UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE
SCUOLA DI DOTTORATO IN INFORMATICA
DIPARTIMENTO DI INFORMATICA
DOTTORATO DI RICERCA IN INFORMATICA, XXXIV CICLO

Unsupervised Generative Models for Data Analysis and Explainable Artificial Intelligence

INF/01 INFORMATICA

TESI DI DOTTORATO DI RICERCA DI:
Ing. Mohanad Abukmeil

RELATORE:
Prof. Vincenzo Piuri

CORRELATORE:
Prof. Fabio Scotti
Dott. Angelo Genovese

DIRETTORE DELLA SCUOLA DI DOTTORATO:
Prof. Paolo Boldi

Anno Accademico 2020/2021

Acknowledgements

I would like to express my loyal gratitude to my supervisor Prof. Vincenzo Piuri for his incessant support, guidance, warm encouragement, and superintending my Ph.D. research activities. With his wisdom and patience, I always had the motivation to cope with obstacles, and I was able to finalize my Ph.D. journey with the exceptional conditions I faced. I am thankful to him and he will be the role model I look up to in my future career.

I also want to thank Prof. Fabio Scotti and Dr. Angelo Genovese for co-supervising this dissertation and offering me an appreciated effort and support. The thank is connected also to Prof. Stefano Ferrari for his valuable collaboration and guidance during my Ph.D. journey. A deep thank goes to Dr. Ruggero Donida Labati for his help and support during the research work, and for facilitating my activities in the laboratory.

I would like to express my deep appreciation to Prof. Cesare Alippi for offering me the opportunity to do an internship at the Università della Svizzera Italiana, I also thank his research team for their collaboration and the accompanying me during the internship. I also thank all I worked with, my colleagues, and all the Professors who helped me finalize my Ph.D. dissertation. Among them I want to mention, with no particular order, Prof. Pierangela Samarati, Prof. Sabrina De Capitani di Vimercati, Prof. Paolo Boldi, Dr. Massimo Walter Rivolta, and Dr. Fatme Hachem. I would also like to thank the administrative staff members of the Università degli Studi di Milano who always assisted me with the bureaucracy and official work procedures. Specifically, I would like to thank Lorena Sala, Giorgio Biacchi, Maria Pia Mericio, Anna Pozella, and Antonella Astori.

A special thank goes to the referees Prof. Konstantinos N. Plataniotis, Prof. Leszek Rutkowski, and Prof. Jianhua Ma for their time spent reviewing my thesis, and for offering me valuable suggestions for smoothing this dissertation.

Finally, my deepest gratitude and thank goes to my mother for her tenderness and belief in my potential. To my wife and children for their confidence and love. To my brothers and sisters for their never-ending support. To my friends and colleagues for their special encouragement. My doctoral dissertation is dedicated to all of them.

Abstract

For more than a century, the methods of learning representation and the exploration of the intrinsic structures of data have developed remarkably and currently include supervised, semi-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, missing, incomplete, unlabeled, or corrupted forms. Consequently, discovering and learning the hidden structure buried inside such data becomes highly challenging.

From this perspective, latent data analysis and dimensionality reduction play a substantial role in decomposing the exploratory factors and learning the hidden structures of data, which encompasses the significant features that characterize the categories and trends among data samples in an ordered manner. That is by extracting patterns, differentiating trends, and testing hypotheses to identify anomalies, learning compact knowledge, and performing many different machine learning (ML) tasks such as classification, detection, and prediction.

Unsupervised generative learning (UGL) methods are a class of ML characterized by their possibility of analyzing and decomposing latent data, reducing dimensionality, visualizing the manifold of data, and learning representations with limited levels of predefined labels and prior assumptions. Furthermore, explainable artificial intelligence (XAI) is an emerging field of ML that deals with explaining the decisions and behaviors of learned models. XAI is also associated with UGL models to explain the hidden structure of data, and to explain the learned representations of ML models. However, the current UGL models lack large-scale generalizability and explainability in the testing stage, which leads to restricting their potential in ML and XAI applications.

To overcome the aforementioned limitations, this thesis proposes innovative methods that integrate UGL and XAI to enable data factorization and dimensionality reduction to improve the generalizability of the learned ML models. Moreover, the proposed methods enable visual explainability in modern applications as anomaly detection and autonomous driving systems. The main research contributions are listed as follows:

- A novel overview of UGL models including blind source separation (BSS), manifold learning (MfL), and neural networks (NNs). Also,

the overview considers open issues and challenges among each UGL method.

- An innovative method to identify the dimensions of the compact feature space via a generalized rank in the application of image dimensionality reduction.
- An innovative method to hierarchically reduce and visualize the manifold of data to improve the generalizability in limited data learning scenarios, and computational complexity reduction applications.
- An original method to visually explain autoencoders by reconstructing an attention map in the application of anomaly detection and explainable autonomous driving systems.

The novel methods introduced in this thesis are benchmarked on publicly available datasets, and they outperformed the state-of-the-art methods considering different evaluation metrics. Furthermore, superior results were obtained with respect to the state-of-the-art to confirm the feasibility of the proposed methodologies concerning the computational complexity, availability of learning data, model explainability, and high data reconstruction accuracy.

Contents

List of Figures

List of Tables

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 [93]. 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 [3, 64, 158].

Typically, for latent data (or big data) analysis, the most difficult tasks include dimensionality reduction and uncovering the hidden structures of these massive datasets, which contain the latent features and characteristics of the data. Furthermore, the causality that explains the behavior of the data or models 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 [3, 354]. Moreover, the demand to disentangle the exploratory factors of data to reduce the processing time and produce interpretable results is currently very important [1, 62]. 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 flexibly to build manageable visualizers, predictors, and classifiers [2, 50, 51, 330].

To build a machine learning (ML) model, it is essential to identify the format that constitutes the data samples [176]. 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 [164]; *(ii)* unstructured data, that lacks the structure as images, videos, and text data [161]; 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 [228]. 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 [29]; 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 [2, 3, 173, 304].

Unsupervised learning (UL) models are a class of ML tools for learning data distributions, discovering the underlying structures of data, removing redundancies, and reducing the dimensionality of datasets with limited levels of prior assumptions and predefined labels [18, 65, 164, 229, 256]. 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" [1, 169]. Alternatively, X can be addressed probabilistically by supposing that the set of samples, X, has a joint distribution $P(X)$ ($X = \{x_i \,|\, x_i \in \mathbb{R}^D,\, i = 1, \ldots, N\}$); then, analysis can be performed based on that joint distribution $P(x_1, x_2, ..., x_N)$ [3, 159]. X can also be statistically represented based on n^{th}-order statistical moments, e.g., the mean, variance, skewness, and kurtosis [29, 171]. Moreover, X can be graphically represented as a set of nodes, $\nu = 1, .., N$, and relations between data samples, X, are represented by edges, ε, to view the data as an undirected or directed graph, $\mathcal{G}\{\nu, \varepsilon\}$, where the graph is characterized by its adjacency matrix $A \in \mathbb{R}^{N \times N}$ [36, 176, 281].

UL is usually applied in combination with generative modeling, where the task is to model the distribution $P(X)$, or the compact space, from which the original data are drawn [251] 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 original data can be generated and reconstructed [129]. Moreover, early UL methods aimed 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 [298].

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 [143]. However, normalization alone is not an efficient means of removing redundancies, reducing data dimensionality, or facilitating representation learning. Therefore, this thesis focuses on the ability of UL-based generative models (UGL) to be used in data analysis to improve the generalizability through dimensionality reduction (Section 4.1), minimize learning and visualization complexity (Section 4.2), and to improve accuracy as well as their potential to produce explainable results to elucidate causality and facilitate decision making (Section 4.3 and Section 4.4) [8, 116].

Recent deep learning (DL) models comprise hundreds of linear and nonlinear layers, which map the input data X to the corresponding target domain Y. The target domain

can be a discrete class label as in the case of DL-based classification (e.g., true/ false classification), or a continuous class label as in the case of DL-based regression (e.g., age, price, salary prediction). Alternatively, the target domain can be a specific object domain as in the DL-based segmentation (e.g., object detection, semantic segmentation, and instance segmentation). Therefore, many different mapping layers are utilized including fully connected layers (dense multiplication), and convolutional layers as in the convolutional neural networks (CNNs) [3, 4, 29]. Accordingly, 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 [128]. Moreover, lacking the explainability when analyzing and learning specific-domain datasets (or models) has led to undermining the potential of recent models to perform tasks in many different real-life application domains.

To mitigate these shortcomings, various approaches have been introduced in the literature and will be highlighted in this thesis including *(i)* unsupervised pretraining by which data and the learning parameters are regulated to be in appropriate forms and ranges concerning redundancy and noise removal [125]; *(ii)* intelligent weights initialization as in using orthogonal weights W, i.e., $W^T W = I$ [140]; *(iii)* minibatch normalization and dropout by deactivating some neurons in accordance with a predefined probability [275]; *(iv)* knowledge transfer and domain adaptation by utilizing representations from related domains [257]; *(v)* and skip connections (residual learning) by adding layers that apply an identity mapping $f(X)$ to copy early layers X [120]; and *(vi)* explainable artificial intelligence (XAI) methods that are introduced to build a level of confidence, or understanding, by explaining and interpreting the learned representations when generalizing the learned model to real-life production (testing) stage, and they are also introduced to improve the model accuracy by improving and incorporating different optimization objectives [4, 188].

Consequently, the generalizability and explainability for unseen data (any data do not consider in model training) are improved according to the above approaches; especially, when building UL models for data visualization and distribution encoding to make a prior knowledge [296, 298], dimensionality reduction to remove the redundancy and address missing data [1, 2], object recognition and detection tasks [227], autonomous driving and semantic segmentation [4], or when proposing new representations learning strategy as in Generative Adversarial Networks GANs based on min-max optimization losses approach [133].

1.1 Relevant Application Domains of UGL Models

Currently, UGL models (including DL models) offer valuable capabilities for exploring and learning datasets from different application domains. These capabilities have led to success in modern 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

Table 1.1: Relevant application domains and finer-grained categorization of the state-of-the-art 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 [291], Multivariate data analysis [321], Dimensionality reduction [5, 94], Pattern recognition [183], Clustering [220], Polynomial feature extraction [258], Signal unmixing and separation [142], Coding and decoding [131]
	Structure	Data mining [51], Hyperspectral image processing and signal separation [64], Explainable artificial intelligence and low-rank approximation [1, 7], Tensor data learning [158], Missing data and collaborative filtering [24], Dimensionality reduction[62], Object recognition [51]
	Probabilistic	Latent data mining [205], NLP and topic modeling [42, 108], Data visualization and topographic mapping [11], Clustering and Dimensionality reduction [343, 350]
MfL	Global	Global data visualization [289], Dimensionality reduction [169], Clustering and interpolation [273, 344], Global joint mapping [74], Graph and multimedia analysis [132], Classification and regression [328], Time series analysis [325], Topological analysis [17]
	Local	Local data visualization [2, 297], Data behavior analysis [244], Dimensionality reduction [298], Clustering and interpolation [221], Local joint mapping [315, 345], Social and biological analysis [327], Object recognition [297], Graph and tensor analysis [179]
NN	Energy	Model pretraining and representation normalization [128], Object reconstruction [6], Object recognition [61, 197], image classification [173], Speech recognition [202], Image and speech interpolation [252], Multimodal learning [276]
	Autoencoder	Anomaly detection [188], Encoding/decoding [248], Image segmentation [326], Collaborative filtering [337], Cybernetics [322], Object recognition [153], Natural language processing [269], Disentangled learning [122], Autonomous driving systems [4, 95]
	Adversarial	Data generation and synthesis [105, 218], Image denoising and deraining [312], Image-to-image translation [356], Multiclass labeling [200], High-resolution image learning [76, 83, 307], Multimodal image synthesis [218], Object recognition and shape learning [323], Natural language processing [182], Graph learning[107], Chest COVID-19 X-ray detection [189]

samples are unstructured, structured, or semi-structured (See section 1).

Table 1.1 highlights the existing research in terms of the application domains and a fine-grained taxonomy among the state-of-the-art UGL models. Specifically, Table 1.1 encompasses three kin families of UGL models including blind source separation (BSS), manifold learning (MfL), and neural network models (NN) due to their potential in big data learning and XAI applications. Additionally, categorizing the models (see the second column in Table 1.1) is implemented based on the model's learning concept and methodology for each subgroup of models. Moreover, the main utilization of such models can be summarized in the application of data analysis and dimensionality reduction, visualization and manifold learning, object detection and segmentation, natural language processing (NLP), machine translation, object recognition/ prediction, XAI-powered systems, and other applications.

1.2 Performed Research and Novelties

Related works that focus on UGL models for data analysis, representation learning (rather than only deep learning models), and XAI have not thoroughly covered the recent aspects and challenges of interest in this field. The main challenge lies in the generalizability, where the learned ML model is generalized for the large-scale production (or testing) stage that is coupled with unseen and diverse data. Another recent challenge is raised due to lacking interpretability when analyzing real-life datasets, or explainability when building DL models without uncovering the black box of learned parameters (weights and biases) to explain how they are learned.

This thesis presents innovative methods for data analysis and XAI applications based on UGL models, which are able to improve generalizability and offer novel explainability techniques for modern ML applications. The realized approaches include BSS, MfL, NNs models based on UGL to facilitate data learning and dimensionality reduction, manifold visualization, computational complexity reduction, and the application of XAI in anomaly detection and autonomous driving system (ADS).

The first goal of this thesis is to review the current state-of-the-art UGL models based on the theoretical, mathematical foundation, application, and development points of view. Indeed, only a few works in the literature have reviewed and discussed these topics, such as [29, 354]. A comprehensive overview of unsupervised representation learning was presented in 2013 [29], which reviewed the probabilistic, MfL, and DL models. By contrast, the overview presented in [354] was brief, and the taxonomy of methods was restricted to traditional and DL models, excluding state-of-the-art models and research challenges. The history of deep learning based on NNs was addressed in 2014 [257] through a review of supervised learning, UL, reinforcement learning, and evolutionary computation without highlighting the current challenges and recent developments among the state-of-the-art UGL models.

Accordingly, a novel overview of UGL models has been performed in this thesis, where the objective 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 works that have targeted specific disciplinary perspectives, e.g., data science excluding representation learning procedures [50], deep-learning-based NNs [227], and GANs [133]. Moreover, this thesis presents the models' foundation, development, research challenges among three kin families of UGL models. The first family of models is BSS and its extensions, which are intended for extracting representative factors from shallow and massive datasets. The second family of models is considered in this thesis consists of MfL-based generative models, which can accommodate data in joint spaces and offer powerful strategies for visualizing low-dimensional embedding of the original data. Finally, this thesis presents a review of generative NNs models from shallow to deep, concerning showing the advantages achieved by integrating these three categories of models for effective representation learning.

The second goal of this thesis is to introduce innovative approaches to improve the generalizability of UGL models concerning the production stage, i.e., testing the model with more data samples than the training ones. Considering that many different approaches were introduced in the literature to improve the generalizability by learning data with additional noise, or by integrating additional penalties to the optimization objective to enforce models to learn new different representations. However, such methods require more computational time, and they did not perform well in reconstruction or generation tasks due to the overfitting when learning data.

Thus, this thesis proposes innovative methods to improve the generalizability of the learned representations by means of data factorization and dimensionality reduction. In this regard, the introduced methods motivate data decomposition by nonnegative data (matrix) factorization (NMF) to decompose data into compact subspaces. Accordingly, the redundancy is removed, and the representations are learned from a rooted version of data to carry out the model training and testing procedures for many different ML tasks, such as reconstruction, generation, and recognition. Moreover, a novel method based on hierarchically reducing the dimensionality of data to minimize the computational complexity when visualizing the manifold of data is presented in this thesis.

The third goal of this thesis is to propose an innovative method to improve the explainability of deep autoencoder models and XAI-based applications. Recently, deep autoencoders were utilized to perform object detection and segmentation tasks concerning modern applications such as anomaly detection and ADS. Also, recent models have introduced visual XAI methods that are induced by attention mapping, which aim to visualize the learned representations and to draw a level of confidence about the model performance. However, recent works were proposed based on offline attention mapping, i.e., after learning the model, neglecting the online explainability in which XAI objectives are integrated into the model optimization objective to give a better visual explainability, and to improve the model generalizability.

Consequently, this thesis offers an innovative XAI method based on the second-order derivative between the latent space of autoencoders with respect to the encoding layers. The proposed method is able to capture the temporal variations of neurons' activations during learning time, which are aggregated to build attention maps to visually explain the learned representations. Besides the explainability, the proposed method showed its

ability to improve the model's performance by incorporating a novel attention loss to the optimization objective. Moreover, the proposed XAI method are used to explain the behavior of the variational autoencoder model (VAE), concerning one-class anomaly detection in which the learned model is able to characterize data samples that are misrepresented from what is normal. Furthermore, the introduced XAI method are utilized to build a multiscale explainability for ADS applications, where the proposed model is able to semantically segment the data samples utilizing a separate attention map at each encoding scale (or layer).

1.3 Formalization, Conventions, and Basic Concepts

Because unsupervised learning has been studied in diverse fields, the terminology and mathematical notations that are reported in Table 1.2 and used in the literature can differ, even when referring to similar (or identical) concepts. To assist the reader, this thesis uses a uniform formalization, stressing the alternative terminologies that can be found in the literature when needed. In addition, although the techniques and learning paradigms reviewed in this thesis 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 $X \subset \mathbb{R}^D$:

$$X = \{x_i \,|\, x_i \in \mathbb{R}^D,\, i = 1, \ldots,\, N\} \tag{1.1}$$

where D is the dimensionality of the original data space, i.e., the number of features in the original data, and N is the number of data samples. Alternatively, the dataset X can be represented as a matrix, where the i^{th} row is the i^{th} element, x_i, and the columns represent the feature points. Unless potentially ambiguous, we will use the same symbol, X, for the matrix described above, where d is the dimensionality of the reduced space and M is the number of mapped features in the embedding space.

