shown in Tables 2, 3, 4, 5, 6, 7, 8, and 9, we will summarize some results reported in the corresponding original papers based on their inheritance and homogeneity. Additionally, we also report many test performances made on standard settings for a fair comparison. More detail can be found in Appendix F, available in the online supplemental material.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Comparisons of State-of-the-Art Semi-Supervised GANs Errors (%) on MNIST, SVHN and CIFAR-10 Datasets

Model	MNIST	SVHN		CIFAR-10	
	$n = 100$	$n = 500$	$n = 1000$	$n = 1000$	$n = 4000$
CatGAN [32]	$1.39(\pm 0.28)$	-	-	-	$19.58(\pm 0.58)$
ImprovedGAN [33]	$0.93(\pm 0.07)$	$18.44(\pm 4.8)$	$8.11(\pm 1.3)$	$21.83(\pm 2.01)$	$18.63(\pm 2.32)$
GoodBadGAN [34]	$0.80(\pm 0.1)$	-	$4.25(\pm 0.03)$	-	$14.41(\pm 0.30)$
Localized GAN [35]	-	$5.48(\pm 0.29)$	$4.73(\pm 0.16)$	$17.44(\pm 0.25)$	$14.23(\pm 0.27)$
CT-GAN [36]	$0.89(\pm 0.13)$	-	-	-	$9.98(\pm 0.21)$
ALI [37]	-	-	$7.42(\pm 0.65)$	$19.98(\pm 0.89)$	$17.99(\pm 1.62)$
Augmented-BiGAN [38]	-	$4.87(\pm 1.6)$	$4.39(\pm 1.2)$	$19.52(\pm 1.5)$	$16.20(\pm 1.6)$
Triple GAN [39]	$0.91(\pm 0.58)$	-	$5.77(\pm 0.17)$	-	$16.99(\pm 0.36)$
Enhanced TGAN [40]	$0.42(\pm 0.03)$	-	$2.97(\pm 0.09)$	-	$9.42(\pm 0.22)$
MarginGAN [41]	$3.53(\pm 0.57)$	$6.07(\pm 0.43)$	-	$10.39(\pm 0.43)$	$6.44(\pm 0.1)$
TriangleGAN [42]	-	-	-	-	$16.80(\pm 0.42)$
Structured GAN [43]	$0.89(\pm 0.11)$	-	$5.73(\pm 0.12)$	-	$17.26(\pm 0.69)$
$\boldsymbol{R}^3$-CGAN [44]	-	-	$2.97(\pm 0.05)$	-	$6.69(\pm 0.28)$

TABLE 3
Semi-Supervised VAE Test Error (%) Benchmarks on MNIST, SVHN, NORB and CIFAR-10 for Randomly Labeled and Evenly Distributed Data Points

Model	MNIST				SVHN	NORB	CIFAR-10	
	$n = 100$	$n = 600$	$n = 1000$	$n = 3000$	$n = 1000$	$n = 1000$	$n = 1000$	$n = 4000$
M1+TSVM [52]	$11.82(\pm 0.25)$	$5.72(\pm 0.049)$	$4.24(\pm 0.07)$	$3.49(\pm 0.04)$	$54.33(\pm 0.11)$	$18.79(\pm 0.05)$	-	-
M2 [52]	$11.97(\pm 1.71)$	$4.94(\pm 0.13)$	$3.6(\pm 0.56)$	$3.92(\pm 0.63)$	-	-	-	-
M1+M2 [52]	$3.33(\pm 0.14)$	$2.59(\pm 0.05)$	$2.40(\pm 0.02)$	$2.18(\pm 0.04)$	$36.02(\pm 0.10)$	-	-	-
ADGM [53]	$0.96(\pm 0.02)$	-	-	-	22.86	$10.06(\pm 0.05)$	-	-
SDGM [53]	$1.32(\pm 0.07)$	-	-	-	$16.61(\pm 0.24)$	$9.40(\pm 0.04)$	-	-
Infinite VAE [54]	$3.93(\pm 0.5)$	-	$2.29(\pm 0.2)$	-	-	-	$8.72(\pm 0.45)$	$7.78(\pm 0.13)$
Disentangled VAE [55]	$9.71(\pm 0.91)$	$3.84(\pm 0.86)$	$2.88(\pm 0.79)$	$1.57(\pm 0.93)$	$38.91(\pm 1.06)$	-	-	-
SDVAE [56]	$2.71(\pm 0.32)$	$1.97(\pm 0.14)$	$1.29(\pm 0.11)$	$1.00(\pm 0.05)$	$29.37(\pm 0.12)$	-	-	-

TABLE 4
Test Performance of Semi-Supervised Learning Models on MNIST, SVHN, CIFAR-10 and CIFAR-100

Model	MNIST		SVHN			CIFAR-10			CIFAR-100
	$n = 100$	$n = 1000$	$n = 250$	$n = 500$	$n = 1000$	$n = 1000$	$n = 2000$	$n = 4000$	10000
Ladder Network [62]	$1.06(\pm 0.37)$	$0.87(\pm 0.08)$	-	-	-	-	-	$20.40(\pm 0.47)$	-
Π Model [63]	$0.55(\pm 0.16)$	-	-	$6.65(\pm 0.53)$	$4.82(\pm 0.17)$	-	-	$12.36(\pm 0.31)$	$39.19(\pm 0.36)$
Temporal Ensembling [64]	-	-	-	$5.12(\pm 0.13)$	$4.42(\pm 0.16)$	-	-	$12.16(\pm 0.24)$	$38.65(\pm 0.51)$
Mean Teacher [31]	-	-	$4.35(\pm 0.50)$	$4.18(\pm 0.27)$	$3.95(\pm 0.19)$	$21.55(\pm 1.48)$	$15.73(\pm 0.31)$	$12.31(\pm 0.28)$	-
VAT [28]	$1.36(\pm 0.03)$	$1.27(\pm 0.11)$	-	-	$6.83(\pm 0.24)$	-	-	$14.87(\pm 0.13)$	-
Dual Student [30]	-	-	$4.24(\pm 0.10)$	$3.96(\pm 0.15)$	-	$15.74(\pm 0.45)$	$11.47(\pm 0.14)$	$9.65(\pm 0.12)$	$33.08(\pm 0.27)$
SWA [65]	-	-	-	-	-	6.6	5.7	5.0	28.0
VAdD [29]	-	$0.99(\pm 0.07)$	-	-	$4.16(\pm 0.08)$	-	-	$11.68(\pm 0.19)$	-
WCP [66]	-	-	$4.29(\pm 0.10)$	$3.75(\pm 0.11)$	$3.58(\pm 0.186)$	$17.62(\pm 1.52)$	$11.93(\pm 0.39)$	$9.72(\pm 0.31)$	-
UDA [27]	-	-	-	-	$2.55(\pm 0.99)$	-	-	$5.29(\pm 0.25)$	-

In order to better understand DSSL, we discuss the assumptions for semi-supervised learning in the Appendix B, available in the online supplemental material. Moreover, classical SSL methods and related learning paradigms are introduced in the Appendices C and D, available in the online supplemental material.

3 Generative Methods

In this section, we will review the deep generative semi-supervised methods based on the GANs framework and the VAEs framework, respectively.

3.1 Semi-Supervised GANs

A typical GAN [45] consists of a generator G and a discriminator D (see Fig. 2(2)). The goal of G is to learn a distribution p_g over data x given a prior on input noise variables $p_z(z)$. The fake samples $G(z)$ generated by the generator G are used to confuse the discriminator D. The discriminator D is used to maximize the distinction between real training samples x and fake samples $G(z)$. As we can see, D and G play the following two-player minimax game with the value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]. \qquad (2)$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 5
Summary of Consistency Regularization Methods

Methods	Techniques	Transformations	Consistency Constraints
Ladder Network	Additional Gaussian Noise in every neural layer	input	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\theta,x+\zeta))$
Π Model	Different Stochastic Augmentations	input	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta_1),f(\theta,x,\zeta_2))$
Temporal Ensembling	Different Stochastic Augmentation and EMA the predictions	input, predictions	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta_1),\mathrm{EMA}(f(\theta,x,\zeta_2)))$
Mean Teacher	Different Stochastic Augmentation and EMA the weights	input, weights	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta),f(\mathrm{EMA}(\theta),x,\zeta))$
VAT	Adversarial perturbation	input	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\theta,x,\gamma^{adv}))$
Dual Student	Stable sample and stabilization constraint	input, weights	$\mathbb{E}_{x\in X}\mathcal{R}(f(\mathrm{STA}(\theta,x_i),\zeta_1),f(\mathrm{STA}(\theta,x_j),\zeta_2))$
SWA	Stochastic Weight Averaging	input, weights	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\mathrm{SWA}(\theta),x,\zeta))$
VAdD	Adversarial perturbation and Stochastic Augmentation (dropout mask)	input, weights	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\epsilon^{s}),f(\theta,x,\epsilon^{adv}))$
UDA	AutoAugment/RandAugment for image; Back-Translation for text	input	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\theta,x,\zeta))$
WCP	Additive perturbation on network weights, DropConnect perturbation for network structure	input, network structure	$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),g(\theta+\zeta,x))$

TABLE 6
Summary of AutoEncoder-Based Deep Graph Embedding Methods

Method	Encoder	Decoder	Similarity Measure	Loss Function	Time Complexity
SDNE [72]	MLP	MLP	$\mathbf{s}_u$	$\sum_{u\in\mathcal{V}}\Vert\mathrm{Dec}(\mathbf{z}_u)-\mathbf{s}_u\Vert_2^2$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$
DNGR [73]	MLP	MLP	$\mathbf{s}_u$	$\sum_{u\in\mathcal{V}}\Vert\mathrm{Dec}(\mathbf{z}_u)-\mathbf{s}_u\Vert_2^2$	$O(\lvert\mathcal{V}\rvert^2)$
GAE [75]	GCN	$\mathbf{z}_u^\top\mathbf{z}_v$	A_{uv}	$\sum_{u\in\mathcal{V}}\Vert\mathrm{Dec}(\mathbf{z}_u)-A_u\Vert_2^2$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$
VGAE [75]	GCN	$\mathbf{z}_u^\top\mathbf{z}_v$	A_{uv}	$\mathbb{E}_{q(\mathbf{Z}\mid X,A)}[\log p(A\mid\mathbf{Z})]-\mathrm{KL}[q(\mathbf{Z}\mid X,A)\Vert p(\mathbf{Z})]$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$
S2S-AE [77]	LSTM	LSTM	$\mathbf{s}_u$	$\sum_{u\in\mathcal{V}}\Vert\mathrm{Dec}(\mathbf{z}_u)-\mathbf{s}_u\Vert_2^2$	$O(\lvert\mathcal{V}\rvert^2)$
DRNE [78]	LSTM	LSTM	$\mathbf{s}_u$	$\sum_{u\in\mathcal{V}}\Vert(\mathbf{z}_u)-\sum_{v\in\mathcal{N}(u)}\mathrm{LSTM}(\mathbf{z}_v)\Vert_2^2$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$
ARVGA [79]	VGAE	$\mathbf{z}_u^\top\mathbf{z}_v$	A_{uv}	$\mathbb{E}_{q(\mathbf{Z}\mid X,A)}[\log p(A\mid\mathbf{Z})]-\mathrm{KL}[q(\mathbf{Z}\mid X,A)\Vert p(\mathbf{Z})]$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$

TABLE 7
Main Differences on GNN-Based Deep Graph Embedding Methods

Method	Improved Operation	Technique	Key Changed Formula	Time Complexity
GCN [76]	Aggregate Operation	Neighborhood Normalization	$\mathbf{m}_{\mathcal{N}(u)}=\sum_{v\in\mathcal{N}(u)}\frac{\mathbf{h}_v}{\sqrt{\lvert\mathcal{N}(u)\rvert\,\lvert\mathcal{N}(v)\rvert}}$	$O(\lvert\mathcal{E}\rvert)$
GraphSAGE [80]	Update Operation	Concatenation	$[\texttt{Update}_{\texttt{base}}(\mathbf{h}_u,\mathbf{m}_{\mathcal{N}(u)})\oplus\mathbf{h}_u]$	$O(\lvert\mathcal{V}\rvert)$
GAT [81]	Aggregate Operation	Neighborhood Attention	$\mathbf{m}_{\mathcal{N}(u)}=\sum_{v\in\mathcal{N}(u)}\alpha_{u,v}\mathbf{h}_v$	$O(\lvert\mathcal{V}\rvert+\lvert\mathcal{E}\rvert)$
JK Net [82]	Update Operation	Jumping Knowledge Conncetions	$\mathbf{z}_u=f_{\mathrm{JK}}(\mathbf{h}_u^{(0)}\oplus\mathbf{h}_u^{(1)}\oplus\ldots\oplus\mathbf{h}_u^{(K)})$	$O(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)$
SGC [83]	Aggregate Operation	Neighborhood Normalization	$\mathbf{h}_u^{(k)}=\sigma(\sum_{v\in\mathcal{N}(u)\cup\{u\}}\frac{\mathbf{h}_v}{\sqrt{\lvert\mathcal{N}(u)\rvert\lvert\mathcal{N}(v)\rvert}})$	$O(\lvert\mathcal{E}\rvert)$
MixHop [84]	Aggregate Operation	Neighborhood Normalization	$\mathbf{H}^{(k)}=\Vert_{j\in P}\sigma(A^j\mathbf{H}^{(k-1)}\mathbf{W}_j^{(i)})$	$O(\lvert\mathcal{E}\rvert)$
DGN [85]	Aggregate Operation	Neighborhood Normalization	$\mathbf{H}^{(k+1)}=\mathbf{H}^{(k)}+\lambda\sum_{i=1}^{g}(\gamma_i(\frac{\mathbf{s}_i^{(k)}\circ\mathbf{H}^{(k)}-\mu_i}{\delta_i})+\beta_i)$	$O(\lvert\mathcal{E}\rvert)$

