A Survey of Unsupervised Generative Models for Exploratory Data Analysis and Representation Learning

MOHANAD ABUKMEIL, STEFANO FERRARI, ANGELO GENOVESE, VINCENZO PIURI, and FABIO SCOTTI, Università degli Studi di Milano, Italy

For more than a century, the methods for data representation and the exploration of the intrinsic structures of data have developed remarkably and consist of supervised and unsupervised methods. However, recent years have witnessed the flourishing of big data, where typical dataset dimensions are high and the data can come in messy, incomplete, unlabeled, or corrupted forms. Consequently, discovering the hidden structure buried inside such data becomes highly challenging. From this perspective, exploratory data analysis (EDA) plays a substantial role in learning the hidden structures that encompass the significant features of the data in an ordered manner by extracting patterns and testing hypotheses to identify anomalies. Unsupervised generative learning (UGL) models are a class of Machine Learning (ML) models characterized by their potential to reduce the dimensionality, discover the exploratory factors, and learn representations without any predefined labels; moreover, such models can generate the data from the reduced factors' domain. The beginner researchers can find in this survey the recent UGL models for the purpose of data exploration and learning representations; specifically, this paper covers three families of methods based on their usage in the era of big data: blind source separation (BSS), manifold learning (MfL), and Neural Networks (NNs), from shallow to deep architectures.

CCS Concepts: • **Computing methodologies → Unsupervised learning**.

Additional Key Words and Phrases: Blind Source Separation, Manifold Learning, Neural Networks, Exploratory Data Analysis, Representation Learning, Explainable Machine Learning, Unsupervised Deep Learning

ACM Reference Format:
Mohanad Abukmeil, Stefano Ferrari, Angelo Genovese, Vincenzo Piuri, and Fabio Scotti. 2021. A Survey of Unsupervised Generative Models for Exploratory Data Analysis and Representation Learning. *ACM Comput. Surv.* 1, 1 (February 2021), 37 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The rapid advancement of ubiquitous computing, in which computerized systems are located everywhere, presents challenges related to the nature of data and their latent structures, which contain significant features; these challenges arise due to the large-scale and high-speed interactions that are encountered in both physical scenarios, such as climate data flows, and virtual environments, such as social and financial data [64]. In particular, the main challenges lie in the massive data volumes of typical datasets; such datasets may consist of billions of observations or entries that are represented by high-dimensional blocks of matrices [43, 104]. Typically, for big data analysis,

This work was supported in part by the EC within the H2020 Program under projects MOSAICrOWN and MARSAL, by the Italian Ministry of Research within the PRIN program under project HOPE, by the Università degli Studi di Milano under project AI4FAO, and by JPMorgan Chase & Co. We thank the NVIDIA Corporation for the GPU donated.
Authors' address: Mohanad Abukmeil, mohanad.abukmeil@unimi.it; Stefano Ferrari, stefano.ferrari@unimi.it; Angelo Genovese, angelo.genovese@unimi.it; Vincenzo Piuri, vincenzo.piuri@unimi.it; Fabio Scotti, fabio.scotti@unimi.it, Università degli Studi di Milano, Via Giovanni Celoria, 18, 20133, Milan, Italy.

0360-0300/2021/2-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

the most difficult tasks include uncovering the hidden structures of these massive datasets, which contain the latent features of the data. Furthermore, the causality that explains the behavior of the data must be interpreted, especially if they include *messy data*, meaning that the organization of data belonging to the same category is not standardized and there is clear variability among entries; *missing data*, meaning that the observations are incomplete and there is inconsistency among the variables; or *corrupted data*, meaning that there is consistency among observations but some entries are damaged [218]. Moreover, the demand to disentangle the exploratory factors of data to reduce the processing time and produce interpretable results is currently very important [42]. Accordingly, exploring data to extract the latent features and factors that constitute their intrinsic structure offers the ability to address data inflation; thus, learning from the data can be achieved in a flexible manner to build manageable visualizers, predictors, and classifiers [36, 37, 203].

To build a machine learning (ML) model, it is essential to identify the format that constitutes the data samples [116]. The data format is categorized into three main groups according to its type including *(i)* structured data, that is organized in a specific form and it has a defined length as in time-series and tabular data [109]; *(ii)* unstructured data, that lacks the structure as images, videos, and text data [106]; and *(iii)* semi-structured data, in which graph representations and adjacency relations among data samples are exploited, as in the social media data that comprises both structured and unstructured forms [148]. Usually, in ML models the learning is carried out from different abstraction spaces, that are constituted from data features and can be detected by a set of representations [19]; where such representations are inverted to generate the original data from the abstraction spaces. The data representations that are commonly used to detect and extract latent features are derived from algebra, probability approximations, statistical analysis, and graph theory; moreover, they are usually integrated with ML models to learn data distributions and structures, differentiate patterns, and facilitate visualization and decision making [18, 115, 187].

Unsupervised learning (UL) models are a class of ML tools for discovering data distributions, learning the underlying structures of data, removing redundancies, and reducing the dimensionality of massive datasets without any predefined labels [10, 44, 109, 149, 161]. In UL models, it is assumed that the set of observations (or samples) X ($X \subset \mathbb{R}^D$, where D is the original dimensionality) can be algebraically represented by special values called "eigenvalues", denoted by Λ, and their corresponding vectors, called "eigenvectors" [112]. Alternatively, X can be addressed probabilistically by supposing that the set of samples X has a joint distribution $P(X)$ ($X = \{x_i \mid x_i \in \mathbb{R}^D,\ i = 1, \ldots, N\}$); then, analysis can be performed based on that joint distribution $P(x_1, x_2, ..., x_N)$ [105]. X can also be statistically represented based on n^{th}-order statistical moments, e.g., the mean μ and variance σ [19, 113]. Moreover, X can be graphically represented as a set of nodes $v = 1, .., N$, and relations between data samples X are represented by edges ε to view the data as an undirected or directed graph $\mathcal{G}\{v, \varepsilon\}$, where the graph is characterized by its adjacency matrix $A \in \mathbb{R}^{N \times N}$ [24, 116, 175].

UL is usually applied in combination with generative modeling, where the task is to model the distribution $P(X)$ from which the original data are drawn [158] utilizing a density function f and a set of model parameters θ, in the form $f(X|\theta)$. Generative models are defined as methods or techniques that are able to transform data into a code or subset of codes from which the data can be reconstructed [85]. Moreover, the aim of early UL methods was to fit generative models to the given data with a high likelihood $P(\theta|X)$, by eliminating redundancy and reducing the dimensionality through data preprocessing, thus allowing useful representations to be obtained [183]. As a statistical preprocessing, normalization utilizes the first- and second-order moments, i.e., the mean (μ, or $E[X]$ for discrete data) and the variance (σ), respectively, to characterize data on a fine-scale or in a white space [94]. However, normalization alone is not an efficient means of removing redundancies, reducing data dimensionality, or facilitating representation learning.

Therefore, we focus on the ability of UL-based generative models (UGL) to be used in exploratory data analysis (EDA), reduce learning complexity, and improve accuracy as well as their potential to produce explainable results to elucidate causality relations and facilitate decision making [5, 74].

Recent deep learning (DL) models comprise hundreds of linear and nonlinear layers, thus different representations at different levels of abstraction spaces can be learned. However, the performance decreases when multiple layers are trained simultaneously, and they are susceptible to diminishing features reuse and overfitting, i.e., learning too many specific details and noise from the data, which can negatively affect the model performance [84]. To mitigate these shortcomings, various approaches have been introduced in the literature and will be highlighted in this survey (also in the supplementary material of this survey) including *(i)* unsupervised pretraining by which data and the learning parameters are regulated to be in appropriate forms and ranges [81]; *(ii)* intelligent weights initialization as in using orthogonal weights W, i.e., $W^TW = I$ [91]; *(iii)* minibatch normalization and dropout by deactivating some neurons in accordance with a predefined probability [173]; *(iv)* knowledge transfer and domain adaptation by utilizing representations from related domains [162]; *(v)* and skip connections (residual learning) by adding layers that apply an identity mapping $f(X)$ to copy early layers X [77]. Consequently, the generalizability for unseen data is improved according to the above approaches; specially, when building UL models for data visualization and distribution encoding to make a prior knowledge [181, 183], dimensionality reduction to remove the redundancy and address missing data [2, 57], recognition tasks [147], or proposing new representations learning strategy as in Generative Adversarial Networks GANs based on min-max losses approach [89].

1.1 Connection to Existing Surveys

Related surveys that focus on unsupervised generative learning (UGL) models for data analysis and representation learning (rather than only deep learning models) have not thoroughly covered the recent aspects and challenges of interest in this field. Indeed, only a few articles have reviewed and discussed these topics, such as [19, 218]. A comprehensive survey of unsupervised representation learning was presented in 2013 [19], which reviewed probabilistic models, manifold learning (MfL), and deep learning models. By contrast, the overview presented in [218] was brief, and the taxonomy of methods was restricted to traditional and deep learning models, excluding state-of-the-art models. The history of deep learning based on neural networks (NNs) was addressed in 2014 [162] through a review of supervised learning, UL, reinforcement learning, and evolutionary computation.

To the best of our knowledge and based on a literature analysis, this paper is the first work with the goal of making interested researchers aware of the current status of the field of unsupervised generative methods by reviewing the most common and state-of-the-art models, in contrast to recent surveys that have targeted specific disciplinary perspectives, e.g., data science excluding representation learning procedures [36], deep-learning-based NNs [147], and GANs [89].

Consequently, we present an overview of three families of UGL models. The first is blind source separation (BSS) and its extensions, which are intended for extracting representative factors from massive datasets. The second family of models considered here consists of MfL-based generative models, which can accommodate data in joint spaces and offer powerful strategies for visualizing low-dimensional embeddings of the original data. Finally, we present a review of generative NNs models, from shallow to deep, and show the advantages achieved by integrating these three categories of models for effective representation learning. Our aim is to present major and prevalent UGL models at a level of detail that will be useful for interested and beginning researchers.

1.2 Relevant Application Domains

Currently, UGL models offer valuable capabilities for exploring and learning datasets. These capabilities have led to success in ML applications in which such models are used at an early stage

to prepare the data and then learn representations thereof regardless of whether the samples are unstructured, structured, or semi-structured (See section 1). Table 1 highlights the existing research in terms of the application domains and a finer-grained taxonomy of the reviewed models. Moreover, we present other application domains and remarks in the supplementary materials.

Table 1. Application domains and finer-grained categorization of the reviewed UGL models, namely, blind source separation (BSS) models, manifold learning (MfL) models, and neural network (NN)-based models.

Family	Category	Application Domains
BSS	Statistical	Missing data [178], Multivariate data analysis [196], Dimensionality reduction [2, 65], Pattern recognition [122], Clustering [144], Polynomial feature extraction [163], Signal unmixing and separation [93], Coding and decoding [87]
	Structure	Data mining [37], Hyperspectral image processing and signal separation [43], Explainable artificial intelligence and low-rank approximation [1, 4], Tensor data learning [104], Missing data and collaborative filtering [16], Dimensionality reduction[42], Object recognition [37]
	Probabilistic	Latent data mining [134], Topic modeling [29, 72], Topographic mapping [6], Clustering and Dimensionality reduction [211, 215]
MfL	Global	Global data visualization [177], Dimensionality reduction [112], Clustering and interpolation [172, 212], Global joint mapping [51], Graph and multimedia analysis [88], Classification and regression [202], Time series analysis [199], Topological analysis [9]
	Local	Local data visualization [182], Data behavior analysis [156], Dimensionality reduction [183], Clustering and interpolation [145], Local joint mapping [192, 213], Social and biological analysis [201], Object recognition [182], Graph and tensor analysis [119]
NN	Energy	Model pretraining and representation normalization [84], Object reconstruction [3], Object recognition [41, 130], Large-scale image classification [115], Speech recognition [133], Image and speech interpolation [159], Multimodal learning [174]
	Autoencoder	Encoding/decoding [157], Dimensionality reduction [84], Image segmentation [200], Collaborative filtering [207], Cybernetics [197], Object recognition [100], Natural language processing [170], Disentangled learning [78]
	Adversarial	Data generation, classification, and synthesis [70, 143], Image denoising and deraining [190], Image-to-image translation [219], Multiclass labeling [131], High-resolution natural image learning [52, 58, 188], Multimodal image synthesis [143], Object recognition and shape learning [198], Natural language processing [121], Graph learning[71]

1.3 Notations and Formalization

UGL models have been studied in several fields; consequently, the formalization used in the literature can differ, even when referring to similar learning concepts. To assist the reader, we will use a uniform formalization and mathematical notations throughout this overview, as summarized in Table 2. In addition, although the techniques and learning paradigms reviewed in this paper can be applied to datasets consisting of elements with a complex structure, our explanations will consider only the simplest form: a dataset of arrays of features. Without loss of generality, the data are given as a dataset $X \subset \mathbb{R}^D$: $X = \{x_i \mid x_i \in \mathbb{R}^D,\ i = 1, \ldots, N\}$, where D is the dimensionality of the original space and N is the number of data samples. Alternatively, the dataset X can be represented as a matrix (or second-order tensor), where the i^{th} row is the i^{th} element, x_i, and each column represents a feature point of a sample. The dataset can also be expressed in a higher-order tensor form as $\mathcal{X}_{i,j,k}$, where a depth variable k is introduced. The UGL models described in Section 2 and Section 3 aim to generate a compact representation of the data in a lower-dimensional subspace $Z \subset \mathbb{R}^d$ through the application of a suitable mapping, $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ (where $d << D$), to the original data as follows: $Z = f(X) = \{z_i = f(x_i) \mid i = 1 \ \ldots, M\}$, where M is the number of mapped features in the embedding space. In Section 4, such a mapping can be seen as an encoding of the data belonging to the original space, as in the case of autoencoder models, which also employ a corresponding decoding function to reconstruct the original data as follows: $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$. Moreover, for the

reviewed models, the learning process is optimized depending on the distribution of the data, for which descriptive statistics of different orders can be exploited to measure the similarity of different data distributions. Additionally, for some models, probabilistic measures are used to optimize the learning process; for example, the Kullback-Leibler (KL) divergence [108] can be utilized to measure how far a reconstructed distribution q is from the original distribution p. The KL divergence is defined as $\mathrm{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)}\, \mathrm{d}x$, which is always nonnegative and tends to zero if and only if p and q are almost equal in the distributions. Because of space limitations, we refer the reader to the supplementary materials that accompany this survey for further mathematical details.

Table 2. List of notations and mathematical symbols used in the paper.

Notation	Definition or Description	Notation	Definition or Description
$X \in \mathbb{R}^{N\times D}$	Dataset, matrix, or second-order tensor	D	Dimensionality of the original space
$\mathcal{X} \in \mathbb{R}^{n\times m\times p}$	Three-way dataset or third-order tensor	n, m, p	Indices of row, column, and depth
N	Number of original data samples	M	Mapped features in the embedded space
$Z \subset \mathbb{R}^d$	Lower-dimensional space or latent space	r	Data rank
$f : \mathbb{R}^D \to \mathbb{R}^d$	Mapping or encoding function	C_X	Covariance matrix
$g : \mathbb{R}^d \to \mathbb{R}^D$	Reconstruction or decoding function	$\tilde{X}$	Reconstructed data after learning
d	Embedding or latent spaces dimension	$\|.\|_F$	Frobenius norm
$E[X]$	Expectation value of a random variable	$\mathrm{KL}(X\|\tilde{X})$	Kullback-Leibler (KL) divergence
U	Left singular vectors, orthonrmal matrix	V	Right singular vectors, orthonrmal matrix
Σ	Diagonal matrix of singular values	I	Identity matrix
$J, H(X)$	Negentropy, differential entropy	A	Mixing or reconstruction coefficients
$\circ$	Composition of linear mappings	$\otimes$	Outer or tensor product
$\mathcal{Z}$	Core tensor	λ	Eigenvalue or scaling factor
Λ	Eigenvalues in a diagonal matrix	$\mathcal{G}$	Weighted or finite graph
ϵ	Radius or variational parameter	W, θ	Learning weights or parameters
h	Hidden variable	$P(V), P(X)$	Prior distributions
$P(v\|h)$	Conditional probability	$P(v, h)$	Joint probability

1.4 Outline and Article Organization

The methodologies built around UGL are quite extensive, and the application domains are numerous; thus, it is impossible to cover all of them in this survey. Moreover, our taxonomy and model selection that are integrated into the treatment of this work reflect our own research interests and backgrounds in this attractive field. Realizing that, our aim is to pave the way for beginner researchers by highlighting the basic models and recent research trends. The remainder of this overview is organized as follows. Section 2 reviews three subsets of BSS models, namely, statistical, structural, and probabilistic-based approaches. Moreover, the same section highlights the importance of such methodologies for disentangling the main and latent factors influencing data. Section 3 summarizes MfL models, separated into global and local methodologies, for constructing low-dimensional embeddings. Additionally, the same section discusses the value of such models in capturing meaningful representations and reconstructing messy and corrupted data. Section 4 presents a review of NNs-based models, from shallow to deep, including energy-, autoencoders-, and adversarial learning-based models. Finally, for sake of space, remarks, open issues, and future research directions are highlighted in the supplementary materials that accompany to this survey.

2 Blind Source Separation Models

Typically, massive-scale datasets are naturally aggregated from several sources; they are also inherently heterogeneous and may include messy, missing, or corrupted data. Accordingly, disentangling the exploratory factors from such datasets to reduce their dimensionality, understand the causality of their behavior and the natural interactions among the data, and visualize the latent features are extremely important tasks for achieving flexible learning based on suitable representations.