UGL algorithms aim to generate a new representation of the data in a lower-dimensional subspace, $Z \subset \mathbb{R}^d$, by applying a suitable mapping, $f : \mathbb{R}^D \to \mathbb{R}^d$ (where $d << D$), to the original data:

$$Z = f(X) = \{z_i = f(x_i) \,|\, i = 1, \ldots,\, M\} \tag{1.2}$$

where Eqn. 1.2 can be seen as an encoding of the data belonging to the original space, as in autoencoder (AE) models (see Section 2.3.2). Moreover, for AE models, the corresponding decoding function can be devised as follows: $g : \mathbb{R}^d \to \mathbb{R}^D$,

$$\tilde{X} = g(Z) = \{\tilde{x}_i = g(z_i) \,|\, i = \ \ldots,\, N\} \tag{1.3}$$

However, when approximations are required (e.g., for noise removal), the composition of the two functions does not produce an identity mapping: $f \circ g \neq I$ and $\tilde{x}_i \neq x_i$ (although they should be similar).

Table 1.2: List of notations and mathematical symbols used in the thesis.

Notation	Description	Notation	Description
$X \in \mathbb{R}^{N \times D}$	Dataset, matrix, or second-order tensor	D	Dimensionality of the original data 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 embedding space
$Z \subset \mathbb{R}^d$	Lower-dimensional space or latent space	r	Data or matrix rank in NMF and SVD
$f : \mathbb{R}^D \to \mathbb{R}^d$	Mapping or encoding function	$C_{\mathrm{X,X}}$	Covariance matrix, X is the original data
$g : \mathbb{R}^d \to \mathbb{R}^D$	Reconstruction or decoding function	$\tilde{X}$	Reconstructed data after learning
d	Embedding or latent spaces dimension	$\|.\|_{\mathrm{F}}$	Frobenius norm used as a reconstruction metric
$E[X]$	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 (ones in the diagonal)
$J, H(X)$	Negentropy, differential entropy	A	Reconstruction coefficients
$\circ$	Composition of linear mappings	$\otimes$	Outer or tensor product
Λ	Eigenvalues in a diagonal matrix	$\mathcal{G}$	Weighted or finite graph
ϵ	Radius or variational parameter	W, B	Learning weights and biases
$\mathcal{Z}$	Core tensor in Tucker decomposition	λ	Eigenvalue or scaling factor
$\|.\|_*$	Nuclear norm is obtained from SVD	Tr	The trace of data matrix
$P(V), P(X)$	Prior distribution of data	h	Hidden or latent variable
$P(v\|h)$	Conditional probability of v given h	$P(v, h)$	Joint probability of v and h
∂	Partial derivative, or rate of change	θ	Learning parameters, i.e., weights and biases

In the statistical context, the features of x_i are called observable, or explanatory variables, while the features of z_i are called latent, or hidden variables or factors. The data are seen as samples that are representative of a larger population and hence can be used to estimate various properties of the whole population. Common statistical indices used to describe a population distribution are the (sample) mean, variance, covariance, and kurtosis [163]. Several methods described in the thesis are based on properties of the distribution that are estimated from the statistics of the sampled data (e.g., the mean of a random variable, $E[x_i]$, can be replaced by the mean of the data, $\mu = 1/N \sum_{i=1}^{N} x_i$), and in the following, the statistics of the sample and the population will be referred to interchangeably. The mean μ describes the central tendency of the population, while the variance σ_x describes the spread of the distribution around the mean. When a sample of data is multivariate (which is typical), the auto-covariance matrix, $\mathrm{C_{X,X}}$, is introduced :

$$\mathrm{C_{X,X}} = \frac{\Sigma_{i=1}^{N}(X - \mu_X)(X - \mu_X)^{\mathrm{T}}}{N - 1} \tag{1.4}$$

where $\mathrm{C_{X,X}}$ describes the linear relationships between every pairs of scalar components. For a zero-mean variable, $\mathrm{C_{X,X}}$ coincides with the autocorrelation matrix ($E[X^{\intercal}X]$). A similar concept is the Pearson correlation coefficient, ρ, of random variables or a matrix, which is the covariance matrix of the standardized X (i.e., the variable obtained by shifting X by its mean and scaling it by its standard deviation) and expresses the linear correlations in terms of a dimensionless index:

$$\rho = \frac{\mathrm{C_{X,X}}}{\sigma_X \sigma_X} \tag{1.5}$$

The calculation of standardized variables is a common preprocessing step in data analysis and is called whitening. In addition to simplifying algorithm procedures, whitening of the data enables a fairer evaluation of the contribution of each component to the phenomenon described by the data. Other indices of interest are the skewness (which measures the asymmetry of a probability distribution and is defined as a third-order statistic), and the kurtosis (which gives information about the shape of the probability distribution and is defined as a fourth-order statistic). In particular, kurtosis is used to measure the similarity of the data distribution to the Gaussian distribution with the same variance. The kurtosis is defined as $\mathrm{kurt}(X) = E[(X - E[X])^4] - 3E[(X - E[X])^2]^2$, which, for zero-mean data, becomes $\mathrm{kurt}(X) = E[X^4] - 3E[X^2]^2$; in the case of whitened data, it simplifies to $E[X^4] - 3$.

Two distributions p (represents original data x_i distribution) and q (represents the distribution of approximated data $\tilde{x_i}$ after learning) can also be compared using the Kullback-Leibler (KL) divergence, which is also known as a relative entropy (since it is a relative, thus no measurement unit is given for the score) [163, 204]:

$$\mathrm{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(\tilde{x})} \, \mathrm{d}x \tag{1.6}$$

which is always nonnegative and tends to zero if and only if p and q are almost equal in the distributions. The KL divergence can also be generalized to enable matrix comparisons:

$$\mathrm{KL}(X\|\tilde{X}) = \sum_{i,j}[X_{i,j}\log\frac{X_{i,j}}{\tilde{X}_{i,j}} - X_{i,j} + \tilde{X}_{i,j}] \tag{1.7}$$

Practically, the KL divergence measures if two distributions p and q are equal (or identical) almost everywhere in the distributions, thus no divergence comparison is required in case when q is equal to 0, or when q contains no information, i.e, $\log\frac{p}{0} \to \infty$. Moreover, the base of the log-term in Eqn. (1.6 and 1.7) can be expressed either in the base-nat (base-e or base-10), or base-bit (Shannons or base-2); whereas, when base-nat is considered, then the output score must be equal to 0 if both p and q identical everywhere in the distributions, otherwise, the score will be positive taking into account that the KL divergence is an asymmetrical divergence, i.e., $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$ [37]. However, considering the base-2 in the log term reflects the number of bits required to encode data or information that is represented in p distribution in a compact or learned form that is drawn in q distribution, taking into account minimum information and loss after encoding. Furthermore, the KL divergence is extended in the Jensen-Shannon divergence JS [181] to enforce the resulted score to be a normalized-symmetric, i.e., $\mathrm{JS}(p||q) = \mathrm{JS}(q||p)$, where the JS is obtained as follows:

$$\mathrm{JS} = \frac{1}{2}(\mathrm{KL}(p||M)) + \frac{1}{2}(\mathrm{KL}(q||M)) \tag{1.8}$$

where $M = \frac{1}{2}(p+q)$.

Another class of divergence metrics that has been used in the literature to quantify the difference between two probabilistic distributions is termed as β-divergences, which is a group of metrics that comprises the Itakura-Saito divergence (IS) when $\beta = 0$, the KL divergence when $\beta = 1$, and the Frobenius norm when $\beta = 2$ [63]. The distributions p and q are referred to x and $\tilde{x}$, respectively, in the context of matrices or images comparison:

$$\mathrm{d}_\beta(p||q) = \begin{cases} \frac{p}{q} - \log\frac{p}{q} - 1, & \beta = 0 \\ p\log\frac{p}{q} - p + q, & \beta = 1 \\ \frac{1}{\beta}(\beta-1)(q^\beta + (\beta-1)q^\beta - \beta p q^\beta), & \text{otherwise} \end{cases} \tag{1.9}$$

The Frobenius norm [222] that is utilized to measure the difference between the two matrices (or data samples), $\|X - \tilde{X}\|_\mathrm{F}$, is also commonly used for quantifying the similarity between two images in the context of image recognition; when the score difference between two images is zero, then it reflects that two images are identical and can be calculated for each image (or matrix) as follows:

$$\|.\|_\mathrm{F} = \sqrt{\sum_{i,j}|x_{ij}|^2} \tag{1.10}$$

Furthermore, a comparison between data distributions can be carried out in a low-dimensional space by utilizing rank, r, truncation methods. The rank is the dimensionality of the data, represented in a matrix, in terms of the number of columns or rows and can be utilized to reduce the data dimensions by eliminating columns that are linearly dependent while retaining only columns that are linearly separable [1].

Finally, besides the abovementioned metrics, the structural similarity index (SSIM) has been proposed in [310] and is mainly used for image quality assessment. Moreover, it measures the quality of an image after encoding or carrying out learning procedure, i.e., it indicates the quality of reconstruction in the AEs models (See section 2.3.2), and it measures the quality of image generation from scratch as in GANs models (see Section 2.3.3.1). Unlike the most widely utilized image quality metrics which reflect the absolute error, such as mean square error:

$$\text{MSE} = \frac{1}{N}\Sigma_{i=1)}^{N}(X - \tilde{X})^2 \tag{1.11}$$

or pixel-to-noise ratio:

$$\text{PSNR} = 10\log(\text{pr}) - 10\log(\text{MSE}) \tag{1.12}$$

where pr is the pixel range in an image (pr = 255 for a gray-scale image), the SSIM estimates its score based on the dependencies between local pixels preserving symmetry property. Furthermore, it measures the similarity by windowing images (e.g. window x from original image and $\tilde{x}$ from the reconstructed one) to simplify the computation:

$$\text{SSIM}(x, \tilde{x}) = \frac{(2\mu_x\mu_{\tilde{x}} + c_1)(2\text{C}_{x,\tilde{x}} + c_2)}{(\mu_x^2 + \mu_{\tilde{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\tilde{x}}^2 + c_1)} \tag{1.13}$$

where μ_x, $\mu_{\tilde{x}}$, σ_x, and $\sigma_{\tilde{x}}$ are the statistical components that are derived from the original and reconstructed window, respectively, and $\text{C}_{x,\tilde{x}}$ is the window covariance matrix; $c1 = (k_1 L)^2$, $c2 = (k_2 L)^2$ are stabilizers components and L is the range of pixel vales and obtained as $2^{pr} - 1$, where both k_1, k_2 are equal to 0.01 and 0.03, respectively.

1.4 Thesis Structure and Organization

The thesis is structured as follows:

- *Chapter* 2 presents a novel overview of the current UGL models that are utilized in data analysis, representation learning, and XAI applications. In particular, it reviews three kin families of models including BSS, MfL, and NNs. Chapter 2 is organized as follows:
  - *Section* 2.1 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 analysis.

  - *Section* 2.2 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* 2.3 presents a review of NNs-based models, from shallow to deep, including energy, autoencoders, and generative adversarial learning-based models, which are utilized to learn the representations among data.

- *Chapter* 3 describes the current open issues, research challenges, and future research direction of the reviewed UGL method at Chapter 2 for data analysis and XAI applications. Chapter 3 is organized as follows:

  - *Section* 3.1 presents the role of BSS in XAI data analysis, where it gives the development of BSS models and future directions for many different data types in the form of tabular, image, video, and signal data.
  - *Section* 3.2 introduces the current role of MfL in preparing data and visualization, where it highlights the developments and future directions for local and global MfL methods. Moreover, the same section shows the role of integrating BSS and MfL methods for real-time big data exploration.
  - *Section* 3.3 reports the considerations, challenges, and future directions of the reliable NNs-based models. Also, it comprises the recent trends of DL models from the architectural and optimization point of view. The same section highlights the role of integrating BSS, MfL, and NNs models to facilitate learning procedures and generalize standard deep UGL models.
  - *Section* 3.4 highlights the current XAI attention mapping methods that are utilized to build visual XAI attentions of the learned deep NN models. Moreover, the same section introduces the research issues and future considerations to build reliable XAI attention.

- *Chapter* 4 describes our proposed investigations and experimental analysis to cope with the generalizability and explainability of recent UGL models. In particular, this chapter introduces NMF rank approximation for data factorization to reduce the dimensionality, and to learn from limited data aiming to improve the generalizability of autoencoders. Moreover, the same chapter contains a novel XAI method based on the online gradient attention mapping to improve the explainability of the deep VAE. Chapter 4 is organized as follows:

  - *Section* 4.1 introduces a novel method to obtain the nonnegative rank, r, that is imposed as a prior for β-NMF data factorization, which is used in explainable dimensionality reduction.
  - *Section* 4.2 highlights an innovative method to improve the generalizability (large-scale testing) of autoencoders learning, where the proposed method depends on the NMF and rank approximation introduced in Section 4.2.

  - *Section* 4.3 presents a novel method to explain the VAE models based on the second-order derivation between the latent space (bottleneck) concerning the encoder layers. The same section contains the experimental analysis of the proposed method in the field of one-class anomaly detection.
  - *Section* 4.4 introduces multiscale variational attention mapping method for explainable image segmentation, where the introduced method expands the role of gradient attention mapping method that is introduced in Section 4.4 for ADS applications.

- *Chapter* 5 summarizes the work and the obtained results, the originality of the contribution, and then presents a series of possible future works.
- *Appendix* A contains the list of publications in which some of the ideas and significant results present in this thesis were published.

2

State of The Art of Unsupervised Generative Learning (UGL) Models

2.1 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 methods and techniques that are valuable for splitting up mixed sources, signals, and data points [357]. Moreover, BSS models are utilized in many different fields including speech signal analysis and recognition, image processing, electroencephalogram (EEG), electrocardiogram (ECG), and musical signal separation as it is shown in Fig. 2.1. 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 [3]. BSS methods are also closely associated with unsupervised representation learning and XAI, through which the hidden structures of the data, signal sources, or learned representations can be discovered, separated, isolated, and interpreted.

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 [25]. In the following, we provide a brief summary of the three main types of BSS methods: those that are derived from statistical analysis in section 2.1.1, those related to structural or matrix decomposition in Section 2.1.2, and those that are derived from probabilistic modeling (PBMs) in Section 2.1.3.

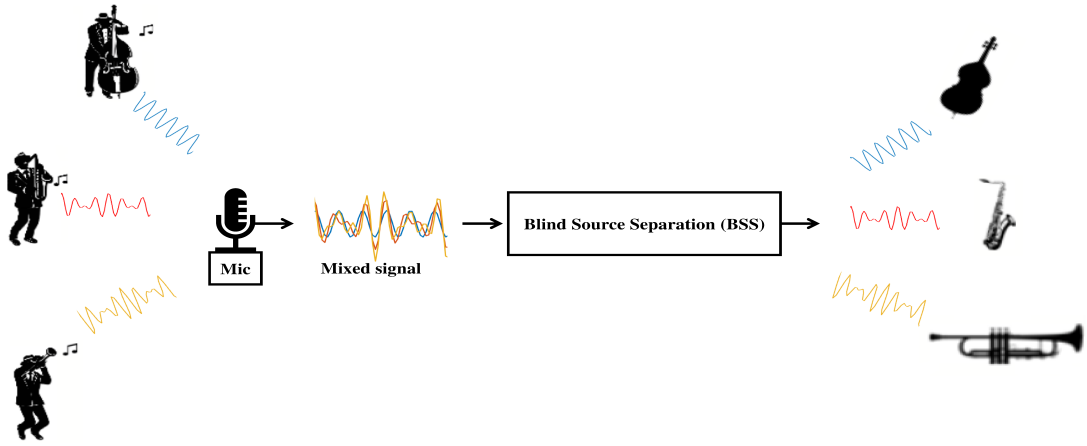
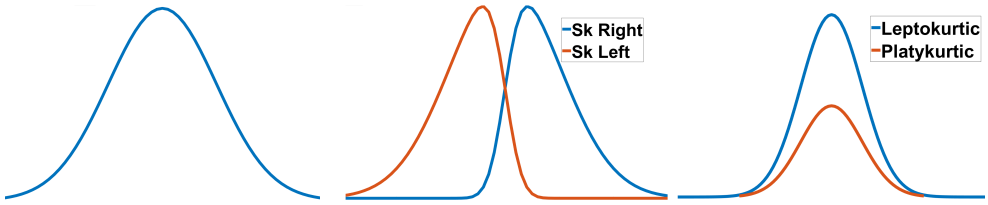


Figure 2.1: A typical BSS model for musical signals separation.



(a) Gaussian Distribution (b) Skewness of Gaussian (c) Kurtosis of Gaussian

Figure 2.2: The graphical representation of the statistical moments, where the first and second moments (μ, σ, respectively) are used to draw the Gaussian (or normal, $X \sim \mathcal{N}(\mu, \sigma^2)$) distribution that is depicted in Fig. 2.2(a); the third moment is the skewness of the data distribution and is shown in Fig. 2.2(b); the fourth moment is the kurtosis of the data distribution is exhibited in Fig. 2.2(c).