TABLE 8
Test Classification Error Rates on MNIST, SVHN, CIFAR-10,CIFAR-100,STL10 and ImageNet

Model	MNIST	SVHN	CIFAR-10	CIFAR-100	STL10		ImageNet	
	$n=100$	$n=1000$	$n=4000$	$n=10000$	$n=1000$	$n=5000$	Top-1	Top-5
Deep Co-training [89]	-	3.61(±0.15)	9.03(±0.18)	38.77(±0.28)	-	-	46.50	22.73
Tri-Net [90]	0.53(±0.10)	3.71(±0.14)	8.45(±0.22)	-	-	-	-	-
S^4L-MOAM [91]	-	-	-	-	-	-	26.79	8.77
MPL [92]	-	1.99(±0.07)	3.89(±0.07)	-	-	-	26.11	-
EnAET [93]	-	2.42	4.18	22.92	8.04	4.52	-	-
SimCLRv2 [94]	-	-	-	-	-	-	19.1	4.5

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 9
Error Rates for SVHN, CIFAR-10,CIFAR-100 and STL10

Model	SVHN				CIFAR-10				CIFAR-100			STL10
	$n = 250$	$n = 500$	$n = 1000$	$n = 4000$	$n = 40$	$n = 250$	$n = 1000$	4000	$n = 400$	$n = 2500$	10000	1000
ICT [111]	4.78(±0.68)	4.23(±0.15)	3.89(±0.04)	-	-	-	15.48(±0.78)	7.29(±0.02)	-	-	-	-
MixMatch [112]	3.78(±0.26)	3.64(±0.46)	3.27(±0.31)	2.89(±0.06)	-	-	-	4.95(±0.08)	-	-	25.88(±0.30)	10.18(±1.46)
ReMixMatch [113]	3.10(±0.50)	-	2.83(±0.30)	2.42(±0.09)	19.10(±9.64)	6.27(±0.34)	5.73(±0.16)	5.14(±0.04)	44.28(±2.06)	27.43(±0.31)	23.03(±0.56)	6.18(±1.24)
FixMatch [114]	2.48(±0.38)	-	2.28(±0.11)	-	11.39(±3.35)	5.07(±0.33)	-	4.26(±0.05)	48.85(±3.01)	28.29(±0.11)	22.60(±0.12)	5.17(±0.63)
FlexMatch [115]	-	-	2.86(±0.91)	-	4.99(±0.16)	4.80(±0.06)	-	3.95(±0.03)	32.44(±1.99)	23.85(±0.23)	19.92(±0.06)	5.56(±0.22)

Since GANs can learn the distribution of real data from unlabeled samples, it can be used to facilitate SSL. There are many ways to use GANs in SSL settings. A simple and efficient SSL method can be provided by combining an unsupervised GAN value function with a supervised classification objective function, e.g., $\mathbb{E}_{(x,y)\in X_l}[\log D(y|x)]$. In the following, we review several representative methods of semi-supervised GANs.

CatGAN. Categorical Generative Adversarial Network (CatGAN) [32] modifies the GAN's objective function to take into account the mutual information between observed examples and their predicted categorical class distributions. The structure is illustrated in Fig. 2(3). It consists of three parts: (1) entropy $H[p(y|x, D)]$ which to obtain certain category assignment for samples; (2) $H[p(y|G(z), D)]$ for uncertain predictions from generated samples; and (3) the marginal class entropy $H[p(y|D)]$ to uniform usage of all classes. This method aims to learn a discriminator which distinguishes the samples into K categories by labeling y to each x, instead of learning a binary discriminator value function. For the labeled data, the supervised loss is also a cross-entropy term between the conditional distribution $p(y|x, D)$ and the samples' true label distribution.

CCGAN. Context-Conditional Generative Adversarial Networks (CCGAN) [46] is proposed to use an adversarial loss for harnessing unlabeled image data based on image in-painting. The architecture of the CCGAN is shown in Fig. 2(4). The main highlight of this work is context information provided by the surrounding parts of the image. The method trains a GAN where the generator is to generate pixels within a missing hole. The discriminator is to discriminate between the real unlabeled images and these in-painted images. More formally, $m \odot x$ as input to a generator, where m denotes a binary mask to drop out a specified portion of an image and $\odot$ denotes element-wise multiplication. Thus the in-painted image $x_I = (1 - m) \odot x_G + m \odot x$ with generator outputs $x_G = G(m \odot x, z)$.

Improved GAN. To overcome the learned representations' bottleneck of CatGAN, Semi-supervised GAN (SGAN) [47] learns a generator and a classifier simultaneously. The classifier network can have $(K + 1)$ output units corresponding to $[y_1, y_2, \ldots, y_K, y_{K+1}]$, where the y_{K+1} represents the outputs generated by G. Similar to SGAN, Improved GAN [33] solves a $(K + 1)$-class classification problem. The structure of Improved GAN is shown in Fig. 2(5). Real examples for one of the first K classes and the additional $(K + 1)$th class consisted of the synthetic images generated by the generator G. This work proposes the improved techniques to train the GANs, i.e., feature matching, minibatch discrimination, historical averaging one-sided label smoothing, and virtual batch normalization, where feature matching is used to train the generator. It is trained by minimizing the discrepancy between features of the real and the generated examples, that is $\|\mathbb{E}_{x\in X}D(x) - \mathbb{E}_{z\sim p(z)}D(G(z))\|_2^2$, rather than maximizing the likelihood of its generated examples classified to K real classes. The loss function for training the classifier becomes

$$\begin{aligned}&\max\nolimits_D \mathbb{E}_{(x,y)\sim p(x,y)} \log p_D(y|x, y \leq K)\\&+ \mathbb{E}_{x\sim p(x)} \log p_D(y \leq K|x) + \mathbb{E}_{x\sim p_G} \log p_D(y = K + 1|x),\end{aligned} \tag{3}$$

where the first term of Eq. (3) denotes the supervised cross-entropy loss, The last two terms of Eq. (3) are the unsupervised losses from the unlabeled and generated data, respectively.

GoodBadGAN. GoodBadGAN [34] realizes that the generator and discriminator in [33] may not be optimal simultaneously, i.e., the discriminator achieves good performance in SSL, while the generator may generate visually unrealistic samples. The structure of GoodBadGAN is shown in Fig. 2(5). Generally, the generated samples, along with the loss function (Eq. (3)), can force the boundary of the discriminator to lie between the data manifolds of different categories, which improves the generalization of the discriminator. Due to the analysis, GoodBadGAN learns a bad generator by explicitly adding a penalty term $\mathbb{E}_{x\sim p_G} \log p(x)\mathbb{I}[p(x) > \epsilon]$ to generate bad samples, where $\mathbb{I}[\cdot]$ is an indicator function and ϵ is a threshold, which ensures that only high-density samples are penalized while low-density samples are unaffected. Further, to guarantee the strong true-fake belief in the optimal conditions, a conditional entropy term $\mathbb{E}_{x\sim p_x} \sum_{k=1}^{K} \log p_D(k|x)$ is added to the discriminator objective function in Eq. (3).

Localized GAN. Localized GAN [35] focuses on using local coordinate charts to parameterize local geometry of data transformations across different locations manifold rather than the global one. This work suggests that Localized GAN can help train a locally consistent classifier by exploring the manifold geometry. The architecture of Localized GAN is shown in Fig. 2(6). Like the methods introduced in [33], [34], Localized GAN attempts to solve the $K + 1$ classification problem that outputs the probability of x being assigned to a class.

CT-GAN. CT-GAN [36] combines consistency training with WGAN [48] applied to semi-supervised classification problems. And the structure of CT-GAN is shown in Fig. 2 (7). Following [49], this method also lays the Lipschitz continuity condition over the manifold of the real data to improve the improved training of WGAN. Formally, since the value function of WGAN is

$$\min_G \max_D \mathbb{E}_{x\sim p_x} D(x) - \mathbb{E}_{z\sim p_z} D(G(z)), \tag{4}$$

where D is one of the sets of 1-Lipschitz function. The objective function for updating the discriminator include: (a) basic

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

Wasserstein distance in Eq. (4), (b) gradient penalty $GP|_{\hat{x}}$ [49] used in the improved training of WGAN, where $\hat{x} = \epsilon x + (1 - \epsilon)G(z)$, and (c) A consistency regularization $CT|_{x,'x''}$. For semi-supervised classification, CT-GAN uses the Eq. (3) for training the discriminator instead of the Eq. (4), and then adds the consistency regularization $CT|_{x,'x''}$.

BiGAN/ALI. Bidirectional Generative Adversarial Networks (BiGANs) [50] is an unsupervised feature learning framework. The architecture of BiGAN is shown in Fig. 2 (8). In contrast to the standard GAN framework, BiGAN adds an encoder E to this framework, which maps data x to z', resulting in a data pair (x, z'). The data pair (x, z') and the data pair generated by generator G constitute two kinds of true and fake data pairs. The BiGAN discriminator D is to distinguish the true and fake data pairs. In this work, the value function for training the discriminator becomes,

$$\begin{aligned} \min_{G,E} \max_{D} V(D, E, G) = \mathbb{E}_{x \sim \mathcal{X}} & \underbrace{\left[\mathbb{E}_{z \sim p_E(\cdot|x)}[\log D(x, z)]\right]}_{\log D(x, E(x))} \\ + \mathbb{E}_{z \sim p(z)} & \underbrace{\left[\mathbb{E}_{x \sim p_G(\cdot|z)}[\log(1 - D(x, z))]\right]}_{\log(1 - D(G(z), z))}. \end{aligned} \tag{5}$$

Similarly, Adversarially Learned Inference (ALI) [37] combines an inference network and a generative model. In semi-supervised settings, the objective function can be rewritten as an extended version similar to Eq. (5). Based on the architecture of BiGAN, Kumar et al. [38] propose an extension of BiGAN called Augmented BiGAN for SSL. The Augmented BiGAN uses feature matching loss [33] to optimize the generator network and the encoder network. Moreover, to avoid the issue of class-switching (the class of $G(E(x))$ changed during the decoupled training), a third pair loss term $\mathbb{E}_{x \sim p(x)}[\log(1 - D(E(x), G_x(E(x))))]$ is added to the objective function Eq. (5).

Triple GAN. Triple GAN [39] is presented to address the issue that the generator and discriminator of GAN have incompatible loss functions, i.e., the generator and the discriminator can not be optimal at the same time [33]. The problem has been mentioned in [34], but the solution is different. As shown in Fig. 2(9), the Triple GAN tackles this problem by playing a three-player game. This three-player framework consists of three parts, a generator G using a conditional network to generate the corresponding fake samples for the true labels, a classifier C that generates pseudo labels for given real data, and a discriminator D distinguishing whether a data-label pair is from the real-label dataset or not. This Triple GAN loss function may be written as

$$\begin{aligned} \min_{C,G} \max_{D} V(C, G, D) = {} & \mathbb{E}_{(x,y) \sim p(x,y)}[\log D(x, y)] \\ & + \alpha \mathbb{E}_{(x,y) \sim p_c(x,y)}[\log(1 - D(x, y))] \\ & + (1 - \alpha)\mathbb{E}_{(x,y) \sim p_g(x,y)}[\log \\ & (1 - D(G(y, z), y))], \end{aligned} \tag{6}$$

where D obtains label information about unlabeled data from the classifier C and forces the generator G to generate the realistic image-label samples.

Enhanced TGAN. Based on the architecture of Triple GAN [42], Enhanced TGAN [40] modifies the Triple-GAN by re-designing the generator loss function and the classifier network. The generator generates images conditioned on class distribution and is regularized by class-wise mean feature matching. The classifier network includes two classifiers that collaboratively learn to provide more categorical information for generator training. Additionally, a semantic matching term is added to enhance the semantics consistency with respect to the generator and the classifier network.

MarginGAN. MarginGAN [41] is another extension framework based on Triple GAN [42]. From the perspective of classification margin, this framework works better than Triple GAN when used for semi-supervised classification. The architecture of MarginGAN is presented in Fig. 2(10). MarginGAN includes three components like Triple GAN, a generator G which tries to maximize the margin of generated samples, a classifier C used for decreasing margin of fake images, and a discriminator D trained as usual to distinguish real samples from fake images. This method solves the problem that performance is damaged due to the inaccurate pseudo label in SSL, and improves the accuracy rate of SSL.

Triangle GAN. Triangle Generative Adversarial Network ($\triangle$-GAN) [42] introduces a new architecture to match cross-domain joint distributions. The architecture of the $\triangle$-GAN is shown in Fig. 2(11). The $\triangle$-GAN can be considered as an extended version of BiGAN [50] or ALI [37]. This framework is a four-branch model that consists of two generators E and G, and two discriminators D_1 and D_2. The two generators can learn two different joint distributions by two-way matching between two different domains. At the same time, the discriminators are used as an implicit ternary function, where D_2 determines whether the data pair is from (x, y') or from $(G(z), y)$, and D_1 distinguishes real data pair (x, y) from the fake data pair $(G(z), y)$. Relied on the Triangle GAN, Liu et al. [44] propose a Class-conditional GAN with Random Regional Replacement (R3-regularization) technique, called R^3-CGAN. In this framework, cross-category instances and real-fake instances are constructed by CutMix [51]. These instances are used to regularize the R^3-CGAN classifier and discriminator D_1. Like the training strategy of Triangle GAN, the class-specific information is effectively used for the downstream.