BSS models are a set of techniques that are valuable for splitting up mixed sources, signals, and data points [220]. Furthermore, BSS methods share the same goal of using the original data and subjecting them to a set of operations such as n^{th}-order statistical analysis, matrix decomposition, and probabilistic estimation. BSS methods are also closely associated with unsupervised representations, through which the hidden structures of the data or data sources can be discovered, separated, and isolated. Consequently, BSS methods can be used to decompose the original data samples or points in a massive-scale dataset into a set of linear combinations, thereby facilitating learning and reducing computational complexity [17]. In the following, we provide a brief summary of the three main types of BSS methods: those that are derived from statistical analysis, those related to structural or matrix decomposition, and those that are derived from probabilistic modeling (PBMs).

2.1 Statistics-Based Models

Statistics-based models (SBMs) are a class of BSS models that depend on statistical moments that provide information about the variability and location (dispersion or spread) of a set of data points for feature X; they also offer insight into the data distribution. Moreover, such moments are widely used in statistical data exploration to describe the characteristics of estimated parameters θ or sets of data points X. For a given dataset (or matrix) $X \subset \mathbb{R}^D$, $X = \{x_i \mid x_i \in \mathbb{R}^D,\ i = 1, \ldots, N\}$, the first moment is the mean of the data, $\mu = \Sigma_1^N \frac{x_i}{N}$; the second moment is the variance, $\sigma = \Sigma(x_i - \mu)^2$; the third moment is the skewness, $sk = \Sigma_1^N \frac{x_i^3}{N}$, which is described in terms of the normalized variance; and the fourth moment is the kurtosis, $kr = \Sigma_1^N \frac{x_i^4}{N}$, which is regarded as a high-order statistic and can be obtained from the normalized version of the third moment [108].

SBMs are utilized in different applications to remove correlations, feature extraction, and dimensionality reduction. In the following, we highlight the two basic methods: principal component analysis (PCA) [144] and independent component analysis (ICA) [45]. These techniques are interrelated with the forthcoming sections, as in the PCA connection to Sections 3.1 and 4.2.

2.1.1 Principal Component Analysis PCA, proposed by Pearson [144], was one of the earliest methods used to determine the correlations among data and to reduce the dimensionality of the original space by projecting the data $X \subset \mathbb{R}^D$ into a lower-dimensional subspace $Z \subset \mathbb{R}^d$, $d << D$. In the formalization defined in the supplementary materials, the PCA attempts to find a mapping f that preserves, to the greatest extent possible, the variance (second moment) of the original data points once they are projected into the reduced space Z. Without loss of generality, in the following, X can be taken to be a zero-mean (normalized) data matrix, corresponding to $X - \mu_X$. Because the variance of a feature can be influenced by the measurement units used to express it, a commonly used preprocessing procedure for X is normalization. PCA is applied in various domains, and its mathematical and statistical properties can be derived using different conceptual tools.

Mathematically, in PCA, the mapping $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ generates a linear transformation matrix $U \in \mathbb{R}^{D \times d}$, whose columns map each data vector x_i to corresponding elements z_i, called the principal components of x_i: $Z = XU$. Because the columns of U are constrained to be unit vectors that are normal to each other ($U^\intercal U = I$, to restrict the orthogonal lengths between singular values and bound the optimization to avoid a quadratic explosion of the optimization), they constitute an orthonormal basis for the mapping domain Z. The projection problem can be treated as an optimization problem, $U = \arg\max_{\hat{U}} \hat{U}^\intercal C \hat{U}$, where C_X is the covariance matrix of X, with the constraint that the variances of the transformed dataset along the U basis directions are sorted in decreasing order *w.r.t.* the basis element order. The features in Z are hence uncorrelated and capture most of the variability of X. It can be shown that the columns of U can be formed from the eigenvectors of C_X (which describe the directions of maximum variance) with the d largest

eigenvalues Λ. The usage of the covariance matrix, $C_X = \frac{1}{N-1} X^\intercal X$, allows PCA to be applied for statistical analysis tasks. Finally, the inverse of the mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$ generates, or reconstruct, the original data X from its mapped space Z as follows: $\tilde{X} = g(Z) = ZU^\intercal = XUU^\intercal$.

PCA can also be derived on the basis of geometrical considerations [98]. Here, the process can be iteratively described as finding one orthonormal basis component u_i at a time. The direction of the first component, u_1, is defined by the best-fit line (which passes through the origin): this is the line such that the average projection ($pro_x u = \frac{x.u}{||x||^2}$, the projection of the data onto u) residual of the original data (i.e., the mean squared distance of the original data from their projections onto u), expressed as $MSD = \frac{1}{N}\Sigma_{i=1}|x_i - pro_x u|^2$, is minimized. The other basis components can be found by generalizing the above approach to a search for the best-fit line for the differences between the original data and their thus-far-computed principal components. The unit vectors along the best-fit lines and the sum of the squared distances correspond to the eigenvectors and eigenvalues, respectively, in the mathematical definition of PCA presented above. Another property of PCA is that U is the projection in a d-dimensional subspace Z that maximizes the expected Euclidean distance between pairs of projected vectors (i.e., the projected vectors are as spread out as possible).

Several extensions of the PCA method have been reported in the literature. Among others, dual PCA [122], kernel PCA [163], and probabilistic relational PCA [118, 178] have been developed. Dual PCA relies on decomposing the right singular vectors obtained through singular value decomposition (SVD) (further details on SVD can be found in Section 2.2.1). Kernel PCA can be used for nonlinear data reduction. In kernel PCA, a mapping function $\phi(\cdot)$ is used to spread the data X throughout the high-dimensional space in which PCA is performed; a proper kernel function involving the inner product in the ϕ space enables the computation of the eigenvectors and eigenvalues of the covariance matrix in that high-dimensional space [163]. In probabilistic relational PCA, it is instead assumed that the data are not independent and identically and distributed (*i.i.d.*).

2.1.2 Independent Component Analysis ICA [45], attempts to find linear representations of non-Gaussian data in an unsupervised manner, is considered to be another powerful technique for source separation and data exploration. In contrast to PCA, in which Gaussianity of the data is assumed and global projection is performed, ICA considers non-Gaussianity and local data projection. It also operates on signals resulting from the superposition of different sources that must be separated. Intuitively, the problem can be explained using the cocktail party analogy, in which several people are speaking while several microphones record their speech, and the goal is to recover the separate speech signals from the recordings. Formally, the problem is to represent a multivariate signal $(x_i^1, x_i^2, ..., x_i^D, i - 1, ..., N)$ as a linear combination of independent components following a non-Gaussian distribution [94]. In the formalization defined in the supplementary materials, the features of x_i represent the recordings from the D sensors at the i^{th} instant and are a linear combination of the d sources described by the components z_i. ICA involves the construction of a generative model $X = ZA + R$, where A is called the mixing matrix (weights) and describes how each source contributes to each feature, and R is the residual. Alternatively, ICA can be formulated as the search for a transformation matrix U such that the columns of Z, where $Z = XU$, are independent [93].

As in PCA, without loss of generality, X can be assumed to be a zero-mean data $X = (X - \mu_X)$, implying that Z has the same property. Because ICA decomposition is possible up to a proper scaling of Z and A, a common hypothesis is that each z_i should have unit variance, which limits the ambiguity of the decomposition to the sign (i.e., a scaling factor of -1). The whitening of the data (i.e., the linear transformation of the data into a zero-mean, with columns that are uncorrelated and have unit variance) simplifies the ICA computation, and hence, it is a common preprocessing step for ICA (notably, PCA can be used for this purpose). Also, because it is not possible to rank the importance of the components and accordingly discard some components after ICA decomposition

has been performed, dimensionality reduction should instead be performed during preprocessing, if at all. However, an absence of correlation is a necessary (but not sufficient) for independence. Specifically, random variables are mutually independent if their joint probability can be expressed as the product of their marginal probabilities, $p_{z_1,\ldots,z_d}(z_1, \ldots, z_d) = \prod_i p_{z_i}(z_i)$, i.e., if the values of any subset of the random variables do not provide any information regarding the values of the other variables. Thus, a correlation merely indicates a linear trend of the variables (as in PCA), and hence, a lack of correlation does not imply independence (unless the variables are Gaussian).

The main tool for computing ICA components is the non-Gaussian property: essentially, it can be shown that choosing the z_i that maximize the non-Gaussianity ensures their independence. The kurtosis is a classical statistical moment used to measure non-Gaussianity and can thus be exploited to estimate the components using several less computationally demanding algorithms: Z can be estimated by minimizing the kurtosis of the components. However, the kurtosis is quite sensitive to outliers and noise [93]; as an alternative, a more robust measure of non-Gaussianity defined in the field of information theory is the negentropy (J) [108]. Among all distributions with the same μ and σ, the Gaussian distribution has the largest entropy $H(X) = -\int p_X(x) \log p_X(x)\, \mathrm{d}x$. The negentropy is defined as the difference between the entropies of the Gaussian distribution and Z, $J(Z) = H(Z_{\text{Gauss}}) - H(Z)$, and can be used as a measure of the non-Gaussianity of Z. However, because the definition of the negentropy involves probability distributions that are usually unknown, estimation of these distributions is required, which makes the algorithms more computationally expensive. Another tool that can be used to measure the independence of the components is the mutual information: $\mathrm{I}(z_1, \ldots, z_d) = \sum_{i=1}^{d} H(z_i) - H(Z)$. When the components are independent, the entropy of their joint distribution is equal to the sum of the entropies of the individual components; when the components are not independent, the entropy of their joint distribution is smaller. Equivalently, independence can be tested using the KL divergence between the joint distribution, $P(z_1, z_2, ..., z_i)$, and the product of the marginal distributions, $P(z_1)P(z_2), ..., P(z_i)$, of the components (see the supplementary materials). Finally, the original data can be generated from the separated components, i.e., $Z = XE$ and $E = A^{-1}$, by mixing the components as $\tilde{X} = g(Z) = ZE^T$ such that $BB^T = I$.

Among the prevalent ICA-based models, low-complexity coding and decoding (lococode) [87] is a nonlinear ICA that is built based on the sparse regularization of an autoencoder (see Section 4.2.1). Because of space limitations, we refer the reader to [93] for more details and other ICA extensions.

2.2 Structure-Based Models

Structure-based models are considered the pillars of UGL and dataset decomposition. The intuition behind such methods is that instead of directly exploring or learning from on the original data, it is desirable to simply decompose the dataset's structure into a set of subcomponents. Then, learning based on the representative factors can be implemented in a flexible manner.

A given dataset $X \subset \mathbb{R}^D$ can be factored into a product of two matrices, $X \approx UV = u_{i1}v_{1j} + u_{i2}v_{2j} + ... + u_{im}v_{nj}$, for $i = 1, ..., m$ and $j = 1, ..., n$, where m and n are the numbers of rows and columns, respectively. Moreover, this expression is valid in several different techniques when diagonal matrices are introduced for various purposes, such as the singular value matrix Σ in SVD, i.e., $\Sigma = \delta_1, \delta_2, ..., \delta_n$; accordingly, the generalized form for the decomposition of a given dataset X:

$$X \approx U_1\delta_1 V_1^{\mathsf{T}} + U_2\delta_2 V_2^{\mathsf{T}} + ... + U_n\delta_n V_n^{\mathsf{T}} \tag{1}$$

where U and V are called the left and right decomposition matrices, respectively, and the δ_n are diagonal scalars. In SVD (Section 2.2.1), these scalars represent the singular values that identify the data rank r (r is the number of feature vectors or columns that are linearly separable); however, $\delta = 1$ in other methods, such as nonnegative matrix factorization (NMF) (Section 2.2.2). In the

following, we provide a brief overview of the major structure-based methods, namely, SVD [101], NMF [141], and tensor decomposition (TD) [104], all of which share similar mathematical concepts.

2.2.1 Singular Value Decomposition SVD is an algebra-driven tool for decomposing a data matrix, X, to extract its singular values Σ. The singular values, Σ, and eigenvalues, Λ, play a substantial role in model reduction, where the reduced version of the data is the result of transforming the original data from one vector space to another, i.e., $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ [48]. The singular values measure the spectrum of the eigenvalues and are used to find the data rank, r, which indicates how many feature vectors are linearly independent (the number of columns) [101]. For a given data matrix $X \in \mathbb{R}^{N \times D}$, SVD generates three matrices ($U \in \mathbb{R}^{N \times N}$, $\Sigma \in \mathbb{R}^{N \times D}$, and $V \in \mathbb{R}^{D \times D}$) such that:

$$X = U \Sigma V^{\intercal} \tag{2}$$

where U and V are orthonormal matrices, called the left and right singular vectors, respectively, that represent the eigenvectors of $XX^{\intercal}$ and $X^{\intercal}X$, respectively. Additionally, Σ is a diagonal matrix that contains the singular values $\{\delta_i \mid i = 1 \ldots, D\}$ of the matrix X and are sorted in decreasing order. If the rank of X is r, the singular values after the first r values are equal to zero (or neglectable). The exact decomposition in Eqn. (2) can be approximated when the nonzero singular values are neglected based on a threshold of small value (truncated SVD). This practice is useful in addressing numerical instability and noise and in reducing the dimensionality of the data. However, dimensionality reduction to a d-dimensional space can be realized by defining U_d as the matrix with the first d columns of U, Σ_d as the $d \times d$ diagonal matrix containing the first d singular values, and V_d as the first d rows of V. Where, U_d represent the distribution of the data points around the directions V_d when they multiplied as: $U_d \Sigma_d$. Moreover, the original data X can be generated (or reconstructed) after the decomposition as $\tilde{X} = U_d \Sigma_d V_d^T$, considering an optimization goal $min \|X - \tilde{X}\|_F$.

Various extensions have been proposed in the literature to generalize SVD, e.g., generalized SVD (GSVD) [142], in which the joint decomposition of two different matrices is performed: $X1, X2 = U \Sigma V^{\intercal}$. For certain values of the second matrix, GSVD reduces to SVD; however, it can be used to formalize a prior knowledge. This approach is extended in restricted SVD (RSVD) [49], which decomposes a given matrix with special constraints encoded in two other matrices.

2.2.2 Nonnegative Matrix Factorization The analysis of the environmental data originating from chemical or physical interactions requires the identification of quantitative scales to conduct reliable measurements. Such analysis also possesses variational errors due to natural interplays. PCA (see Section 2.1.1) is based on the covariance C_X decomposition and faces different challenges in disentangling the representative factors from big datasets, due to imposing the centering around the origin that leads to the loss of some important information among the original data and prevents the interpretability. NMF proposed in [141], addresses the limitations of PCA in determining the interpretable factors by forcing non-negativity onto the elements resulting from the data decomposition. In contrast to PCA and ICA, no assumption on the statistical dependency of the components is made. NMF provides a generative model for the data: for a given dataset $X \in \mathbb{R}^{N \times D}$, the algorithm decomposes the data matrix as $X = ZA + R$, where $Z \in \mathbb{R}^{N \times d}$ is the bases subspace, $A \in \mathbb{R}^{d \times D}$ is the reconstruction coefficients subspace, $R \in \mathbb{R}^{N \times D}$ is the factorization residual, with the constraints of that Z, $A \geq 0$, $d = r$ is the rank, and minimum residual R. The model provided by NMF tends to be sparser (fewer elements in the Z rows are nonzero) and more part-based (few elements in the A rows are nonzero) than PCA. Thus, because all the components are nonnegative and the combination is additive, thus NMF offers what is called a part-based representations, where the data reconstruction ($g : \mathbb{R}^d \rightarrow \mathbb{R}^D$) is realized by assembling different parts, i.e., $\tilde{X} = Z^{N \times d} A^{d \times D}$.

NMF can be optimized to minimize the residual R using various approaches. For example, in [141], a weighted Frobenius norm of R was introduced, in which the inverse of the variance of each element

of X is used to weight the corresponding element of R. In [113], the objective function was related to the likelihood of reconstructing X by adding Poisson noise to ZA: $R = \sum_{i,j} \|X_{(i,j)}\ log(ZA)_{(i,j)} - (ZA)_{(i,j)}\|$. Another way to minimize R was proposed in [114]; in this case, the modification lay in introducing a composite objective function to update both the basis Z and the coefficients A. The first term of this composite function is the square of the Frobenius norm, $R_{Euc} = \|X - ZA\|_F^2$, and the second is the KL divergence, given by $KL(X||ZA) = \sum_{(i,j)} X_{(i,j)}\ log\frac{X_{(i,j)}}{ZA(i,j)} - X_{(i,j)} + ZA_{(i,j)}$. Finally, R is optimized by minimizing $R_{Euc} + KL(X||ZA)$ because both sides of this function vanish when $X = ZA$ and the lower bound is zero. Additionally, because the product ZA is invariant under a scaling factor, additional constraints can be added to make the optimization numerically stable or regularized. For instance, the rows of Z can be restricted to be of unit norm [113], or the orthogonality of the basis or an altered factorization (e.g., kernel mapping) can be addressed [189].

The main advantages of NMF are the nonnegativity in the factorized subspaces, i.e., Z and A, and the fact that it is applied in the data space, i.e., X. However, it cannot be used to address data in the kernel Hilbert space. Concept factorization (CF) is a variant of NMF that can be used in either the kernel space or the original data space due to its resilient formalization. Specifically, in CF, the data matrix X is decomposed into a product of three matrices, $X = XZA + R$, where, as in NMF, the residual R are minimized and Z, $A \geq 0$ (XZ approximates the basis and A corresponds to the new representations) [216]. Among the generations of CF, Robust Flexible Auto-weighted Local-coordinate Concept Factorization (RFA-LCF) was proposed in [215, 217] for high-dimensional data clustering. RFA-LCF is based on data cleaning and the learning of a local sparse projection of the data, followed by flexible CF in the projected subspace. The RFA-LCF is followed in the Deep Self-representative Concept Factorization Network (DSCF-Net) framework [211], in which the data are factorized in the subspace into which they are projected. DSCF-Net also improves the locality of representations using a matrix of coefficients as adaptive reconstruction weights.