2.1.1 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 (or sample) 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 (mean, variance, etc.) or sets of data points X_i. For a given dataset (or matrix) $X \subset \mathbb{R}^D$, $X = \{x_i \,|\, x_i \in \mathbb{R}^D, i = 1, \ldots, N\}$, the first moment (first statistical parameter) 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, $\text{Sk} = \Sigma_1^N \frac{x_i^3}{N}$, which is described in terms of the normalized variance; and the fourth moment is the kurtosis, $\text{Kr} = \Sigma_1^N \frac{x_i^4}{N}$, which is regarded as a high-order statistic and can be obtained from the normalized version of the third moment [163]. Fig. 2.2 depicts the graphical representation of the statistical moments.

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) [220] in Section 2.1.1.1 and independent component analysis (ICA) [67] in Section 2.1.1.2. These techniques are interrelated with the forthcoming sections, as in the PCA connection to Sections 2.2.1 and 2.3.2.

2.1.1.1 Principal Component Analysis

PCA, proposed by Pearson [220], 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 Section 1.3, 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:

$$Z = XU \tag{2.1}$$

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 estimation of the projection can be treated as an optimization problem to find the principal components that achieve the highest variance among data points:

$$U = \arg\max_{\hat{U}} \hat{U}^\intercal C_X \hat{U} \tag{2.2}$$

where $\mathrm{C_X}$ is the covariance matrix of X and is given as:

$$C_X = \frac{1}{N-1} X^\intercal X \tag{2.3}$$

where N is the total number of the data points in a feature vector (or sample), and it is utilized as a normalization factor. The optimization is constrained such that the variances of the transformed dataset along the U basis directions are sorted in decreasing order *w.r.t.* the basis element order, i.e., the first column of U matrix corresponds to the highest eigenvalue of C_X, the second column of U corresponds to the second largest eigenvalues of C_X, and so on. The features in Z are hence uncorrelated and capture most of the variability of X. Thus, the usage of the covariance matrix $\mathrm{C_X}$ allows PCA to be applied for statistical analysis tasks. Fig. 2.3 shows the

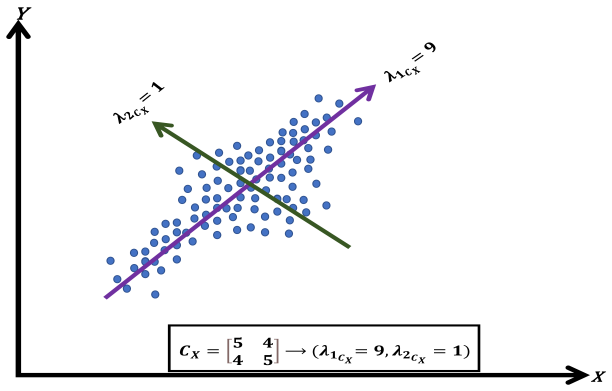


Figure 2.3: The graphical representation of two principal components that are taken from the decomposition of the covariance matrix C_X, where the purple line arrow is the direction of the first principal component which is corresponded to the largest eigenvalue ($\lambda = 9$), and the green line arrow reflects the second principal component direction which is associated with the second largest eigenvalue ($\lambda = 1$).

graphical representation of two principal components. Moreover, 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. Finally, the inverse of the mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$ generates, or reconstructs, the original data X from its mapped space Z as follows:

$$\tilde{X} = g(Z) = ZU^\intercal = XUU^\intercal \tag{2.4}$$

PCA can also be derived on the basis of geometrical considerations [147]. 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, and the projection of the data x_i onto u_i is given as follows;

$$\text{pro}_{\text{xu}} = \frac{x_i.u_i}{||x_i||^2} \tag{2.5}$$

where the line is optimized such that the average projection residual of the data onto u (i.e., the mean squared distance of the original data from their projections onto u) is minimized by using mean square displacement (MSD) as follows:

$$\text{MSD} = \frac{1}{N}\Sigma_{i=1}|x_i - \text{pro}_{xu}|^2 \tag{2.6}$$

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 [183], kernel PCA [258], and probabilistic relational PCA [178, 291] 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 2.1.2.1). 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 [258]. In the probabilistic PCA [291], the optimization enforces each basis u to follow a certain Gaussian to ensure that all principal components are independent. However, in probabilistic relational PCA [178], it is instead assumed that the data are not independent and identically distributed (*i.i.d.*). In this regards, two subsets of the data follow (*i.i.d.*) when the values of data points in a subset do not affect the values of the other subset (independent), and changing the values setting doesn't change μ and σ for each subset (identical).

2.1.1.2 Independent Component Analysis

ICA [67], 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 (Section 2.1.1.1, 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; such sources can be formed from multi-microphone with different wavelengths in audio signal systems, or from the electromagnetic spectrum (range of frequencies) that build hyperspectral images. Fig. 2.4 shows the graphical representation of ICA for speech separation, where a similar graphical representation is utilized for RGB and hyperspectral images unmixing.

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 [143]. In the formalization defined in Section 1.3, 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 as follows:

$$X = ZA + R \tag{2.7}$$

where A is called the mixing matrix (weights) and describes how each source contributes to each feature, and R is the factorization, decomposition, or unmixing residual. Al-

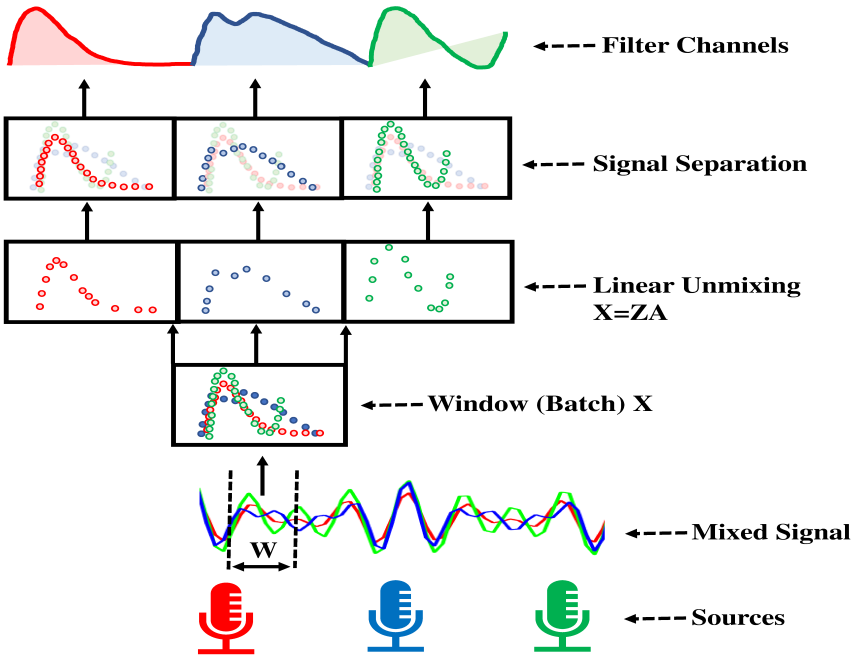


Figure 2.4: A typical ICA block diagram for speech processing, where W represents the window size in a period of time. Moreover, a similar block diagram and learning procedure are applied for image channel separation.

ternatively, ICA can be formulated as the search for a transformation matrix U such that the columns of Z, where $Z = XU$, are independent [142].

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 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 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 as follows:

$$P_{z_1,\ldots,z_d}(z_1, \ldots, z_d) = \prod_i P_{z_i}(z_i) \tag{2.8}$$

where, 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), and there are many different considerations are taken to achieve the independence and optimize ICA factorization.

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 (Section 2.1.1) 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 [142]; as an alternative, a more robust measure of non-Gaussianity defined in the field of information theory is the negentropy (J) [163]. Among all statistical distributions with the same μ and σ, the Gaussian distribution has the largest entropy:

$$H(X) = -\int P(x) \log P(x) \, \mathrm{d}x \tag{2.9}$$

The negentropy is defined as the difference between the entropies of the Gaussian distribution and Z as follows:

$$J(Z) = H(Z_{\text{Gauss}}) - H(Z) \tag{2.10}$$

where, $J(Z)$ 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:

$$\mathrm{I}(z_1;\ \ldots;\ z_d) = \sum_{i=1}^{d} H(z_i) - H(Z) \tag{2.11}$$

where for two variables z_1 and z_2,the mutual information is computed as:

$$I(z_1; z_2) = H(z_1)H(z_1|z_2) \tag{2.12}$$

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 (Eqn. (1.6)) 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. Eventually, the original data can be generated from the separated components, i.e., $Z = XE$ and $E = A^{-1}$, by mixing the components as:

$$\tilde{X} = g(Z) = ZE^{\intercal}$$

such that $EE^T = I$.

Among the prevalent ICA-based models, low-complexity coding and decoding (lococode) [131] is a nonlinear ICA that is built based on the sparse regularization of an autoencoder (see Section 2.3.2.1). Finally, we refer the reader to [142] for more details and other ICA extensions.

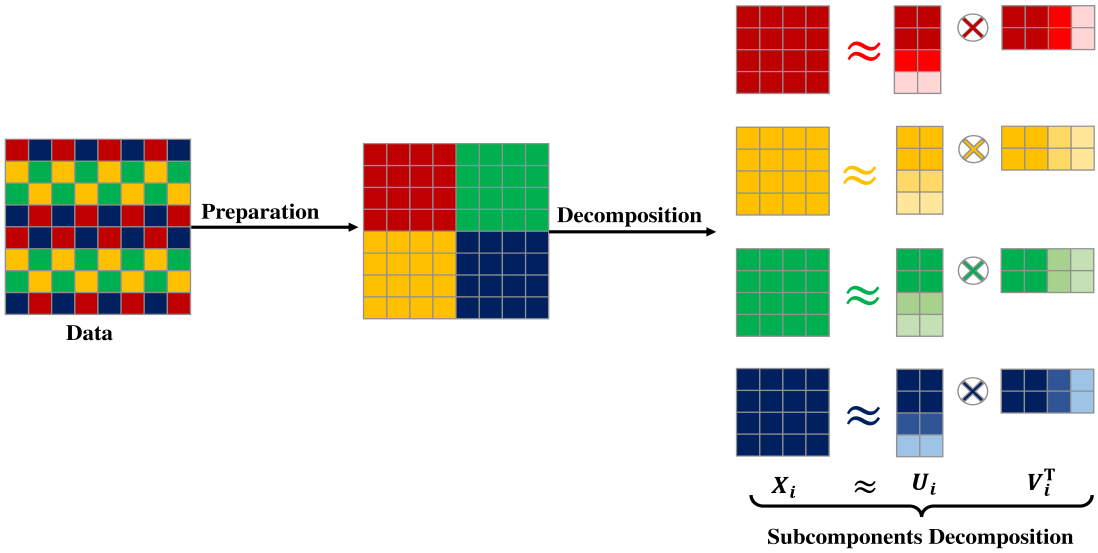


Figure 2.5: The graphical representation of dataset decomposition, where the colored cells reflect numeric values if the data comes in a tabular format; they reflect pixel's intensities if the data is represented in an image format, and they reflect nodes values if the data is introduced in the context of graph learning. Also, the preparation step is an optional stage and can be exploited for data standardization and pre-clustering.

2.1.2 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 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 flexibly. Fig. 2.5 depicts the graphical representation of the structural decomposition of data, where the dataset can be introduced in the tabular, image pixels, or graph format.

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:

$$X \approx U_1\delta_1V_1^\intercal + U_2\delta_2V_2^\intercal + ... + U_n\delta_nV_n^\intercal \tag{2.13}$$

where U and V are called the left and right decomposition matrices, respectively, and the δ_n are diagonal scalars. In SVD (Section 2.1.2.1), 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 2.1.2.2). In the following, we provide a brief overview of

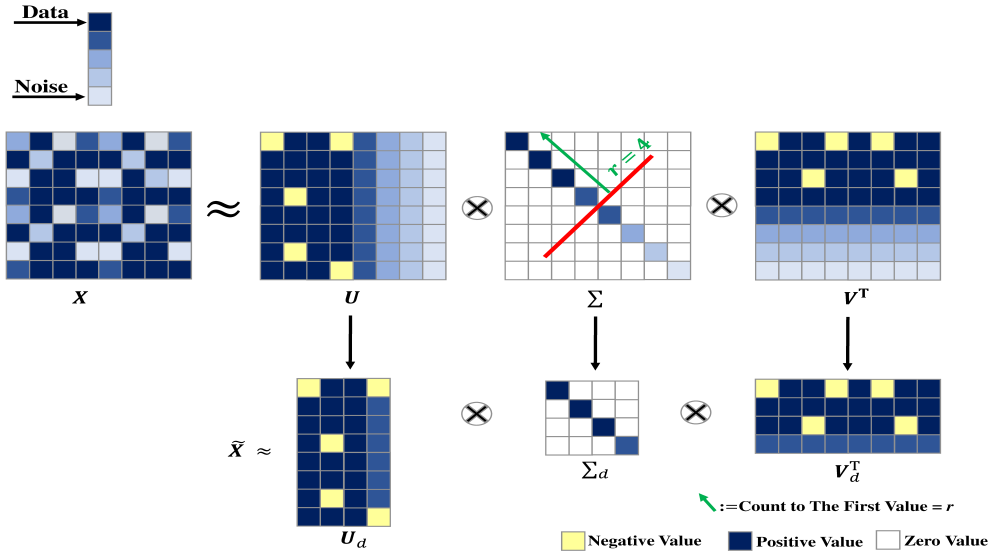


Figure 2.6: The graphical representation of truncated SVD, where data X is formed in colored cells that reflect specific intensities of image pixels, or they give numeric measurement values when data comes in a tabular form. Moreover, X is the original data, $\tilde{X}$ is the reconstructed data, and the rank r reflects the dimensionality in the mapped space Z and can be obtained by counting the highest singular values of the matrix Σ.

the major structure-based methods, namely, SVD [155] in Section 2.1.2.1, NMF [215] in Section 2.1.2.2, and tensor decomposition (TD) [158] in Section 2.1.2.3, all of which share similar mathematical concepts.

2.1.2.1 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., $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$ [71]. 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, or rows in a compact space) [155]. 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:

$$X = U \Sigma V^{\intercal} \tag{2.14}$$

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. Fig. 2.6 depicts the graphical representation of SVD decomposition, where the original data is represented by X, and the reconstructed data after decomposition is provided by $\tilde{X}$.

Additionally, Σ is a diagonal matrix that contains the singular values $\{\delta_i \,|\, i = 1 \ldots, D\}$ 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. (2.14) can be approximated when the nonzero singular values are neglected based on a threshold of small value, i.e., 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 is obtained from the rank, r; the rank gives the dimension of the column space of a matrix U, and it gives the dimension of the row space of a matrix V. Furthermore, the dimensionality reduction 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 (see Fig. 2.6). Also, U_d represents the distribution of the data points around the directions V_d when they are multiplied by the singular values as $U_d\Sigma_d$. Moreover, the original data X can be generated (or reconstructed) after the decomposition as follows:

$$\tilde{X} = U_d \Sigma_d V_d^{\intercal} \tag{2.15}$$

considering an optimization goal:

$$\min \|X - \tilde{X}\|_{\mathrm{Er}}. \tag{2.16}$$

where Er is the reconstruction error metric that can be measure by MSE (Eqn. (1.11), Frobenius norm (Eqn. (1.10)), or KL divergence (Eqn. (1.6)).

Various extensions have been proposed in the literature to generalize SVD, e.g., generalized SVD (GSVD) [216], in which the joint decomposition of two different matrices is performed:

$$X1, X2 = U_1 U_2 \Sigma_1 \Sigma_2 V^{\intercal} \tag{2.17}$$

where the first matrix is given by $X_1 = U_1\Sigma_1 V^{\intercal}$, and the second matrix is obtained from $X_2 = U_2\Sigma_2 V^{\intercal}$. For certain values of the second matrix, GSVD reduces to SVD; however, it can be used to formalize prior knowledge. This approach is extended in restricted SVD (RSVD) [72], which decomposes a given matrix with special constraints encoded in two other matrices.

2.1.2.2 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 2.1.1.1) is based on the covariance $\mathrm{C_X}$ 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 [215], 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

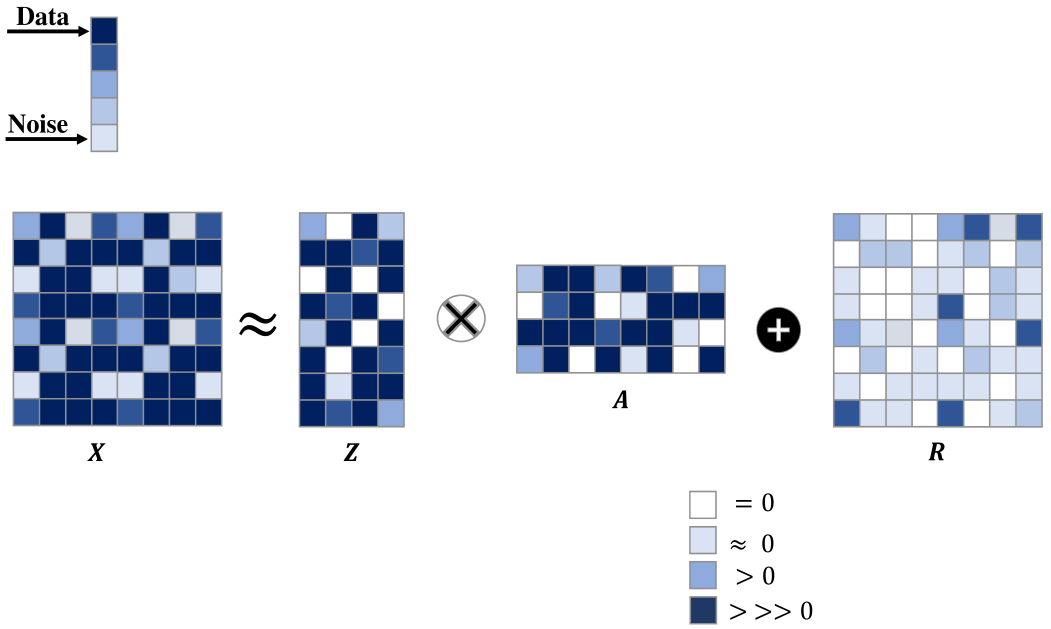


Figure 2.7: The graphical representation of NMF, where data X is formed in colored cells that reflect specific intensities of image pixels, or they give numeric measurement values when data comes in a tabular form. Moreover, all values of the factorized subspaces are positive (> 0), where Z is the bases subspace that hides the features of data, A represents the reconstruction coefficients (weights), and R is the factorization residual.