Structured GAN. Structured GAN [43] studies the problem of semi-supervised conditional generative modeling based on designated semantics or structures. The architecture of Structured GAN (see Fig. 2(12)) is similar to Triangle GAN [42]. Specifically, Structured GAN assumes that the samples x are generated conditioned on two independent latent variables, i.e., y that encodes the designated semantics and z contains other variation factors. Training Structured GAN involves solving two adversarial games that have their equilibrium concentrating at the true joint data distributions $p(x, z)$ and $p(x, y)$. The synthesized data pair $(x,' y)$ and $(x,' z)$ are generated by generator $G(y, z)$, where $(x,' y)$ mix real sample pair (x, y) together as input for training discriminator $D(x, y)$ and $(x'z)$ blends the E's outputs pair (x, z') for discriminator $D(x, z)$.

Summary. Comparing with the above discussed Semi-GANs methods, we find that the main difference lies in the

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

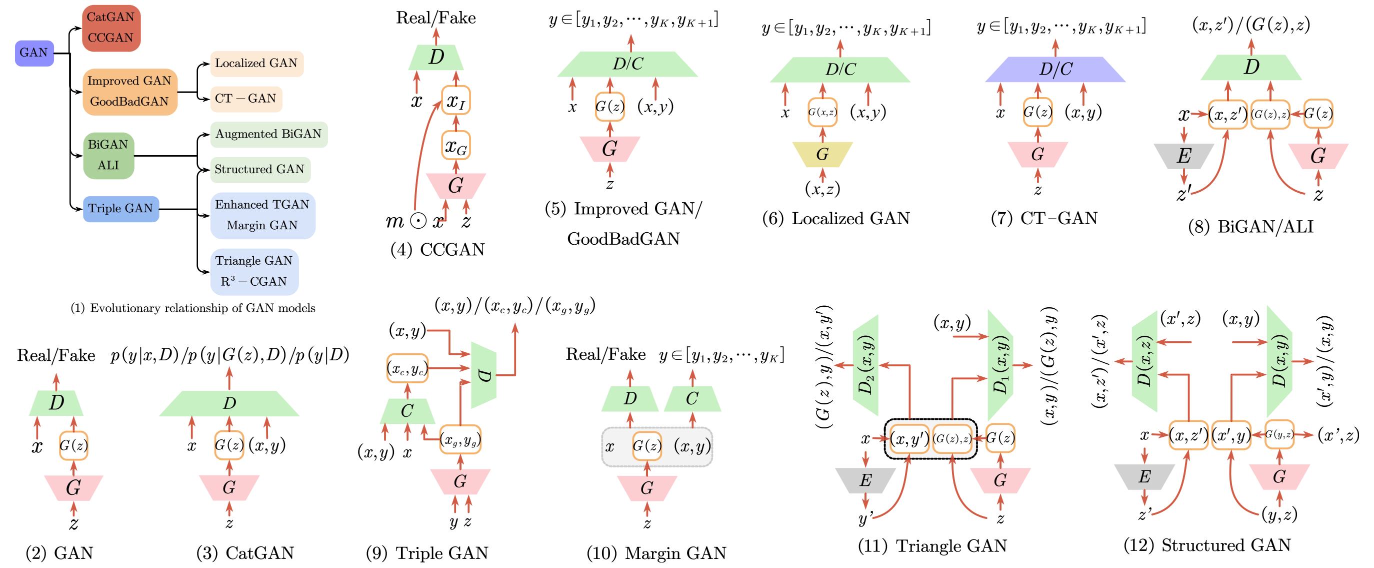


Fig. 2. A glimpse of the diverse range of architectures used for GAN-based deep generative semi-supervised methods. The characters “D, G” and “E” represent *Discriminator*, *Generator* and *Encoder*, respectively. In Figure (6), Localized GAN is equipped with a local generator $G(x,z)$, so we use the yellow box to distinguish it. Similarly, in CT-GAN, the purple box is used to denote a discriminator that introduces consistency constraint.

number and type of the basic modules, such as generator, encoder, discriminator, and classifier. As shown in Fig. 2(1), we find the evolutionary relationship of the Semi-GANs models. Overall, CatGAN [32] and CCGAN [46] extend the basic GAN by including additional information in the model, such as category information and in-painted images. Based on Improved GAN [33], Localized GAN [35] and CT-GAN [36] consider the local information and consistency regularization, respectively. BiGAN [50] and ALI [37] learn an inference model during the training process by adding an Encoder module. In order to solve the problem that the generator and discriminator can not be optimal at the same time, Triple-GAN [39] adds an independent classifier instead of using a discriminator as a classifier.

3.2 Semi-Supervised VAE

Variational AutoEncoders (VAEs) [57], [58] are flexible models which combine deep autoencoders with generative latent-variable models. The generative model captures representations of the distributions rather than the observations of the dataset, and defines the joint distribution in the form of $p(x,z) = p(z)p(x|z)$, where $p(z)$ is a prior distribution over latent variables z. The architecture of VAEs is a two-stage network, an encoder to construct a variational approximation $q(z|x)$ to the posterior $p(z|x)$, and a decoder to parameterize the likelihood $p(x|z)$. The variational approximation of the posterior maximizes the marginal likelihood, and the evidence lower bound (ELBO) may be written as

$$\log p(x) = \log \mathbb{E}_{q(z|x)}[\frac{p(z)p(x|z)}{q(z|x)}] \geq \mathbb{E}_{q(z|x)}[\log \frac{p(z)p(x|z)}{q(z|x)}]. \tag{7}$$

In the following, we review several representative latent variable methods for semi-supervised learning.

SSVAEs. SSVAEs denotes the VAE-based generative models with latent encoder representation proposed in [52]. The first one, i.e., the latent-feature discriminative model, referred to M1 [52], can provide more robust latent features with a deep generative model of the data. As shown in Fig. 3 (1), $p_\theta(x|z)$ is a non-linear transformation, e.g., a deep neural network. Latent variables z can be selected as a Gaussian distribution or a Bernoulli distribution. An approximate sample of the posterior distribution on the latent variable $q_\phi(z|x)$ is used as the classifier feature for the class label y. The second one, namely Generative semi-supervised model, referred to M2 [52], describes the data generated by a latent class variable y and a continuous latent variable z, is expressed as $p_\theta(x|z,y)p(z)p(y)$ (as depicted in Fig. 3(2)). $p(y)$ is the multinomial distribution, where the class labels y are treated as latent variables for unlabeled data. $p_\theta(x|z,y)$ is a suitable likelihood function. The inferred posterior distribution $q_\phi(z|y,x)$ can predict any missing labels.

Stacked generative semi-supervised model, called M1 +M2, uses the generative model M1 to learn the new latent representation z_1, and uses the embedding from z_1 instead of the raw data x to learn a generative semi-supervised model M2. As shown in Fig. 3(3), the whole process can be abstracted as follows:

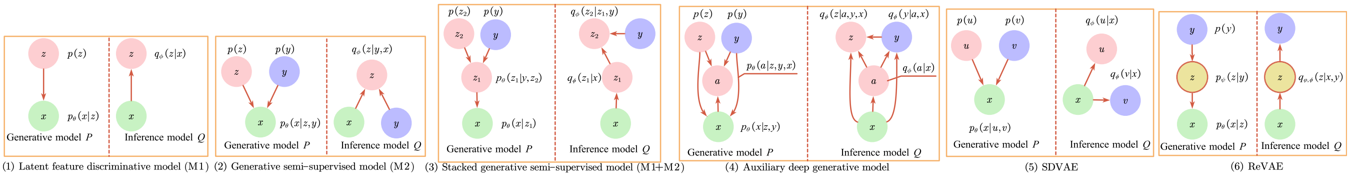
$$p_\theta(x,y,z_1,z_2) = p(y)p(z_2)p_\theta(z_1|y,z_2)p_\theta(x|z_1), \tag{8}$$

where $p_\theta(z_1|y,z_2)$ and $p_\theta(x|z_1)$ are parameterised as deep neural networks. In all the above models, $q_\phi(z|x)$ is used to approximate the true posterior distribution $p(z|x)$, and following the variational principle, the boundary approximation lower bound of the model is derived to ensure that the approximate posterior probability is as close to the true posterior probability as possible.

ADGM. Auxiliary Deep Generative Models (ADGM) [53] extends SSVAEs with auxiliary variables, as depicted in Fig. 3(4). Adding the auxiliary variable a leaves the generative model of x, y unchanged while significantly improving the representative power of the posterior approximation. An additional inference network is introduced such that

$$q_\phi(a,y,z|x) = q_\phi(z|a,y,x)q_\phi(y|a,x)q_\phi(a|x). \tag{9}$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.



(1) Latent feature discriminative model (M1) (2) Generative semi–supervised model (M2) (3) Stacked generative semi–supervised model (M1+M2) (4) Auxiliary deep generative model (5) SDVAE (6) ReVAE

Fig. 3. A glimpse of probabilistic graphical models used for VAE-based deep generative semi-supervised methods. Each method contains two models, the generative model P and the inference model Q. The variational parameters θ and ϕ are learned jointly by the incoming connections (i.e., deep neural networks).

And the framework has the generative model p defined as $p_\theta(a)p_\theta(y)p_\theta(z)p_\theta(x|z,y)$, where a, y, z are the auxiliary variable, class label, and latent features, respectively. The auxiliary unit a actually introduces a class-specific latent distribution between x and y, resulting in a more expressive distribution $q_\phi(y|a,x)$. Formally, the ADGM employs a similar variational lower bound $\mathbb{E}_{q_\phi(a,z|x)}[\log p_\theta(a,x,y,z) - \log q_\phi(a,z|x,y)]$ on the marginal likelihood, with $q_\phi(a,z|x,y) = q_\phi(a|x)q_\phi(z|y,x)$. Interestingly, by reversing the direction of the dependence between x and a, a model called Skip Deep Generative Model (SDGM) [53] is recovered, with skipping connections from the second stochastic layer and the labels. In this case the generative model is affected. This model is able to be trained end-to-end using stochastic gradient descent (SGD).

Infinite VAE. Infinite VAE [54] proposes an infinite mixture model of variational autoencoders by utilizing a non-parametric Bayesian approach. This model can adapt to suit the input data by mixing coefficients by a Dirichlet process. It combines Gibbs sampling and variational inference that enables the model to learn the input's underlying structures efficiently. Formally, Infinite VAE employs the mixing coefficients to assist SSL by combining the unsupervised generative model and a supervised discriminative model. The infinite mixture generative model as,

$$p(c,\pi,x,z) = p(c|\pi)p_\alpha(\pi)p_\theta(x|c,z)p(z), \tag{10}$$

where c denotes the assignment matrix for each instance to a VAE component where the VAE-i can best reconstruct instance i. π is the mixing coefficient prior for c, drawn from a Dirichlet distribution with parameter α. Each latent variable z_i in each VAE is drawn from a Gaussian distribution.

Disentangled VAE. Disentangled VAE [55] attempts to learn disentangled representations using partially-specified graphical model structures and distinct encoding aspects of the data into separate variables. It explores the graphical model for modeling a general dependency on observed and unobserved latent variables with neural networks, and a stochastic computation graph [59] is used to infer with and train the resultant generative model. For this purpose, importance sampling estimates are used to maximize the lower bound of both the supervised and semi-supervised likelihoods. Formally, this framework considers the conditional probability $q_{y,z|x}$, which has a factorization $q_\phi(y,z|x) = q_\phi(y|x,z)q_\phi(z|x)$ rather than $q_\phi(y,z|x) = q_\phi(z|x,y)q_\phi(y|x)$ in [52], which means we can no longer compute a simple Monte Carlo estimator by sampling from the unconditional distribution $q_\phi(z|x)$. Thus, the variational lower bound for supervised term expand below,

$$\mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|y,z)p(y)p(z) - q_\phi(y,z|x)]. \tag{11}$$

SDVAE. Semi-supervised Disentangled VAE (SDVAE) [56] incorporates the label information to the latent representations by encoding the input disentangled representation and non-interpretable representation. The disentangled variable captures categorical information, and the non-interpretable variable consists of other uncertain information from the data. As shown in Fig. 3(5), SDVAE assumes the disentangled variable v and the non-interpretable variable u are independent condition on x, i.e., $q_\phi(u,v|x) = q_\phi(u|x)q_\phi(v|x)$. This means that $q_\phi(v|x)$ is the encoder for the disentangled representation, and $q_\phi(u|x)$ denotes the encoder for non-interpretable representation. Based on those assumptions, the variational lower bound is written as

$$\mathbb{E}_{q(u|x),q(v|x)}[\log p(x|u,v)p(v)p(u) - \log q(u|x)q(v|x)]. \tag{12}$$

ReVAE. Reparameterized VAE (ReVAE) [60] develops a novel way to encoder supervised information, which can capture label information through auxiliary variables instead of latent variables in the prior work [52]. The graphical model is illustrated in Fig. 3(6). In contrast to SSVAEs, ReVAE captures meaningful representations of data with a principled variational objective. Moreover, ReVAE carefully designed the mappings between auxiliary and latent variables. In this model, a conditional generative model $p_\psi(z|y)$ is introduced to address the requirement for inference at test time. Similar to [52] and [53], ReVAE treats y as a known observation when the label is available in the supervised setting, and as an additional variable in the unsupervised case. In particular, the latent space can be partitioned into two disjoint subsets under the assumption that label information captures only specific aspects.

Summary. As the name indicates, Semi-supervised VAE applies the VAE architecture for handling SSL problems. An advantage of these methods is that meaningful representations of data can be learned by the generative latent-variable models. The basic framework of these Semi-supervised VAE methods is M2 [52]. On the basis of the M2 framework, ADGM [53] and ReVAE [61] consider introducing additional auxiliary variables, although the roles of the auxiliary variables in the two models are different. Infinite VAE [54] is a hybrid of several VAE models to improve the performance of the entire framework. Disentangled VAE [55] and SDVAE [56] solve the semi-supervised VAE problem by different disentangled methods. Under semi-supervised conditions, when a large number of labels are unobserved, the key to this kind of method is how to deal with the latent variables and label information.