2.2.3 Tensor Decomposition A tensor is a generalization of a vector or matrix, often represented as a d-way array, where d_w denotes the number of dimensions (or the order); for example, a 1-way tensor is a vector, while a 2-way tensor is a matrix. Accordingly, a tensor is of high order when $d \geq 3$, as in the case of RGB and hyperspectral images [104]. The aforementioned BSS methods, including both statistics- and structure-based techniques (Sections 2.1 and 2.2), are considered to be fundamental data factorization techniques; however, they are limited to 2-way representations because they are based on concepts of linear algebra. By contrast, TD methods and their variants are a set of techniques developed on the basis of multilinear algebra to decompose massive-scale datasets regardless of their dimensionality, for which matrix operations and optimization procedures are utilized. For three given vectors $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$, and $w \in \mathbb{R}^p$, a 3-way tensor $\mathcal{X}$ can be formed by the outer product of these vectors, $\mathcal{X} = u \otimes v \otimes w$, where $\mathcal{X} \in \mathbb{R}^{n \times m \times p}$ and $\mathcal{X}_{i,j,k} = u_i v_j w_k$. This principle can be extended to d-way tensors when $d > 3$; here, however, we limit the description to $d = 3$ to keep the notation simple. The decomposition process generates low-order components to discover the hidden structure of the data. For instance, the SVD procedure described in Section 2.2.1 can be described by reframing Eqn. (2) in the tensor context, where X is a 2^{nd}-order tensor, $\tilde{X} = \sum_{i=1}^{r} \sigma_i U_i \otimes V_i$, with U_i and V_i being the i^{th} columns of the matrices U and V, respectively; accordingly, 2^{nd}-order tensors can be obtained.

Canonical polyadic decomposition (CPD) [74] and Tucker decomposition [179] are commonly used tensor factorization methods and serve as the foundation for much related work. CPD is often used for latent variable estimation, while Tucker decomposition can be used for dimensionality reduction and compression. In CPD, a d-way tensor is expressed as a sum of r tensor products of d 1-way tensors, and under certain mild conditions, this decomposition is unique. For a 3-way dataset $\mathcal{X} \in \mathbb{R}^{n \times m \times p}$, CPD generates three matrices, $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{m \times r}$, and $W \in \mathbb{R}^{p \times r}$, such that

$$\mathcal{X} = \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i = [\![U, V, W]\!] \tag{3}$$

where U_i, V_i, and W_i are the columns of the corresponding factor matrices. When U_i, V_i, and W_i are normalized, their scaling factors can be factorized as a vector $\lambda \in \mathbb{R}^r$, and the original tensor is generated from the decomposed matrices considering $[\![\lambda; U, V, W]\!]$ as $\tilde{\mathcal{X}} = \sum_{i=1}^{r} \lambda_r U_i \otimes V_i \otimes W_i$.

Tucker decomposition can be regarded as multiway PCA when applied to different views (or sides) of the data, and it can also be generalized to higher-order SVD to capture the best rank for a given set of data. Tucker decomposition involves factorizing d-way tensor data $\mathcal{X}$ into d factor matrices (which represent the orthogonal basis of the space of each dimension of $\mathcal{X}$, i.e., the "principal components") and a d-way core tensor, $\mathcal{Z}$, consisting of the projections of the data onto those bases. For a 3-way dataset $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$, given the factor matrices $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times s}$, and $W \in \mathbb{R}^{p \times t}$ and the 3-way core tensor $\mathcal{Z} \in \mathbb{R}^{r \times s \times t}$, the Tucker decomposition generates approximation such that:

$$\tilde{\mathcal{X}} = \sum_{i=1}^{r} \sum_{j=1}^{s} \sum_{k=1}^{t} \mathcal{Z}_{i,j,k} U_i \otimes V_j \otimes W_k = [\![\mathcal{Z}; U, V, W]\!] \tag{4}$$

where U_i, V_j, and W_k are the columns of the factor matrices. Similar to PCA, because r, s, and t are usually smaller than m, n, and p, respectively, $\mathcal{Z}$ is an encoding of the data $\mathcal{X}$.

2.3 Probabilistic-Based Models

Probabilistic-based models (PBMs) are a class of UL models that utilize the advantages of probabilistic estimation to learn ML models. Many recent shallow and deep learning models have been built based on probabilistic modeling, where the goal is to model the generation of data $P_{model}(X)$ [105]. The following methods share a similar concept in using the inference $P(X|\theta_Z)$ to predict a variable (or set of variables) given any other variables, i.e, given a set of latent parameters θ_Z (or H_i in the context of hidden variables in NNs), the data X can be predicted in an unsupervised fashion [97]. Moreover, they are able to model data generation and prediction by coupling a set of input data $X \subset \mathbb{R}^D$, $X = \{x_i \,|\, x_i \in \mathbb{R}^D,\ i = 1, \ldots, N\}$, with a set of latent variables $Z \subset \mathbb{R}^d$, $Z = \{z_i \,|\, z_i \in \mathbb{R}^d,\ i = 1, \ldots, M\}$ through a set of parameters θ_Z to form the structure of the PBM:

$$P_{model} = \mathbb{E}_z\, P(X|\theta_Z) P(\theta_Z) \tag{5}$$

where $P(\theta_Z)$ is the prior probability, and $\mathbb{E}_z$ is the expectation (discrete averaging) among all possible configurations (update) of the latent parameters θ_Z. In the following, we recapitulate two categories of the PBMs: those are derived to latent variables (LVs) modeling, and those related to mixture modeling; however, some models comprise both latent and mixture views.

2.3.1 Latent Variable Models LVs are unobservable random variables, which introduce a latent generation process among observed data [135]. Latent variables models (LVMs) quantify different characteristics or phenomenons (such as intelligence, happiness, and document understanding), which are unable to be measured directly by a quantitative scale [134]. Moreover, the standard ML methods such as regression and discriminative NNs are not useful in this regard; because of missing data and handling uncertainty. Accordingly, the LVs are introduced to build links between all data points (features' components), thus constructing a final parameterized model. Such models affect Eqn. (5) to be more tractable, i.e., they factorize the joint probability into small probabilities:

$$P_{model} = \mathbb{E}_z\, P(x_1|\theta_Z) P(x_2|\theta_Z) \ldots P(x_N|\theta_Z) P(\theta_Z) \tag{6}$$

Consequently, the LVMs reduce the model complexity by shrinking the total number of parameters and edges, i.e., the connections between data points and latent variables without losing the model

flexibility. In the following, we summarize common LVMs, including Latent Dirichlet Allocation [29], hierarchical Latent Dirichlet Allocation [72] as well as their relevant extensions, is given.

2.3.1.1 Latent Dirichlet Allocation (LDA) is a typical UL method that is capable of detecting words and phrases in a corpus, scanning and clustering documents, and analyzing textual information. Recently, it has been received a high popularity in topic modeling, i.e., finding the distribution over words [29]. LDA builds a probabilistic model considering a multinomial distribution $P(X|\theta)$ of each document over latent/hidden topics, and each latent topic is represented as a multinomial distribution of words. Moreover, the LDA is named from Dirichlet distribution Dir: a continuous multivariate distribution that is parameterized by a non-negative and summed to 1 vector α. The Dirichlet prior $Dir(\theta)$ is considered a conjugate to the multinomial likelihood, or distribution over count, $P(X|\theta) = \frac{n!}{x_1,..,x_k}\theta_1^{x_1},..,\theta_K^K$ when the posterior $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$ belongs to the a similar distribution of the prior. Initially, the corpus is represented as a word-document matrix (as a bag of words), where each matrix element reflects the number of documents D in the corpus, vocabulary $V = \{1, 2, .., N\}$ of N words, and the number of topics Z of the documents. The LDA assumes that each topic distribution through all documents participate in a common Dirichlet prior $P(\theta_d) = Dir(\theta_d|\alpha) = \frac{1}{B(\alpha)}\Pi_{d=1}^{D}\theta_d^{\alpha_d-1}$. Similarly, the distribution of the words in topics Z share a joint prior $P(\theta_z) = Dir(\theta_z|\eta) = \frac{1}{B(\eta)}\Pi_{z=1}^{Z}\theta_z^{\eta_z-1}$, thus given both α and η the Dirichlet distribution for document d and Z topics is drawn as $Dir(\theta_d|\alpha)$, and for a topic z over N words as $Dir(\theta_z|\eta)$.

The LDA is attributed to the UGl due to its ability to describe a document in a corpus D, by repeating the process D times. Also, it is able to generate a posterior distribution $P(\theta|X)$ for a model with unknown latent variables and parameters, i.e., given the corpus it estimates the model posterior. Subsequently, it utilizes model inference for parameters optimization and model fittings, by using variational inference, Monte Carlo simulation, or Gipps sampling [59]. Finally, the model output comprises two sub-matrices; the first one of size $D \times Z$ and shows the probability of the topic over documents, where the other of size $Z \times N$ and reflects the probability of words over topics. The LDA-dual was proposed in [167] is considered an extension of the LDA for multi-topic modeling in the same document, and to solve the name-sharing problem in bibliography websites.

2.3.1.2 hierarchical Latent Dirichlet Allocation Mining modern multi-dimensional datasets (as in social media data) that are composed of hierarchical relations among data entries, becomes more challenging nowadays [123]. Such datasets come in unformalized or unstructured forms, and can be explored by probabilistic generative models. The LDA [29] captures the correlation between words in a document, where it neglects the correlation among topics because of the single distribution of the topic in each document. Accordingly, hierarchical Latent Dirichlet Allocation (hLDA) has been proposed in [72] to model the hierarchical correlation among topics of documents in a corpus D, by using a tree structure to break the independency among all topics.

Th hLDA is built based on the nested Chinese restaurant process (nCRP), which offers a distribution of an infinite number of customers m to sit in an infinite number of tables i in a restaurant according to the distribution $P(table_i) = \frac{mi}{\gamma m-1}$, γ is a parameter [27]. The nCRP constitutes topics hierarchy equivalently to assigning a probability for customers to visit L restaurants for L days, thus the L restaurants construct a path from the root to the L level from an infinite tree. For documents in a corpus D, vocabulary $V = \{1, 2, .., N\}$ of N words, and Z topics; the hLDA selects a root topic z_1 sits in the top-level l_1, then it passes to the next levels $l \in \{2, ..., L\}$ utilizing the distribution drawn from $P(table_i)$, and it chooses the topic z_1 from z_{l-1}. Moreover, it draws the L-dimensional topic proportions to a document d from $Dir(\theta_d|\alpha)$, and for N words it draws a topic z as $Dir(\theta_z|\eta)$.

The hLDA is attributed to UGl because it is able to generate documents using L-level path from the top root to the bottom leaves; however, it fixes the path for each document when sampling

topics (each node represents a single topic). Accordingly, a supervised hierarchical latent Dirichlet allocation (SHLDA) was proposed in [140] to allow a document to obtain a multi-path through tree structure, and a semi-supervised hierarchical topic model (SSHLDA) was proposed in [146] based on a supervised path labeling method to search a topic automatically in the data space.

Another kin model to the hLDA is termed as Pachinko Allocation Model (PAM) and was proposed in [117], where it differs from the hLDA in that the correlation among the topics in the PAM is captured by a Directed Cyclic Graph (DAG) instead of using a tree as in the hLDA. Moreover, in the PAM the topic is modeled considering the distribution through all topics, not only over words. Two steps are necessary in the PAM to generate a document d from a topic z_i: *(i)* the first step lies in sampling the parameters $\theta^d_{z_i}$ from $Dir(\alpha_i)$, *(ii)* the topic path is sampled, for each word v_i, in the second step utilizing $\theta^{(d)}_{z_{n(i-1)}}$; thus the word v_i is generated from $\theta^{(d)}_{z_{n(L_n)}}$. Finally, the model can be inferred using sampling methods such as Gibbs sampling, or variational inference [28, 123].

2.3.2 Mixture-Based Models MBMs are a set of UL models that defines a suitable probability density function for data in the input or latent spaces, and they explain data clusters in terms of a sequence of density distributions, e.g., a sequence of Gaussian or Student-t distributions [145]. They also share a similar concept in modeling data distributions based on centroids or prototypes and Euclidian distance. The MBMs restrict the mixture components to follow a certain shape for model optimization, e.g., Gaussian-spherical or Gaussian-full, etc, and they are optimized using the expectation-maximization (EM) to maximize the model likelihood [125]. For a given dataset $X \subset \mathbb{R}^D, X = \{x_i \mid x_i \in \mathbb{R}^D,\ i = 1, \ldots, N\}$, the mixture expresses data utilizing the joint distribution between data and latent variables in the latent space, $Z \subset \mathbb{R}^d, Z = \{z_j \mid z_j \in \mathbb{R}^d,\ j = 1, \ldots, M\}$:

$$P(X, Z) = \prod_{i=1}^{N} \prod_{j=1}^{M} [P_j(x_i)^{\pi_j z_{ij}}] \tag{7}$$

Eqn. (7) is a mixture model with M components, and π_j is the component weight. In the following, we highlight typical MBMs including Gaussian mixture models (GMMs) [155], probablisic self-organizing map (PSOM) [38], generative topographic mapping (GTM) [25], and their extensions.

2.3.2.1 Gaussian Mixture Models Many real-life datasets comprise different clusters, i.e. more than one subgroup of data samples, thus fitting a probabilistic model for such data can not be evaluated by one distribution like a Gaussian. Intuitively, one Gaussian distribution fits all data points to a shape, e.g., a circle or ellipsoid, and it assigns a higher probability to the center of the shape which is not applicable for multi-cluster datasets with multi-centers [155]. Accordingly, multi-class datasets can be modeled by multi-Gaussian (a sequence or mixture) to meet the number of clusters; by using GMM that is able to model the densities among data points according to a predefined weighted sum form the mixture densities (components).

The GMM is a probabilistic model that implies that all data points (or samples) are generated in a form of a limited number of Gaussian distributions (a linear combination of distributions), which are paramertrized by θ_j, hidden or latent variables $Z = z_i \in \mathbb{R}^M$, $z_{ij} \in \{0, 1\}$, and $\Sigma_j z_{ij} = 1$ [66]. It also estimates the mixture components M in Eqn. (7) in terms of multivariate Gaussian with components centroids μ_M (which reflects the locations of the Gaussians), and a positive-definite (all eigenvalues Λ are positive) covariance matrix C_M (which reflects the shapes of the Gaussians). The number of components M is a free parameter, and $P_j(x_i)$ indicates the probability of the data sample x_i if it is to be sampled from a certain mixture component j from M, i.e., $P(x_i) = P(\theta_j)P(x_i|\theta_j)$; where $P(x_i|\theta_j) = \frac{1}{(2\pi_j\sigma_j^2)} \exp\{-\frac{||xi-\mu_j}{2\sigma_j^2}\}$, and the model posterior is generated using Bayes' rule as

$P(\theta_j|x_i) = \frac{P(x_i|\theta_j)P(\theta_j)}{P(x_i)}$. Assuming that all samples are independent given θ_j, thus the likelihood is factorized and the latent variables are marginalized by a discrete sum among all values of Z:

$$P(X) = \prod_i \Sigma_{z_j \in Z} \prod_j [P_j(x_i)^{\pi_j z_{ij}}] \tag{8}$$

The GMM is optimized to find an optimal set of parameters which expresses the density of each set of data points, i.e, $\theta_j = \{\pi_j, \mu_j, \sigma_j\}$, by adapting the parameters values to maximize the likelihood $P(x_i|\theta_j)$ [150]. EM is an optimization algorithm that is utilized to optimize GMM parameters, by assigning the data points to clusters c given a certain probability [155]. Two associated steps in the EM algorithm proceed iteratively, *(i)* Expectation or E-step which starts by allocating different clusters with different parameters θ_j, then assigning a responsibility probability r_{cj} that recognizes if the data point i belongs to cluster c_j (or z_j); *(ii)* Maximization or M-step in which r_{cj} is fixed and the mixture components parameters θ_j are updated. The model loss is minimized considering the logarithm which is normalized by the number of samples N as $\mathcal{L} = \frac{1}{N}\Sigma_{n=1}^{N} log(\Sigma_j \pi_j P_j(x_n))$. More details and model extensions about the GMM and EM algorithm are introduced in [136, 155].

2.3.2.2 Probabilistic Self-Organizing Map Self-Organizing Map (SOM) is a ML model that is trained in an unsupervised fashion to cluster and visualize data. Moreover, it constitutes feature detectors that are spatially formed and represented in a gradually increasing array [102]. A SOM contains a layer (like a layer of neurons) that is parameterized by a set of weights, which are trained using an online algorithm to fit a set of prototypes c (similar to centroids c of clusters) to the data X in the latent space Z, by assuming that the prototypes in both latent and data spaces are nearby. For a given dataset $X \in \mathbb{R}^{N \times D}$, the SOM selects a random sample x_i a time, and the distances between the sample and all prototypes are computed $||x_i - c_r|| = min||x_i - c_j||$, where c_r is the location of the center neuron that is maximized to be indicated as a winner neuron. Subsequently, the weights are updated as $w_i^{new} = (1 - \alpha.n_b)w_i^{old} + \alpha.n_b.x_i$, where α is the learning rate. The neighbors for the i^{th} neuron are identified such that $nb = \exp(\frac{||c_i - c_r||^2}{\omega^2})$, ω is the variance of n_b.

The stranded SOM neglects the loss function form the model optimization [103]; conversely, it is considered in the probabilistic self-organizing map (PbSOM) that has been introduced in view of Gaussian-full mixture components M [38]. The PbSOM implies a similar prior probability $P(\theta_j)$ is shared among all mixture components, and it partitions the prototypes $P = \{c_1, .., c_M\}$ as a function of mixture M; utilizing the objective function $\mathbf{O}(P, \theta) = \Sigma_{i=1}^{N}\Sigma_{x_i \in c_j}\Sigma_{j=1}^{M}(n_b \log P(x_i|\theta_j))$, which seeks to maximize the likelihood considering each partition P. The EM is used to optimize and update θ_j, by introducing a new step "C-step" to assigns a sample x_i to each mixture component j, which obtains the highest responsibility r_{c_j}, i.e., $c_j = \{argmax\ r_{c_j} = P(x_i|\theta_j)\}$. Lastly, the M-step re-estimates the mean μ_j and variance σ_j for each mixture component j, thus its θ_j is optimized to maximize the likelihood. For the sake of space, we refer to [103] for more details and extensions.