ICA, no assumption on the statistical dependency of the components is made, where PCA implies the Gaussianity of data samples and components, but ICA requires the non-Gaussianity among samples and factors. 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:

$$X = ZA + R \tag{2.18}$$

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. Fig. 2.7 exhibits the graphical representation of NMF factorization; unlike SVD (see Fig. 2.6), all subspaces Z, A, and R elements values are positive.

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. Because all the components are nonnegative and the combination is additive, thus NMF offers what is called part-based representations, where the data reconstruction ($g : \mathbb{R}^d \to \mathbb{R}^D$) is realized by assembling different parts:

$$\tilde{X} = Z^{N \times d} A^{d \times D} \tag{2.19}$$

NMF can be optimized to minimize the residual R using various approaches. For example, in [215], 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 [171], the objective function was related to the likelihood of reconstructing X by adding Poisson noise to ZA:

$$R = \sum_{i,j} \|X_{(i,j)} \log(ZA)_{(i,j)} - (ZA)_{(i,j)}\| \tag{2.20}$$

Another way to minimize R was proposed in [172]; in this case, the modification lays in introducing a composite objective (loss) 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:

$$R_{\text{Euc}} = \|X - ZA\|_{\text{F}}^2 \tag{2.21}$$

and the second loss is the KL divergence, given by:

$$\text{KL}(X||ZA) = [\sum_{(i,j)} X_{(i,j)} \log \frac{X_{(i,j)}}{ZA(i,j)}] - [X_{(i,j)} + ZA_{(i,j)}] \tag{2.22}$$

Finally, R is optimized by minimizing $R_{\text{Euc}} + \text{KL}(X||ZA)$ 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 [171], or the orthogonality of the basis or an altered factorization (e.g., kernel mapping) can be addressed [308].

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:

$$X = XZA + R \tag{2.23}$$

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) [351]. Among the generations of CF, Robust Flexible Auto-weighted Local-coordinate Concept Factorization (RFA-LCF) was proposed in [350, 352] 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 [343], in which the data are factorized in the subspace into which they are projected hierarchically, i.e, the projected subspace is factorized at layer L_1, then Z_1 subspace is factorized at layer L_2, and so on for the deepest (consecutive) layers L_{i+1} for the corresponding Z_i. DSCF-Net also improves the locality of representations using a matrix of coefficients as adaptive reconstruction weights.

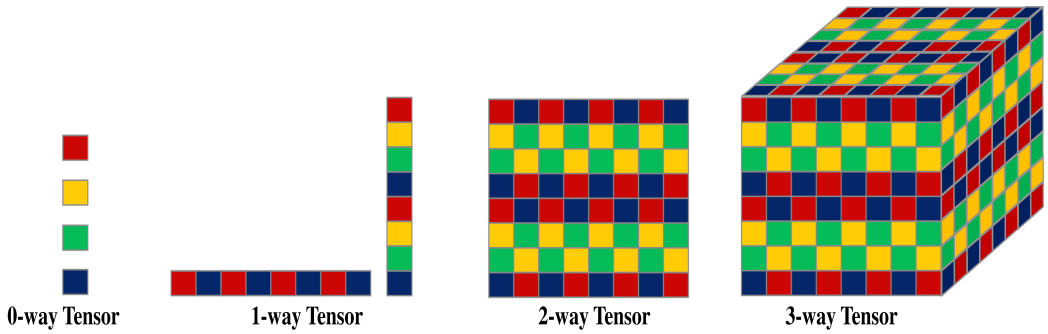


Figure 2.8: The graphical representation of tensor orders, where data X is formed in colored cells that reflect specific intensities of image pixels, or they give numeric measurement values when data comes in a tabular form. Moreover, four different orders of the tensor are considered: *(i)* 0-way tensor that is considered any scalar value; *(ii)* 1-way tensor which is a row or a column vector; *(iii)* 2-way tensor that is represented by a 2-d matrix; and *(iv)* 3-way tensor that stores data in a 3-dimensional cube as in the RGB images.

2.1.2.3 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 [158]. Fig. 2.8 shows the graphical representation of tensor orders as a function of dimensionality (number of ways).

The aforementioned BSS methods, including both statistics- and structure-based techniques (Sections 2.1.1 and 2.1.2), 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 based on 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 (a product that yields a matrix as a result of two vectors multiplication) of these vectors:

$$\mathcal{X} = u \otimes v \otimes w \tag{2.24}$$

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 2.1.2.1 can be described by reframing Eqn. (2.14) 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) [116] and Tucker decomposition [293] are

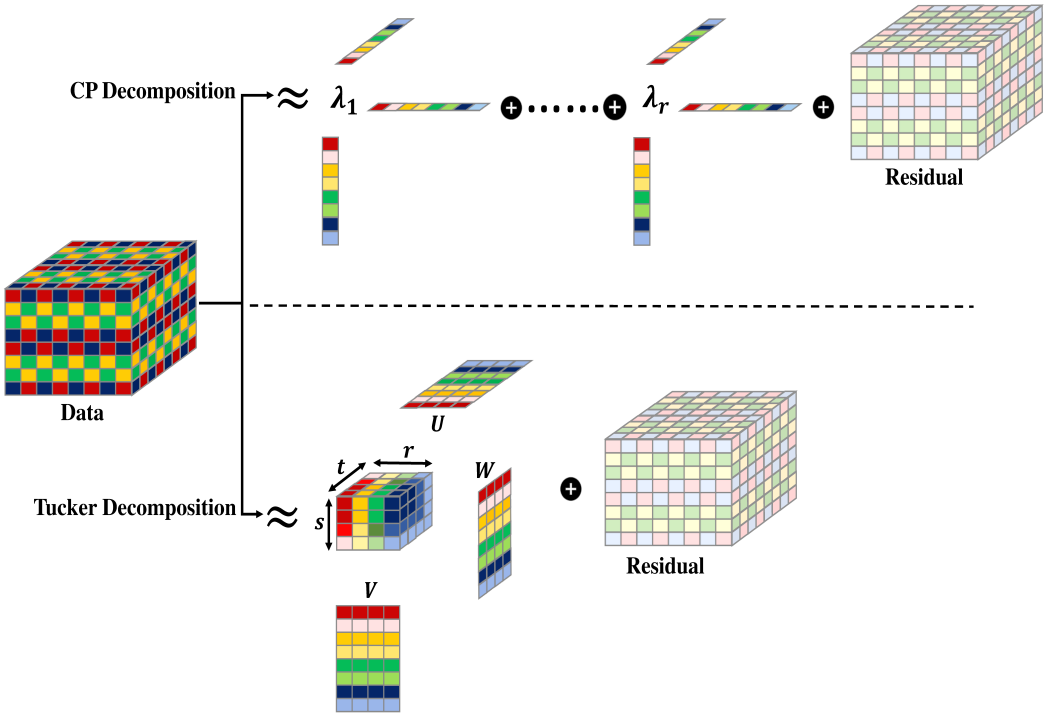


Figure 2.9: The graphical comparison between CPD and Tucker decomposition of a 3-way tensor data, where r is the decomposition rank, λ is the scaling factor (eigenvalue), and "Residual" is the factorization residual that must be minimized.

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 rank, r, tensor products of d 1-way tensors as shown in Fig. 2.9, 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:

$$\mathcal{X} = \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i = [\![U, V, W]\!] \tag{2.25}$$

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 $[\![\lambda; U, V, W]\!]$:

$$\tilde{\mathcal{X}} = \sum_{i=1}^{r} \lambda_r U_i \otimes V_i \otimes W_i \tag{2.26}$$

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 as depicted in Fig. 2.9.

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 $\mathcal{Z} \in \mathbb{R}^{r \times s \times t}$ (ranks of $\mathcal{Z}$), the Tucker decomposition generates approximation such that:

$$\tilde{\mathcal{X}} = \sum_{i=1}^{r} \sum_{j=1}^{s} \sum_{k=1}^{t} \mathcal{Z}_{i,j,k} U_i \otimes V_j \otimes W_k = [\![\mathcal{Z}; U, V, W]\!] \tag{2.27}$$

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 (compact version) of the data $\mathcal{X}$.

2.1.3 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 $P_{model}(X)$ [159]. 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 [146]. 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:

$$P_{model} = \mathbb{E}_z \, P(X|\theta_Z) P(\theta_Z) \tag{2.28}$$

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. Fig. 2.10 shows the graphical representation of a PBM that estimates the index of the merry of life (as an example), where six feature points, or parameters, are connected to each other (based on relative correlations) to form the least structure of the PBM. Furthermore, assigning a probabilistic value to each variable leads to parametrize the whole model, and the final probability result among all configurations of the parameters reflects the index of the merry of life, which is given according to any liner function, e.g., the exponential of linear regression:

$$P(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{\exp(-\theta_Z^{\intercal} X)}{\langle Z \rangle} \tag{2.29}$$

where $\langle Z \rangle$ is the partition factor that normalizes the final probabilistic result among all configurations, or learning iterations. Moreover, the model can be expanded as a function of the number of the model's parameters; however, the complexity of averaging, $\langle Z \rangle$, among all model configurations will be increased.

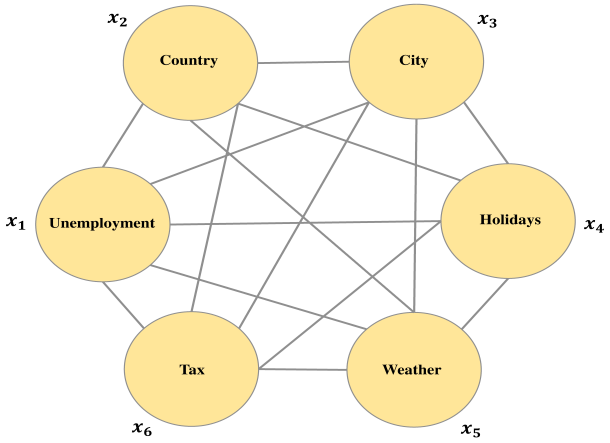


Figure 2.10: The graphical representation of a PBM, where 6 random variables are considered to form the model's structure that estimates the index of the merry of life (as an example). Moreover, the connection is drawn based on the correlation between the parameters, and the final probabilistic result is given according to Eqn. (2.29).

In the following, we recapitulate two categories of the PBMs: those are derived to latent variables (LVs) modeling in Section 2.1.3.1, and those related to mixture modeling in Section 2.1.3.2; however, some models comprise both latent and mixture modeling perspectives.

2.1.3.1 Latent Variable Models

LVs are unobservable random variables, which introduce a latent generation process among observed data [206]. 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 [205]. 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. (2.28) to be more tractable, i.e., they factorize the joint probability into small probabilities:

$$P_{\text{model}} = \mathbb{E}_z \, P(x_1|\theta_Z)P(x_2|\theta_Z)...P(x_N|\theta_Z)P(\theta_Z) \tag{2.30}$$

Practically, LVs are integrated into the PBMs to form the connections between the model's parameters, where each LV acts as a focal point in the model inference; such integration can affect each parameter's value, and LV can be measured according to a predefined scale α_H. Considering Fig. 2.10 that mimics the estimation of the index of the merry of life, the connections among all parameters can be reduced by introducing an LV that can affect all parameters (e.g., Happiness, Z_H). Also, it assigns the probability for every pair of variables, i.e., given the happiness what is the value of x_i, $P(x_i|Z_H)$, where Fig. 2.11 shows the integration of the latent variable, H_Z, into

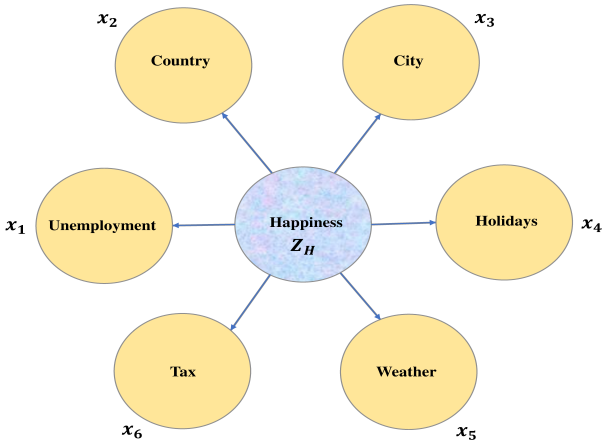


Figure 2.11: The graphical representation of a LVM, where an LV "Happiness" is integrated into the PBM that is depicted in Fig. 2.10. Also, the connection between Z_H and each x_i is directed, i.e., the probability is gained given H_Z, and not vice versa.

the PBM that is depicted in Fig. 2.10. Moreover, Eqn. (2.29) can be factorized by neglecting $\langle Z \rangle$ as follows:

$$P_{\text{model}} = \sum_{Z_H=1}^{\alpha_H} P(x_1|Z_H)...P(x_6|Z_H)P(Z_H) \tag{2.31}$$

where α_H is the maximum limit of a predefined happiness scale, and $P(Z_H)$ is the prior probability; such a prior can be obtained according to the available data.

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 [42], hierarchical Latent Dirichlet Allocation [108] as well as their relevant extensions, is given.

2.1.3.1.1 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 high popularity in topic modeling, i.e., finding the distribution over words [42]. 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(θ), is considered a conjugate to the multinomial likelihood, or distribution over count, when the posterior, $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$, belongs to a similar distribution of the prior:

$$P(X|\theta) = \frac{n!}{x_1, .., x_k}\theta_1^{x_1}, .., \theta_K^K \tag{2.32}$$

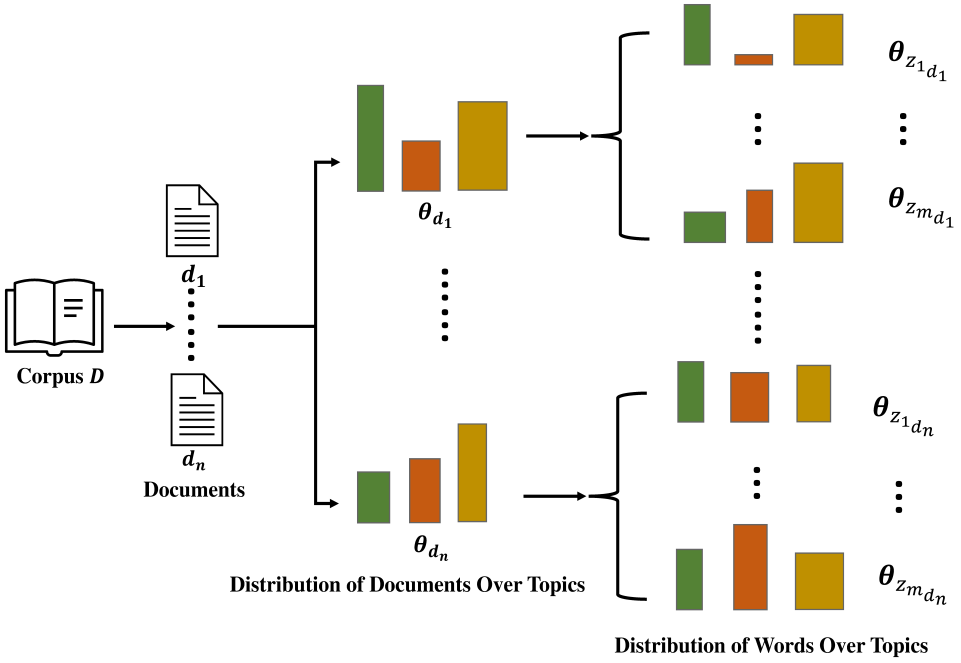


Figure 2.12: The graphical representation of the latent Dirichlet allocation model, where d_n refers to the number of documents d in a corpus, θ_{d_n} is the distribution of a document d_n over topics θ_d, and $\theta_{z_{m_{d_n}}}$ is the distribution of word z_m over topics in a document d_n.

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. Fig. 2.12 illustrates the graphical representation of the LDA, which assumes that each document distribution through all topics participates in a common Dirichlet prior:

$$P(\theta_d) = \text{Dir}(\theta_d|\alpha) = \frac{1}{B(\alpha)} \Pi_{d=1}^{D} \theta_d^{\alpha_d - 1} \tag{2.33}$$

where α is a parameter for each document (a prior distribution of topics), B is a distribution of words represented by a $K \times V$ matrix, and K is the dimensionality of the Dirichlet distribution. Similarly, the distribution of the topics over words Z share a joint prior:

$$P(\theta_z) = \text{Dir}(\theta_z|\eta) = \frac{1}{B(\eta)} \Pi_{z=1}^{Z} \theta_z^{\eta_z - 1} \tag{2.34}$$

where η is a parameter for each topic (a prior distribution of words). Thus given both α and η the Dirichlet distribution for document d and Z topics is drawn as $\text{Dir}(\theta_d|\alpha)$, and for a topic z over N words as $\text{Dir}(\theta_z|\eta)$.