4 Consistency Regularization

The most common structure of consistency regularization SSL methods is the Teacher-Student structure. As a student,

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

the model learns as before, and as a teacher, the model generates targets simultaneously. Let θ' denote the weight of the target, and θ denote the weights of the basic student. The consistency constraint is defined as

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),\mathcal{T}_x), \tag{13}$$

where $f(\theta,x)$ is the prediction from model $f(\theta)$ for input x. $\mathcal{T}_x$ is the consistency target of the teacher. $\mathcal{R}(\cdot,\cdot)$ measures the distance between two vectors and is usually set to Mean Squared Error (MSE) or KL-divergence. Different consistency regularization techniques vary in how they generate the target. There are several ways to boost the target $\mathcal{T}_x$ quality. One strategy is to select the perturbation rather than additive or multiplicative noise carefully. Another technique is to consider the teacher model carefully instead of replicating the student model.

Ladder Network. Ladder Network [62] is the first successful attempt towards using a Teacher-Student model that is inspired by a deep denoising AutoEncoder. The structure of the Ladder Network is shown in Fig. 4(1). In Encoder, noise ζ is injected into all hidden layers as the corrupted feedforward path $x+\zeta\rightarrow\underset{f(\cdot)}{\underline{\text{Encoder}}}\rightarrow\tilde{z}_1\rightarrow\tilde{z}_2$ and shares the mappings $f(\cdot)$ with the clean encoder feedforward path $x\rightarrow\underset{f(\cdot)}{\underline{\text{Encoder}}}\rightarrow z_1\rightarrow z_2\rightarrow y$. The decoder path $\tilde{z}_1\rightarrow\tilde{z}_2\rightarrow\underset{g(\cdot,\cdot)}{\underline{\text{Decoder}}}\rightarrow\hat{z}_2\rightarrow\hat{z}_1$ consists of the denoising functions $g(\cdot,\cdot)$ and the unsupervised denoising square error $\mathcal{R}$ on each layer consider as consistency loss between $\hat{z}_i$ and z_i. Through latent skip connections, the ladder network is differentiated from regular denoising AutoEncoder. This feature allows the higher layer features to focus on more abstract invariant features for the task. Formally, the ladder network unsupervised training loss $\mathcal{L}_u$ or the consistency loss is computed as the MSE between the activation of the clean encoder z_i and the reconstructed activations $\hat{z}_i$. Generally, $\mathcal{L}_u$ is

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),g(f(\theta,x+\zeta))). \tag{14}$$

Π *Model.* Unlike the perturbation used in Ladder Network, Π Model [63] is to create two random augmentations of a sample for both labeled and unlabeled data. Some techniques with non-deterministic behavior, such as randomized data augmentation, dropout, and random max-pooling, make an input sample pass through the network several times, leading to different predictions. The structure of the Π Model is shown in Fig. 4(2). In each epoch of the training process for Π Model, the same unlabeled sample propagates forward twice, while random perturbations are introduced by data augmentations and dropout. The forward propagation of the same sample may result in different predictions, and the Π Model expects the two predictions to be as consistent as possible. Therefore, it provides an unsupervised consistency loss function,

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta_1),f(\theta,x,\zeta_2)), \tag{15}$$

which minimizes the difference between the two predictions.

Temporal Ensembling. Temporal Ensembling [64] is similar to the Π Model, which forms a consensus prediction under different regularization and input augmentation conditions. The structure of Temporal Ensembling is shown in Fig. 4(3). It modifies the Π Model by leveraging the Exponential Moving Average (EMA) of past epochs predictions. In other words, while Π Model needs to forward a sample twice in each iteration, Temporal Ensembling reduces this computational overhead by using EMA to accumulate the predictions over epochs as $\mathcal{T}_x$. Specifically, the ensemble outputs Z_i is updated with the network outputs z_i after each training epoch, i.e., $Z_i\leftarrow\alpha Z_i+(1-\alpha)z_i$, where α is a momentum term. During the training process, the Z can be considered to contain an average ensemble of $f(\cdot)$ outputs due to Dropout and stochastic augmentations. Thus, consistency loss is

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta_1),\mathrm{EMA}(f(\theta,x,\zeta_2))). \tag{16}$$

Mean Teacher. Mean Teacher [31] averages model weights using EMA over training steps and tends to produce a more accurate model instead of directly using output predictions. The structure of Mean Teacher is shown in Fig. 4(4). Mean Teacher consists of two models called Student and Teacher. The student model is a regular model similar to the Π Model, and the teacher model has the same architecture as the student model with exponential moving averaging the student weights. Then Mean Teacher applied a consistency constraint between the two predictions of student and teacher

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\zeta),f(\mathrm{EMA}(\theta),x,\zeta')). \tag{17}$$

VAT. Virtual Adversarial Training [28] proposes the concept of adversarial attack for consistency regularization. The structure of VAT is shown in Fig. 4(5). This technique aims to generate an adversarial transformation of a sample, which can change the model prediction. Specifically, the adversarial training technique is used to find the optimal adversarial perturbation γ of a real input instance x such that $\gamma\leq\delta$. Afterward, the consistency constraint is applied between the model's output of the original input sample and the perturbed one, i.e.,

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),g(\theta,x+\gamma^{adv})), \tag{18}$$

where $\gamma^{adv}=\mathrm{argmax}_{\gamma;\|\gamma\|\leqslant\delta}\mathcal{R}(f(\theta,x),g(\theta,x+\gamma))$.

Dual Student. Dual Student [30] extends the Mean Teacher model by replacing the teacher with another student. The structure of Dual Student is shown in Fig. 4(6). The two students start from different initial states and are optimized through individual paths during training. The authors also defined a novel concept, "stable sample," along with a stabilization constraint to avoid the performance bottleneck produced by a coupled EMA Teacher-Student model. Hence, their weights may not be tightly coupled, and each learns its own knowledge. Formally, Dual Student checks whether x is a stable sample for student i

$$\mathcal{C}_x^i=\{p_x^i=p_{\bar{x}}^i\}_1\&(\{\mathcal{M}_x^i>\xi\}_1\|\{\mathcal{M}_{\bar{x}}^i>\xi\}_1), \tag{19}$$

where $\mathcal{M}_x^i=\|f(\theta^i,x)\|_\infty$, and the stabilization constraint

$$\mathcal{L}_{sta}^i=\begin{cases}\{\varepsilon^i>\varepsilon^j\}_1\mathcal{R}(f(\theta^i,x),f(\theta^j,x)) & \mathcal{C}^i=\mathcal{C}^j=1\\ \mathcal{C}^i\mathcal{R}(f(\theta^i,x),f(\theta^j,x)) & \text{otherwise}\end{cases}. \tag{20}$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

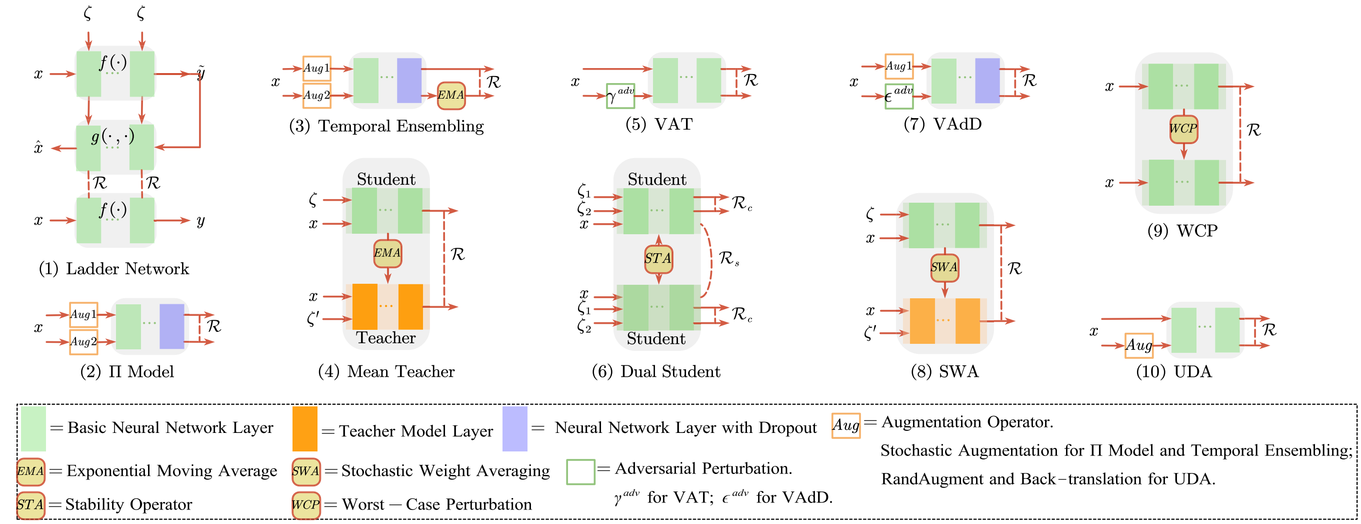

(1) Ladder Network
(2) Π Model
(3) Temporal Ensembling
(4) Mean Teacher
(5) VAT
(6) Dual Student
(7) VAdD
(8) SWA
(9) WCP
(10) UDA
Student
Teacher
Aug1
Aug2
Aug
EMA
STA
SWA
WCP
$f(\cdot)$
$g(\cdot,\cdot)$
x
$\tilde{y}$
$\hat{x}$
y
ζ
ζ'
ζ_1
ζ_2
γ^{adv}
ϵ^{adv}
$\mathcal{R}$
$\mathcal{R}_c$
$\mathcal{R}_s$
= Basic Neural Network Layer
= Teacher Model Layer
= Neural Network Layer with Dropout
Aug = Augmentation Operator.
Stochastic Augmentation for Π Model and Temporal Ensembling;
RandAugment and Back−translation for UDA.
EMA = Exponential Moving Average
SWA = Stochastic Weight Averaging
= Adversarial Perturbation.
γ^{adv} for VAT; ϵ^{adv} for VAdD.
STA = Stability Operator
WCP = Worst − Case Perturbation

Fig. 4. A glimpse of the diverse range of architectures used for consistency regularization semi-supervised methods. In addition to the identifiers in the figure, ζ denotes the perturbation noise, and $\mathcal{R}$ is the consistency constraint.

SWA. Stochastic Weight Averaging (SWA) [65] improves generalization than conventional training. The aim is to average multiple points along the trajectory of stochastic gradient descent (SGD) with a cyclical learning rate and seek much flatter solutions than SGD. The consistency-based SWA [67] observes that SGD fails to converge on the consistency loss but continues to explore many solutions with high distances apart in predictions on the test data. The structure of SWA is shown in Fig. 4(7). The SWA procedure also approximates the Teacher-Student approach, such as Π Model and Mean Teacher with a single model. The authors propose fast-SWA, which adapts the SWA to increase the distance between the averaged weights by using a longer cyclical learning rate schedule and diversity of the corresponding predictions by averaging multiple network weights within each cycle. Generally, the consistency loss can be rewritten as follows:

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\mathrm{SWA}(\theta),x,\zeta)). \tag{21}$$

VAdD. In VAT, the adversarial perturbation is defined as an additive noise unit vector applied to the input or embedding spaces, which has improved the generalization performance of SSL. Similarly, Virtual Adversarial Dropout (VAdD) [29] also employs adversarial training in addition to the Π Model. The structure of VAdD is shown in Fig. 4(8). Following the design of Π Model, the consistency constraint of VAdD is computed from two different dropped networks: one dropped network uses a random dropout mask, and the other applies adversarial training to the optimized dropout network. Formally, $f(\theta,x,\epsilon)$ denotes an output of a neural network with a random dropout mask, and the consistency loss incorporated adversarial dropout is described as

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x,\epsilon^s),f(\theta,x,\epsilon^{adv})), \tag{22}$$

where $\epsilon^{adv}=\mathrm{argmax}_{\epsilon;\|\epsilon^s-\epsilon\|_2\le\delta H}\mathcal{R}(f(\theta,x,\epsilon^s),f(\theta,x,\epsilon))$; $f(\theta,x,\epsilon^{adv})$ represents an adversarial target; ϵ^{adv} is an adversarial dropout mask; ϵ^s is a sampled random dropout mask instance; δ is a hyperparameter controlling the intensity of the noise, and H is the dropout layer dimension.

WCP. A novel regularization mechanism for training deep SSL by minimizing the Worse-case Perturbation (WCP) is presented by Zhang et al. [66]. The structure of WCP is shown in Fig. 4(9). WCP considers two forms of WCP regularizations – additive and DropConnect perturbations, which impose additive perturbation on network weights and make structural changes by dropping the network connections, respectively. Instead of generating an ensemble of randomly corrupted networks, the WCP suggests enhancing the most vulnerable part of a network by making the most resilient weights and connections against the worst-case perturbations. It enforces an additive noise on the model parameters ζ, along with a constraint on the norm of the noise. In this case, the WCP regularization becomes,

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),g(\theta+\zeta,x)). \tag{23}$$

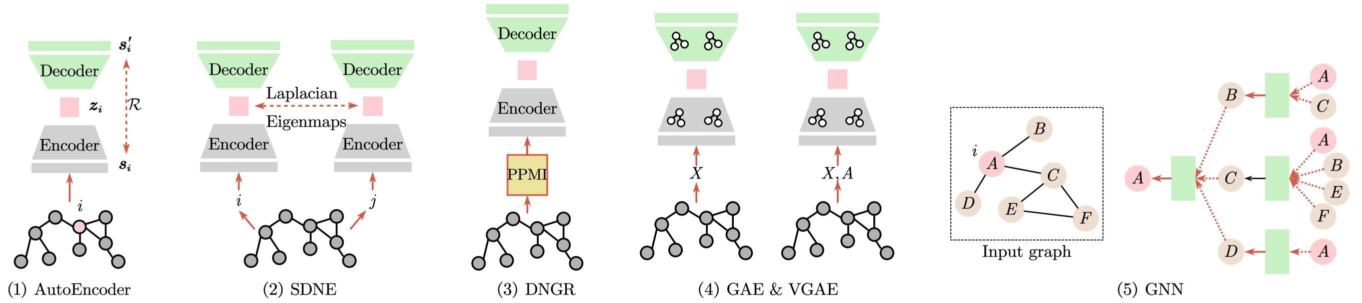
The second perturbation is at the network structure level by DropConnect, which drops some network connections. Specifically, for parameters θ, the perturbed version is $(1-\alpha)\theta$, and the $\alpha=1$ denotes a dropped connection while $\alpha=0$ indicates an intact one. By applying the consistency constraint, we have

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f((1-\alpha)\theta,x)). \tag{24}$$

UDA. UDA stands for Unsupervised Data Augmentation [27] for image classification and text classification. The structure of UDA is shown in Fig. 4(10). This method investigates the role of noise injection in consistency training and substitutes simple noise operations with high-quality data augmentation methods, such as AutoAugment [68], RandAugment [69] for images, and Back-Translation [70], [71] for text. Following the consistency regularization framework, the UDA [27] extends the advancement in supervised data augmentation to SSL. As discussed above, let $f(\theta,x,\zeta)$ be the augmentation transformation from which one can draw an augmented example (x,ζ) based on an original example x. The consistency loss is

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta,x),f(\theta,x,\zeta)), \tag{25}$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.