2.3.2.3 Generative Topographic Mapping GTM was proposed in [25] as a probabilistic nonlinear latent space discovery method, which constructs relations between data points X and a mapping space Z. The GTM is built based on a normal distribution among samples and has been introduced to be an alternative of the SOM, that implies a uniform distribution in the mapped space (output mapping array). For a given dataset $X \in \mathbb{R}^{N \times D}$, the GTM generates a parametric mapping from data space to a latent space $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$; constraining the distribution to follow a Gaussian centered among the mixture components which are drawn as $P(x|z, W, \beta) = (\frac{\beta}{2\pi})^{-D/2}\exp\{-\frac{\beta}{2}||x - y(z, W)||^2\}$, where W are the weights and β^{-1} is the variance. The prior of Z is approximated using the summation of M equally-weighted delta function on agrid as $P(z) = \frac{1}{M}\Sigma_{j=1}^{M}\delta(z - z_j)$, which is employed to measure the likelihood after fitting the model $P(x|W, \beta) = \frac{1}{M}\Sigma_{j=1}^{M}P(x|z_j, W, \beta)$.

The EM maximizes the likelihood considering the same variance β^{-1} among the components M; however, it could be learned among the mixture components as $\beta = \frac{1}{\sigma^2}$ [26]. In the E-step, the responsibility or posterior is estimated $r_{cj} = P(\theta_j|x_i, W, B)$, and each component j is linked with a point in the latent space z_j. Subsequently, the M-step maximizes the complete likelihood for the data as $\mathcal{L} = \Sigma_{i=1}^{N}\Sigma_{j=1}^{M} r_{ci}(\log P(x_i|z_j, W, \beta))$ concerning β and W. The GTM uses a batch learning approach to update the model parameters θ_j, thus visualizing big datasets by standard GTM faces computational burdens: the complexity is increased because the M-step is performed among all data points. Accordingly, the incremental learning procedure has been proposed in [26] to overcome the computational complexity issues, by using small batches and update the model gradually. We refer to [6] for more details and extensions of the GTM models.

Finally, because of space limitations, we refer the reader to the supplementary materials that accompany this survey, which offer remarks, open research issues, and it gives further discussions.

3 Manifold Learning Models

High-order tensors and multidimensional datasets consist of a large number of observations and data samples, and the direct exploitation of the raw data for feature extraction or visualization is usually difficult for ML models. Interesting information that can be extracted is structured in low-dimensional manifolds, with data in the same submanifold sharing properties that can be formalized as a smooth conditional distribution $P(Y|X_i)$, where the X_i are the data points and Y is any prior condition [181]. The concept of a manifold is widely used in geometry; it represents a low-dimensional topological space that locally has the topological properties of a low-dimensional Euclidean space immersed in a higher-dimensional space, e.g., a folded sheet in 3D space [31].

MfL models are a set of methods that aim to discover (learn) from a dataset the embedded low-dimensional features over which the data are supposed to lie. Accordingly, these models can be utilized for dimensionality reduction and data visualization tasks. The MfL can be described as a search for an optimal mapping function f between the original dataset X of points in $\mathbb{R}^D$ and the corresponding feature points Z in the d-dimensional parametric space of a manifold ($d << D$).

The general framework of the methods described in this section consists of stating a similarity or dissimilarity relation over the data in both domains (X and Z) and selecting a mapping function f that preserves the abovementioned relation while optimizing the given constraints; for instance, a common dissimilarity relationship is a distance (Euclidean or other). In this section, MfL methods for dimensionality reduction and visualization are described. Such methods are commonly divided into two categories, namely, global and local methods, depending on the domain over which distance matching is performed (i.e., whether f is estimated on the whole dataset or on local subsets).

3.1 Global Models

Global models are a set of MfL methods known as full spectral methods. These methods can capture the global structure of a manifold, discover the noise in the manifold's general structure, and adapt to cases with multiple manifolds corresponding to different forms of data [183].

In the following, a brief review of several common and well-known global MfL methods, namely, multidimensional scaling (MDS) [107], Isomap [177], and nonlocal manifold tangent learning (NLMTL) [21] as well as their relevant extensions, is given.

3.1.1 Multidimensional Scaling MDS is a baseline method of linear MfL that generates a map that preserves the distances between pairs of data points in a dataset, in contrast to PCA (see Section 2.1.1), which attempts to preserve the correlations or variance of nearby data samples [107]. MDS calculates the distance between every pair of data points. Several functions for representing the distance between data points, such as the Euclidean-, the mean of the absolute log-fold differences-,

the Hamming-, and the Manhattan-distances, can be used for this purpose. When the Euclidean distance is adopted, the results are similar to those of PCA [195], because maximizing the linear correlation is identical to minimizing the linear distance. The distance that is computed between each pair of points x_i and x_j reflects the relation between the data samples, and these distances form the dissimilarity matrix $D_X(i,\ j) = \|x_i - x_j\|^2$. The mapping to Z is chosen so as to minimize the error $\sum_{i,j}(D_X(i,\ j) - D_Z(i,\ j))^2$, where D_Z is defined similarly to D_X but in terms of the corresponding elements of the feature domain. The generative process can be expressed as $Z = U\Lambda^{1/2}$, where U is the $N \times d$ matrix formed by the largest d eigenvectors of $X^\intercal X$ (orthogonal matrix) and the diagonal matrix (Λ) contains the eigenvalues corresponding to the eigenvectors. Alternatively, mapping can be generated using $-\frac{1}{2}HD_XH$, where $H = I - \frac{1}{N}\mathbf{1}^\intercal\mathbf{1}$, with H being the centering matrix, and $\mathbf{1}$ is an $N \times 1$ column vector in which each entry is one. Moreover, the original data can be generated from Z space considering a kernel $A = D_Z^{-1}X$, and $\tilde{X} = D_Z^T A$.

The MDS approach can be extended by adding a weight to each distance to modulate the contribution of each pair of points to the error function when generating a map, i.e., $\sum_{i,j} w_{i,j}(D_X(i,\ j) - D_Z(i,\ j))^2$, where the nonnegative weight $w_{i,j}$ refers to the distance between the i^{th} and j^{th} points. MDS can also be expressed in a nonmetric form, in which the relationships between the data samples and the dissimilarity matrix are defined by exploiting the isotonic regression function [46].

The MDS and its extensions concern linear data representations, i.e., cases in which the manifold is linear and can be obtained as a scaled version of the data, a condition that often does not hold in many real-world problems. In the following, the principal methods devised to address nonlinear manifolds are described. In such methods, a sparse graph is built over the data, and the graph information is exploited to approximate the geometrical and topological properties of the manifold.

3.1.2 Isomap Isometric feature mapping (Isomap) is a global MfL method that combines the capabilities of PCA and MDS to discover linear manifolds; however, it is also able to capture the global relations among the data points when learning nonlinear manifolds [177]. The primary characteristic of Isomap is its use of geodesic distances, which mirror the geometry of the lower-dimensional nonlinear manifold, while PCA and classical MDS employ Euclidean straight-line distances [54]. Hence, instead of D_Z being matched with the Euclidean distance D_X, it is matched with an estimate of the geodesic distance, D_G, which is built as follows. Given a certain neighborhood relation (common choices are the k nearest points or those within a suitable radius ϵ), a weighted graph $\mathcal{G}$ is generated, where the vertices are the points and the edges connect neighboring points, with the corresponding distances as weights. The D_G can then be defined as the shortest path in $\mathcal{G}$ that passes through all k neighbors. The final step recalls MDS over the D_G matrix to generate a mapping Z (either by decomposing the matrix $X^\intercal X$ or by using the centering matrix in the form $-\frac{1}{2}HD_XH$; see Section 3.1.1), and an optimization procedure is applied to optimize the distances in the Euclidean space. Moreover, the original data X reconstruction from space Z can be done similarly to the MDS, where inverting the k-neighbors distances and a kernel matrix A is required.

There multiple variants of Isomap learning. As a prevalent method, C-Isomap is a technique that follows the same procedure as standard Isomap in generating map Z, except that the weighted distances are obtained in the form $D_X(i,j) = \|x_i - x_j\|/\sqrt{m_i \times m_j}$, where m_i is the average distance between x_i and its k neighbors [168]. Marginal Isomap (M-Isomap) [212] uses pairwise constraints to identify and weigh the neighborhood relations among the data points (specifically,there are a Cannot-Link constraint for interclass clusters and Must-Link constraint for intraclass clusters). Consequently, the M-Isomap tends to preserve the margins between interclass and intraclass clusters to unfold a multimodal discriminative (supervised) nonlinear manifold. Moreover, Isomap has been improved to take an incremental form rather than considering fixed patches of data to the incremental form in [112], while it has been stacked in a hybrid manner to extract the

intrinsic structures of hyperspectral images in [8]. Also, node-weighted multidimensional scaling (NWMS) is an extension of MDS and Isomap that was proposed in [172]; NWMS partitions the data in the geodesic space by means of weight matrix that is fitted to the data using EM algorithm [125]. Recently, semi-supervised multimanifold Isomap SSMM-Isomap was proposed in [210] for semi-supervised feature extraction, where it utilizes explicit projection for the new data to be embedded and requires the k-neighbors of each sample within the same class to be fixed.

3.1.3 Nonlocal Manifold Tangent Learning NLMTL [21] was proposed to overcome the intrinsic weakness of methods that estimate the manifold geometry as a linear combination of points within a close neighborhood; namely, these estimates can easily be affected by noise in the data, especially when the data are scarce or the manifold has a high curvature. NLMTL better addresses such cases because the manifold model is more general and because data from a wider neighborhood are considered during the learning process. The core idea of the NLMTL technique is to estimate the tangent plane as a matrix function (mapping) $F : \mathbb{R}^D \rightarrow \mathbb{R}^{d \times D}$ such that $F(x)$, modeled as a standard single-hidden-layer neural network, outputs a basis for the tangent plane at the point x. To learn F, for each point x_i, a suitable neighborhood $\mathcal{N}_i$ is selected (e.g., the k nearest neighbors), and the vectors $\{x_i - x_j \,|\, x_j \in \mathcal{N}_i\}$ are considered targets for $F(x_i)$ (i.e., noisy estimates of the elements in the tangent plane). A point x_j on the tangent plane at x_i can then be expressed as $x_i + w_{i,j}^{\intercal} F(x_i)$, where $w_{i,j} \in \mathbb{R}^d$ is the local coordinate of x_j in the basis provided by $F(x_i)$. The learning procedure is based on a dual optimization problem that alternates between the learning of F and the updating of the optimal value of $w_{i,j}$ to minimize a loss function called the relative projection error: $R(F,\, w) = \sum_i \sum_{j \in \mathcal{N}_i} \frac{\| w_{i,j}^{\intercal} F(x_i) - (x_i - x_j) \|^2}{\| x_i - x_j \|^2}$. The contribution of each example is weighted by the distance from the reference point to decrease the importance of more distant neighbors.

The NLMTL approach for learning the global manifold function is used in the nonlocal manifold Parzen windows (NLMPW) technique [20], in which the manifold distribution is described in terms of a global covariance matrix C_x. NLMTL is also inspired by neighborhood component analysis [68]; by contrast, N-LMPW is generalized and its learning inspired by the manifold Parzen window [185], in which a Gaussian distribution is considered to model local behavior of data.

Geometric structures have been considered to capture the intrinsic embeddings of data in various ways, such as charting a manifold [33], in which, similar to NLMPW, a mixture of Gaussian densities (chosen by minimizing the information losses in the context of posterior and prior inference) is used to model the manifold, but a closed-form mapping can be computed. Furthermore, in local coordinate coding (LCC) [205], the local point geometry is extracted based on anchors representing the low-dimensional coordinates. In the LCC method developed in [204], the number of anchors is optimized by considering a local tangent plane, as in NLMTL.

3.2 Local Models

Local models are a class of MfL, known as sparse spectral methods, and are superior in exploring the local relationships among nearby data samples. In contrast to global methods, local models use local optimization to determine the manifold description. Moreover, they able to capture the fine structure (local) of the manifold as the main objective, and preserve the global properties [183].

In the following, we discuss local MfL methods, including locally linear embedding (LLE) [156], Laplacian eigenmaps (LEs) [15], maximum variance unfolding (MVU) [192], t-distributed stochastic neighbor embedding (t-SNE) [181], and their relevant extensions.

3.2.1 Locally Linear Embedding The intuition behind the LLE technique is based on the idea that the local geometries of a nonlinear manifold can be captured by locally linear functions in small patches of the data points, thus allowing the geometries over all local patches to be considered and mapped to

a lower-dimensional space (global coordinates). In LLE, the low-dimensional mapping Z is generated and optimized by computing the local distances through the k nearest data points rather than by estimating the geodesic or Euclidean distances between the widely separated patches; accordingly, it preserves the locally linear structure of the manifold [156]. A local approximation for each data point x_i is identified as a linear combination of its k neighbors, and the mapping is generated by constraining the mapped values to the same linear combination of the mapped neighbors. Formally, the manifold is described by the weights $W_{i,j}$ obtained to minimize $\sum_i \|x_i - \sum_j W_{i,j}x_j\|^2$ under the constraints that the $W_{i,j}$ are zero for nonneighboring points and that they sum to one to ensure that all neighbors are in the same convex hull. The low-dimensional embedded coordinates are then generated by minimizing the cost function $\sum_i \|z_i - \sum_j W_{i,j}z_j\|^2$. In practice, the problem can be reframed as the computation of the $d+1$ bottom eigenvectors of an $N \times N$ matrix derived from the weights W. Then, the sparse structure of W can be exploited to speed up the computations, moreover, the original data is generated from the embedded space Z by inverting W as $\tilde{X} = W^{-1}Z$.

Various extensions of LLE have been proposed in the literature, e.g., the locally linear coordination (LLC) method proposed in [176], which is based on viewing a nonlinear manifold as a mixture (See section 2.3.2.1) of subspace models in order to generate the coordinates in the lower-dimensional space Z. The core idea of LLC is to construct the global coordinates based on the statistics of the mixture such that the mixture components can subsequently be satisfactorily aligned in the lower-dimensional coordinates and the number of optimization computations can be reduced. The LLC method was later modified in [50] to fix the outlier issue encountered when embedding new data by considering the manifold as a mixture of t-distributed subspaces, as the t-distribution has a larger variance and longer tails than the Gaussian or normal distribution [145]. Moreover, nonnegative adaptive feature extraction (NAFE) [209] is a method akin to LLE (they use similar optimization formulations) that is able to embed new data. NAFE is built on the basis of sparse NMF (see Section 2.2.2) to correct outliers, and weight learning is then performed over the new representations. Thus, NAFE learns the manifold on the projected space Z, unlike LLE, which learns the manifold from the original data space X and requires precalculating the learning weights W.

3.2.2 Laplacian Eigenmaps In the LE method, a mapping is computed using the eigenvectors of the Laplacian matrix of the adjacency graph $\mathcal{G}$ built from the data points [15]. The Laplace Beltrami operator (LBO) is defined on a manifold to measure the divergence of the gradient of a function, where this gradient can be considered a measure of the dispersion of nearby points in the mapping. The purpose of using this operator is to discover the local minima and maxima on the manifold surface in terms of the vectors' gradient divergences [99]. The LBO is related to the modeling of a heat distribution and results in a new function for estimating the distances in the graph induced by the data points. Thus, the core idea of the LE method is to use the LBO, which provides the ability to use the heat kernel to adjust the weight matrix and control its decay.

The algorithm starts from a graph of the data points, which is formalized as an adjacency matrix. The weight of the edge connecting two neighboring nodes can be defined using the heat kernel distance $W_{(i,j)} = \exp(-\frac{\|x_i - x_j\|^2)}{t})$, where t is a parameter that identifies the scale and data point pools. The Laplacian matrix L is then computed as $L = D - W$, where D is a diagonal matrix whose elements are the sum of the weights of each node: $D_{i,i} = \sum_j W_{i,j}$. The mapping is obtained from the solution of the eigenvector problem $Ly = \lambda Dy$ as the components of the 2^{nd} to $d+1^{th}$ eigenvectors (the first eigenvalue being zero), $Z = [Y_2, \ldots, Y_{d+1}]$, with $Y_j \in \mathbb{R}^{D\times 1}$ being the j^{th} eigenvector. The optimization procedure can be viewed as analogous to spectral clustering (SC) in the context of Laplacian graph cuts and segmentations [166]. Both the LE and SC methods involve optimization in the feature domain rather than the original data domain. Accordingly, the Laplacian graph can be optimized using the standard graph cut technique as $L = (D - W)$, where $\tilde{L} = D^{-1/2} \times L \times D^{-1/2}$

is the normalized version of L in the form of a symmetric positive semidefinite matrix, i.e., the diagonal entries are positive or zero. Finally, the Z components are taken to be the eigenvectors corresponding to the second smallest eigenvalue of $\tilde{L}$.

The LE approach has been utilized in several recent works; among the most popular methods are Hessian eigenmaps and Hessian-LLE (hLLE), which were introduced in [55] and possess the properties of both LLE and LE for identifying the k neighbors of a point and locally analyzing a nonlinear manifold. Here, the Hessian operator defines the second-order derivative of a multivariate function that is the eigenfunction of the tangent plane; thus, the Hessian matrix embeds the local curvature information. While the Laplacian is the second-order derivative and returns the sum of the derivative as a single scalar value for each neighbor (or subset of data), the Hessian operator outputs a matrix with all possible combinations [34]. Another extension of the LE method, introduced in [214], utilizes the projection of the principal curves [75], and the Hessian operator is applied in the projection space to obtain the low-dimensional coordinates.