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. Moreover, it can 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 Gibbs sampling [86]. 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 [263] 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.

2.1.3.1.2 Hierarchical Latent Dirichlet Allocation Mining modern complex datasets (as in social media data) that are composed of hierarchical relations among data entries, becomes more challenging nowadays [184]. Such datasets come in unformalized or unstructured forms and can be explored by probabilistic generative models. The LDA [42] (see Section 2.1.3.1.1) 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 [108] to model the hierarchical correlation among topics of documents in a corpus D, by using a tree structure to break the independence among all topics. Moreover, the hLDA is built based on the nested Chinese restaurant process (nCRP), which offers a distribution of an infinite number of customers n to sit in an infinite number of tables i in a restaurant according to the following distribution:

$$\begin{aligned} P(\text{Occupied Table i} \mid \text{Previous Customers}) &= \frac{n_i}{\gamma + n - 1} \\ P(\text{Next Occupied Table} \mid \text{Previous Customers}) &= \frac{\gamma}{\gamma + n - 1} \end{aligned} \tag{2.35}$$

where n_i reflects the total number of customers that are sitting at a table i, and γ is a parameter that regulates how often a customer selects a specific table versus sitting in the other tables. [40]. 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, see Fig. 2.13. For documents in a corpus, D, a vocabulary, $V = \{1, 2, .., N\}$, of N words, and Z topics; the hLDA selects a root topic, z_1, which sits in the top-level, l_1, then it passes to the next levels $l \in \{2, ..., L\}$ utilizing the distribution drawn from Eqn. (2.35), and it chooses the topic, z_1, from z_{l-1} as what is shown in Fig. 2.13. Moreover, it draws the L-dimensional topic proportions to a document d from $\text{Dir}(\theta_d|\alpha)$, and for N words it draws a topic z as $\text{Dir}(\theta_z|\eta)$.

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 [212] to allow a document to obtain a multi-path through a tree structure, and a semi-supervised hierarchical topic model (SSHLDA) was proposed in [223] based on a supervised path labeling method to search a topic automatically in the data space.

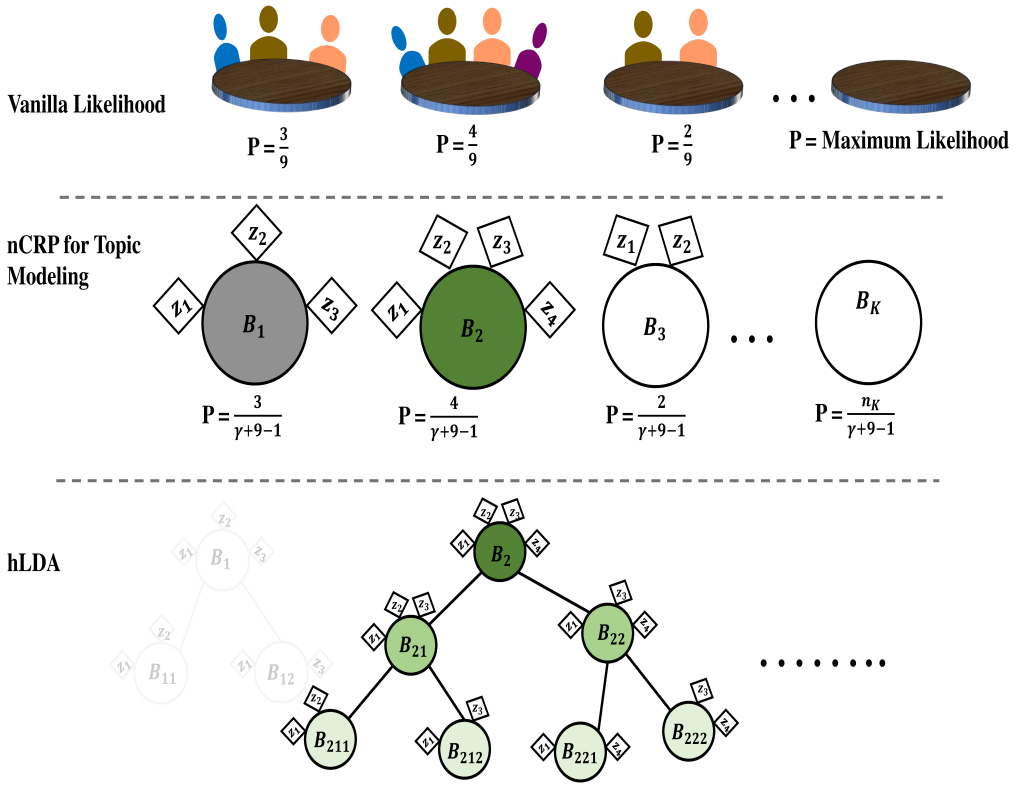


Figure 2.13: The graphical representation of the hierarchical latent Dirichlet allocation model. The first row is attributed to the "Vanilla Likelihood" shows how the distribution of customers sitting in a restaurant; in the middle row the "nCRP for Topic Modeling", the same analogy from the first row is utilized but with a different distribution that is drawn from Eqn. (2.35); and in the bottom row, the hLDA is depicted.

Another kin model to the hLDA is termed as Pachinko Allocation Model (PAM) and was proposed in [177], 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 $\text{Dir}(\alpha_i)$, *(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 [41, 184].

2.1.3.2 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 [221]. 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 [190]. For a given dataset $X \subset \mathbb{R}^D$, $X = \{x_i \,|\, x_i \in \mathbb{R}^D, i = 1, \ldots, N\}$, the mixture expresses data utilizing the joint distribution between data and latent variables in the latent space, $Z \subset \mathbb{R}^d$, $Z = \{z_j \,|\, z_j \in \mathbb{R}^d, j = 1, \ldots, M\}$:

$$P(X, Z) = \prod_{i=1}^{N} \prod_{j=1}^{M} [P_j(x_i)^{\pi_j z_{ij}}] \tag{2.36}$$

Eqn. (2.36) 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) [243], probablisic self-organizing map (PSOM) [57], generative topographic mapping (GTM) [38], and their extensions.

2.1.3.2.1 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 [243]. Accordingly, multi-class datasets can be analyzed by a multi-Gaussian (a sequence or mixture) model to meet the number of clusters; by using a GMM that is able to model the densities among data points according to a predefined weighted sum from 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), $X \sim \mathcal{N}(\theta_j)$, which are paramertrized by $\theta_j = \{\pi_j, \mu_j, \sigma_j^2\}$, hidden or latent variables $Z = z_i \in \mathbb{R}^j$, $z_{ij} \in \{0, 1\}$, and $\Sigma_j z_{ij} = 1$ [96]. It also estimates the mixture components j in Eqn. (2.36) in terms of multivariate Gaussian with components centroids μ_j (which reflects the locations of the Gaussians), and a positive-definite (all eigenvalues Λ are positive) covariance matrix $\mathrm{C_X}$ (which reflects the shapes of the Gaussians). Fig. 2.14 depicts an example of a GMM model with an order of three, i.e., three different Gaussian distributions with their corresponding μ_j and σ_j to model three different clusters or populations of balls in a dataset.

The number of components, j, 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, i.e., $P(x_i) = P(\theta_j)P(x_i|\theta_j)$, and the model inference is given as:

$$P(x_i|\theta_j) = \frac{1}{(2\pi_j\sigma_j^2)} \exp\{-\frac{||xi - \mu_j||}{2\sigma_j^2}\} \tag{2.37}$$

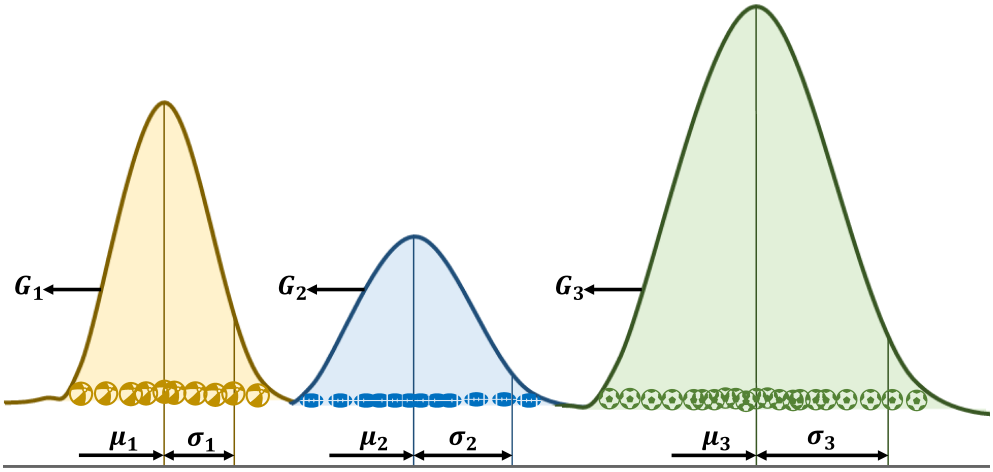


Figure 2.14: The graphical representation of a Gaussian mixture model of three clusters, where each Gaussian distribution (G_1, G_2, or G_3) refers to a certain cluster, c_j, or publication of balls in a dataset. Usually, in the real-life datasets, the distributions are interfering due to the interclass intervention among data classes, and the optimization process must optimize the model parameters, θ_j, which separate each distribution (or each mixture component) considering the interclass and intraclass variations of data samples.

where the model posterior is generated using Bayes' rule as:

$$P(\theta_j|x_i) = \frac{P(x_i|\theta_j)P(\theta_j)}{P(x_i)} \tag{2.38}$$

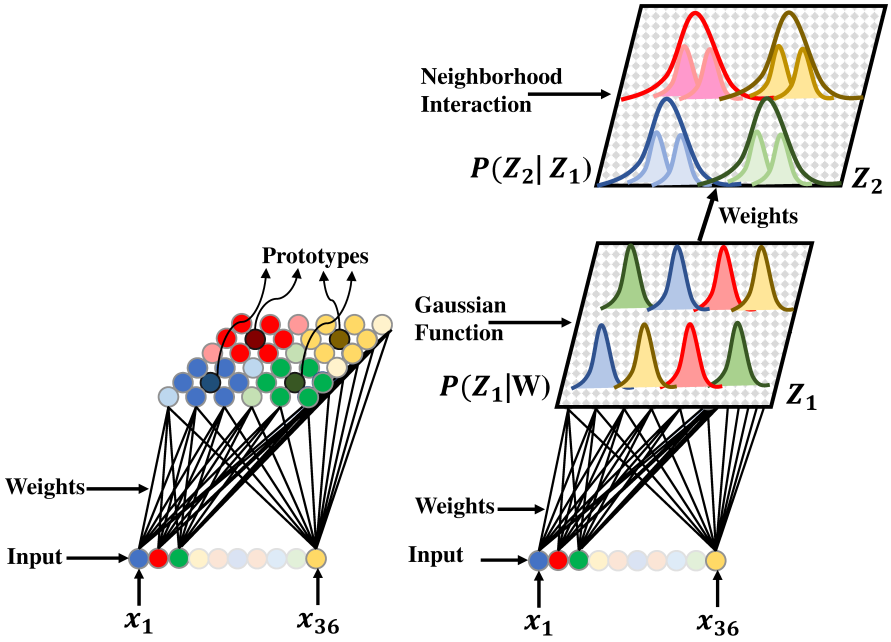
Assuming that all samples are independent given θ_j, thus the likelihood is factorized and the latent variables are marginalized by a discrete sum among all values of Z:

$$P(X) = \prod_i \Sigma_{z_j \in Z} \prod_j [P_j(x_i)^{\pi_j z_{ij}}] \tag{2.39}$$

The GMM is optimized to find an optimal set of parameters which expresses the density of each set of data points, i.e, $\theta_j = \{\pi_j, \mu_j, \sigma_j^2\}$, by adapting the parameters' values to maximize the likelihood $P(x_i|\theta_j)$ [238]. EM is an optimization algorithm that is utilized to optimize GMM parameters, by assigning the data points to clusters c (or z) given a certain probability [243]. 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:

$$\mathcal{L} = \frac{1}{N} \Sigma_{n=1}^{N} \log(\Sigma_j \pi_j P_j(x_n)) \tag{2.40}$$

More details and model extensions about the GMM and EM algorithm are introduced in [207, 243].



(a) SOM (b) PbSOM

Figure 2.15: The graphical representation of the SOM (Fig. 2.15(a)) vs PbSOM (Fig. 2.15(b)). In the SOM, the prototypes are imposed and then are reorganized to fit the data; however, in the PbSOM, the prototypes are considered the center of the Gaussians and the model can be formed by multilayer to fit data.

2.1.3.2.2 Probabilistic Self Organizing Map Self-Organizing Map (SOM) is an ML model that is trained in an unsupervised fashion to cluster and visualize data. Also, it constitutes feature detectors that are spatially formed and represented in a gradually increasing array [156]. In Fig. 2.15(a), 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:

$$||x_i - c_r|| = \min||x_i - c_j|| \tag{2.41}$$

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:

$$w_i^{\text{new}} = (1 - \alpha.n_b)w_i^{\text{old}} + \alpha.\text{n}_\text{b}.x_i \tag{2.42}$$

where α is the learning rate. The neighbors for the i^{th} neuron are identified such that $\text{nb} = \exp(\frac{||c_i - c_r||^2}{\omega^2})$, where ω is the variance of n_b.

The stranded SOM neglects the loss function throughout the model optimization [157]; conversely, the loss function is considered in optimizing the probabilistic self-organizing map (PbSOM) that has been introduced in view of Gaussian-full mixture components M [57] as it is shown in Fig. 2.15(b). 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 by utilizing the following objective optimization:

$$\mathbf{O}(P, \theta) = \Sigma_{i=1}^{N} \Sigma_{x_i \in c_j} \Sigma_{j=1}^{M} (\mathrm{n_b} \log P(x_i|\theta_j)) \tag{2.43}$$

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 probability r_{c_j}, i.e., $c_j = \{\text{argmax } r_{c_j} = P(x_i|\theta_j)\}$. 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[157].

2.1.3.2.3 Generative Topographic Mapping GTM was proposed in [38] 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, which 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 $f : \mathbb{R}^D \to \mathbb{R}^d$; constraining the distribution to follow a Gaussian centered among the mixture components which are drawn as:

$$P(x_i|z_j, w_i, \beta) = (\frac{\beta}{2\pi})^{-D/2} \exp\{-\frac{\beta}{2}||x_i - \sigma(z_j; w_i)||^2\} \tag{2.44}$$

where x_i is a data point in the original data space, z_j is a mapping point in the corresponding mapping space, w_i is the learning weight for the data point x_i, β^{-1} is the variance, and σ is the sigmoidal activation function. The prior of Z is approximated using the summation of M equally-weighted delta function on a grid as:

$$P(z) = \frac{1}{M} \Sigma_{j=1}^{M} \delta(z - z_j) \tag{2.45}$$

where the M delta function reflects the spacing in the grid elements of the mapping space, and it is employed also to measure the likelihood after fitting the model:

$$P(X|W, \beta) = \frac{1}{M} \Sigma_{j=1}^{M} P(x_i|z_j, W, \beta) \tag{2.46}$$

Fig. 2.16 depicts the logical structure of the GTM model, where different Gaussian distributions are centered in the original data space aiming to optimize the mapping space, Z, probabilistically. Moreover, the optimization algorithm optimizes the model's

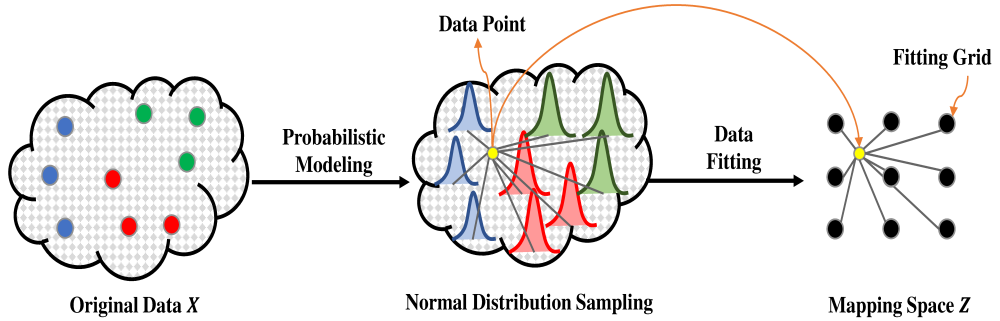


Figure 2.16: The graphical representation of the generative topographic mapping model, where the original data is considered with 9 colored-RGB points, and the yellow points are the data point in the original data space and its representation in the mapping space. Also, the model samples from the structure of the data (probabilistically), then fits the data through maximizing the likelihood in mapping space Z.

likelihood, $P(X|W,\beta)$, such that the distances between the original data points and their neighbors are preserved in the mapping space.

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 $\beta = \frac{1}{\sigma^2}$ [39]. In the E-step, the responsibility or posterior is estimated $r_{cj} = P(\theta_j|x_i, w_i, \beta)$, 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 concerning β and w_i:

$$\mathcal{L} = \Sigma_{i=1}^{N}\Sigma_{j=1}^{M} r_{cj}(\log P(x_i|z_j, w_i, \beta)) \tag{2.47}$$

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 [39] to overcome the computational complexity issues, by using small batches and update the model gradually. We refer to [11] for more details and extensions of the GTM models.

2.1.4 Summary

A brief, non-exhaustive but major review of prevalent BSS methods for data exploration and representation learning, facilitating the dimensionality reduction of massive datasets and the discovery of the hidden structures characterizing them, has been introduced in Section 2.1. SBMs methods (Section 2.1.1) including PCA and ICA, are derived from a statistical point of view; in particular, PCA depends on the second moment, while ICA is based on the fourth moment [142].