(1) AutoEncoder (2) SDNE (3) DNGR (4) GAE & VGAE (5) GNN

Fig. 5. A glimpse of the diverse range of architectures used for graph-based semi-supervised methods. Specifically, in figure (3), "PPMI" is short for positive pointwise mutual information. In figure (4), A denotes the adjacency matrix, and in figure (5), pink A represents node A.

where ζ represents the data augmentation operator to create an augmented version of an input x.

Summary. The core idea of consistency regularization methods is that the output of the model remains unchanged under realistic perturbation. As shown in Table 5, Consistency constraints can be considered at three levels: input dataset, neural networks and training process. From the input dataset perspective, perturbations are usually added to the input examples: additive noise, random augmentation, or even adversarial training. We can drop some layers or connections for the networks, as WCP [66]. From the training process, we can use SWA to make the SGD fit the consistency training or EMA parameters of the model for some training epochs as new parameters.

5 Graph-Based Methods

In this section, we will review graph embedding SSL methods, and the principal goal is to encode the nodes as small-scale vectors representing their role and the structure information of their neighborhood. Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the node that has been embedded is a mapping result of $f_{\mathbf{z}} : v \rightarrow \mathbf{z}_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$ such that $d \ll |\mathcal{V}|$, and the $f_{\mathbf{z}}$ function retains some of the measurement of proximity, defined in the graph $\mathcal{G}$. Two classes can be identified among all deep embedding methods: AutoEncoder-based methods and Graph Neural Network (GNN)-based methods.

5.1 AutoEncoder-Based Methods

Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, every node $i \in \mathcal{V}$ is correlated to a neighborhood vector $\mathbf{s}_i \in \mathbb{R}^{|\mathcal{V}|}$. The $\mathbf{s}_i$ vector contains i's similarity with all other graph nodes and acts as a high-dimensional vector representation of i in the neighborhood. The aim of the autoencoding technique is to embed nodes using hidden embedding vectors like $\mathbf{s}_i$ in so as to reconstruct the original information from such embeddings (Fig. 5(1)), the loss for these methods takes the following form,

$$\mathcal{L} = \sum_{i \in \mathcal{V}} \|\mathrm{Dec}(\mathbf{z}_i) - \mathbf{s}_i\|_2^2, \tag{26}$$

where $\mathrm{Dec}(\mathrm{Enc}(\mathbf{s}_i)) = \mathrm{Dec}(\mathbf{z}_i) \approx \mathbf{s}_i$.

SDNE. Structural deep network embedding (SDNE) [72] propose a DSSL model to capture the highly non-linear network structure. In detail, it develops a deep autoencoder architecture to maintain both the first-order and second-order network proximities (Fig. 5(2)). The framework is composed of unsupervised and supervised components. In the unsupervised component, the second-order proximity is used to capture global network structures by rebuilding the neighbors of each vertex. In the supervised component, the first-order proximity describes the pairwise proximity between vertexes.

DNGR. Deep neural networks for graph representations (DNGR) [73] combines random surfing with autoencoders (Fig. 5(3)). The model includes three components: random surfing, positive pointwise mutual information (PPMI) calculation, and stacked denoising autoencoders. The random surfing design is used to create a stochastic matrix equivalent to the similarity measure matrix in HOPE [74] on the input graph. The matrix is transformed into a PPMI matrix and fed into a stacked denoising autoencoder to obtain the embedding. The PPMI matrix input ensures that the autoencoder model captures a higher proximity order. The use of stacked denoising autoencoders also helps make the model robust in the event of noise in the graph, as well as in seizing the internal structure needed for downstream tasks.

GAE/VGAE. Inspired by the idea of autoencoders, graph auto-encoders (GAE) [75] learns the distribution of nodes through encoder, then samples the vectors representation of the nodes which is to reconstruct the graph by a decoder. The encoder is

$$\mathrm{Enc}(A, X) = \mathrm{GraphConv}(\sigma(\mathrm{GraphConv}(A, X))), \tag{27}$$

where $\mathrm{GraphConv}(\cdot)$ is a graph convolutional layer defined in [76], $\sigma(\cdot)$ is the activation function, and A, X is adjacency matrix and attribute matrix, respectively. The decoder is $\mathrm{Dec}(\mathbf{z}_u, \mathbf{z}_v) = \mathbf{z}_u^T \mathbf{z}_v$. Variational GAE (VGAE) [75] learns about the distribution of the data in which the variation lower bound $\mathcal{L}$, is optimized directly by reconstructing the adjacency matrix.

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z} \mid X, A)}[\log p(A \mid \mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z} \mid X, A) \| p(\mathbf{Z})], \tag{28}$$

where $\mathrm{KL}[q(\cdot) \| p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$.

DRNE. S2S-AE [77] modifies long short term memory (LSTM) Auto-Encoders, rather than the MLP used in SDNE and DNGR. This architecture accommodates both the representations of the nodes and their neighborhood information. Deep recursive network embedding (DRNE) [78] also reconstructs the node from its neighbors via LSTM. By learning recursively the embeddings of nodes, DRNE can approximate the embedding of a target node by

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

aggregating the embeddings of its neighbors. The loss function,

$$\mathcal{L} = \sum_{u \in \mathcal{V}} \|(\mathbf{z}_u) - \sum_{v \in \mathcal{N}(u)} \mathrm{LSTM}(\mathbf{z}_v)\|_2^2, \tag{29}$$

is different from the one in S2S-AE. The learned embedding of a node can preserve the local structure of its neighbors in one recursive step, and then it can be updated iteratively, incorporating structural information into the learned node representations in the same manner.

ARGA/ARVGA. ARGA/ARVGA [79] employs a variant of graph convolutional network (GCN) [76]as a graph encoder to represent both graph structure and node content. In the decoder part, there is also a graph convolutional layer or a combination of link prediction layer. A prior Gaussian or uniform distribution, $p(A|\mathbf{Z})$, is used as the basis for enforcing the adversarial training principle. For optimization, the graph autoencoder can be updated by Eq. (28)'s first term for ARGA, and Eq. (28) for ARVGA.

Summary. It should be noted from Eq. (26) that the encoder unit does depend on the specific $\mathbf{s}_i$ vector, which gives the crucial information relating to the local community structure of v_i. Table 6 sums up the main components of these methods, and their architectures are compared as Fig. 5.

5.2 GNN-Based Methods

Several deep embedding strategies are designed to resolve major autoencoder-based method disadvantages by building certain specific functions that rely on the local community of the node but not necessarily the whole graph (Fig. 5 (5)). The GNN, which is widely used in state-of-the-art deep embedding approaches, can be regarded as a general guideline for the definition of deep neural networks on graphs. GNN-based methods usually consists of two primary operations, the aggregate operation and the update operation. In this subsection, we will review basic GNN and some popular GNN extensions with a view of enhancing each process.

Basic GNN. Based on the work of [86], the essential feature of a basic GNN is that it uses neural networks to exchange and update messages between each pair of nodes. Specifically, a hidden embedding $\mathbf{h}_u^{(k)}$ in each neural message passing through the GNN basic iteration is updated according to message or information from the neighborhood within $\mathcal{N}(u)$ according to each node u. This general message can be expressed according to the update rule,

$$\begin{aligned} &\mathbf{h}_u^{(k+1)} \\ &= \text{ Update }^{(k)}(\mathbf{h}_u^{(k)}, \text{ Aggregate }^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})) \\ &= \text{ Update }^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right), \end{aligned} \tag{30}$$

where $\mathbf{m}_{\mathcal{N}(u)}^{(k)} = \text{ Aggregate }^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})$. It is worth noting that the functions *Update* and *Aggregate* must generally be differentiable in Eq. (30). The new state is generated in accordance with Eq. (30) when the neighborhood message is combined with the previous hidden embedding state. After a number of iterations, the last hidden embedding state converges so that each node's final status is created as the output. Formally, we have, $\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$. The basic GNN message passing is defined as follow,

$$\mathbf{h}_u^{(k)} = \sigma(\mathbf{W}_{\text{self}}^{(k)} \mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}), \tag{31}$$

where $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)}$ are trainable parameters and σ is the activation function. In principle, the messages from the neighbors are summarized first. Then, the neighborhood information and the previously hidden node results are integrated by an essential linear combination. Finally, the joint information uses a nonlinear activation function. It is worth noting that the GNN layer can be easily stacked together following Eq. (31). The last layer's output in the GNN model is regarded as the final node embedding result to train a classifier for the downstream SSL tasks.

GCN. As mentioned above, the most simple neighborhood aggregation operation only calculates the sum of the neighborhood encoding states. The most popular method is to use the following symmetric normalization Eq. (32) in the graph convolutional network (GCN) [76],

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}. \tag{32}$$

GCN fully uses the uniform neighborhood grouping technique. Consequently, the GCN model describes the update function as

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right). \tag{33}$$

As it is indirectly specified in the update function, no aggregation operation is defined. Simple Graph Convolution (SGC) [83] is a simplified linear model to reduce the excess complexity of GCNs by repeatedly eliminating the non-linearities. In linearizaiton step, the nonlinear transition functions between layers are consequently removed and the final softmax is kept only to obtain probabilistic output.

MixHop. GCN often fails to learn a generalized class of neighborhood with various mixing relationships. In order to overcome this limitation, MixHop [84] is proposed to learn these relationships by repeatedly mixing feature representations of neighbors at various distances. MixHop replaces the Graph Convolution (GC) layer defined in Eq. (34) with

$$\mathbf{H}^{(k)} = \|_{j \in P} \sigma(A^j \mathbf{H}^{(k-1)} \mathbf{W}_j^{(i)}), \tag{34}$$

where the hyper-parameter P is a set of integer adjacency powers and $\|$ denotes column-wise concatenation. Specifically, by setting $P = \{1\}$, it exactly recovers the original GC layer. In fact, MixHop is interested in higher-order message passing, where each node receives latent representations from their immediate (one-hop) neighbors and from further N-hot neighbors.

GraphSAGE. Over-smoothing is an obvious concern for GNN, which is almost unavailable after several message iterations. A fair way to lessen this concern is skip connection for preserving information directly from the previous rounds of the *update*. GraphSAGE [80] utilizes a

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

concatenation vector to contain more node-level information during a message passage process,

$$\text{Update}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}}(u)) = \left[\text{Update}_{\text{base}}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) \oplus \mathbf{h}_u\right], \quad (35)$$

where the output from the *update* function is concatenated with the previous layer representation of the node.

DGN. To further mitigate the over-smoothing issue in GCN, DGN [85] also applies a new operation between the successive graph convolutional layers. Taking each embedding matrix $H^{(k)}$ generated from the k^{th} graph convolutional layer as the input, DGN assigns each node into different groups and normalizes them independently to output a new embedding matrix for the next layer. Formally, we have,

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \lambda \sum_{i=1}^{g} \left(\gamma_i (\frac{\mathbf{s}_i^{(k)} \circ \mathbf{H}^{(k)} - \mu_i}{\delta_i}) + \beta_i \right), \quad (36)$$

where $\mathbf{H}^{(k)}$ is the k^{th} layer of the hidden embedding matrix, $\mathbf{s}_i$ is the similarity measure and g is the total number of groups. In particular, μ_i and δ_i denote the vectors of running mean of group i, respectively, and γ_i and β_i denote the trainable scale and shift vectors, respectively.

GAT. Neighborhood attention mechanism [87] give each neighbor a weight or value of significance, which is used to weight the influence of this neighbor during the aggregation process. Graph Attention Network (GAT) [81] uses attention weights to describe a weighted neighboring amount,

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v, \quad (37)$$

where $\alpha_{u,v}$ denotes the attention on neighbor $v \in \mathcal{N}(u)$ with aggregating information at node u. The attention weights are,

$$\alpha_{u,v} = \frac{\exp(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v' \in \mathcal{N}(u)} \exp(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}, \quad (38)$$

where **a** is a trainable attention vector, **W** denotes the concatenation operation, which is a trainable matrix.

JK Net. [82] uses the combinations on every layer of the message passing rather than the final layer's output for increasing the effectiveness of final node representations. In a more formal way,

$$\mathbf{z}_u = f_{\text{JK}}(\mathbf{h}_u^{(0)} \oplus \mathbf{h}_u^{(1)} \oplus \cdots \oplus \mathbf{h}_u^{(K)}), \quad (39)$$

where f_{JK} function denotes the identity function for various applications. It indicates that JK Net essentially perform a simple concatenation for the node embedding from each layer.