3.2.3 Maximum Variance Unfolding In MVU [192][175], initially proposed as semidefinite embedding (SDE), a manifold is learned under the constraint of maximizing the variance between the mapped data samples. MVU is based on semidefinite programming optimization [184] and global isometric mapping [177]. MVU attempts to preserve the local geometry (distances and angles in the local neighborhood), unlike Isomap, which attempts to preserve the geodesic distances. This condition (local isometry) can be characterized by the constraint $\|z_i - z_j\|^2 = \|x_i - x_j\|^2$. The MVU assumes that to unfold the manifold, the variances in the mapping space must be maximized, which can be expressed as $\max \sum_{i,j} \|z_i - z_j\|^2$. To make the search for the mapping simpler and more robust, the problem can be reframed as a semidefinite programming problem, in which the solution is found by maximizing the trace of the Gram matrix $Z^\intercal Z$, where Z is the embedding space.

Numerous methods have been developed to learn manifolds by utilizing semidefinite programming. Among the state-of-the-art methods, the kernel matrix method proposed in [193] for learning a low-dimensional manifold depends on maximizing the variance in the feature space and is able to preserve the distances and angles among the members of each k-neighborhood. The minimum volume embedding (MVE) method proposed in [164] has a different objective function than SDE; specifically, the intent of MVE is to maximize the energy for the top d dimensions of the eigenspectrum to maintain the local distances among k neighbors. Moreover, to ensure that the structural and topological characteristics of the original data are retained, structure preserving embedding (SPE), proposed in [165], can be utilized for loss monitoring and visualization.

3.2.4 t-Distributed Stochastic Neighbor Embedding Maaten *et al.* [181] proposed t-SNE, which is considered a promising recent model for embedded unfolding and dimensionality reduction. Its main focus is also to preserve the local properties and global structure of the nonlinear manifold in the mapping space. The t-SNE model captures the outline of the global structure in terms of multiscale clustering; moreover, it preserves local properties by constructing a weight matrix derived from the local probability density of the local kernels or data point pools.

The t-SNE is built based on the stochastic neighbor embedding (SNE) method proposed in [83] and starts by identifying a similarity matrix based on an asymmetric joint distribution. The main differences between SNE and t-SNE include the type and shape of the distribution. SNE considers an asymmetric conditional probability (i.e., $P(i|j) \neq P(j|i)$) and a different standard deviation, σ_i, for each data point depending on its k neighbors. In contrast, in t-SNE, the conditional probability is converted into a symmetric joint distribution and a fixed standard deviation derived from all data points in the same kernel. Moreover, t-SNE and SNE differ in the statistical distribution used; SNE uses the Gaussian distribution, which suffers from the curse of dimensionality in the mapping

space, whereas t-SNE employs the t-distribution, which has a large variance and long tails and drops to zero in a smoother manner than the Gaussian. Thus, t-SNE offers a larger volume in the mapping space to accommodate higher dimensions (see Fig. 2 in the supplamentarity materials).

In t-SNE, the joint distribution in the data space between each data point x_i and its neighbor x_j is defined as $P = p(i, j)$, and a similar distribution is defined in the mapping space as $Q = q(i, j)$. An objective function is then introduced in terms of the divergence between these two distributions, using the KL divergence and gradient descent optimization: $min\ KL(P||Q) = \sum_i \sum_j p(i, j)(\frac{p(i,j)}{q(i,j)})$. Further extensions of the t-SNE technique include parameterized t-SNE [126], nonmetric visualization [182], and t-SNE acceleration by quadtrees to reduce the computational cost [180].

Finally, because of space limitations, we refer the reader to the supplementary materials that accompany this survey, which offer remarks, open research issues, and it gives further discussions.

4 Neural Network Models

Generative models based on NNs play a substantial role in learning representations of data. These models can be divided into two categories. Models in the first category are called directed generative models and require the prior distribution $P(V)$ (or $P(X)$) to define the states of the hidden units h; such methods also require a separate set of predefined parameters to define the states of the visible or input units (V or X) given the states of the hidden variables ($P(v|h)$ for single visible and hidden units) [79, 105]. The second category consists of undirected models, in which joint probabilities $P(v, h; w)$ between the visible and hidden states are defined given the weight parameters W. Furthermore, NNs models are inherently probabilistically undirected or directed; however, different models may comprise both directed and undirected graph schemes. Moreover, these models usually involve estimating the probability density function (pdf) of the approximated data (X^m, where m is the sample index) and comparing the pdf with the data prior $P(X)$. The posterior distribution $P(h|v)$ is also approximated to optimize the learning for unseen data [5].

From this perspective, there are three approaches that can be used to combine any two different pdfs: *(i)* a mixture, which can be obtained by taking a weighted average of the samples, where the resulting pdf is no smoother than the individual pdfs; *(ii)* a composition, in which the latent or hidden values are considered as the input to the next layer, provided that the connection between the layers is undirected; and *(iii)* a product, which is considered a powerful strategy, in which the pdfs are multiplied together and the resulting distribution is then normalized [80, 82]. Furthermore, either a product or a composition between the input elements v_i and the corresponding h_i defines a combined density distribution that is obtained by multiplying the input data by corresponding weights, followed by an activation function to append nonlinear properties to the whole mapping process, $h_i = \sigma(w_i^\top v_i)$. Thus, the first hidden units feed into subsequent units, and so on. Additionally, the following methods learn representations of the original data based on the Jacobian spectrum, which is the first derivative of the activation matrix that aggregates all activations across all of the network's neurons. Moreover, the subsequent methods are dissimilar to the manifold learning models which need to identify neighborhood relations between batches of data points in the early stage to generate mapping spaces. [19], see Section 3. However, the combinations between the MfL, BSS, and NNs able to bring interesting and obedient representations among the data to facilitate the learning and reduce the computational complexity. In the following, we review three classes of models derived from the NNs family, namely, energy-based models (EBMs), autoencoder models (AE), and adversarial learning models (GANs).

4.1 Energy-Based Models

EBMs encode the dependencies between the input data and the hidden units by modeling their joint probability, $P(v, h)$. Such a model is composed of several units bidirectionally connected to each other (using a schema that characterizes the model family), which can be partitioned into units representing the visible variables v and units representing the hidden variables h and can follow either a Bernoulli (binary) or Gaussian distribution [39] (for notational coherence, we use X to denote the input data; v to denote the visible layer, which receives the data; and h to denote the hidden layer). For each configuration of the units, an energy property can be defined and encoded in the weights of the unit connections W as well as the input. The model learning process is driven by an attempt to minimize the global energy of the network, a condition that is obtained when, for each configuration, its energy is inversely proportional to its joint probability, $P(v) \propto e^{-E(v)}$, according to the distribution of the training data [81]. More specifically, the probabilities for the visible and hidden units in an EBM are assigned in accordance with the Boltzmann distribution:

$$P(v,\ h;\ W) = \frac{1}{Z} e^{-E(v,\ h;\ W)} \tag{9}$$

where $Z = \sum_{v_i, h_j} e^{-E(v_i, h_j; w)}$ is the partition function that is used to normalize the approximated distribution over the possible values of the distribution variables. Consequently, such models learn and reconstruct the latent representations by capturing the interactions of v and h units; the results obtained are shared representations that can be generalized to unseen data. After training, such a model can provide a closed-form representation of the distribution of the training set and the hidden variables. In the following section, we highlight typical EBMs, including restricted Boltzmann machines (RBMs) [80], conditional restricted Boltzmann machines (CRBMs) [130, 132], deep belief networks (DBNs) [82], deep Boltzmann machines (DBMs) [159], and their relevant extensions.

4.1.1 Restricted Boltzmann Machines RBMs [169][80] were among the earliest models for data reconstruction based on Boltzmann machines [3] and are considered one of the principal families of generative undirected stochastic models. Because connections between units are allowed only between units of different layers (i.e., no connections are permitted between visible units or hidden units in the same layer), an RBM can be structured as a two-layer shallow network, where the first layer is the visible layer v and is associated with the input data X, while the second layer is the hidden layer h and is associated with the feature representation. The RBM can then be interpreted as a deterministic network that learns representations in the feature domain based on the expected activations of the hidden units [60]. The connections are characterized by the weight W and the two biases c and b for the visible and hidden units, respectively. For binary or Bernoulli units, the configurations (v, h) between the hidden and visible units have the following energy function:

$$E(v,\ h;\ W) = -\sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} c_j h_j - \sum_{i,j} v_i h_j W_{i,j} \tag{10}$$

Because there are no intralayer connections, the visible units are influenced only by the hidden units and vice versa, i.e., when h units are given, v units are independent: $P(v|h) = \prod_{i \in \text{hidden}} P(v_i|h)$. This also applies to the hidden units: $P(h|v) = \prod_{i \in \text{visible}} P(h_i|v)$. These properties simplify the training algorithm because variables of the same type (in the same layer) can be sampled simultaneously.

Probability approximation is performed by computing both the energy function $E(v, h)$ and the joint probability $P(v, h)$, and the activation between the v and h units is calculated from the conditional probabilities $P(v|h)$ and $P(h|v)$. Hence, the RBM assigns joint probabilities for pairs of v and h units in accordance with Eqn. (9). Additionally, the reconstruction error in the RBM is estimated using the k-step contrastive divergence [80] (see the supplementary materials for more details) to optimize the model parameters $\{W, b, c\}$ and maximize the log-likelihood probability.

Accordingly, the learning process is repeated several times while varying both the biases and the weights until the reconstruction error is minimized; further guidelines are given in [81]. RBMs are used for various tasks, including multiclass classification, collaborative filtering, and information retrieval. They are also utilized as pretraining facilitators for autoencoder networks [84].

4.1.2 Conditional Restricted Boltzmann Machines Despite the plethora of applications based on RBMs, RBMs do not perform well on structured data, such as in multi-label classification and time-series applications, where the output configuration space can be of large dimensionality [18]. To this end, in [132], the CRBM was proposed. The CRBM is similar to the RBM except that a conditional variable u is introduced to retain the context information (*e.g.,* the historical values of the inputs v) and acts as the gateway to control the v and h activations. Analytically, the CRBM models the energy function $E(v, h, u)$ and provides $P(v, h|u)$, which can be used to obtain $P(v|u)$ (by marginalizing over h). The CRBM is an undirected model, similar to the RBM, where in addition to the v and h layers coupled by the weight matrix W with the biases and activation (or conditional) probabilities, an additional layer u is coupled to v and h by directed connections and weight matrices A and B, for v and h, respectively. During the learning, the pairs (u, v) are used as input (visible layer), while after training, only the u variables are known, and v has to be estimated from $P(v|u)$.

There are various extensions of CRBMs; among the most popular are Gaussian-Bernoulli restricted Boltzmann machines (GBRBMs) [40], which follow the same principle as CRBMs but with differences in the probability distributions. Whereas the visible units in a GBRBM follow a Gaussian distribution, the hidden units are forced to take the form of a Bernoulli logistic distribution [18, 39]. Another CRBM model, proposed in [120], integrates a label layer instead of the historical unit layer u as the conditional layer. Moreover, in a gated restricted Boltzmann machine (GRBM) [130], the variables are conditioned to interact multiplicatively with modulated filters. A GRBM can be regarded as a high-order RBM because multiplication can be performed among three units or more, with the weight matrix W being a 3-way tensor among the input, hidden, and output layers. CRBMs are used in motion recognition, multiclass simultaneous labeling, and nonlinear time-sequence analysis; recently, interesting results have been achieved for gait recognition [41].

4.1.3 Deep Belief Networks Deep learning has become increasingly popular in the ML community since the breakthrough work presented in 2006 by Hinton *et al.* [82], which offers a new approach for successfully training deep NNs. The main challenges leading to unsuccessful training lie in random initializations and standard gradient optimization, which had previously resulted in failure to train most multilayer deep NNs [67]. Hinton *et al.* introduced the correct recipe for efficient representation learning utilizing a complementarity prior to overcoming the above challenges. These authors were inspired by [61] when conceiving of the deep architecture as a sequential model. A DBN is a probabilistic model constructed by sequentially combining several simple blocks that are forced to extract different representations. The learning process described in [82] is performed by means of greedy layerwise training, i.e., training one layer at a time. In terms of structure, a DBN is identical to a multilayer perceptron (MPL); however, they differ in their training procedures. DBNs are hybrid generative models that are composed of multiple directed sigmoid layers except for the top layers, which form an RBM model; the joint distribution of the variables in the layers of a DBN can be factorized into the conditional probabilities of variables in two successive layers.

The training of a DBN is performed through a recursive procedure that starts with the training of an RBM. The weight matrix between the visible and hidden layers is then frozen, and its (transposed) replica is used to initialize the weight matrix of a new layer. The hidden variables of the first layer then become the visible variables of the second layer, and training is performed only on the second weight matrix. The complete training procedure can be found in [82, 84, 158].

Convolutional deep belief networks (CDBNs), which were proposed to address large-scale image processing applications, are an extension of DBNs. A CDBN follows the same principles as a DBN; however, it uses a stack of convolutional filters to satisfy the image processing requirements [115]. DBNs are employed in various application domains, including regression and classification on highly structured data, human motion analysis, and character recognition, and speech recognition [133]. Notably, DBN models are distinct from DBMs [159], of which we give a brief overview below.

4.1.4 Deep Boltzmann Machines With the deep learning revolution, simpler RBM models have been combined to form DBM models [82, 158, 159]. A DBM differs from a DBN in that all of the connections among layers are undirected. DBMs are undirected graph models that approximate the gradient of the likelihood by the variational expectations over dependent data instead of randomly approximating the expectations as in RBMs. Additionally, DBMs use Markov chain Monte Carlo (MCMC) sampling to estimate the gradient of the intractable partition function Z. Variational learning has been proposed to maximize the log-likelihood $P(v; W)$ and minimize the KL divergence between the approximated distribution $Q(h|v; \mu)$ and the exact posterior $P(h|v; \mu)$, utilizing a naïve mean-field method involving factorization of the approximated posterior [194].

The motivation behind DBMs is that interpolation applications, such as speech and image, require the estimation of the posterior over the hidden units to be performed in a single mode; accordingly, the naïve mean-field method attempts to perform unimodal approximation. Moreover, DBMs differ from DBNs in that both sample generation and inference are performed within the network. After the first bottom-up pass is performed, the approximated distribution is used in the top-down pass, exploiting the complementary information of the intermediate features. The model training for general DBMs begins with the greedy layer-by-layer pretraining of each RBM; then, the RBMs are stacked together to form a deep architecture (where attention to duplicated visible units in the bottom-up and top-down passes alleviates the double counting problem that is strongly present during training and inference) [159]. After pretraining, the DBM can be trained using contrastive divergence and the mean-field method. By contrast, the need for pretraining is avoided in the learning method presented in [69], where multimodal DBM learning inspired by [174] is introduced. Because of space limitations, we refer the reader to [147] for further details.

4.2 Autoencoder Models

Autoencoder (AE) models are UGL models that can be constructed with either shallow or deep architectures; they also do not depend on partition functions [109], as EBMs do (see Section 4.1). AE models share a similar goal of capturing the hidden structures of data by performing sample reconstruction, benefiting from the advantages of distinct encoding and decoding stages. For a given data sample $x_i \in \mathbb{R}^D$, the encoding stage produces a mapping $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$, $0 < d < D$, to the corresponding encoded data $z_i = f(x_i; \theta_e)$, while the decoding stage produces a mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$, which reconstructs an approximation of the original data: $\tilde{x}_i = g(z_i; \theta_d)$. The problem lies in finding the encoding/decoding parameters that minimize the reconstruction error:

$$\tilde{\theta}_e, \tilde{\theta}_d = \arg\min \|x_i - (g(\cdot; \theta_d) \circ f(\cdot; \theta_e))(x_i)\|^2 \tag{11}$$

Consequently, the optimal reconstruction error allows one to capture rich representations of the hidden structures buried in data, which is considered a key element of establishing a model's reliability and robustness [10]. Commonly, f and g can be composed of several encoding and decoding stages, with a high degree of symmetry, in which the mapping from the j^{th} stage is parameterized by a weight matrix W_j and a bias b_j. In the following, we review the development of classical autoencoders [157], denoising autoencoders (DAEs) [186], contractive autoencoders (CAEs) [152], variational autoencoders (VAEs) [100], and their extensions.

4.2.1 Classical Autoencoders One early type of autoencoder, previously known as a circuit, was proposed in [157] as a shallow model consisting of 3 layers, i.e., input, output, and hidden layers, where the size of the hidden layer is restricted to log_2 of the input layer size. The main role of such a circuit is to encode and decode a sequence of bits by utilizing the backpropagation error to learn internal representations of the data. Shallow autoencoders learn representations of the original data based on two associated steps [86]: *(i) encoding*, in which the features of the data are encoded in the hidden layer using the weight matrix W, and *(ii) decoding*, in which the encoded features are reconstructed by utilizing the same weights used in the encoding stage. Stated otherwise, the encoder is parameterized by weights and biases, whereas the decoder is parameterized only by biases because it uses the transpose of weights passed to it by the encoder. With the deep learning revolution, autoencoders have been stacked together to form a new generation of networks called stacked autoencoders [200]. These models have gained popularity due to their success in various tasks, including dimensionality reduction and discovering the underlying structures of data [84].

The difference between autoencoder models and EBMs, such as RBMs, lies in the complexity of the posterior distribution approximations calculated from the input data in EBMs as the depth of the stacks of RBMs increases. Additionally, the features extracted in EBMs are derived from the data distribution and do not actually represent usable feature vectors. By contrast, in autoencoder models, the process begins with directly mapping or encoding the inputs to their representations, generating parameterized feature vectors [19]. The aforementioned differences are related to the encoding stage, whereas the decoding phase is described by the regularization term, which is already included in RBMs as the stochastic binary activation of the neurons. Moreover, the idea behind regularization is to impose restrictions on the learning process to enhance the generalizability of the learned representations and minimize the reconstruction error for unseen examples. Although the regularization of autoencoder models can take different forms, the original form is a bottleneck architecture, which is formed by restricting the hidden layer to have fewer units than the input and output layers. Moreover, the role of such a bottleneck in autoencoder models can be viewed similarly to the role of PCA (see Section 2.1.1), i.e., reducing the dimensionality [162].