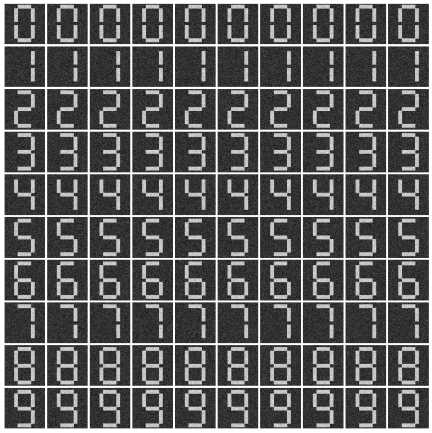
PCA (Section 2.1.1.1) is derived from a Gaussian concept, yielding a global projection and indicating the order of the principal components. The Gaussian (or normal) distribution $N(\mu\,, \sigma^2)$ is a commonly used continuous density distribution, which gives

the probability of an event (or data point) falling between two real number boundaries on a curve that approaches zero on every side of the mean value (μ). The area underneath the Gaussian distribution curve is always equal to 1 due to data standardization (normalization). The curve also has the axis of symmetry in X-axis, i.e., $X = \mu$, $\mu = 0$, and $\sigma = 1$. Conversely, ICA (Section 2.1.1.2) assumes non-Gaussianity among the variables, produces local projections, and cannot be used to order the components. Fig. 2.17(a) shows a synthesized seven-segment dataset, where it comprises 100 images with a dimensionality of each image equal to 128×128. Both PCA and ICA, Fig. 2.17(b), and Fig. 2.17(c), respectively, have been employed for the dataset factorization and only 9 principal/ independent components have been considered. As it is noticed from both subfigures, PCA gives the average or brightness of the global features of the images for each class, and it also seeks minimal correlations among the data points. By contrast, ICA identifies the local edges and features in each image and looks for maximal independence among data components.

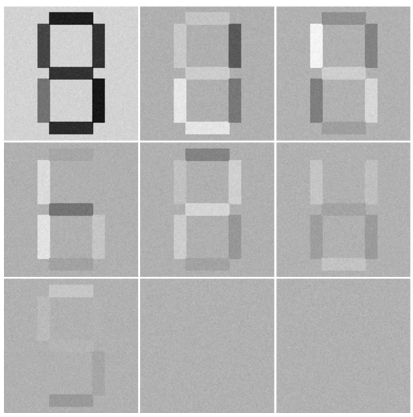
Structure-based methods (Section 2.1.2) including SVD and NMF, are commonly used in data decomposition and tend to produce manageable components of the original messy data. SVD (Section 2.1.2.1) offers the ability to approximate the data's rank and to measure the spectrum of the eigenvalues; moreover, it yields approximate factors that are unique and consist of both positive and negative values. In practice, when using SVD, it is important to ensure that the data matrices are generative, i.e., the matrices are invertible; otherwise, approximation with SVD will fail. In contrast, NMF (Section 2.1.2.2) restricts the decomposition to only positive values and outputs unique factors; this method is very useful for interpreting natural phenomena due to the nonnegativity restriction [2]. Utilizing the same dataset that is depicted in Fig. 2.17(a), the SVD and NMF show different abilities in the dataset decomposition. Additionally, SVD in Fig. 2.17(d) retains a similar decomposition result to PCA that is appeared in Fig. 2.17(b), where NMF in Fig. 2.17(e) shows a better result than the others and it preserves part-based decomposition and sparsity among the decomposed classes.

TD (Section 2.1.2.3) is a powerful mathematical tool for factorizing high-dimensional datasets, which are represented as high-order tensors, into significant components; thus, it is utilized in building efficient big data exploration algorithms, along with defining error metrics that are enhanced by properly decomposing such tensors. Furthermore, the recent trends in reliable big data analysis and exploration lie in integrating the above-mentioned BSS methods with TD, especially in combination with NMF [64], due to its robustness in producing smooth, sparse, and nonnegative components, thus leading to interpretable results for the given model.

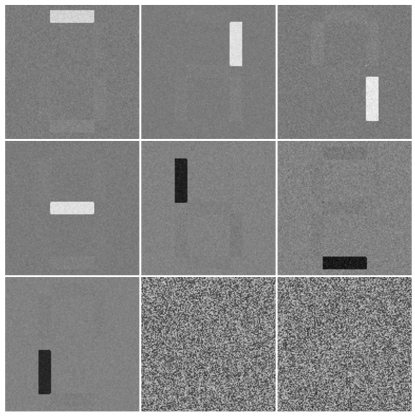
Furthermore, there is the possibility of obtaining approximations of the results of one of these methods using the others; for instance, by subtracting the means from the dataset X and then dividing by its variances, followed by multiplication (or orthogonalization) in the form $XX^{\intercal}$, we can produce the data covariance matrix $\mathrm{C_{X,X}}$ that is used for PCA projection, see Eqn. 1.4 in Section 1.3. Similarly, if we apply SVD to X, then the left singular vectors contained in the U matrix will be typical of the result that is obtained by projecting covariance matrix $\mathrm{C_{X,X}}$ by PCA, and SVD already sorts



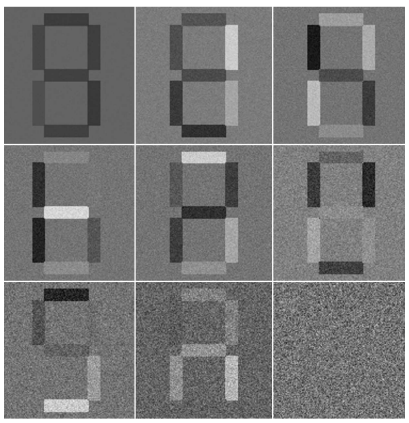
(a) Seven-Segment Dataset



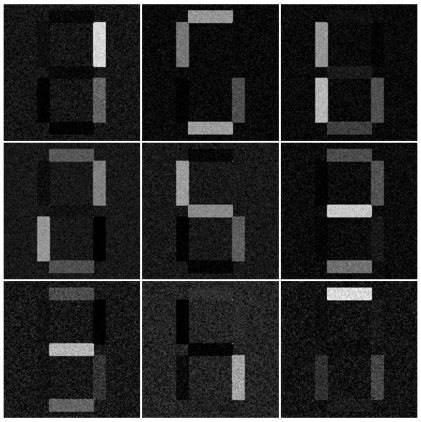
(b) PCA



(c) ICA



(d) SVD



(e) NMF

Figure 2.17: Decomposition (or factorization) examples. The set of images in panel (a) is decomposed using PCA, ICA, SVD, and NMF. The images are composed of $D = 16,384$ pixels. For each method, $d = 9$ (or $r = 9$) components are identified as dimensions of the latent space and used to reconstruct the original dataset. In panels (b) to (e), the bases identified via PCA, ICA, SVD, and NMF are presented. Because the components in panels (b) to (d) can have negative values, the gray levels have been remapped: middle gray represents zero, while bright and dark shades represent positive and negative values, respectively.

the principal components in decreasing order. Consequently, it is possible to use SVD to obtain the direction of the principal components as in PCA (which is known as dual PCA), thereby also finding their spectrum, i.e., the singular values [183].

PBMs (Section 2.1.3) including LVMs and MBMs are derived from probabilistic estimations of latent (or hidden) variables and distributional parameters among data samples. LVMs (Section 2.1.3.1) including LDA, hLDA, and their relevant extensions are summarized. Such models are built based on latent variables (LVs) modeling to extract different characteristics which are buried inside data and cannot be obtained directly by a quantitative scale, e.g., document interpreting and understanding, level of intelligence among students in a class, or happiness, etc. The main utilization of LDA and hLDA are in natural language processing (NLP) applications; specifically, they are used for document understanding, detecting topics in a corpus, and text analysis. Moreover, they use a continuous multinomial probabilistic distribution termed as " Dirichlet distribution" to model the distribution among words, thus clustering the topics of documents in a corpus and classifying words for document understanding are carried out according to the Dirichlet distribution.

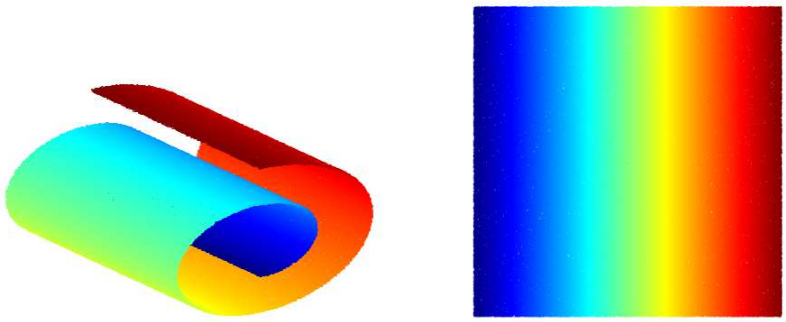
Addressing modern datasets that contain hierarchical relations between data samples or observations (as in social media data) was the intuition behind introducing hLDA; it is also able to model the hierarchal relations among the whole dataset classes or samples (the corpus can be viewed as a dataset), which is not obtainable from ordinal LDA. The hLDA views the dataset as a tree to capture the correlations between topics of documents; also, relations between nodes and links are modeled based on a probabilistic process termed as "nested Chinese restaurant process" (nCRP) to model documents hierarchy, thus generating documents from the drawn tree paths. The hLDA approach has been followed in the Pachinko allocation model (PAM), which utilizes a directed cyclic graph (DAG) to model topic hierarchy. Moreover, it also models a topic distribution considering the distribution among all other topics, unlike the hLDA which considers only the distribution of a topic among words of a document.

To handle semantic confusion issue that is raised when a topic is distributed among different documents in a corpus, matrix factorization, or decomposition, methods which belong to the structure-based methods (Section 2.1.2) have been exploited besides the PBMs, which has been let to capture smooth representations from a corpus structure to model high-level topics. NMF (Section 2.1.2.2) factorizes a data into two subspaces, namely, Z, A, (or U, V) and has been utilized for topic modeling, by adding a sparseness constraint L_1, $(\Sigma_{(i=1)}^{n} U_i)$, to the optimization objective function as a regularization term, where each column of U subspace reflects a specific topic, and meaningful topics have been recovered by adding such sparseness restriction [136]. Moreover, sparseness constraint has been combined with soft orthogonality ($U^T U = I$) constraint among the decomposed subspaces in [56], where the authors introduced the SONMFSR model for a dynamic topic tracking. Also, spelling top-level topics, also known as super-topic, into subtopic (HOSC) has been investigated in [278] by using NMF with orthogonality and sparseness constraints to model hierarchy relations among topics. HOSC model uses global and local independence between topics, it also employs special constraints

termed as consistency constraints (sub-topics that are inherited from the same super-topic should share the same semantic) to ensure that topics are clustered and split correctly [186]. Recently, the distributed latent Dirichlet allocation (dLDA), i.e., the inference is distributed, and NMF have been integrated into the same model in [214], to produce a distributed and a hybrid topic learning model. Moreover, dLDA is able to model and extract topics from a corpus utilizing auxiliary labels, followed by NMF and hierarchical clustering to rank features; consequently, such a combination is assisted to regulate the topics into compact and less fragmented themes.

Other PBMs are built based on a sequence or mixture modeling (MBMs) among high-dimensional and multiclass data (Section 2.1.3.2)). Such models are used for dimensionality reduction, clustering, and visualization; they also generate a mapping from data space to an embedding space $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$, or vice-versa, based on a probabilistic estimation among each mixture component, e.g., for Gaussian data, the basic estimation is to find the distribution parameters $\theta = \{\mu, \sigma\}$. Additionally, MBMs have been utilized for nonlinear BSS modeling, unlike ICA which able to split signals based on linear transformations and high-order statistical moments. In this regard, the self-organizing map (Section 2.1.3.2.2) has been used to construct a mapping from data to embedded space, respecting the output components to be independent for separating nonlinear signals which were modulated as mixed Gaussian components [217].

The advantage of the MBMs is that they can be used to perform many different tasks, e.g., dimensionality reduction, and feature ranking. Similar to what we have described above about obtaining the PCA components from SVD decomposition, Generative topographic mapping (GTM) (Section 2.1.3.2.3) can be used for visualization and feature ranking simultaneously, i.e., besides obtaining a visualization map, it produces principal components; by exploiting the weights matrix W that is used to draw the distributions of the mixture components $P(X|W, \beta)$. The W matrix can be used to initiate the principal components by decomposing the data covariance matrix $\mathrm{C_{X,X}}$, then considering the eigenvectors that are corresponding to the largest d eigenvalues; followed by minimizing the error between the projected components, or points in the GTM mapping space and the projection components from PCA. Moreover, there are different metrics that can be utilized to measure how the error between the mapping components of GTM and PCAs is minimized, by considering the sum of square Frobenius norm, mean square errors, or divergence metrics such as KL and β divergence [2]. Practically, to use the GTM for visualization and feature ranking, it is important to initialize the value of β^{-1} to be greater than either the $d + 1$ eigenvalues of the decomposed covariance matrix, or greater than the square of the half of the grid spacing used in the GTM to project the PCA on it [39].



(a) Swiss-Rolle Manifold. (b) Unfolded The Manifold.

Figure 2.18: The manifold of the Swiss roll synthesized data (Fig. 2.18(a)) vs. unfolded the manifold of the data (Fig. 2.18(b)).

2.2 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 $P(Y|X)$, where X represents the data points and Y is any prior condition [296]. 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 [44].

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. 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 ($d << D$). Fig. 2.18(a) shows an example of a synthesized manifold of 300000 data points that take a shape of a Swiss roll-cake in $\mathbb{R}^D$ space, $D = 3$, where the corresponding unfolding of the manifold is considered in $\mathbb{R}^d$ space, $d = 2$, and it is depicted in Fig. 2.18(b).

The general framework of the methods described in this section consists of stating a similarity or dissimilarity relations over the data points in both domains (X and Z) to identify the neighborhood connections, and then 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).

2.2.1 Global Models

Global models are a set of MfL methods known as full spectral techniques. These methods are based on global neighborhood relations and 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 [298].

In the following, a brief review of several common and well-known global MfL methods, namely, multidimensional scaling (MDS) [162] in Section 2.2.1.1, Isomap [289] in Section 2.2.1.2, and nonlocal manifold tangent learning (NLMTL) [31] in Section 2.2.1.3 as well as their relevant extensions, is given.

2.2.1.1 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 2.1.1.1), which attempts to preserve the correlations or variance of nearby data samples [162]. 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 [318], 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:

$$D_X(i, j) = \|x_i - x_j\|^2 \tag{2.48}$$

where the mapping to Z is chosen so as to minimize the error:

$$\mathcal{L} = \sum_{i,j} (D_X(i, j) - D_Z(i, j))^2 \tag{2.49}$$

where D_Z is defined similarly to D_X but in terms of the corresponding elements of the feature in Z domain. Fig. 2.19 illustrates the graphical representation of MDS learning, where we restrict our example to 12 data souses in $\mathbb{R}^D$, i.e., $D = 5$. Moreover, each source can be seen as a vector of pixel values in an image, or any other measurements which are mapped to $\mathbb{R}^d$, i.e., $d = 2$.

The generative process of the mapping domain Z lies in decomposing the dissimilarity matrix to find a suitable set of eigenvectors that constitute the domain's basis, and it can be expressed as:

$$Z = U\Lambda^{1/2} \tag{2.50}$$

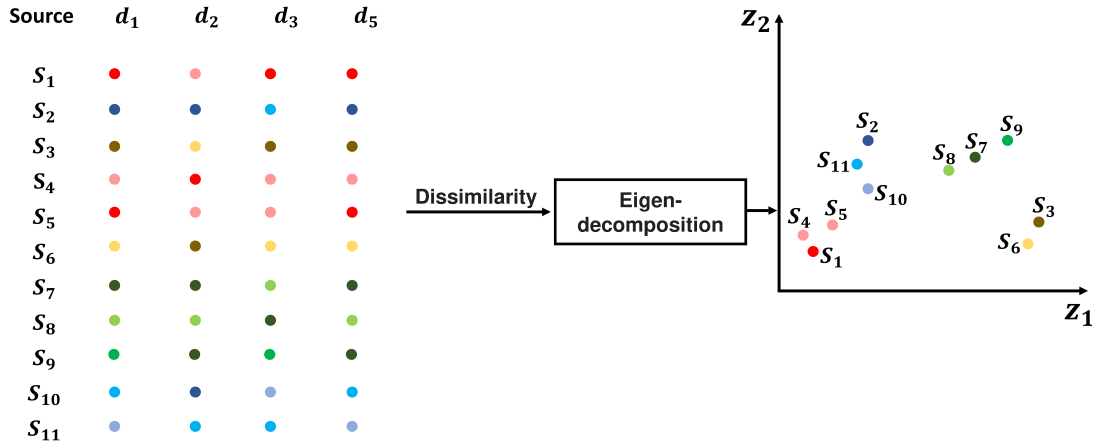


Figure 2.19: The graphical representation of MDS learning, where we consider 5 data points from 12 sources; such sources can be sensors values, pixels in images, nods in a graph, or any other measurements. Also, the dissimilarity D_X is considered for each pair of data points, then MDS decomposes the D_X matrix to fined the corresponding z_1 and z_2 vectors that describe the manifold of the data.

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 as:

$$Z = -\frac{1}{2} H D_X H \tag{2.51}$$

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 as:

$$\tilde{X} = D_Z^T A \tag{2.52}$$

where A is the reconstruction kernel, $A = D_Z^{-1} X$.