CGPN. Contrastive Graph Poisson Networks (CGPN) [88] designs a new graph Poisson network (GPN), which allows for a flexible propagation of limited labels through the graph by exploiting structural information. Moreover, contrastive learning is incorporated into variational inference framework to explore additional supervision information from unlabeled examples to help CGPN training.

Summary. We summarize the main differences on GNN-based deep graph embedding methods in Table 7. The main point of graph-based models for DSSL is to perform label inference on a constructed similarity graph so that the label information can be propagated from the labeled samples to the unlabeled ones by incorporating both the topological and feature knowledge.

6 Pseudo-Labeling Methods

In this section, we review the pseudo-labeling methods. There are two main patterns, one is to improve the performance of the whole framework based on the disagreement of views or multiple networks, and the other is self-training. Moreover, the success of self-supervised learning in unsupervised domain makes some self-training self-supervised methods realized.

6.1 Disagreement-Based Models

The idea of disagreement-based SSL is to train multiple learners for the task and exploit the disagreement during the learning process [95]. In such model designs, two or three different networks are trained simultaneously and label unlabeled samples for each other.

Deep Co-Training. Co-training [19] framework assumes each data x in the dataset has two different and complementary views, and each view is sufficient for training a good classifier. Let v_1 and v_2 as two different views of data such that $x = (v_1, v_2)$ (see Fig. 6(1)). Co-training assumes that $\mathcal{C}_1$ as the classifier trained on View-1 v_1 and $\mathcal{C}_2$ as the classifier trained on View-2 v_2 have consistent predictions on $\mathcal{X}$. In the objective function, the Co-training assumption can be model as

$$\begin{aligned}\mathcal{L}_{ct} = H(&\frac{1}{2}(\mathcal{C}_1(v_1) + \mathcal{C}_2(v_2))) \\ &- \frac{1}{2}(H(\mathcal{C}_1(v_1)) + H(\mathcal{C}_2(v_2))),\end{aligned} \quad (40)$$

where $H(\cdot)$ denotes the entropy, the Co-training assumption is formulated as $\mathcal{C}(x) = \mathcal{C}_1(v_1) = \mathcal{C}_2(v_2), \forall x = (v_1, v_2) \sim \mathcal{X}$. On the labeled dataset X_L, the supervised loss function can be the standard cross-entropy loss

$$\mathcal{L}_s = H(y, \mathcal{C}_1(v_1)) + H(y, \mathcal{C}_2(v_2)), \quad (41)$$

where $H(p, q)$ is the cross-entropy between distribution p and q.

The key to the success of Co-training is that the two views are different and complementary. However, the loss function $\mathcal{L}_{ct}$ and $\mathcal{L}_s$ only ensure the model tends to be consistent for the predictions on the dataset. To address this problem, [89] forces to add the View Difference Constraint to the previous Co-training model, and formulated as

$$\exists \mathcal{X}' : \mathcal{C}_1(v_1) \neq \mathcal{C}_2(v_2), \forall x = (v_1, v_2) \sim \mathcal{X},' \quad (42)$$

where $\mathcal{X}'$ denotes the adversarial examples of $\mathcal{X}$, thus $\mathcal{X}' \cap \mathcal{X} = \emptyset$. In the loss function, the View Difference Constraint can be model by minimizing the cross-entropy between $\mathcal{C}_2(x)$ and $\mathcal{C}_1(g_2(x))$, where $g(\cdot)$ denotes the adversarial examples generated by the generative model. Then, this part loss function is

$$\mathcal{L}_{dif}(x) = H(\mathcal{C}_1(x), \mathcal{C}_2(g_1(x))) + H(\mathcal{C}_2(x), p_1(g_2(x))). \quad (43)$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

Tri-Net. Tri-net [90], a deep learning-based method inspired by the tri-training [96]. The tri-training learns three classifiers from three different training sets, which are obtained by utilizing bootstrap sampling. The framework (as shown in Fig. 6(2)) of tri-net can be intuitively described as follows. Output smearing [97] is used to add random noise to the labeled sample to generate different training sets and help learn three initial modules. The three models then predict the pseudo-label for unlabeled data. With the predictions of two modules for unlabeled instances consistent, the pseudo-label is considered to be confident and stable. The labeled sample is added to the training set of the third module, and then the third module is fine-tuned on the augmented training set. During the augmentation process, the three modules become more and more similar, so the three modules are fine-tuned on the training sets respectively to ensure diversity.

Summary. The disagreement-based SSL methods exploit the unlabeled data by training multiple learners, and the "disagreement" among these learners is crucial. When the data has two sufficient redundancy and conditional independence views, Deep Co-training [89] improves the disagreement by designing a View Difference Constraint. Tri-Net [90] obtains three labeled datasets by bootstrap sampling and trains three different learners. These methods in this category are less affected by model assumptions, non-convexity of the loss function and the scalability of the learning algorithms.

6.2 Self-Training Models

Self-training algorithm leverages the model's own confident predictions to produce the pseudo labels for unlabeled data. Entropy Minimization [98] is a strategy of entropy regularization, which can prevent the decision boundary from passing through a high-density data points region. It can be used to realize SSL by encouraging the model to make low-entropy predictions for unlabeled data and then adding the pseudo-labeled samples into labeled data set for a standard supervised learning setting.

Pseudo-Label. Pseudo-label [99] proposes a simple and efficient formulation of training neural networks in a semi-supervised fashion, in which the network is trained in a supervised way with labeled and unlabeled data simultaneously. As illustrated in Fig. 6(3), the model is trained on labeled data in a usual supervised manner with a cross-entropy loss. For unlabeled data, the same model is used to get predictions for a batch of unlabeled samples. The maximum confidence prediction is called a pseudo-label, which has the maximum predicted probability. That is, the pseudo-label model trains a neural network with the loss function $\mathcal{L}$,

$$\mathcal{L} = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{K}\mathcal{R}(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{K}\mathcal{R}(y_i^{'m}, f_i^{'m}), \tag{44}$$

where n is the number of mini-batch in labeled data for SGD, n' for unlabeled data, f_i^m is the output units of m's sample in labeled data, y_i^m is the label of that, $y_i^{'m}$ for unlabeled data, $y_i^{'m}$ is the pseudo-label of that for unlabeled data, $\alpha(t)$ is a coefficient balancing the supervised and unsupervised loss terms.

Noisy Student. Noisy Student [100] proposes a semi-supervised method inspired by knowledge distillation [101] with equal-or-larger student models. The framework is shown in Fig. 6(4). The teacher EfficientNet [102] model is first trained on labeled images to generate pseudo labels for unlabeled examples. Then a larger EfficientNet model as a student is trained on the combination of labeled and pseudo-labeled examples. These combined instances are augmented by RandAugment [103]. Dropout and stochastic depth are also incorporated in the student model during training.

S^4 L. Self-supervised Semi-supervised Learning (S^4 L) [91] tackles the problem of SSL by employing self-supervised learning [104] techniques to learn useful representations from the image databases. The architecture of S^4 L is shown in Fig. 6(5). Predicting image rotation [105] is a pretext task that anticipates an angle of the rotation transformation applied to an input example. In S^4 L, there are four rotation degrees $\{0°, 90°, 180°, 270°\}$ to rotate input images. The S^4 L-Rotation loss is the cross-entropy loss on outputs predicted by those rotated images. S^4 L-Exemplar introduces an exemplar loss which encourages the model to learn a representation that is invariant to heavy image augmentations. Specifically, eight different instances of each image are produced by inception cropping [106], random horizontal mirroring, and HSV-space color randomization as in [107]. Following [104], the loss term on unsupervised images uses the batch hard triplet loss [108] with a soft margin.

MPL. Meta Pseudo Labels (MPL) [92] designs a teacher model that assigns distributions to input examples to train the student model. Throughout the course of the student's training, the teacher observes the student's performance on a held-out validation set, and learns to generate target distributions so that if the student learns from such distributions, the student will achieve good validation performance. The training procedure of MPL involves two alternating processes. As shown in Fig. 6(6), the teacher $g_\phi(\cdot)$ produces the conditional class distribution $g_\phi(x)$ to train the student. The pair $(x, g_\phi(x))$ is then fed into the student network $f_\theta(\cdot)$ to update its parameters θ from the cross-entropy loss. After the student network updates its parameters, the model evaluates the new parameters θ' based on the samples from the held-out validation dataset.

EnAET. Different from the previous semi-supervised methods and S^4 L [91], EnAET [93] trains an Ensemble of Auto-Encoding Transformations to enhance the learning ability. The core part of this framework is that EnAET integrates an ensemble of spatial and non-spatial transformations to self-train a good feature representation [109]. EnAET incorporates four spatial transformations and a combined non-spatial transformation. The spatial transformations are Projective transformation, Affine transformation, Similarity transformation and euclidean transformation. The non-spatial transformation is composed of different colors, contrast, brightness and sharpness with four strength parameters. As shown in Fig. 6(7), EnAET learns an encoder $E : x \rightarrow E(x), t(x) \rightarrow E(t(x))$ on an original instance and its transformations. Meanwhile, a decoder $D : [E(x), E(t(x))] \rightarrow \hat{t}$ is learned to estimate $\hat{t}$ of input transformation. The AutoEncoding Transformation (AET) loss is,

$$\mathcal{L}_{AET} = \mathbb{E}_{x,t(x)}\|D[E(x), E(t(x))] - t(x)\|^2. \tag{45}$$

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

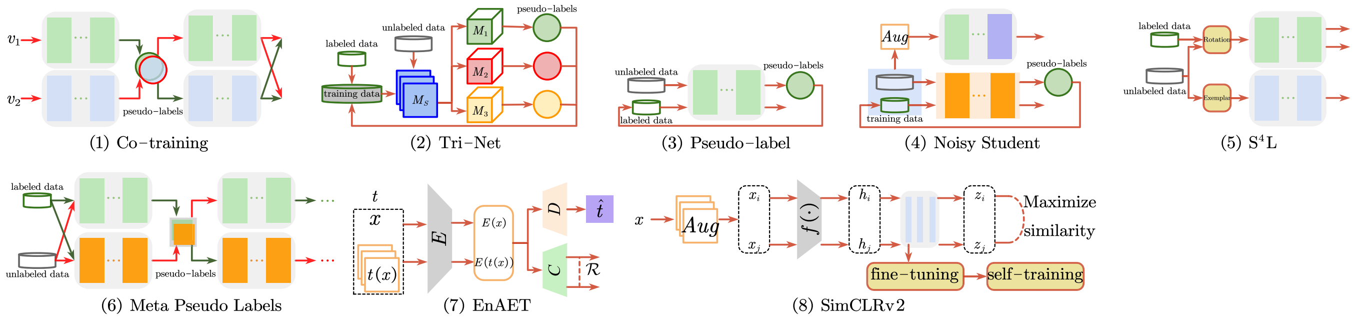


Fig. 6. A glimpse of the diverse range of architectures used for pseudo-label semi-supervised methods. The same color and structure have the same meaning as shown in Fig. 4. M_s denotes shared module, M_1, M_2 and M_3 are three different modules in Tri-Net. “Rotation” and “Exemplar” represent S^4 L-Rotation and S^4 L-Exemplar, respectively.

Apart from the AET loss, EnAET explore a pseudo-labeling consistency by minimizing the KL divergence between $P(y|x)$ on an original sample x and $P(y|t(x))$ on a transformation $t(x)$.

SimCLRv2. SimCLRv2 [94] modifies SimCLR [110] for SSL problems. Following the paradigm of supervised fine-tuning after unsupervised pretraining, SimCLRv2 uses unlabeled samples in a task-agnostic way, and shows that a big (deep and wide) model can superisingly effective for semi-supervised learning. As shown in Fig. 6(8), the SimCLRv2 can be summarized in three steps: unsupervised or self-supervised pre-training, supervised fine-tuning on 1% or 10% labeled samples, and self-training with task-specific unlabeled examples. In the pre-training step, SimCLRv2 learns representations by maximizing the contrastive learning loss function in latent space, and the loss function is constructed based on the consistency of different augmented views of the same example. The contrastive loss is

$$\ell_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2\,N} \mathbb{I}_{[k\neq i]}\exp(sim(z_i, z_k)/\tau)}, \tag{46}$$

where (i, j) is a pair of positive example augmented from the same sample. $sim(\cdot,\cdot)$ is cosine similarity, and τ is a temperature parameter. In the self-training step, SimCLRv2 uses task-specific unlabeled samples, and the fine-tuning network acts as a teacher model to minimize the following distillation loss,

$$\mathcal{L}^{\text{distill}} = -\sum_{x_i\in X}\left[\sum_{y} P^T(y|x_i;\tau)\log P^S(y|x_i;\tau)\right], \tag{47}$$

where $P^T(y|x_i;\tau)$ and $P^S(y|x_i;\tau)$ are produced by teacher network and student network, respectively.

Summary. In general, self-training is a way to get more training data by a series of operations to get pseudo-labels of unlabeled data. Both EntMin [98] and Pseudo-label [99] use entropy minimization to get the pseudo-label with the highest confidence as the ground truth for unlabeled data. Noisy Student [100] utilizes a variety of techniques when training the student network, such as data augmentation, dropout and stochastic depth. S^4 L [91] not only uses data augmentation techniques, but also adds another 4-category task to improve model performance. MPL [92] modifies Pseudo-label [99] by deriving the teacher network's update rule from the feedback of the student network. Emerging techniques (e.g., rich data augmentation strategies, meta-learning, self-supervised learning) and network architectures (e.g., EfficientNet [102], SimCLR [110]) provide powerful support for the development of self-training methods.