An alternative approach based on imposing regularization by adding sparsity, which can be conceptualized as increasing the number of hidden units to be greater than the number of units in the input layer, is similar to duplicating the inputs to better capture the hidden structure of the data without increasing the data processing time. The measure used to exploit the effect of sparsity is also called a sparsity penalty, which can be implemented by forcing the average neuron activations to be near 0. In practice, this can be done by identifying a sparsity hyperparameter, ρ, to be used as the condition for the activation of a specific neuron and then estimating the average activation of that neuron as $\tilde{\rho}$. Consequently, the KL divergence, $KL(\rho||\tilde{\rho})$, can be introduced to measure how ρ and $\tilde{\rho}$ are related [137]. However, the effects of regularization on representation consistency have led to the emergence of many versions of autoencoder models to achieve model generalization [22, 137]. Among the most popular models are DAEs [186], CAEs [152], and VAEs [100], all of which have been proposed as the results of various developments in regularization.

4.2.2 Denoising Autoencoders DAEs were proposed in [186] to obtain robust representations. The underlying principle is to add noise to or corrupt the input data and attempt to train an autoencoder to reconstruct the encoded data to obtain the clean or original data. The added noise can take different forms, e.g., Poisson or Gaussian noise [30]. Additionally, the progression of deep learning based on stacking different shallow autoencoders, as discussed in [110], has allowed DAEs to achieve a deep form; stacked denoising autoencoders (SDAEs) follow the same principle as DAEs except that the data reconstructed in the first DAE are fed as clean data to the next DAE, and so on

[187]. Recently, DAEs have been employed for various tasks, including machinery fault diagnosis, medical image processing, and collaborative filtering [207].

4.2.3 Contractive Autoencoders In a CAE [152], a modified optimization objective function is used to regularize the reconstruction. The matrix aggregating all activations throughout the encoder after one forward pass is known as the activation matrix, and its derivative is the Jacobian matrix, which is used for gradient optimization. The Frobenius norm of the encoder's Jacobian matrix is added to the optimization objective function of the decoder (on the generation side); thus, the reconstruction is forced to contract toward the original training data. In other words, the difference loss between the reconstructed and original data is penalized and increased by the Frobenius norm of the Jacobian matrix; thus, the form of the regularization offers the ability to discover intermediate structures in the data. Recently, CAEs have been used in radar imaging and domain adaptation and have been extended to aeronautical cybernetics [197].

4.2.4 Variational Autoencoders Variational inference (VI) methods belong to the Bayesian family of analysis methods; such methods can be used to approximate an intractable posterior over a large dataset using a simpler variational distribution to obtain the solution to an optimization problem [206]. By observing the encoder output, also known as the results of AE inference, one can find the approximate posterior distribution function $q(z|x)$ that parameterizes the latent distribution z according to the input data. Specifically, a VAE [100] is characterized by forcing the latent distribution to follow a unit Gaussian with certain μ and σ vectors, which can be regarded as the regularized form of the distribution. As the first step, the prior distribution of the latent variable $P(z)$ must be defined according to a unit Gaussian distribution. Accordingly, the generated distribution $q(z|x)$ and the prior distribution $P(z)$ can be compared using the KL divergence [151]. Moreover, adding noise to the approximated distributions by varying the standard deviations and training the autoencoder for reconstruction in accordance with the true prior is called the reparameterization trick, in which several different new distributions are generated for learning the prior. Thus, from the application of this reparameterization trick, this method acquired the name *"variational"*.

Consequently, the evidence lower bound (ELBO) of the log-likelihood can be optimized through direct backpropagation, by optimizing the covariance and mean parameters as a function of two losses. The evidence is considered to be the marginal likelihood probability of the data and is expressed as $P(x) = \int P(x,z)\,\mathrm{d}z$. The ELBO is defined as the minimum bound on the evidence, $ELBO = -\mathbb{E}_q[P(x,z) - q(z|x)]$ and can be utilized in the AE training process in the form $\log P(x) = ELBO + KL(p(z)||q(z|x))$, considering that $KL(p(z)||q(z|x))$ must be minimized and reach zero, whereas the ELBO must be maximized [206]. Here, the first loss is the latent loss assessed in terms of the KL divergence for each reparameterized approximated posterior $q(z|x)$ and the prior $P(z)$ and is used to judge the extent to which the latent variables of the reparameterized sample satisfy a unit Gaussian. The second loss is the generative loss $P(x|z)$, which measures the accuracy of the reconstruction as the average of the negative expectation of the log-likelihood. This metric (generative loss) represents the ability of the decoder to reconstruct the encoded data; when the decoder is highly accurate, the negative log terms will tend to be near zero [105].

There are several extensions of VAEs, the most popular being sparse-coding VAEs, which differ from VAEs in that the Gaussian distribution restriction on the encoder output is replaced with sparse coding [14]. In a denoising VAE, as proposed in [95], noise is added during the recognition or encoding stage, and the decoder is trained to reconstruct the original input. Another extension is an importance-weighted autoencoder (IWAE), which can optimize the ELBO based on derived significance weights [35]. Furthermore, a posterior collapse VAE is another version that can optimize degenerate local minima during learning [76]; in addition, the beta-VAE model has been developed to generate interpretable factorized representations and facilitate disentangled learning [78].

4.2.5 Diversified/Other Autoencoders The emergence of regularized AE variants has led to a flourishing of AE frameworks for different applications. Because of space limitations (See the supplementary materials), we discuss only major frameworks here. Recursive AEs (RAEs) were proposed in [171] and extended in [170] for natural language processing; the underlying idea of an RAE is to utilize a sparse tree to represent the general structure of sentences, including nodes and words. Moreover, at each node, the RAE learns the phrase features, and consequently it is trained to reconstruct the parents' representations, then it builds a similarity matrix between the branches and nodes. The similar work to (RAEs) was proposed for the natural language processing by Bowman in [32] and dedicated for the word sequence generation. Convolutional AEs (CoAEs) follow the same principles as DAEs; however, the weights are shared among all layers to preserve spatial locality, and CoAEs have been applied for image reconstruction [90, 129]. Another variant, which is specialized for Boolean applications, is called a Boolean AE (BAE) [11]. Finally, complex-valued AEs (CvAEs) were proposed in [12] to encode data from the complex domain, involving both real and imaginary data forms.

4.3 Generative Adversarial Learning Models

Generative adversarial learning models are a class of ML models that attempt to generate new data from scratch based on the concept of reducing the accuracy of discriminating between original and generated samples and mining the original data distribution. Such models are utilized for several tasks, including image/video synthesis, dimensionality reduction, recognition and visualization [70]. Although EBMs and autoencoders learn the latent parameters of data via undirected or directed probabilistic modeling, they obtain different representations on high-dimensional data. Such models face challenges in approximating the intractable posterior distributions ($P(z|x)$ or $P(y|x)$) of massive datasets [100] and in their inability to leverage linear transformations. Conversely, adversarial learning models, which consist of two separate networks participating in the complete learning cycle, are able to overcome these challenges experienced by previous generative models based on NNs. The first network is called the generator, **G**, and is able to generate new samples and noise. The second one is called the discriminator, **D**, and is trained to distinguish between real and fake samples. In the following models, these two networks share a similar optimization:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} V(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{X \sim P_X(x)}[\log \mathbf{D}(x)] + \mathbb{E}_{Z \sim P_{Z(z)}}[\log(1 - \mathbf{D}(\mathbf{G}(z)))] \tag{12}$$

Here, $\log \mathbf{D}(x)$ is the logarithm of the marginal probability that describes **D** loss on the real data. $\mathbb{E}$ is the expectation, which refers to a sum of discrete random probabilities. Additionally, it approximates the global loss for a set of possible random likelihood probabilities among the data after running n iterations or during learning. The term $\log(1-\mathbf{D}(\mathbf{G}(z)))$ represents the complementary loss of **D** for the generated data or distributions. The above optimization equation is common among adversarial learning models and is built upon the concept of a min-max game for the function $V(\mathbf{D}, \mathbf{G})$ to optimize the two opposing losses of **G** and **D**. However, different variants of this optimization function can be found in the literature, which characterize different models.

We will now describe the family of adversarial learning models, including standard GANs [70], adversarial autoencoders (AAEs) [127], conditional generative adversarial networks (CGANs) [131], and their relevant extensions.

4.3.1 Generative Adversarial Networks GANs, which were introduced in [70], are state-of-the-art generative models based on NNs. A GAN model is able to map imposed noise to a desired output and consists of two NN blocks. The first block is the generator **G**, which generates noisy samples based on gradient optimization to supply the discriminator **D** with samples that are similar to previously misclassified samples; thus, **G** can be used as a tool to determine **D**'s performance. The

other block is the discriminator **D**, which recognizes whether samples are drawn from the generator distribution $P_Z(z)$ or from the real data distribution $P_X(x)$ on the basis of a probability measure. The model is trained through stochastic backpropagation optimization and regularized via dropout [173]. A GAN is similar to a VAE [100] in that it uses a similar propagation method for optimization; however, they differ in their optimization goals. A VAE performs optimization with respect to the latent space, whereas as GAN performs optimization and differentiation based on the input space.

The learning process starts with the definition of the prior noise distribution $q(z)$; then, the imposed noise is passed through the generator network and multiplied by the weights to be mapped to the data. The discriminator **D** receives data from two distributions, namely, the real data distribution $P_X(x)$ and the generated data distribution $P_Z(z)$; thus, the GAN loss is composed of two probability components, as in Eqn. (12). Moreover, after the first pass is completed, the generator **G** samples the discriminator's distribution to adapt the variances of subsequent samples. Accordingly, **G** iteratively adjusts the samples it generates until the min-max optimization process converges to the global optimum and reaches "equilibrium", i.e., when the real data converge with the data produced by the generator **G**, $P_Z(z) = P_X(x)$, and the discriminator probability for both sides of Eqn. (12) is $\mathbf{D}(\mathbf{G}(z)) = 1/2 = \mathbf{D}(x)$. GANs have been considered promising models in the last five years, even though they suffer from challenges due to the gradient vanishing problem for the generator and the possibility of nonconvergence leading to instability [160].

Consequently, recent works have devoted attention to optimizing GAN learning. From the perspective of learning optimization, a bidirectional GAN was proposed in [53] considering the joint distribution between the real data and their latent features $X = (x, h_x)$ as well as that between the generated samples and their latent space $Z = (z, h_z)$; thus, **D** can discriminate between X and Z based on the assigned joint distributions. To force **G** to produce samples that are close to the real boundary (the manifold of the real data), in a least-squares GAN, as introduced in [128], the sigmoid cross-entropy loss is replaced with a least square error, i.e., the log term is removed from the objective function. Another extension aimed at forcing a GAN to learn the distribution of the target data through the low-dimensional manifold is called a Wasserstein GAN, proposed in [7] and improved in [73]. The boundary equilibrium GAN (BEGAN) proposed in [23] focuses on the error distribution rather than on directing matching the distributions of the real and generated samples.

The abovementioned works cover GAN-based learning (optimization); in the following, we review advances related to the model architecture, separating the models into two classes to enhance the understanding. Additional details can be found in a recent survey of GAN models in [89].

4.3.2 Adversarial Autoencoders VI is utilized in VAEs to approximate the intractable posterior [100]; similarly, inference can be performed using GANs to regularize AEs. AAEs were proposed in [127] and can be conceptualized as consisting of two interconnected networks. The first network is the AE net, which maps the data $X \in \mathbb{R}^D$ into the latent space $Z \in \mathbb{R}^d$, through a set of transformations θ_e to produce the latent distribution $q(z_i|x_i)$ and subsequently reconstructs the latent features in accordance with the posterior $P(\tilde{x_i}|z_i)$ and transformations θ_d. The second network is the adversarial net, which contains **G** (which is identical to the encoder of the first network because the AE encodes both real and generated samples) and **D**. Once the real and generated samples are mapped to their own latent spaces, the two networks are jointly trained via stochastic gradient descent in two phases. The first stage focuses on optimizing the reconstruction error of the AE; the latent codes from the aggregated posterior $q(z)$ are passed through the **D** to increase the confidence in the real and generated data distributions. The second stage is concerned with regularizing the AE network by forcing the aggregated posterior $q(z)$ to match the randomly drawn prior (noise) $P(z)$, as the decoder acts as the generator's teacher; note that by contrast, in a VAE, the aggregated posterior $q(z)$ is matched with the real data prior $P(x)$.

In an AAE model, the aim is to produce a regularized AE such that the aggregated posterior $q(z)$ matches the randomly imposed prior $P(z)$ by replacing the latent loss assessed in terms of the KL divergence in the VAE learning with an adversarial loss, as expressed in Eqn. (12). Conversely, the reconstruction loss $P(x|z)$ has been replaced from the VAE with the adversarial loss that is taken from **D** of the GAN sub-module in [111]. Whereas the authors have been combined the VAE and GAN models as a single model to guide the VAE by the learned representations at **D**, i.e., the learned representations at **D** side are considered the basis for the VAE sub-module. The idea behind learning both VAE and GAN jointly in [111] is that the element-wise reconstruction error taken from the VAE is not sufficient for invariant signals and images, thus exploiting the VAE to encode the data and GAN for a better data generations tasks, and better similarities measures in the data space. Consequently, the similarity metric can be taken from the **D** side, alleviating the difficulties of choosing a similarity measure (for optimization) that plays the main role in training GANs.

The combination between GAN and VAE has been utilized to build a conditional fine-grained image generation model (CVAE-GAN) [13], which consists of four sub-modules: *(i)* The Encoder that maps data to the latent space z, *(ii)* **G** to generate data by sampling from the learned distribution $P(x|z)$, *(iii)* **D** to distinguish between the real and generated samples, *(iv)* adversarial classification to measure the class probability of the data, i.e., posterior $P(c|x)$, c is a class label. The CVAE-GAN has been proposed for structure-preserving data generation, in which the encoder and generator are utilized to map the real data x to the synthesis one $\tilde{x}$, utilizing pair-wise feature matching and pixel-wise l_2 loss. The CVAE-GAN architecture followed in Zero-VAE-GAN model [62], to generate high-resolution features for generalized Zero-Shot learning. The Zero-VAE-GAN exploits class-level semantic embedding s as a condition for feature generation, i.e. $s \in \mathbb{R}^{d_s}$ and d_s is the embedding dimensionality. Moreover, it differs from CVAE-GAN of that the Zero-VAE-GAN generates features (instead of images) with a similar distribution of the real data for unseen classes. Accordingly, the semantic embedding of the data is considered as an auxiliary input instead of the class label to capture finer details for feature generation. Additionally, the Zero-VAE-GAN utilizes perceptual loss as a similarity metric (loss) for feature generation, by employing an intermediate output between the discriminator and classification sub-modules, as $||f_D(x) - f_D(\tilde{x})||_2^2 + ||f_C(x) - f_C(\tilde{x})||_2^2$.

Finally, literature includes many other works based on the integration of AEs with GANs; in [154], the intractable log-likelihood is substituted with a synthetic likelihood, the discriminator is utilized for VI when generating the implicit posterior, and the pixel ratio trick is applied to measure how the real and generated distributions are related. Based on the denoising AAEs (DAAEs) inspired by [92, 95], Creswell *et al.* [47] proposed two extensions of DAAEs that combine both denoising and regularization terms. Whereas the first extension involves matches the aggregated posterior of the corrupted samples, $q(z|\tilde{x})$, with the predefined prior $q(z)$, the other involves matching the posterior of the reconstructed samples, $P(\tilde{x}|z)$, with the prior of the generated noise, $P(z)$.

4.3.3 Convolutional Adversarial Networks Despite the success of the many ML models that have been developed for image recognition and computer vision tasks, they have not proven their proficiency for synthesis applications such as image/video and multiclass labeling, i.e., for one-to-many mappings [132]. The CGAN model proposed in [131], which was the first GAN to address the above issues, is considered the nucleus of many GANs; it is constructed by introducing an additional input, or layer, into **D** and **G** to condition their learning. In a CGAN, the class labels are used for conditioning; therefore, such a model is classified as a supervised GAN. Improvements to CGANs followed in [52] with the proposal of a Laplacian Pyramid GAN (LAPGAN), which was introduced to generate high-resolution natural images. A LAPGAN is a stack of different levels, with each level consisting of a discriminator, a generator, and a Laplacian pyramid (LAP) operator. In each level, the image is processed with the LAP operator to obtain a downsampled version of

the image from the previous layer; thus, the noise can be reconstructed in accordance with the processed image versions and the real samples.

Further improvement of the CGAN followed in [96], therein introducing a general framework termed as pix2pix for image-to-image translation through conditioning both **G** and **D**, with the image structure leveraging the structure loss during learning. Also, in [96] the **G** built based on skip connection approach (or Residual learning [77], see Section 4 in our supplementary materials) and U-Net model [153], while the **D** termed as *patch*GAN and built based on L_1 norm (loss) to model only high frequencies, i.e., by penalizing the patches at their scales with a sparse loss. The main limitation of the pix2pix model is that the learning process requires imposing pair of aligned images, i.e., to map the data from domain X to Z then reconstruct it as $\tilde{X}$, a set of aligned images pairs must be provided (one form domain X and the other from the corresponding Z (ground-truth)). Cycle-Consistent GAN (Cycle-GAN) proposed in [219] for image-to-image translation under unpaired conditions, where two discriminators $\mathbf{D_X}$ and $\mathbf{D_Z}$ employed to enforce the **G** to translate X into output unrecognized by domain Z. Moreover, the Cycle-GAN uses two cycle consistency losses: forward loss ($x \rightarrow G(x) \rightarrow F(G(x)) \rightarrow \approx \tilde{x}$), and backward loss ($z \rightarrow G(z) \rightarrow F(G(z)) \rightarrow \approx \tilde{z}$).