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:

$$\mathcal{L} = \sum_{i,j} w_{i,j} (D_X(i,j) - D_Z(i,j))^2 \tag{2.53}$$

where the nonnegative weight $w_{i,j}$ refers to the weighted 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 [68].

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

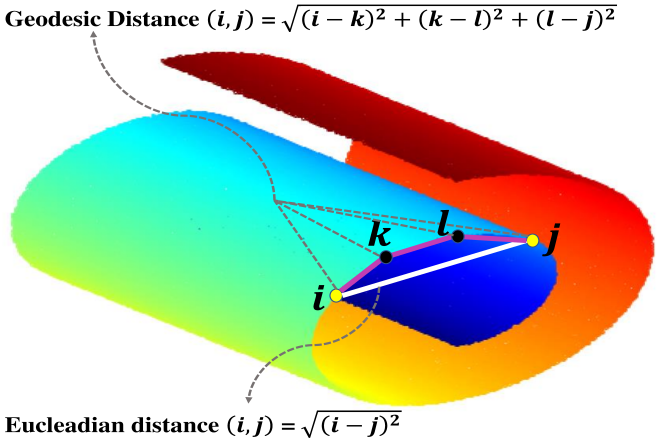


Figure 2.20: Geodesic vs. Euclidean distance between data points i and j that are pointed by yellow color.

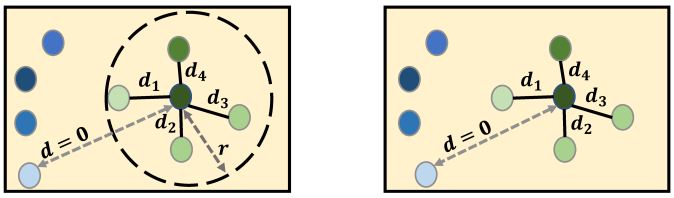
methods devised to address nonlinear manifolds are described. In such methods, a sparse graph is built over the data, and the graph information is exploited to approximate the geometrical and topological properties of the manifold.

2.2.1.2 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 [289]. 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 [78]. Fig. 2.20 shows the difference between the geodesic and Euclidean distance among two data points in a manifold.

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 neighbor ($k$nn) points or those within a suitable radius, ϵ, as it is shown in Fig. 2.21), 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. Moreover, the D_G can then be defined as the shortest path in $\mathcal{G}$ that passes through all k neighbors, which can practically be identified by using Dijkstra or Floyd-Warshall algorithms that order all D_G distances from the shortest to the longest according to the neighborhood relations [289], i.e., after computing all D_G distances, a new sampled manifold can be produced in which the distances are ordered.

The final step recalls MDS over the D_G matrix to generate a mapping Z (either by decomposing the matrix $D_G^{\intercal} D_G$ or by using the centering matrix in the form $-\frac{1}{2} H D_G H$; see Section 2.2.1.1), and an optimization procedure is applied to optimize the distances in the Euclidean space. Also, the original data, X, reconstruction from space Z can



(a) Radius, $\epsilon = r$. (b) $k$nn, $k = 4$.

Figure 2.21: A graphical comparison between radius (Fig. 2.21(a)) and $k$nn (Fig. 2.21(b)) methods to identify the neighborhood relations among data points, where the colored circles reflect different values of data points.

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 a special form as:

$$D_X(i,j) = \|x_i - x_j\| / \sqrt{m_i \times m_j} \tag{2.54}$$

where m_i is the average distance between x_i and its k neighbors [264]. Marginal Isomap (M-Isomap) [344] uses pairwise constraints to identify and weigh the neighborhood relations among the data points (specifically, there is a Cannot-Link constraint for interclass clusters and a 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 [169], while it has been stacked in a hybrid manner to extract the intrinsic structures of hyperspectral images in [16]. Also, node-weighted multidimensional scaling (NWMS) is an extension of MDS and Isomap that was proposed in [273]; NWMS partitions the data in the geodesic space employing weight matrix that is fitted to the data using EM algorithm [190]. Recently, semi-supervised multi manifold Isomap SSMM-Isomap was proposed in [342] 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.

2.2.1.3 Nonlocal Manifold Tangent Learning

NLMTL [31] 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 mapping function $F : \mathbb{R}^D \rightarrow \mathbb{R}^{d \times D}$

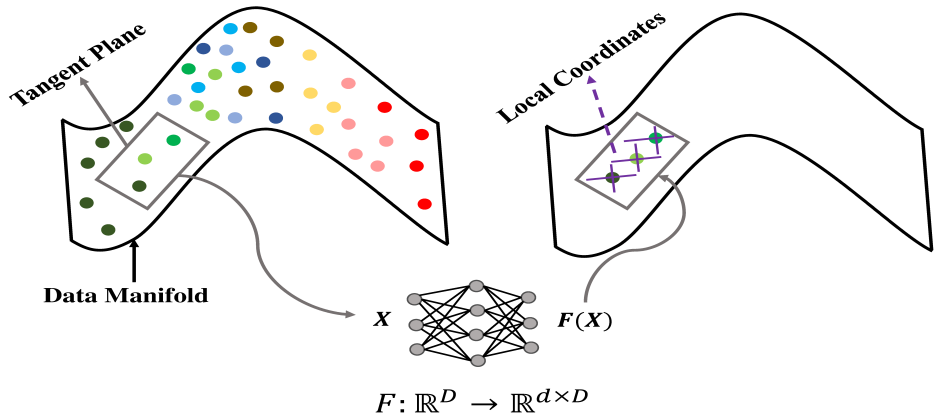


Figure 2.22: A graphical representation of NLMT, where two layers NNs are used to map the tangent plane (as a matrix) to its corresponding coordinates.

such that $F(x)$, modeled as a standard single-hidden-layer neural network, outputs a basis for the tangent plane at the point x. Fig. 2.22 depicts the tangent plane mapping of a manifold in $\mathbb{R}^D$ to its corresponding local coordinates in $\mathbb{R}^d$ space. 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:

$$\mathcal{L}(F,\, w) = \sum_i \sum_{j \in \mathcal{N}_i} \frac{\|w_{i,j}^{\intercal} F(x_i) - (x_i - x_j)\|^2}{\|x_i - x_j\|^2} \tag{2.55}$$

where the contribution of each example (data point) is weighted by the distance from the reference point, i.e., $w_{i,j}^{\intercal}$, 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 [30], in which the manifold distribution is described in terms of a global covariance matrix $\mathrm{C_{X,X}}$. NLMTL is also inspired by neighborhood component analysis [103]; by contrast, NLMPW is generalized and its learning inspired by the manifold Parzen window [302], 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 [46], 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) [332], the local point geometry is extracted based on anchors representing the low-dimensional

coordinates. In the LCC method developed in [331], the number of anchors is optimized by considering a local tangent plane, as in NLMTL.

2.2.2 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 [298].

In the following, we discuss local MfL methods, including locally linear embedding (LLE) [244], Laplacian eigenmaps (LEs) [23], maximum variance unfolding (MVU) [315], t-distributed stochastic neighbor embedding (t-SNE) [296], and their relevant extensions.

2.2.2.1 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' overall 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 and weighing the local distances through the k nearest data points (see Fig. 2.21(b)) rather than by estimating the geodesic or Euclidean distances between the widely separated patches; accordingly, it preserves the locally linear structure of the manifold [244]. 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. Fig. 2.23 illustrates the two main stages of the LLE algorithm, where the distances between local neighbors are weighed after obtaining the neighborhood relations, and the same weights are used in the low coordinates mapping space Z.

Formally, the manifold is described by the weights, $W_{i,j}$, obtained to minimize the error (or distance) using a special form of the MSE metric in the manifold data space:

$$\text{MSE}_x = \sum_i \|x_i - \sum_j W_{i,j} x_j\|^2 \tag{2.56}$$

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, i.e., $\Sigma_j W_{ij} = 1$. The low-dimensional embedded coordinates are then generated by minimizing the cost function:

$$\text{MSE}_z = \sum_i \|z_i - \sum_j W_{i,j} z_j\|^2 \tag{2.57}$$

where the same weights utilized in Eqn. (2.56) are employed, and $\frac{1}{n} Z^\intercal Z = I$ as an optimization objective. In practice, the problem can be reframed as the computation

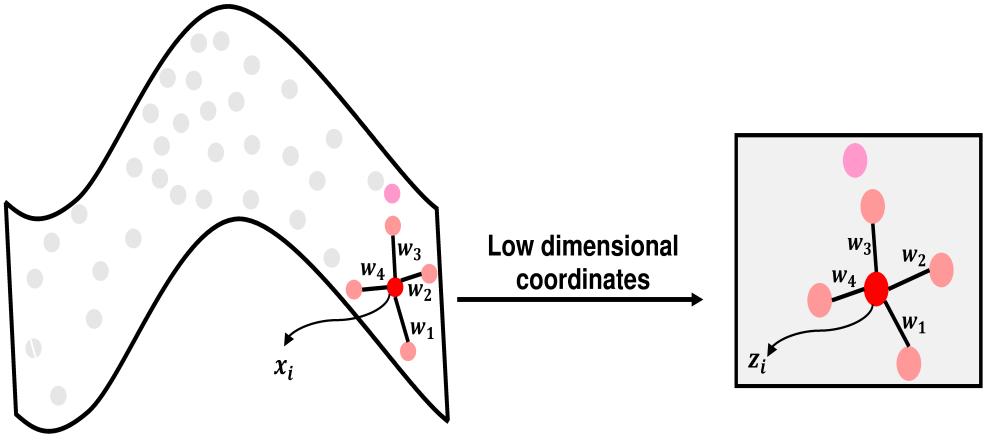


Figure 2.23: Neighborhoods weighing of LLE, where 4nn are considered to build the neighborhood relations between the data points in a local batch. Moreover, the same weights (e.g., $w_1 - w_4$) which reconstruct each data point from its neighbors are shared between the data manifold, X, and the low coordinate space, Z, to optimize the mapping procedure.

of the $d+1$ bottom eigenvectors (corresponding to the $d+1$ smallest eigenvalues) 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:

$$\tilde{X} = W^{-1}Z \tag{2.58}$$

Various extensions of LLE have been proposed in the literature, e.g., the locally linear coordination (LLC) method proposed in [288], which is based on viewing a nonlinear manifold as a mixture (see Section 2.1.3.2.1) of subspace models 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 [73] 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 [221]. Moreover, nonnegative adaptive feature extraction (NAFE) [341] is a method akin to LLE (they use similar optimization formulations) that is able to embed new data. NAFE is built based on sparse NMF (see Section 2.1.2.2) to correct outliers, and weights 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.

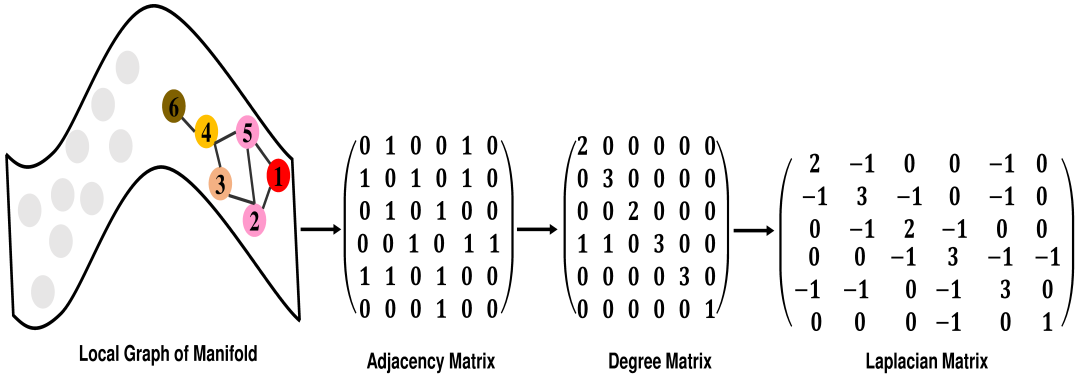


Figure 2.24: The mathematical representation of a local graph in the manifold, where the adjacency matrix, A, is obtained from the connected nodes (if two nodes are connected the value is 1, and 0 otherwise); the degree matrix, D, is obtained by summing up each row of the adjacency matrix and form the result of summation among all rows in a diagonal matrix; and the Laplacian matrix is estimated as $L = D - A$. Moreover, L contains 0 eigenvalues if the graph is connected, however, it contains n eigenvalues for a graph contains n branches.

2.2.2.2 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 [23]. 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 [148]. The LBO is related to the modeling of 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 weights matrix and control its decay.

The algorithm starts from a graph of the data points, which is formalized as an adjacency matrix. Fig. 2.24 depicts the procedure of obtaining the mathematical representation of a graph in the manifold, where the numerical values of 6 nodes are considered, as an example, in the manifold to obtain the Laplacian matrix. For every pair of connected nodes, the relations (or distance of edges) can be identified as 1, or 0 for the disconnected ones. Also, the weight of the edge connecting two neighboring nodes can be defined using the heat kernel distance:

$$W_{(i,j)} = \exp(-\frac{\|x_i - x_j\|^2)}{t}) \tag{2.59}$$

where t is a parameter that identifies the scale and data point pools. The Laplacian matrix L is then computed using W instead of using the adjacency matrix A as:

$$L = D - W \tag{2.60}$$

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:

$$Ly = \lambda Dy \tag{2.61}$$

as the components of the 2^{nd} to $d + 1^{th}$ eigenvectors corresponding the eigenvalues (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. Practically, the algorithm uses the trace loss to simplify th optimization:

$$\mathcal{L} = \min(\mathrm{Tr}(Z^\intercal LZ)) \tag{2.62}$$

where $Z^\intercal Z = I$ is considered as an optimization objective. Moreover, the optimization procedure can be viewed as analogous to spectral clustering (SC) in the context of Laplacian graph cuts and segmentations [262]. 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 the 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 eigenvalues 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 [79] 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 [47]. Another extension of the LE method, introduced in [349], utilizes the projection of the principal curves [117], and the Hessian operator is applied in the projection space to obtain the low-dimensional coordinates.

2.2.2.3 Maximum Variance Unfolding

In MVU [315][281], initially proposed as a 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 [299] and global isometric mapping [289]. 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) preserves local distances and can be characterized by the constraint:

$$\|z_i - z_j\|^2 = \|x_i - x_j\|^2 \tag{2.63}$$

The MVU assumes that to unfold the manifold, the variances in the mapping space must be maximized (pair wise distances in Z), which can be expressed as:

$$\text{MUV} = \max \sum_{i,j} \|z_i - z_j\|^2 \tag{2.64}$$

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 [316] for learning a low-dimensional manifold depends on maximizing the variance in the feature space and can preserve the distances and angles among the members of each k-neighborhood. The minimum volume embedding (MVE) method proposed in [260] has a different objective function than SDE; specifically, MVE intends to maximize the energy for the top d dimensions of the eigenspectrum to maintain the local distances among k neighbors. Furthermore, to ensure that the structural and topological characteristics of the original data are retained, structure-preserving embedding (SPE), proposed in [261], can be utilized for loss monitoring and visualization.

2.2.2.4 T Distributed Stochastic Neighbor Embedding

Maaten *et al.* [296] 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 weights 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 [127], which starts by identifying a similarity matrix based on an asymmetric joint distribution. For two data points, x_i, x_j, in a manifold $X \in \mathbb{R}^D$, SNE identifies the probabilistic similarity relations in the manifold space as:

$$P_{(j|i)} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i)}{\Sigma_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i)} \tag{2.65}$$

where k refers to the index of all k-neighbors points of the data point x_i. Equivalently, SNE measures the same probabilistic similarity relations in the embedding space, $Z \in \mathbb{R}^d$, as:

$$Q_{(j|i)} = \frac{\exp(-||z_i - z_j||^2/2\sigma_i)}{\Sigma_{k \neq i} \exp(-||z_i - z_k||^2/2\sigma_i)} \tag{2.66}$$

where k refers to the index of all k-neighbors points of the embedding space point z_i. Finally, the SNE optimizes the mapping from space X to spaces Z to visualize the manifold, throughout optimizing the distributions in both spaces and minimizing the KL divergence (see Eqn. (1.6)) as follows:

$$\text{KL}(P, Q) = \Sigma_i \Sigma_j P_{(j|i)} \log \frac{P_{(j|i)}}{Q_{(j|i)}} \tag{2.67}$$

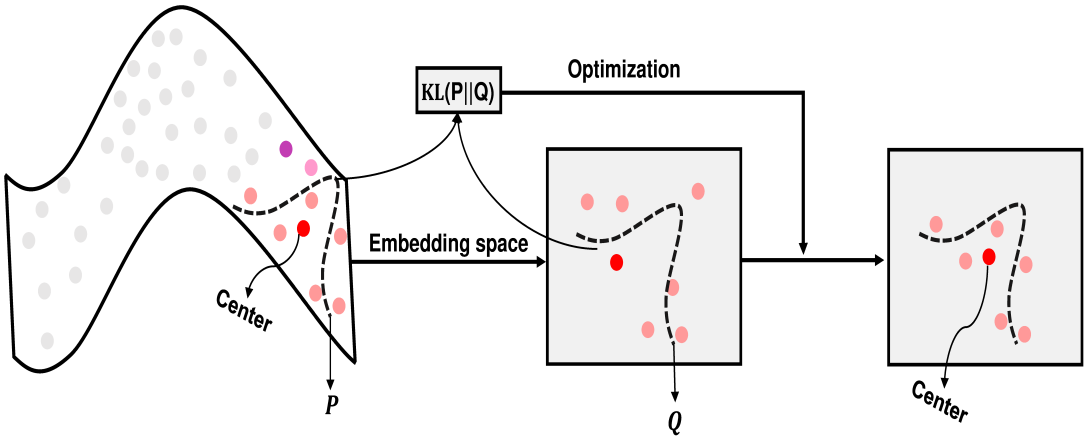


Figure 2.25: The graphical representation of the SNE embedding learning, where both distributions P and Q are optimized by minimizing the KL divergence.