7 Hybrid Methods

In recent years, many hybrid methods have been proposed, which combine ideas, such as consistency regularization, data augmentation, entropy minimization, and pseudo-labeling. In this section, we review a series of hybrid methods from MixMatch [112] to FlexMatch [115]. In those hybrid methods, a simple, data-agnostic data augmentation approach, called Mixup [116], is introduced. Formally, Mixup constructs virtual training examples,

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j, \tag{48}$$

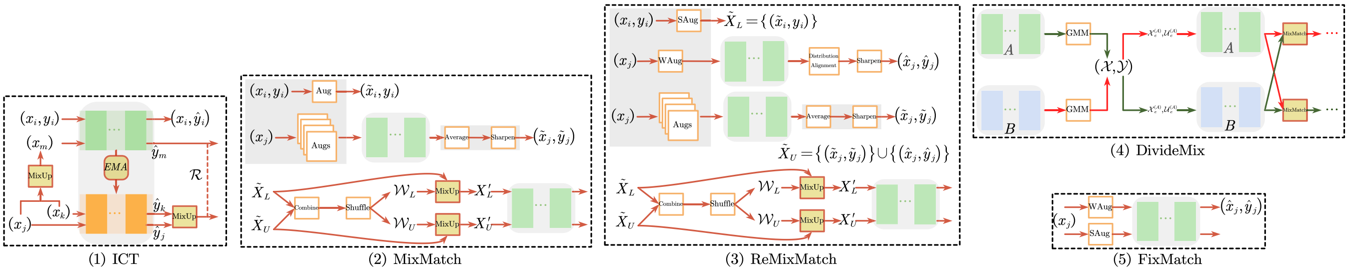
where (x_i, y_i) and (x_j, y_j) are two instances from the training data, and $\lambda \in [0, 1]$. Therefore, Mixup extends the training data set by a hard constraint that linear interpolations of samples should lead to the linear interpolations of the corresponding labels.

ICT. Interpolation Consistency Training (ICT) [111] regularizes SSL by encouraging the prediction at an interpolation of two unlabeled examples to be consistent with the interpolation of the predictions at those points. The architecture is shown in Fig. 7(1). ICT can achieve broad margin decision boundaries under the low-density separation assumption. It is done by training the model $f(\cdot)$ to predict the fake label $\text{Mix}_\lambda(f(\theta,' x_i), f(\theta,' x_j))$ at location $\text{Mix}_\lambda(x_i, x_j)$, where the Mixup operation is $\text{Mix}_\lambda(a, b) = \lambda a + (1-\lambda)b$, and θ' is a moving average of θ, like [31]. Thus, the ICT term is

$$\mathbb{E}_{x\in X}\mathcal{R}(f(\theta, \text{Mix}_\lambda(x_i, x_j)), \text{Mix}_\lambda(f(\theta,' x_i), f(\theta,' x_j))). \tag{49}$$

MixMatch. MixMatch [112] combines consistency regularization and entropy minimization in a unified loss function. As shown in Fig. 7(2), MixMatch produces an augmentation of each labeled instance (x_i, y_i) and K weakly augmented versions of each unlabeled instance (x_j). Then, it generates a pseudo-label $\bar{y}_j$ for each x_j by computing the average prediction across the K augmentations. The pseudo-label distribution is then sharpened by adjusting temperature scaling to get the final pseudo-label $\tilde{y}_j$. After the data augmentation, the batches of augmented labeled examples and unlabeled with pseudo-label examples are combined, then the whole group is shuffled. This group is

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.



(1) ICT (2) MixMatch (3) ReMixMatch (4) DivideMix (5) FixMatch

Fig. 7. A glimpse of the diverse range of architectures used for hybrid semi-supervised methods. “Mixup” is Mixup operator [116]. “MixMatch” is MixMatch [112] in figure (2). “GMM” is short for Gaussian Mixture Model. “SAug” and “WAug” represent Strong Augmentation and Weak Augmentation, respectively.

divided into two parts: the first L samples were taken as $\mathcal{W}_L$, and the remaining taken as $\mathcal{W}_U$. The group $\mathcal{W}_L$ and the augmented labeled batch $\tilde{X}_L$ are fed into the Mixup algorithm to compute examples $(x,' y')$ where $x' = \lambda x_1 + (1 - \lambda)x_2$ for $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$. Similarly, Mixup is applied between the remaining $\mathcal{W}_U$ and the augmented unlabeled group $\tilde{X}_U$. MixMatch conducts traditional fully-supervised training with a standard cross-entropy loss for a supervised dataset and a mean square error for unlabeled data given these mixed-up samples.

ReMixMatch. ReMixMatch [113] extends MixMatch [112] by introducing distribution alignment and augmentation anchoring. Distribution alignment encourages the marginal distribution of aggregated class predictions on unlabeled data close to the marginal distribution of ground-truth labels. Augmentation anchoring replaces the consistency regularization component of MixMatch. As the procedure of ReMixmatch presented in Fig. 7(3), an “anchor” is generated by applying weak augmentation to a given unlabeled example and then K strongly-augmented versions of the same unlabeled example by using a variant of AutoAugment [68] dubbed “CTAugment”.

DivideMix. DivideMix [117] presents a new DSSL framework to improve MixMatch by utilizing co-refinement and co-guessing, and handles the problem of learning with noisy labels. As shown in Fig. 7(4), DivideMix proposes co-divide, a process that trains two networks simultaneously. For each network, a dynamic Gauss Mixed Model (GMM) is fitted on the loss distribution of each sample to divide the training set into labeled data and unlabeled data. The separated data sets are then used to train the next epoch’s networks.

FixMatch. FixMatch [114] combines consistency regularization and pseudo-labeling while vastly simplifying the overall method. As shown in Fig. 7(5), given an instance x_j, FixMatch first generates pseudo-label $\hat{y}_j$ for weakly-augmented unlabeled instances $\hat{x}_j$. Then, the model trained on the weakly-augmented samples is used to predict pseudo-label in the strongly-augmented version of x_j. It attempts to minimize the difference between the two types of augmentations. In FixMatch, weak augmentation is a standard flip-and-shift augmentation strategy, randomly flipping images horizontally with a probability. For strong augmentation, there are two approaches which are based on [68], i.e., RandAugment [103] and CTAugment [113]. Moreover, Cutout [118] is followed by either of these strategies.

FlexMatch. In FixMatch, unlabeled data for each category is selected by a pre-defined constant threshold, which does not take into account for different learning statuses and learning difficulties. In response, FlexMatch [115] designs a curriculum learning approach to utilize unlabeled examples in accordance with the model’s learning state. Basically, this strategy renders the pseudo labels by adjusting thresholds for each class at every step of the process.

Summary. For vision, hybrid methods reach state-of-the-art performances in most benchmark data sets, such as SVHN, CIFAR-10, CIFAR-100, and STL 10. MixMatch [112] is the most basis framework in this series. These hybrid methods seamlessly reduce entropy while maintaining consistency and compatibility with traditional regularization techniques. In recent SSL approaches, data augmentation has been adopted in order to take full advantage of the consistency training framework in both consistency regularization methods and hybrid methods. In Table 10, we summarize the techniques that can be utilized in the consistency training process.

8 Challenges and Future Directions

Although exceptional performance and achieved promising DSSL progress, there are still several open research questions for future work. We outline some of these issues and future directions below.

Theoretical Analysis. Existing semi-supervised approaches predominantly use unlabeled samples to generate constraints and then update the model with labeled data and these constraints. However, the internal mechanism of DSSL and the role of various techniques, such as data augmentations, training methods and loss functions,*etc.*, are not clear. Generally, there is a single weight to balance the supervised and unsupervised loss, which means that all unlabeled instances are equally weighted. However, not all unlabeled data is equally appropriate for the model in practice. To counter this issue, [119] considers how to use a different weight for every unlabeled example. For consistency regularization SSL, [67] investigates how loss geometry interacts with training process. [120] experimentally explores the effects of data augmentation and labeled dataset size on pre-training and self-training, as well as the limitations and interactions of pre-training and self-training. [121] analyzes the property of the consistency regularization methods when data instances lie in the neighborhood of low-dimensional manifolds, especially in the case of efficient data augmentations or perturbations schemes.

Incorporation of Domain Knowledge. Most of the SSL approaches listed above can obtain satisfactory results only in ideal situations in which the training dataset meets the designed assumptions and contains sufficient information to learn an

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

TABLE 10
Summary of Input Augmentations and Neural Network Transformations in Both Consistency Regularization Methods and Hybrid Methods

	Techniques	Methods
Input augmentations	Additional Noise	Ladder Network [62], WCP [66]
	Stochastic Augmentation	Π Model [63], Temporal Ensembling [64], Mean Teacher [31], Dual Student [30], MixMatch [112], ReMixMatch [113], FixMatch [114]
	Adversarial perturbation	VAT [28], VAdD [29]
	AutoAugment	UDA [27], Noisy Student [100]
	RandAugment	UDA [27], FixMatch [114]
	CTAugment	ReMixMatch [113], FixMatch [114]
	Mixup	ICT [111], MixMatch [112], ReMixMatch [113], DivideMix [117]
Neural network transformations	Dropout	Π Model [63], Temporal Ensembling [64], Mean Teacher [31], Dual Student [30], VAdD [29],Noisy Student [100]
	EMA	Mean Teacher [31], ICT [111]
	SWA	SWA [67]
	Stochastic depth	Noisy Student [100]
	DropConnect	WCP [66]

insightful learning system. However, in practice, the distribution of the dataset is unknown and does not necessarily meet these ideal conditions. When the distributions of labeled and unlabeled data do not belong to the same one or the model assumptions are incorrect, the more unlabeled data is utilized, the worse the performance will be. Therefore, we can attempt to incorporate richer and more reliable domain knowledge into the model to mitigate the degradation performance.

Learning With Noisy Labels. In this survey, models discussed typically consider the labeled data is generally accurate to learn a standard cross-entropy loss function. An interesting consideration is to explore how SSL can be performed for cases where corresponding labeled instances with noisy initial labels [122]. For example, the labeling of samples may be contributed by the community, so we can only obtain noisy labels for the training dataset. One solution to this problem is [123], which augments the prediction objective with consistency where the same prediction is made given similar percepts. Based on graph SSL, [124] introduces a new L_1-norm formulation of Laplacian regularization inspired by sparse coding. [125] deals with this problem from the perspective of memorization effects, which proposed a learning paradigm combining co-training and mean teacher.

Imbalanced Semi-Supervised Learning. The problem of class imbalance is naturally widespread in real-world applications. When the training data is highly imbalanced, most learning frameworks will show bias towards the majority class, and in some extreme cases, may completely ignore the minority class [126], as a result, the efficiency of predictive models will be significantly affected. Nevertheless, to handle the semi-supervised problem, it is commonly assumed that training dataset is uniformly distributed over all class labels. Recently, more and more works have focused on this problem. [127] aligns pseudo-labels with the desirable class distribution in the unlabeled data for SSL with imbalanced labeled data. Based on graph-based semi-supervised methods, [128] copes with various degrees of class imbalance in a given dataset.

Robust Semi-Supervised Learning. The common of the latest state-of-the-art approaches is the application of consistency training on augmented unlabeled data without changing the model predictions. One attempt is made by VAT [28] and VAdD [29]. Both of them employ adversarial training to find the optimal adversarial example. Another promising approach is data augmentation (adding noise or random perturbations, CutOut [118], RandomErasing [129], HideAndSeek [130] and GridMask [131]), especially advanced data augmentation, such as AutoAugment [68], RandAugment [103], CTAugment [113], and Mixup [116] which also can be considered as a form of regularization.

Safe Semi-Supervised Learning. In SSL, it is generally accepted that unlabeled data can help improve learning performance, especially when labeled data is scarce. Although it is remarkable that unlabeled data can improve learning performance under appropriate assumptions or conditions, some empirical studies [132], [133], [134] have shown that the use of unlabeled data may lead to performance degeneration, making the generalization performance even worse than a model learned only with labeled data in real-world applications. Thus, safe semi-supervised learning approaches are desired, which never significantly degrades learning performance when unlabeled data is used.

9 Conclusion

Deep semi-supervised learning is a promising research field with important real-world applications. The success of deep learning approaches has led to the rapid growth of DSSL techniques. This survey provides a taxonomy of existing DSSL methods, and groups DSSL methods into five categories: deep generative methods, consistency regularization methods, graph-based methods, pseudo-labeling methods, and hybrid methods. We provide illustrative figures to compare the differences between the approaches within the same category. Finally, we discuss the challenges of DSSL and some future research directions that are worth further studies.

References

[1] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] O. Chapelle, B. Schölkopf, and A. Zien, "Introduction to semi-supervised learning," in *Semi-Supervised Learning*. Cambridge, MA, USA: The MIT Press, 2006, pp. 1–12.

[4] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*. San Rafael, CA, USA: Morgan & Claypool, 2009.

[5] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3239–3250.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

[6] A. K. Agrawala, "Learning with a probabilistic teacher," *IEEE Trans. Inf. Theory*, vol. IT-16, no. 4, pp. 373–379, Jul. 1970.

[7] S. C. Fralick, "Learning to recognize patterns without a teacher," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 57–64, Jan. 1967.

[8] H. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 3, pp. 363–371, Jul. 1965.

[9] D. J. Miller and H. S. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *Proc. 9th Int. Conf. Neural Inf. Process. Syst.*, 1996, pp. 571–577.

[10] K. Nigam, A. McCallum, S. Thrun, and T. M. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, no. 2/3, pp. 103–134, 2000.

[11] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 200–209.

[12] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1998, pp. 368–374.

[13] Z. Xu, R. Jin, J. Zhu, I. King, and M. R. Lyu, "Efficient convex relaxation for transductive support vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1641–1648.

[14] Z. Xu, R. Jin, J. Zhu, I. King, M. R. Lyu, and Z. Yang, "Adaptive regularization for transductive support vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 2125–2133.

[15] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 912–919.

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

[17] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 19–26.

[18] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst.*, 2003, pp. 321–328.

[19] A. Blum and T. M. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.

[20] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA, USA: The MIT Press, 2006.