The Cycle-GAN losses have recently been utilized for image deraining in the DerainCycleGAN model, which was proposed in [190] as a model for single-image deraining (SID). The Derain-CycleGAN model consists of an unsupervised attention-guided rain streak extractor (U-ARSE), two generators ($\mathbf{G_R}$ and $\mathbf{G_N}$) for generating rainy and rain-free images, and two discriminators ($\mathbf{D_R}$ and $\mathbf{D_N}$) corresponding to the generators. Similar to DerainCycleGAN, Semi-DerainGAN is a semisupervised model proposed for SID applications in [191], where both real and synthetic datasets were utilized to build the model. Unlike in DerainCycleGAN, a semisupervised rain streak mask learner (RSSML) is utilized in Semi-DerainGAN to learn rain streaks. Moreover, three generators ($\mathbf{G_r}$, $\mathbf{G_s}$, and $\mathbf{G_r^{'}}$) are used to address synthetic and real rainy images, and three corresponding discriminators ($\mathbf{D_r}$, $\mathbf{D_s}$, and $\mathbf{D_p}$) are used for multiscale image transformations and paring tasks.

Moreover, the CGAN [131] and pix2pix [96] methodologies in conditioning the generation stage, by utilizing the residual learning [77] and Patch-GAN, have been followed in [124]. By introducing two GAN models that are termed as attribute-GAN and CA-GAN. Both models share similar goals in guiding **G** to the synthesis of images with specific attributes and different supervision conditions, by using multi-discriminators frameworks. The attribute-GAN comprises a generator $\mathbf{G} : \mathbb{R}^X \rightarrow \mathbb{R}^Y$, two discriminators including real/ fake $\mathbf{D}_c : \mathbb{R}^Y \rightarrow \{0, 1\}$, and attribute $\mathbf{D}_{attri} : \mathbb{R}^Y \rightarrow \{1, ..., M\}$ or allocation $\mathbf{D}_{collc} : \mathbb{R}^X \times \mathbb{R}^Y \rightarrow \{0, 1\}$ (one for attribute and one for allocation but not both are used). However, the CA-GAN contains a generator $\mathbf{G} : \mathbb{R}^{X+1} \rightarrow \mathbb{R}^Y$, three discriminators including categorical discriminator $\mathbf{D}_{cate} : \mathbb{R}^Y \rightarrow \{0, ..., c-1\}$, also similar attribute and collocation discriminators which are used in the attribute-GAN. Moreover, both GAN models in [124] have been proposed for collocating clothes and cross-domain translation, due to the utilization of latent compatibility principles based on attributes, and they formulate the outfit collocation as a conditional factor for image generation, i.e., both models locate a mapping function from one domain to another.

The pix2pix model has not proven its superiority for high-resolution images with dimensions of $> 256 \times 256$; thus, in [188], the pix2pix framework was enhanced to a high-definition version called pix2pixHD by utilizing a robust adversarial loss, coarse-to-fine generators, and multiscale discriminators. In pix2pixHD, two generators (a global generator, $\mathbf{G_1}$, and a local enhancer, $\mathbf{G_2}$) and three discriminators ($\mathbf{D_1} - \mathbf{D_3}$) with identical architectures that operate at different scales are used to enhance the photorealism of the generated images. The utilization of different discriminators operating at the same scale had previously been proposed for an unconditional GAN [58], but in pix2pixHD, this approach is extended for the modeling of high-resolution images. The GAN loss

expressed in Eqn. (12) is improved in pix2pixHD by incorporating a least square error loss (*LSE*) to match the image features based on the discriminator. Recently, pix2pixHD has been further developed in [143] by using the same discriminators but replacing the *LSE* loss with a hinge loss and spatially adaptive (de)normalization (SPADE) for multimodal synthesis.

The limitations of the LAPGAN and CGAN in generating high-resolution natural images was addressed also in [149], by introducing the deep convolutional GAN "(DCGAN), which is utilized for different image generation tasks. The proposed model can scale the GAN using a specific CNN architecture by replacing the spatial pooling with a stride convolution, therein neglecting the fully connected layers and using different ReLU activation functions. Thus, if a noise vector with 100 dimensions is passed to the DCGAN, then both agents are learned jointly to reconstruct that vector as a 64×64 image. This methodology was utilized to build the first 3D GAN model in [198].

The structure of the DCGAN has been exploited with the Resnet [77] and self-attention network architectures [208] in [63], to build a self-attention driven adversarial similarity learning network (SAASLN). The idea behind proposing the SAASLN is that most similarity learning algorithms distinguish objects, lacking the explainability of the obtained similarity scores. The SAASLN neglects employing the whole object to obtain a similarity score, instead, it exploits the advantage of the self-attention mechanism [63] that retains the semantic information from parts-based representations to ensure that the final similarity scores are discriminative. Also, the SAASLN is an interpretable semantic similarity model that was proposed for information retrieval applications, where it comprises four sub-modules including *(i)* representations learning model that is taken from the Resnet, *(ii)* self-attention mechanism that is obtained from the self-attention network, *(iii)* similarity learning model, and *(iv)* generator-discriminator model that is taken from the DCGAN [149]. Moreover, the whole model is trained jointly based on a hybrid loss function, which considers all four sub-modules to optimize a final explainable similarity score.

Recently, many GAN models have been proposed to address the limitations of the original CGAN in term of new metrics, e.g., the major deep perceptual similarity metrics (DeePSiM) proposed in [56] as a new similarity metric. DeepSiM attempts to measure the distance between the generated image and the original image in the feature space instead of in the image space, by imposing the prior of the natural image and carrying out the measurements with respect to the transformed prior in the mapping space. Moreover, the DeePSiM metric is synthesized using the preferred inputs for the neurons in the deep generator network [139], which was used for the activation maximization (AM) to find which neuron is maximally activated when learning the image. Accordingly, by optimizing the imposed prior for the hidden distribution, one could stimulate the neurons to detect the preferred images and their features. A similar conditional prior for the AM was proposed in [138] with different generator outputs by enforcing the additional diversity prior in the latent space.

Finally, because of space limitations, we refer the reader to the supplementary materials that accompany this survey, which offer remarks, open research issues, and it gives further discussions.

5 Conclusion

We reviewed three families of unsupervised learning models that aim to capture the best representations from various datasets. The selected models and the paper structure were chosen to satisfy the demands of modern big data trends, in which the data size is exploding and their dimensions are high-order tensors. Our aim was to help beginners researchers be aware of where we are in unsupervised generative learning and how their role can affect the next generation of ML models. Moreover, the method for introducing this overview is to simplify the review process for interested researchers in this attractive field, thus motivating them to pursue further studies and introduce novel methodologies for learning representations on massive datasets.

From this perspective ("the current state of research"), we can conclude that, from the analyzed literature, it seems that explainable and interpretable ML models for real-time processing and big data scenarios have yet to be developed. An important discrepancy is found here: whereas the size of datasets is currently exploding in the big data era, the majority of recent work on practical methods has focused on implementation on small datasets. Generally, attempts to generalize the available methods to real-time applications in which the available data are messy, incomplete, or corrupted have not been fully considered by the research community. Finally, it will be important to develop conscientious approaches for decomposing and disentangling the main representative factors from reliable and higher-dimensional datasets. Such efforts should be followed by the development of countermeasures to overcome data incompleteness and the definition of suitable evaluation metrics for evaluating the extracted representations and overall performance in a standardized manner to facilitate learning and reduce complexity.

References

[1] M. A. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti. 2020. On approximating the non-negative rank: Applications to image reduction. In *Proc. of CIVEMSA*.

[2] M. A. M. Abukmeil, H. Elaydi, and M. Alhanjouri. 2015. Palmprint Recognition via Bandlet, Ridgelet, Wavelet and Neural Network. *Journal of Computer Sciences and Applications* 3, 2 (2015), 23–28.

[3] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science* 9, 1 (1985), 147–169.

[4] B. Alexandrov, V. V. Vesselinov, and H. N. Djidjev. 2018. *Non-negative Tensor Factorization for Robust Exploratory Big-Data Analytics.* Technical Report. Los Alamos National Lab. (LANL), Los Alamos, NM (United States).

[5] E. Alpaydin. 2020. *Introduction to machine learning.* MIT press.

[6] A. O. Andrade, S. Nasuto, P. Kyberd, and C. M. Sweeney-Reed. 2005. Generative topographic mapping applied to clustering and visualization of motor unit action potentials. *Biosystems* 82, 3 (2005), 273–284.

[7] M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein Generative Adversarial Networks *(Proc. of ICML).* 214–223.

[8] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina. 2005. Exploiting manifold geometry in hyperspectral imagery. *IEEE Trans. on Geoscience and Remote Sensing* 43, 3 (2005), 441–454.

[9] M. Balasubramanian and E. L. Schwartz. 2002. The isomap algorithm and topological stability. *Science* 295 (2002).

[10] P. Baldi. 2012. Autoencoders, Unsupervised Learning, and Deep Architectures. In *Proc. of the 2011 Int. Conf. on Unsupervised and Transfer Learning workshop.* 37–49.

[11] P. Baldi. 2012. Boolean autoencoders and hypercube clustering complexity. *Designs, Codes and Cryptography* 65, 3 (2012), 383–403.

[12] P. Baldi and Z. Lu. 2012. Complex-valued autoencoders. *Neural Networks* 33 (2012), 136–147.

[13] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. 2017. CVAE-GAN: fine-grained image generation through asymmetric training. In *Proc. of ICCV*. 2745–2754.

[14] G. Barello, A. Charles, and J. Pillow. 2018. Sparse-Coding Variational Auto-Encoders. *bioRxiv* (2018), 399246.

[15] M. Belkin and P. Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15, 6 (2003), 1373–1396.

[16] R. M. Bell and Y. Koren. 2007. Lessons from the Netflix prize challenge. *ACM Sigkdd Explorations* 9, 2 (2007), 75–79.

[17] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. 1997. A blind source separation technique using second-order statistics. *IEEE Trans. on Signal Processing* 45, 2 (1997), 434–444.

[18] Y. Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* (2009).

[19] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[20] Y. Bengio, H. Larochelle, and P. Vincent. 2006. Non-local manifold parzen windows. In *Proc. of NIPS*. 115–122.

[21] Y. Bengio and M. Monperrus. 2005. Non-local manifold tangent learning. In *Proc. of NIPS*. 129–136.

[22] Y. Bengio, L. Yao, G. Alain, and P. Vincent. 2013. Generalized denoising auto-encoders as generative models. In *Proc. of NIPS*. 899–907.

[23] D. Berthelot, T. Schumm, and L. Metz. 2017. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* (2017).

[24] F. M. Bianchi, D. Grattarola, and C. Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. (2020).

[25] C. M. Bishop, M. Svensén, and C. K. Williams. 1998. GTM: The generative topographic mapping. *Neural Computation* 10, 1 (1998), 215–234.
[26] C. M. Bishop, M. Svensén, and C. K. I. Williams. 1998. Developments of the generative topographic mapping. *Neurocomputing* 21, 1-3 (1998), 203–224.
[27] D. M. Blei, T. L. Griffiths, and M. I. Jordan. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM* 57, 2 (2010), 1–30.
[28] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. 2017. Variational inference: A review for statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3 (January 2003), 993–1022.
[30] C. Boncelet. 2009. Image noise models. In *The Essential Guide to Image Processing*. Elsevier, 143–167.
[31] W. M. Boothby. 1986. *An introduction to differentiable manifolds and Riemannian geometry*. Vol. 120. Academic press.
[32] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. 2016. Generating Sentences from a Continuous Space. In *Proc. of CoNLL*. 10–21.
[33] M. Brand. 2003. Charting a manifold. In *Proc. of (NIPS*. 985–992.
[34] R. Brualdi and H. Ryser. 1991. *Combinatorial Matrix Theory*. Cambridge University Press.
[35] Y. Burda, R. B. Grosse, and R. Salakhutdinov. 2016. Importance Weighted Autoencoders. *CoRR* abs/1509.00519 (2016).
[36] L. Cao. 2017. Data science: a comprehensive overview. *Comput. Surveys* 50, 3 (2017), 43.
[37] L. Cao and C. Zhang. 2008. Domain driven data mining. In *Data Mining and Knowledge Discovery Technologies*. IGI Global, 196–223.
[38] S.-S. Cheng, H.-C. Fu, and H.-M. Wang. 2009. Model-based clustering by probabilistic self-organizing maps. *IEEE Trans. on Neural Networks* 20, 5 (2009), 805–826.
[39] K. H. Cho, A. Ilin, and T. Raiko. 2011. Improved learning of Gaussian-Bernoulli restricted Boltzmann machines. In *Proc. of ICANN*. 10–17.
[40] K. H. Cho, T. Raiko, and A. Ilin. 2013. Gaussian-bernoulli deep boltzmann machine. In *Proc. of IJCNN*. 1–7.
[41] E. Chong and F. C. Park. 2017. Movement prediction for a lower limb exoskeleton using a conditional restricted Boltzmann machine. *Robotica* 35, 11 (2017), 2177–2200.
[42] A. Cichocki. 2018. Tensor networks for dimensionality reduction, big data and deep learning. In *Advances in Data Analysis with Computational Intelligence Methods*. Springer, 3–49.
[43] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. 2009. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons.
[44] A. Coates, A. Ng, and H. Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proc. of AISTATS*. 215–223.
[45] P. Comon. 1994. Independent component analysis, a new concept? *Signal processing* 36, 3 (1994), 287–314.
[46] M. A. Cox and T. F. Cox. 2008. Multidimensional scaling. In *Handbook of data visualization*. Springer, 315–347.
[47] A. Creswell and A. A. Bharath. 2018. Denoising adversarial autoencoders. *IEEE Trans. on Neural Networks and Learning Systems* (2018), 1–17.
[48] L. De Lathauwer, B. De Moor, and J. Vandewalle. 2000. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications* 21, 4 (2000), 1253–1278.
[49] B. L. R. De Moor and G. H. Golub. 1991. The restricted singular value decomposition: properties and applications. *SIAM J. Matrix Anal. Appl.* 12, 3 (1991), 401–425.
[50] D. de Ridder and V. Franc. 2003. *Robust manifold learning*. Czech Technical University.
[51] P. DeLellis, G. Polverino, G. Ustuner, N. Abaid, S. Macrì, E. M. Bollt, and M. Porfiri. 2014. Collective behaviour across animal species. *Scientific Reports* 4 (2014), 3723.
[52] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. 2015. Deep Generative Image Models using a Laplacian pyramid of Adversarial Networks. In *Proc. of NIPS*. 1486–1494.
[53] J. Donahue, P. Krähenbühl, and T. Darrell. 2017. Adversarial Feature Learning. In *Proc. of ICLR*.
[54] D. L. Donoho and C. Grimes. 2002. When does geodesic distance recover the true hidden parametrization of families of articulated images?. In *Proc. of ESANN*. 199–204.
[55] D. L. Donoho and C. Grimes. 2003. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. of the National Academy of Sciences* 100, 10 (2003), 5591–5596.
[56] A. Dosovitskiy and T. Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *Proc. of NIPS*. 658–666.
[57] S. Dray and J. Josse. 2015. Principal component analysis with missing values: a comparative survey of methods. *Plant Ecology* 216, 5 (2015), 657–667.
[58] I. P. Durugkar, I. Gemp, and S. Mahadevan. 2017. Generative Multi-Adversarial Networks. In *Proc. of ICLR*.

[59] A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. 2012. A brief survey of automatic methods for author name disambiguation. *ACM Sigmod Record* 41, 2 (2012), 15–26.
[60] A. Fischer and C. Igel. 2014. Training restricted Boltzmann machines: An introduction. *Pattern Recognition* 47, 1 (2014).
[61] Y. Freund. 1995. Boosting a weak learning algorithm by majority. *Information and Computation* 121, 2 (1995), 256–285.
[62] R. Gao, X. Hou, J. Qin, J. Chen, L. Liu, F. Zhu, Z. Zhang, and L. Shao. 2020. Zero-VAE-GAN: Generating Unseen Features for Generalized and Transductive Zero-Shot Learning. *IEEE Trans. on Image Processing* 29 (2020), 3665–3680.
[63] X. Gao, Z. Zhang, T. Mu, X. Zhang, C. Cui, and M. Wang. 2020. Self-attention driven adversarial similarity learning network. *Pattern Recognition* (2020), 107331.
[64] A. E. Gawęda, J. Kacprzyk, L. Rutkowski, and G. G. Yen. 2017. *Advances in data analysis with computational intelligence methods: dedicated to Professor Jacek Żurada*. Vol. 738. Springer.
[65] A. Genovese, V. Piuri, K. N. Plataniotis, and F. Scotti. 2019. PalmNet: Gabor-PCA convolutional networks for touchless palmprint recognition. *IEEE Trans. on Information Forensics and Security* 14, 12 (2019), 3160–3174.
[66] A. Gepperth and B. Pfülb. 2020. A Rigorous Link Between Self-Organizing Maps and Gaussian Mixture Models. In *Proc. of ICANN*.
[67] X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*. 249–256.
[68] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. 2005. Neighbourhood components analysis. In *Proc. of NIPS*. 513–520.
[69] I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio. 2013. Multi-prediction deep Boltzmann machines. In *Proc. of NIPS*. 548–556.
[70] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Proc. of NIPS*. 2672–2680.
[71] D. Grattarola, D. Zambon, L. Livi, and C. Alippi. 2019. Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. *IEEE Trans. on Neural Networks and Learning Systems* 31, 6 (2019), 1856–1869.
[72] T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum, and D. M. Blei. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Proc. of NIPS*. 17–24.
[73] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. 2017. Improved training of wasserstein gans. In *Proc. of NIPS*. 5767–5777.
[74] R. A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. (1970).
[75] T. Hastie and W. Stuetzle. 1989. Principal curves. *J. Amer. Statist. Assoc.* 84, 406 (1989), 502–516.
[76] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. 2019. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. *arXiv preprint arXiv:1901.05534* (2019).
[77] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*. 770–778.
[78] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Proc. of ICLR*.
[79] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine* 29, 6 (November 2012), 82–97.
[80] G. E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 8 (2002), 1771–1800.
[81] G. E. Hinton. 2012. A practical guide to training restricted Boltzmann machines. 599–619.
[82] G. E. Hinton, S. Osindero, and Y.-W. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 7 (2006), 1527–1554.
[83] G. E. Hinton and S. T. Roweis. 2002. Stochastic neighbor embedding. In *Proc. of NIPS*. 857–864.
[84] G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[85] G. E. Hinton, T. J. Sejnowski, and T. A. Poggio. 1999. *Unsupervised learning: foundations of neural computation*. MIT.
[86] G. E. Hinton and R. S. Zemel. 1994. Autoencoders, minimum description length and Helmholtz free energy. In *Proc. of NIPS*. 3–10.
[87] S. Hochreiter and J. Schmidhuber. 1997. Low-Complexity Coding and Decoding. In *Theoretical Aspects of Neural Computation*. 297–306.
[88] H. Hong. 2007. Multimodal Discovering and Fusion for Semantics multimedia analysis. In *Proc. of ALPIT*. 155–158.
[89] Y. Hong, U. Hwang, J. Yoo, and S. Yoon. 2019. How generative adversarial networks and their variants work: An overview. *Comput. Surveys* 52, 1 (2019), 1–43.
[90] H. Huang, X. Hu, Y. Zhao, M. Makkie, Q. Dong, S. Zhao, L. Guo, and T. Liu. 2018. Modeling task fMRI data via deep convolutional autoencoder. *IEEE Trans. on Medical Imaging* 37, 7 (2018), 1551–1561.