[21] G. Qi and J. Luo, "Small data challenges in Big Data era: A survey of recent progress on unsupervised and semi-supervised methods," 2019, *arXiv:1903.11260*.

[22] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

[23] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," 2020, *arXiv:2006.05278*.

[24] H. Cevikalp, B. Benligiray, Ö. N. Gerek, and H. Saribas, "Semi-supervised robust deep neural networks for multi-label classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 9–17.

[25] H. Cevikalp, B. Benligiray, and Ö. N. Gerek, "Semi-supervised robust deep neural networks for multi-label image classification," *Pattern Recognit.*, vol. 100, 2020, Art. no. 107164.

[26] Z. Song, X. Yang, Z. Xu, and I. King, "Graph-based semi-supervised learning: A comprehensive review," 2021, *arXiv:2102.13303*.

[27] Q. Xie, Z. Dai, E. H. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 525.

[28] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[29] S. Park, J. Park, S. Shin, and I. Moon, "Adversarial dropout for supervised and semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3917–3924.

[30] Z. Ke, D. Wang, Q. Yan, J. S. J. Ren, and R. W. H. Lau, "Dual student: Breaking the limits of the teacher in semi-supervised learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6727–6735.

[31] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1195–1204.

[32] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2016.

[33] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2226–2234.

[34] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov, "Good semi-supervised learning that requires a bad GAN," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6510–6520.

[35] G. Qi, L. Zhang, H. Hu, M. Edraki, J. Wang, and X. Hua, "Global versus localized generative adversarial nets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1517–1525.

[36] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "Improving the improved training of wasserstein GANs: A consistency term and its dual effect," in *Proc. Int. Conf. Learn. Representations*, 2018.

[37] V. Dumoulin et al., "Adversarially learned inference," in *Proc. Int. Conf. Learn. Representations*, 2017.

[38] A. Kumar, P. Sattigeri, and T. Fletcher, "Semi-supervised learning with GANs: Manifold invariance with improved inference," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5534–5544.

[39] C. Li, T. Xu, J. Zhu, and B. Zhang, "Triple generative adversarial nets," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4088–4098.

[40] S. Wu, G. Deng, J. Li, R. Li, Z. Yu, and H. Wong, "Enhancing TripleGAN for semi-supervised conditional instance synthesis and classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10091–10100.

[41] J. Dong and T. Lin, "MarginGAN: Adversarial training in semi-supervised learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10440–10449.

[42] Z. Gan et al., "Triangle generative adversarial networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5247–5256.

[43] Z. Deng et al., "Structured generative adversarial networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3899–3909.

[44] Y. Liu, G. Deng, X. Zeng, S. Wu, Z. Yu, and H.-S. Wong, "Regularizing discriminative capability of cgans for semi-supervised generative learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5719–5728.

[45] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[46] E. L. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks," 2016, *arXiv:1611.06430*.

[47] A. Odena, "Semi-supervised learning with generative adversarial networks," 2016, *arXiv:1606.01583*.

[48] M. Arjovsky, S. Chintala, and L. Bottou, "GAN Wasserstein," 2017, *arXiv:1701.07875*.

[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[50] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[51] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6022–6031.

[52] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[53] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary deep generative models," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1445–1453.

[54] M. E. Abbasnejad, A. R. Dick, and A. van den Hengel, "Infinite variational autoencoder for semi-supervised learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 781–790.

[55] S. Narayanaswamy et al., "Learning disentangled representations with semi-supervised deep generative models," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5925–5935.

[56] Y. Li, Q. Pan, S. Wang, H. Peng, T. Yang, and E. Cambria, "Disentangled variational auto-encoder for semi-supervised learning," *Inf. Sci.*, vol. 482, pp. 73–85, 2019.

[57] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations*, 2014.

[58] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

[59] J. Schulman, N. Heess, T. Weber, and P. Abbeel, “Gradient estimation using stochastic computation graphs,” in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3528–3536.

[60] T. Joy, S. M. Schmon, P. H. S. Torr, S. Narayanaswamy, and T. Rainforth, “Capturing label characteristics in VAEs,” in *Proc. Int. Conf. Learn. Representations*, 2021.

[61] T. Joy, S. M. Schmon, P. H. S. Torr, N. Siddharth, and T. Rainforth, “Rethinking semi-supervised learning in VAEs,” 2020, *arXiv:2006.10102*.

[62] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, “Semi-supervised learning with ladder networks,” in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3546–3554.

[63] M. Sajjadi, M. Javanmardi, and T. Tasdizen, “Regularization with stochastic transformations and perturbations for deep semi-supervised learning,” in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1163–1171.

[64] S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in *Proc. Int. Conf. Learn. Representations*, 2017.

[65] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, “Averaging weights leads to wider optima and better generalization,” in *Proc. 34th Conf. Uncertainty Artif. Intell.*, 2018, pp. 876–885.

[66] L. Zhang and G. Qi, “WCP: Worst-case perturbations for semi-supervised deep learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3911–3920.

[67] B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson, “There are many consistent explanations of unlabeled data: Why you should average,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[68] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “AutoAugment: Learning augmentation strategies from data,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.

[69] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical data augmentation with no separate search,” 2019, *arXiv:1909.13719*.

[70] R. Sennrich, B. Haddow, and A. Birch, “Improving neural machine translation models with monolingual data,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 86–96.

[71] S. Edunov, M. Ott, M. Auli, and D. Grangier, “Understanding back-translation at scale,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2018, pp. 489–500.

[72] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1225–1234.

[73] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.

[74] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1105–1114.

[75] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, *arXiv:1611.07308*.

[76] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. Int. Conf. Learn. Representations*, 2017.

[77] Y. Jin and J. F. JáJá, “Learning graph-level representations with gated recurrent neural networks,” 2018, *arXiv:1805.07683*.

[78] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, “Deep recursive network embedding with regular equivalence,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2357–2366.

[79] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, and C. Zhang, “Learning graph embedding with adversarial training methods,” *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2475–2487, Jun. 2020.

[80] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[81] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *Proc. Int. Conf. Learn. Representations*, 2018.

[82] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 5449–5458.

[83] F. Wu, A. H. Souza Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[84] S. Abu-El-Haija et al., “MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 21–29.

[85] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, “Towards deeper graph neural networks with differentiable group normalization,” in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 413.

[86] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[87] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. Int. Conf. Learn. Representations*, 2015.

[88] S. Wan, Y. Zhan, L. Liu, B. Yu, S. Pan, and C. Gong, “Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 6316–6327.

[89] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. L. Yuille, “Deep co-training for semi-supervised image recognition,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 142–159.

[90] D. Chen, W. Wang, W. Gao, and Z. Zhou, “Tri-net for semi-supervised deep learning,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2014–2020.

[91] L. Beyer, X. Zhai, A. Oliver, and A. Kolesnikov, “S4L: Self-supervised semi-supervised learning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1476–1485.

[92] H. Pham, Q. Xie, Z. Dai, and Q. V. Le, “Meta pseudo labels,” 2020, *arXiv:2003.10580*.

[93] X. Wang, D. Kihara, J. Luo, and G. Qi, “EnAET: A self-trained framework for semi-supervised and supervised learning with ensemble transformations,” *IEEE Trans. Image Process.*, vol. 30, pp. 1639–1647, 2021.

[94] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, “Big self-supervised models are strong semi-supervised learners,” in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1865.

[95] Z. Zhou and M. Li, “Semi-supervised learning by disagreement,” *Knowl. Inf. Syst.*, vol. 24, no. 3, pp. 415–439, 2010.

[96] Z. Zhou and M. Li, “Tri-training: Exploiting unlabeled data using three classifiers,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.

[97] L. Breiman, “Randomizing outputs to increase prediction accuracy,” *Mach. Learn.*, vol. 40, no. 3, pp. 229–242, 2000.

[98] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization,” in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, 2004, pp. 529–536.

[99] D.-H. Lee, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” *Proc. Workshop Challenges Representation Learn.*, vol. 3, no. 2, p. 896, 2013.

[100] Q. Xie, M. Luong, E. H. Hovy, and Q. V. Le, “Self-training with noisy student improves ImageNet classification,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10684–10695.

[101] G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, *arXiv:1503.02531*.

[102] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[103] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 3008–3017.

[104] A. Kolesnikov, X. Zhai, and L. Beyer, “Revisiting self-supervised visual representation learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1920–1929.

[105] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” in *Proc. Int. Conf. Learn. Representations*, 2018.

[106] C. Szegedy et al., “Going deeper with convolutions,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[107] A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with convolutional neural networks,” in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 766–774.

[108] A. Hermans, L. Beyer, and B. Leibe, “In defense of the triplet loss for person re-identification,” 2017, *arXiv:1703.07737*.

[109] L. Zhang, G. Qi, L. Wang, and J. Luo, “AET vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2542–2550.

[110] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, “A simple framework for contrastive learning of visual representations,” in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.

[111] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3635–3641.

[112] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "MixMatch: A holistic approach to semi-supervised learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5050–5060.

[113] D. Berthelot et al., "ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring," in *Proc. Int. Conf. Learn. Representations*, 2020.

[114] K. Sohn et al., "FixMatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 51.

[115] B. Zhang et al., "FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 18408–18419.

[116] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018.

[117] J. Li, R. Socher, and S. C. H. Hoi, "DivideMix: Learning with noisy labels as semi-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[118] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[119] Z. Ren, R. A. Yeh, and A. G. Schwing, "Not all unlabeled data are equal: Learning to weight data in semi-supervised learning," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21786–21797, 2020.

[120] B. Zoph et al., "Rethinking pre-training and self-training," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 323.

[121] A. Ghosh and A. H. Thiery, "On data-augmentation and consistency-based semi-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2021.

[122] C. Zhang et al., "ACE: A coarse-to-fine learning framework for reliable representation learning against label noise," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2022, pp. 1–8.

[123] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. Int. Conf. Learn. Representations*, 2015.

[124] Z. Lu and L. Wang, "Noise-robust semi-supervised learning via fast sparse coding," *Pattern Recognit.*, vol. 48, no. 2, pp. 605–612, 2015.

[125] B. Han et al., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8536–8546.

[126] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, 2019, Art. no. 27.

[127] J. Kim, Y. Hur, S. Park, E. Yang, S. J. Hwang, and J. Shin, "Distribution aligning refinery of pseudo-label for imbalanced semi-supervised learning," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1221.

[128] J. Deng and J. Yu, "A simple graph-based semi-supervised learning approach for imbalanced classification," *Pattern Recognit.*, vol. 118, 2021, Art. no. 108026.

[129] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 13001–13008.

[130] K. K. Singh, H. Yu, A. Sarmasi, G. Pradeep, and Y. J. Lee, "Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond," 2018, *arXiv:1811.02545*.

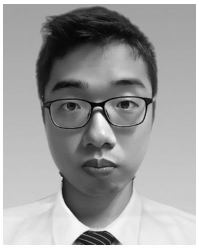
[131] P. Chen, S. Liu, H. Zhao, and J. Jia, "GridMask data augmentation," 2020, *arXiv:2001.04086*.

[132] A. Singh, R. D. Nowak, and X. Zhu, "Unlabeled data: Now it helps, now it doesn't," in *Proc. 21st Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1513–1520.

[133] T. Yang and C. E. Priebe, "The effect of model misspecification on semi-supervised classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 10, pp. 2093–2103, Oct. 2011.

[134] N. V. Chawla and G. I. Karakoulas, "Learning from labeled and unlabeled data: An empirical study across techniques and domains," *J. Artif. Intell. Res.*, vol. 23, pp. 331–366, 2005.

Xiangli Yang is currently working toward the PhD degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. Her research interests are semi-supervised learning, data mining, and machine learning.



Zixing Song received the BEng degree in computer science and engineering from Southeast University, Nanjing, China. He is currently working toward the PhD degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests are machine learning on graphs, semi-supervised learning, and social computing.

Irwin King (Fellow, IEEE) received the BSc degree in engineering and applied science from the California Institute of Technology (Caltech), Pasadena, and the MSc and PhD degrees in computer science from the University of Southern California (USC), Los Angeles. He is the chair and professor of computer science & engineering with The Chinese University of Hong Kong. His research interests include machine learning, social computing, AI, web intelligence, data mining, and multimedia information processing. In these research areas, he has more than 300 technical publications in journals and conferences. He is an associate editor of the *Journal of Neural Networks (NN)*. He is an ACM distinguished member, and a fellow of Hong Kong Institute of Engineers (HKIE). He has served as the president of the International Neural Network Society (INNS), general co-chair of The WebConf 2020, ICONIP 2020, WSDM 2011, RecSys 2013, ACML 2015, and in various capacities in a number of top conferences and societies such as WWW, NIPS, ICML, IJCAI, AAAI, APNNS, *etc.* He is the recipient of the ACM CIKM 2019 Test of Time Award, the ACM SIGIR 2020 Test of Time Award, and 2020 APNNS Outstanding Achievement Award for his contributions made in social computing with machine learning. In early 2010 while on leave with AT&T Labs Research, San Francisco, he taught classes as a visiting professor with UC Berkeley.

Zenglin Xu (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Chinese University of Hong Kong. He is currently a full professor with the Harbin Institute of Technology, Shenzhen. He has been working with Michigan State University, Cluster of Excellence with Saarland University & Max Planck Institute for Informatics, Purdue University, and University of Electronic Science & Technology of China. His research interests include machine learning and its applications in computer vision, health informatics, and natural language processing. He currently serves as an associate editor of *Neural Networks* and *Neurocomputing*. He is the recipient of the outstanding student paper honorable mention of AAAI 2015, the best student paper runner up of ACML 2016, and the 2016 young researcher award from APNNS.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: NIST Virtual Library (NVL). Downloaded on October 04,2023 at 17:08:20 UTC from IEEE Xplore. Restrictions apply.