[91] L. Huang, X. Liu, B. Lang, A. Yu, Y. Wang, and B. Li. 2018. Orthogonal Weight Normalization: Solution to Optimization Over Multiple Dependent Stiefel Manifolds in Deep Neural Networks. In *Proc. of AAAI*.
[92] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. 2017. Stacked generative adversarial networks. In *Proc. of CVPR*. 5077–5086.
[93] A. Hyvärinen, J. Karhunen, and E. Oja. 2004. *Independent component analysis*. Vol. 46. John Wiley & Sons.
[94] A. Hyvärinen and E. Oja. 2000. Independent component analysis: algorithms and applications. *Neural Networks* 13, 4-5 (2000), 411–430.
[95] D. I. J. Im, S. Ahn, R. Memisevic, and Y. Bengio. 2017. Denoising criterion for variational auto-encoding framework. In *Proc. of AAAI*.
[96] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proc. of CVPR*. 1125–1134.
[97] E. T. Jaynes. 2003. *Probability theory: The logic of science*. Cambridge University Press.
[98] I. Jolliffe. 2011. *Principal component analysis*. Springer.
[99] J. Jost and J. Jost. 2008. *Riemannian geometry and geometric analysis*. Vol. 42005.
[100] D. P. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. of ICLR*.
[101] V. Klema and A. Laub. 1980. The singular value decomposition: Its computation and some applications. *IEEE Trans. on Automatic Control* 25, 2 (1980), 164–176.
[102] T. Kohonen. 1990. The self-organizing map. *Proc. of the IEEE* 78, 9 (1990), 1464–1480.
[103] T. Kohonen. 2013. Essentials of the self-organizing map. *Neural networks* 37 (2013), 52–65.
[104] T. G. Kolda and B. W. Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[105] D. Koller, N. Friedman, and F. Bach. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.
[106] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*. 1097–1105.
[107] J. B. Kruskal. 1964. Nonmetric multidimensional scaling: a numerical method. *Psychometrika* 29, 2 (1964), 115–129.
[108] S. Kullback. 1997. *Information theory and statistics*. Courier Corporation.
[109] M. Längkvist, L. Karlsson, and A. Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42 (2014), 11–24.
[110] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. 2009. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research* 10 (January 2009), 1–40.
[111] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. 2016. Autoencoding beyond Pixels Using a Learned Similarity Metric. In *Proc. of ICML*. 1558–1566.
[112] M. H. C. Law and A. K. Jain. 2006. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28, 3 (March 2006), 377–391.
[113] D. D. Lee and H. S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.
[114] D. D. Lee and H. S. Seung. 2001. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*. 556–562.
[115] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. of ICML*. 609–616.
[116] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. 2017. Feature selection: A data perspective. *Comput. Surveys* 50, 6 (2017), 1–45.
[117] W. Li and A. McCallum. 2006. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proc. of ICML*. 577–584.
[118] W.-J. Li, D.-Y. Yeung, and Z. Zhang. 2009. Probabilistic relational PCA. In *Proc. of NIPS*. 1123–1131.
[119] X. Li, S. Lin, S. Yan, and D. Xu. 2008. Discriminant locally linear embedding with high-order tensor data. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38, 2 (2008), 342–352.
[120] X. Li, F. Zhao, and Y. Guo. 2015. Conditional Restricted Boltzmann Machines for Multi-label Learning with Incomplete Labels. In *Proc. of AISTATS*. 635–643.
[121] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. 2017. Adversarial ranking for language generation. In *Proc. of NIPS*. 3155–3165.
[122] S. Lipovetsky. 2009. PCA and SVD with nonnegative loadings. *Pattern Recognition* 42, 1 (2009), 68–76.
[123] L. Liu, L. Tang, L. He, W. Zhou, and S. Yao. 2016. An overview of hierarchical topic modeling. In *Proc. of IHMSC*. 391–394.
[124] L. Liu, H. Zhang, X. Xu, Z. Zhang, and S. Yan. 2019. Collocating clothes with generative adversarial networks cosupervised by categories and attributes: a multidiscriminator framework. *IEEE Trans. on Neural Networks and Learning Systems* (2019).
[125] E. López-Rubio. 2010. Probabilistic self-organizing maps for continuous data. *IEEE Trans. on Neural Networks* 21, 10 (2010), 1543–1554.

[126] L. Maaten. 2009. Learning a parametric embedding by preserving local structure. In *Proc. of AISTATS*. 384–391.
[127] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).
[128] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. 2017. Least squares generative adversarial networks. In *Proc. of ICCV*. 2794–2802.
[129] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proc. of ICANN*. 52–59.
[130] R. Memisevic and G. Hinton. 2007. Unsupervised learning of image transformations. In *Proc. of (CVPR*. 1–8.
[131] M. Mirza and S. Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014).
[132] V. Mnih, H. Larochelle, and G. E. Hinton. 2011. Conditional Restricted Boltzmann Machines for Structured Output Prediction. In *Proc. of UAI*. 514–522.
[133] A.-r. Mohamed, G. E. Dahl, and G. Hinton. 2012. Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Processing* 20, 1 (2012), 14–22.
[134] B. Muthen. 2001. Latent variable mixture modeling. *New developments and techniques in structural equation modeling* 2 (2001), 1–33.
[135] B. Muthén. 2004. Latent variable analysis. *The Sage handbook of quantitative methodology for the social sciences* 345, 368 (2004), 106–109.
[136] I. Nabney. 2002. *NETLAB: algorithms for pattern recognition*. Springer Science & Business Media.
[137] A. Ng and others. 2011. Sparse autoencoder. *CS294A Lecture notes* 72, 2011 (2011), 1–19.
[138] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. 2017. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proc. of CVPR*. 4467–4477.
[139] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proc. of NIPS*. 3387–3395.
[140] V.-A. Nguyen, J. L. Ying, and P. Resnik. 2013. Lexical and hierarchical topic regression. In *Proc. of NIPS*. 1106–1114.
[141] P. Paatero and U. Tapper. 1994. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* 5, 2 (1994), 111–126.
[142] C. C. Paige and M. A. Saunders. 1981. Towards a generalized singular value decomposition. *SIAM J. Numer. Anal.* 18, 3 (1981), 398–405.
[143] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proc. of CVPR*. 2337–2346.
[144] K. Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572.
[145] D. Peel and G. J. McLachlan. 2000. Robust mixture modelling using the t distribution. *Statistics and computing* 10, 4 (2000), 339–348.
[146] Y. Petinot, K. McKeown, and K. Thadani. 2011. A Hierarchical Model of Web Summaries. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 670–675.
[147] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar. 2018. A Survey on Deep Learning: Algorithms, Techniques, and Applications. *Comput. Surveys* 51, 5 (2018), 92.
[148] S. Pouyanfar, Y. Yang, S.-C. Chen, M.-L. Shyu, and S. Iyengar. 2018. Multimedia big data analytics: A survey. *Comput. Surveys* 51, 1 (2018), 1–34.
[149] A. Radford, L. Metz, and S. Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
[150] D. A. Reynolds. 2009. Gaussian Mixture Models. *Encyclopedia of biometrics* 741 (2009).
[151] D. J. Rezende, S. Mohamed, and D. Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proc. of ICML*.
[152] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. of ICML*. 833–840.
[153] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of MICCAI*. 234–241.
[154] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. 2017. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987* (2017).
[155] S. Roweis and Z. Ghahramani. 1999. A unifying review of linear Gaussian models. *Neural Computation* 11, 2 (1999), 305–345.
[156] S. T. Roweis and L. K. Saul. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290, 5500 (2000), 2323–2326.
[157] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[158] R. Salakhutdinov. 2015. Learning deep generative models. *Review of Statistics and its Application* 2 (2015), 361–385.
[159] R. Salakhutdinov and G. Hinton. 2009. Deep boltzmann machines. In *Proc. of AISTATS*. 448–455.
[160] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved techniques for training gans. In *Proc. of NIPS*. 2234–2242.
[161] L. K. Saul and S. T. Roweis. 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research* 4 (Jun 2003), 119–155.
[162] J. Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.
[163] B. Schölkopf, A. Smola, and K.-R. Müller. 1997. Kernel principal component analysis. In *Proc. of ICANN*. 583–588.
[164] B. Shaw and T. Jebara. 2007. Minimum volume embedding. In *Proc. of AISTATS*. 460–467.
[165] B. Shaw and T. Jebara. 2009. Structure preserving embedding. In *Proc. of ICML*. 937–944.
[166] J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.
[167] L. Shu, B. Long, and W. Meng. 2009. A latent topic model for complete entity resolution. In *Proc. of ICDE*. 880–891.
[168] V. D. Silva and J. B. Tenenbaum. 2003. Global versus local methods in nonlinear dimensionality reduction. In *Proc. of NIPS*. 721–728.
[169] P. Smolensky. 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. Chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, 194–281.
[170] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. of NIPS*. 801–809.
[171] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*. 151–161.
[172] R. Souvenir and R. Pless. 2005. Manifold clustering. In *Proc. of ICCV*. 648–653 Vol. 1.
[173] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[174] N. Srivastava and R. R. Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Proc. of NIPS*. 2222–2230.
[175] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. 2006. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM review* 48, 4 (2006), 681–699.
[176] Y. W. Teh and S. T. Roweis. 2003. Automatic alignment of local representations. In *Proc. of NIPS*. 865–872.
[177] J. B. Tenenbaum, V. De Silva, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319–2323.
[178] M. E. Tipping and C. M. Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61, 3 (1999), 611–622.
[179] L. R. Tucker and others. 1964. The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology* 110119 (1964).
[180] L. Van der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research* 15, 1 (2014), 3221–3245.
[181] L. Van der Maaten and G. Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (November 2008), 2579–2605.
[182] L. Van der Maaten and G. Hinton. 2012. Visualizing non-metric similarities in multiple maps. *Machine Learning* 87, 1 (2012), 33–55.
[183] L. Van der Maaten, E. Postma, and J. Van den Herik. 2009. Dimensionality reduction: a comparative review. *Technical Report TiCC-TR 2009-005, Tilburg University,* (2009).
[184] L. Vandenberghe and S. Boyd. 1996. Semidefinite programming. *SIAM review* 38, 1 (1996), 49–95.
[185] P. Vincent and Y. Bengio. 2003. Manifold parzen windows. In *Proc. of NIPS*. 849–856.
[186] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*. 1096–1103.
[187] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising. *Journal of Machine Learning Research* 11 (2010), 3371–3408.
[188] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc. of CVPR*. 8798–8807.
[189] Y. Wang and Y. Zhang. 2013. Nonnegative Matrix Factorization: A Comprehensive Review. *IEEE Trans. on Knowledge and Data Engineering* 25, 6 (June 2013), 1336–1353.
[190] Y. Wei, Z. Zhang, J. Fan, Y. Wang, S. Yan, and M. Wang. 2019. DerainCycleGAN: An Attention-guided Unsupervised Benchmark for Single Image Deraining and Rainmaking. *arXiv preprint arXiv:1912.07015* (2019).

[191] Y. Wei, Z. Zhang, H. Zhang, J. Qin, and M. Zhao. 2020. Semi-DerainGAN: A New Semi-supervised Single Image Deraining Network. *arXiv preprint arXiv:2001.08388* (2020).
[192] K. Q. Weinberger and L. K. Saul. 2006. Unsupervised Learning of Image Manifolds by Semidefinite Programming. *Int. Journal of Computer Vision* 70, 1 (01 Oct 2006), 77–90.
[193] K. Q. Weinberger, F. Sha, and L. K. Saul. 2004. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proc. of ICML*. 106.
[194] M. Welling and G. E. Hinton. 2002. A new learning algorithm for mean field Boltzmann machines. In *Proc. of ICANN*. 351–357.
[195] C. K. Williams. 2002. On a connection between kernel PCA and metric MDS. *Machine Learning* 46, 1-3 (2002), 11–19.
[196] S. Wold, K. Esbensen, and P. Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.
[197] E. Q. Wu, G.-R. Zhou, L.-M. Zhu, C.-F. Wei, H. Ren, and R. S. Sheng. 2019. Rotated Sphere Haar Wavelet and Deep Contractive Auto-Encoder Network With Fuzzy Gaussian SVM for Pilot's Pupil. *IEEE Trans. on Cybernetics* (2019).
[198] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proc. of NIPS*. 82–90.
[199] F. Xiong, O. I. Camps, and M. Sznaier. 2011. Low order dynamics embedding for high dimensional time series. In *Proc. of ICCV*. 2368–2374.
[200] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi. 2016. Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE Trans. on Medical Imaging* 35, 1 (2016), 119–130.
[201] W. Xu, X. Jiang, X. Hu, and G. Li. 2014. Visualization of genetic disease-phenotype similarities by multiple maps t-SNE with Laplacian regularization. *BMC Medical Genomics* 7, 2 (2014), S1.
[202] M.-H. Yang. 2002. Face recognition using extended isomap. In *Proc. of ICIP*.
[203] C. H. Yu. 1977. Exploratory data analysis. *Methods* 2 (1977), 131–160.
[204] K. Yu and T. Zhang. 2010. Improved Local Coordinate Coding using Local Tangents. In *Proc. of ICML*. 1215–1222.
[205] K. Yu, T. Zhang, and Y. Gong. 2009. Nonlinear learning using local coordinate coding. In *Proc. of NIPS*. 2223–2231.
[206] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. 2019. Advances in Variational Inference. *IEEE Trans. on Pattern Analysis & Machine Intelligence* 41, 08 (August 2019), 2008–2026.
[207] C. Zhang, T. Li, Z. Ren, Z. Hu, and Y. Ji. 2019. Taxonomy-aware collaborative denoising autoencoder for personalized recommendation. *Applied Intelligence* 49, 6 (June 2019), 1–18.
[208] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. 2019. Self-attention generative adversarial networks. In *Proc. of ICML*. 7354–7363.
[209] Y. Zhang, Z. Zhang, S. Li, J. Qin, G. Liu, M. Wang, and S. Yan. 2018. Unsupervised nonnegative adaptive feature extraction for data representation. *IEEE Trans. on Knowledge and Data Engineering* 31, 12 (2018), 2423–2440.
[210] Y. Zhang, Z. Zhang, J. Qin, L. Zhang, B. Li, and F. Li. 2018. Semi-supervised local multi-manifold Isomap by linear embedding for feature extraction. *Pattern Recognition* 76 (2018), 662–678.
[211] Y. Zhang, Z. Zhang, Z. Zhang, M. Zhao, L. Zhang, Z. Zha, and M. Wang. 2020. Deep Self-representative Concept Factorization Network for Representation Learning. In *Proc. of ICDM*. 361.
[212] Z. Zhang, T. W. Chow, and M. Zhao. 2012. M-Isomap: Orthogonal constrained marginal isomap for nonlinear dimensionality reduction. *IEEE Trans. on Cybernetics* 43, 1 (2012), 180–191.
[213] Z. Zhang, F. Li, M. Zhao, L. Zhang, and S. Yan. 2017. Robust neighborhood preserving projection by nuclear/l2, 1-norm regularization for image feature extraction. *IEEE Trans. on Image Processing* 26, 4 (2017), 1607–1622.
[214] Z. Zhang and H. Zha. 2004. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing* 26, 1 (2004), 313–338.
[215] Z. Zhang, Y. Zhang, S. Li, G. Liu, D. Zeng, S. Yan, and M. Wang. 2019. Flexible Auto-weighted Local-coordinate Concept Factorization: A Robust Framework for Clustering. *IEEE Trans. on Knowledge and Data Engineering* (2019).
[216] Z. Zhang, Y. Zhang, G. Liu, J. Tang, S. Yan, and M. Wang. 2019. Joint label prediction based semi-supervised adaptive concept factorization for robust data representation. *IEEE Trans. on Knowledge and Data Engineering* 5 (2019), 952–970.
[217] Z. Zhang, Y. Zhang, L. Zhang, and S. Yan. 2020. A Survey on concept factorization: From shallow to deep representation learning. *arXiv preprint arXiv:2007.15840* (2020).
[218] G. Zhong, L.-N. Wang, X. Ling, and J. Dong. 2016. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science* 2, 4 (2016), 265–278.
[219] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of ICCV*. 2223–2232.
[220] M. Zibulevsky and B. A. Pearlmutter. 2001. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation* 13, 4 (2001), 863–882.