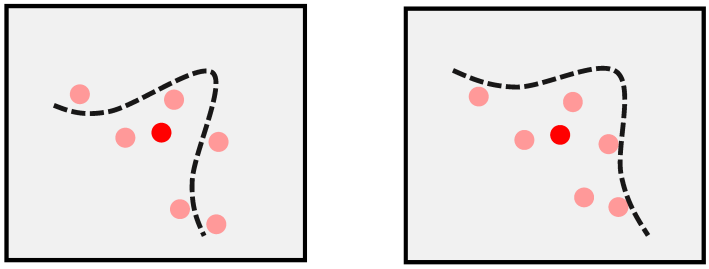
Fig. 2.25 shows the graphical representation of the SNE learning, where it fixes a Gaussian in the data manifold, thereafter it draws another Gaussian with random points in the embedding space to optimize the two distributions by using the KL divergence.

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 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 in the mapping space Z; SNE uses the Gaussian distribution, which suffers from the curse of dimensionality, whereas t-SNE employs the t-distribution (or student's t-distribution), which has a large variance and long-heavy 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 data in higher dimensions. Fig. 2.26 depicts the difference between using the Gaussian distribution (Fig. 2.26(a)) and t-distribution (Fig. 2.26(b)) to accommodate data in the embedding space, where a similar data points configuration is exploited from Fig. 2.25.

As in SNE, for two data points, x_i, x_j, in a manifold $X \in \mathbb{R}^D$, t-SNE identifies the probabilistic similarity relations in the manifold space using the probability relation that is presented in Eqn. (2.65); however, in t-SNE, the probability is symmetrized as follows:

$$P_{ij} = \frac{P(j|i) + P(i|j)}{2N} \tag{2.68}$$

where N represents the dimensionality of the original data manifold, $P_{ij} = P_{ji}$, $P_{ii} = 0$, and $\Sigma_{i,j} P_{i,j} = 1$. Moreover, t-SNE measures the same probabilistic similarity relations of Eqn. (2.69) in the embedding space, $Z \in \mathbb{R}^d$, but a heavy-tailed distribution is used



(a) Gaussian Distribution. (b) T-Distribution.

Figure 2.26: Gaussian distribution (Fig. 2.26(a)) vs. t-distribution (Fig. 2.26(b)) in the embedding learning, where it is noticed that the t-distribution is able to accommodate more data points than the Gaussian due to its heavy-tailed and large variance, see Fig. 2.25.

to allow dissimilar points in the mapping space to be far away from the similar ones:

$$Q_{ij} = \frac{(1 + ||z_i - z_j||^2)^{-1}}{\Sigma_{k \neq i}(1 + ||z_k - z_l||^2)^{-1}} \tag{2.69}$$

where $Q_{ii} = 0$ and it is matched iteratively with P_{ij} by minimizing the KL divergence (see Eqn. (2.67)) through gradient descent optimization.

Further extensions of the t-SNE technique include parameterized t-SNE [191], non-metric visualization [297], and t-SNE acceleration by quadtrees to reduce the computational cost [295].

2.2.3 Summary

Humans' perception of big data varies more than a computer's attention and is limited to low dimensionality; furthermore, increasing the dimensions to $d > 3$ leads to uninterpretable observations when visualizing data [287]. We highlighted the prevalent and well-known MfL models, including both global (Section 2.2.1) and local (Section 2.2.2) techniques that can be utilized for dimensionality reduction in visualization and data exploration, to facilitate representation learning and the determination of interpretable information and decisions. These methods share similar intuitions in transforming the original data structures expressed in a high dimensional space, $X \in \mathbb{R}^D$, $D > 3$, into a sampled space of the manifold (as in constructing neighborhood graph $\mathcal{G}$ from X) and then performing additional estimations, e.g., computing and decomposing geodesic or weights matrices in the sampled manifold to obtain the final mapping space, $Z \in \mathbb{R}^d$, $d <<< D$; thereby enabling data visualization, exploration, and feature extraction to facilitate model learning [44].

The core idea of employing such techniques in building machine learning models lies in mapping high-dimensional data into a low-dimensional joint space. Consequently, such techniques can be used in a bidirectional manner to provide insight into the data by looking at the joint mapping space. Moreover, they offer the ability to generate or

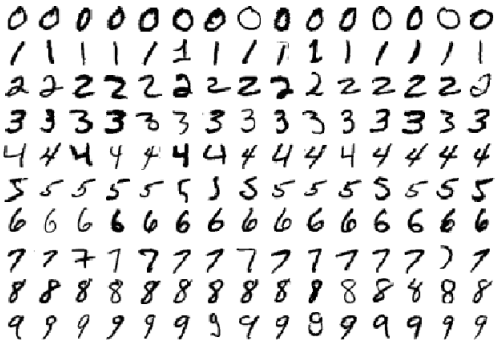
reconstruct the data from the joint space, thus overcoming the challenges presented by corruption, outliers, and session variability in subsequent data samples.

By comparing the local and global models for MfL, it is easy to identify a trade-off between the two types of methods. Methods in the global class, such as MDS (Section 2.2.1.1), Isomap (Section 2.2.1.2), and NLMTL (Section 2.2.1.3), seek to preserve the general curvature and global characteristics of the manifold and data points. Local methods, such as LLE (Section 2.2.2.1), LE (Section 2.2.2.2), and t-SNE (Section 2.2.2.4), attempt to maintain the local neighbor relations between data points. Recently, robust neighborhood preserving projection (2DNPP) [345] has been proposed as a local method for extracting representations from 2D images, where it is built based on preserving neighborhoods during projection. As a result of this difference in their focus, global methods tend to be affected by the shortcut edge problem (points that are considered neighbors but belong to different regions of the manifold), which induces errors in the estimation of the manifold topology, whereas local methods do not propagate such errors [52].

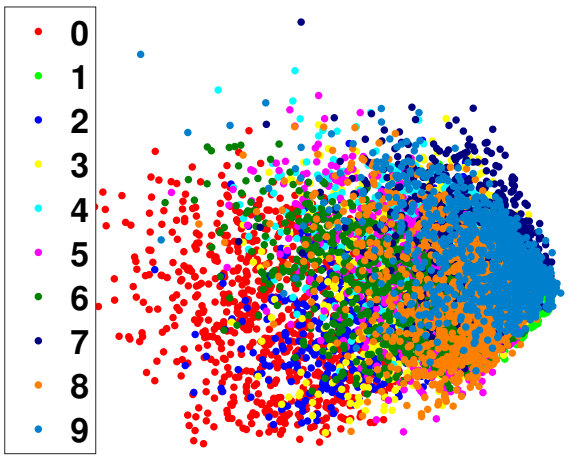
In particular, among all recent modalities, t-SNE appears to be superior due to its ability to preserve both local and global structures [3, 296]. Furthermore, both global and local methods can suffer from the out-of-sample problem when embedding new high-dimensional data into the previously obtained mapping space. Various works have attempted to overcome this issue, as in [32]; however, there is a demand for further investigations regarding very-large-scale online data streaming, manifolds with holes, and the nonconvexity of neighboring points (i.e., *isometric submanifolds*). Fig. 2.27 presents a comparison between the most widely used MfL methods, where we show the ability of each method to visualize the embeddings of 10000 images from the MNIST digits dataset [75].

Furthermore, note that PCA (described in Section 2.1.1.1) is applied directly to the data matrix, whereas MDS requires the computation of the dissimilarity matrix and thus incurs a higher computational cost; we refer to [52, 250] for a more detailed comparison among manifold methods in terms of computational complexity[1]. Moreover, Isomap, LLE, and LE transform the original data to special forms before carrying out the eigendecomposition to visualize the embeddings, e.g., Isomap estimates geodesic distances matrix, LLE approximates weights sparse matrix, and LE estimates Laplacian matrix of the original data. Table 2.1 summarizes the relationship between PCA, MDS, Isomap, LLE, and LE to the eigendecomposition, where all methods apply the decomposition considering different transformations among data and different locations among the eigenvectors of the decomposed matrices. Furthermore, it is important to mention that here, we have attempted to focus on how MfL models have developed and how they can be utilized, rather than presenting an exhaustive list of investigations into each technique; we refer the reader to [29, 174, 298] for relevant complementary reviews.

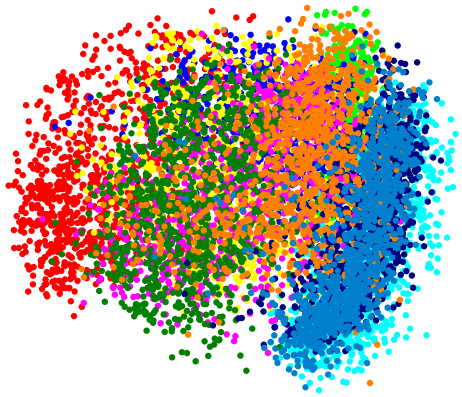
[1] https://scikit-learn.org/stable/modules/manifold.html



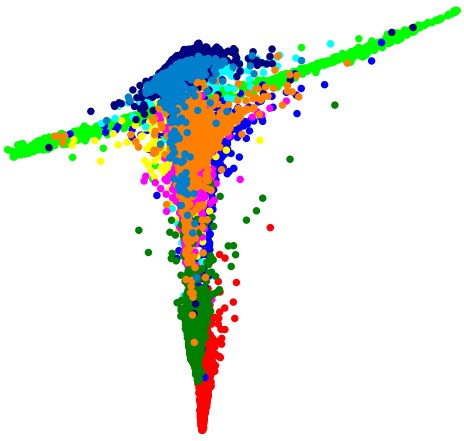
(a) MNIST Digits.



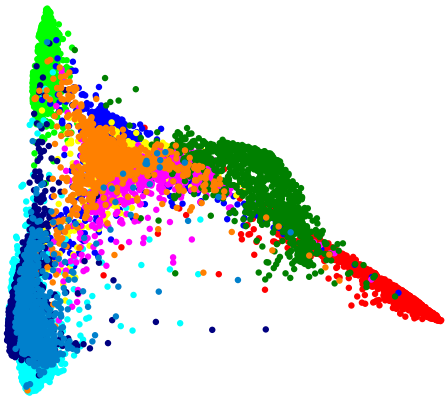
(b) MDS (Section 2.2.1.1).



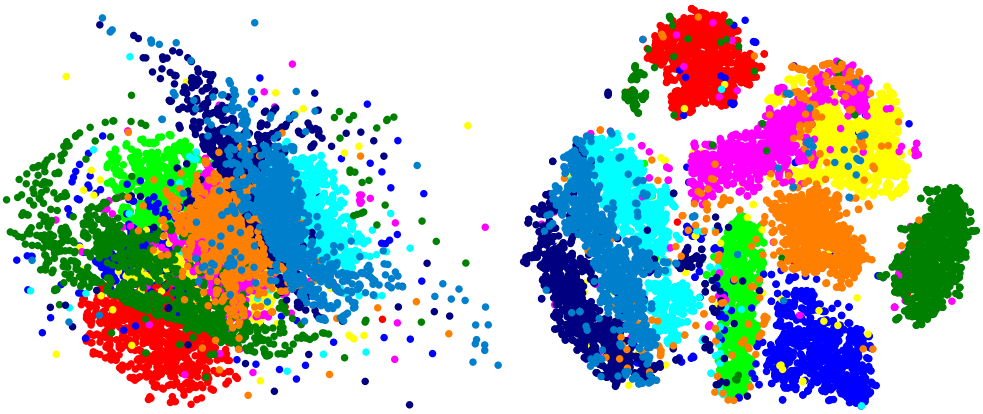
(c) Isomap (Section 2.2.1.2).



(d) LLE (Section 2.2.2.1).



(e) LE (Section 2.2.2.2).



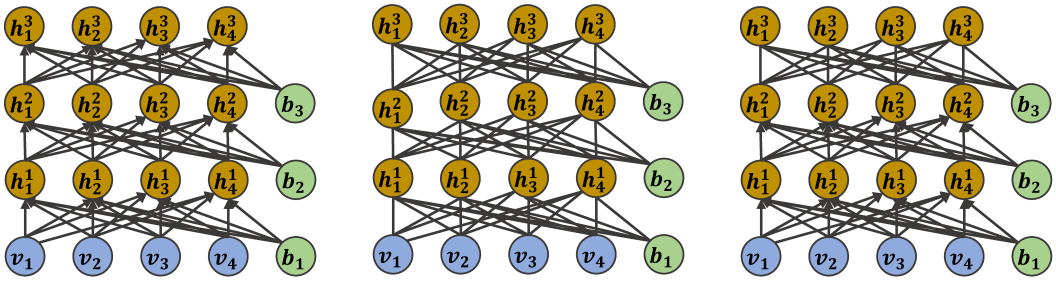
(f) SNE (Section 2.2.2.4). (g) t-SNE (Section 2.2.2.4).

Figure 2.27: A comparison between (b) MDS, (c) Isomap, (d) LEE, (e) LE, (f) SNE, and (g) t-SNE. From the six subfigures, it is noticed that t-SNE outperforms the other methods because it depends on the t-distribution, which offers a high volume to accommodate the high-dimensional embedding of the manifold into the mapping space. Moreover, because Isomap attempts to preserve geodesic distances, the data appear as thick scattered curves, whereas MDS and LLE force the mapping space parameters to be centered around the origin, which leads to overlapping clusters. Moreover, LE is a geometrically motivated method that preserves the geometric properties among data classes, and it clusters similar subgroups to the same region in the embeddings space; whereas SNE attempts to preserve the local neighborhood relation among data points, but it suffers from the crowding problem in the mapping space (class overlapping because the algorithm is unable to accommodate the moderate-near data points), thus t-SNE has been introduced in [296] to break the crowding problem by using t-distribution with heavy-tailed in the mapping space.

Table 2.1: The relationship between PCA, MDS, Isomap, LLE, and LE to the eigen-decomposition. Whereas PCA is applied to the original data form, all other methods decompose transformed forms of the data (not the original data).

Method	Decomposition Space	Note
PCA	C_X	Covariance matrix C_X: Ordinal PCA is obtained from $XX^\intercal$, and Dual PCA is obtained from $X^\intercal X$. The mapping space is visualized considering the top eigenvectors corresponding to the highest eigenvalues of the decomposed matrix
MDS	$-\frac{1}{2}HD_XH$	Dissimilarity matrix D_X with double centering matrix H. The mapping space is visualized considering the top eigenvectors corresponding to the highest eigenvalues of the decomposed matrix
Isomap	$-\frac{1}{2}HD_GH$	Geodesic matrix D_G with double centering matrix H. The mapping space is visualized considering the top eigenvectors corresponding to the highest eigenvalues of the decomposed matrix
LLE	$\lambda_{\text{Max}}(I-M)$, or M^{-1}	$M=(I-W)(I-W)^\intercal$ where $I-W$ is a form of Laplacian, and M^{-1} is pseudo inverse that gives the same eigenvectors but the eigenvalues are flipped. The mapping space is visualized considering the bottom eigenvectors corresponding to the smallest eigenvalues of the decomposed sparse matrix, from 2 to $d+1$ eigenvectors (the smallest eigenvalue is zero)
LE	$\lambda_{\text{Max}}(I-M)$, or M^{-1}	$M=(I-D)(I-D)^\intercal$ where $I-D$ is a form of Laplacian, and M^{-1} is pseudo inverse that gives the same eigenvectors but the eigenvalues are flipped. The mapping space is visualized considering the bottom eigenvectors corresponding to the smallest eigenvalues of the decomposed sparse matrix, from 2 to $d+1$ eigenvectors (the smallest eigenvalue is zero)

Nowadays, Learning from data manifold is evolved with the prosperity of UGL based-NNs and deep learning models (see Section 2.3), throughout using modern graph neural networks [107] and GANs [105] to learn graph data, embeddings, and Riemannian manifold. Variational graph auto-encoder (VGAE) has been proposed in [154] based on VAE (see Section 2.3.2.4), where the authors investigated the link prediction [34] in a graph data utilizing non-Euclidean representations. A distance preserving embedding among data has been investigated in [101], whereas the Gaussian random weights were used for a deep network to preserve a small angle between relative data within the same class. In [213], Poincarè embedding has been employed to capture latent hierarchical representations among complex symbolic data, by embedding the data into a hyperbolic space based on using a Riemannian optimization. The graph structure has been divided into small batches in [265] for the sake of molecule generation of a graph structure using graph-VAE, where such a model encodes nodes and links (or edges) from a graph structure. Moreover, it is also able to reconstruct the graph from the embedding space; conversely, in [230] and [154] the contents of the nodes and topological structure have been encoded as a single graph. The concept of AAE (see Section 2.3.3.2) in combining a GAN and AE in a single model has been followed in [160], by employing autoencoding and adversarial objective functions to learn the manifold from data that preserves a smooth and local mapping in the embedding space. Moreover, the AAE has been utilized recently in [107] to detect the dynamical stationary points (stationary points are equivalent to the representative point of k-neighbors) in graph streams, by exploiting the embeddings that are obtained from constant-curvature Riemannian manifolds.



(h) Directed NN. (i) Undirected NN. (j) Hybrid NN.

Figure 2.28: The graphical representation of the probabilistic graphical models of NNs, where we consider (a) directed, (b) undirected, and (c) hybrid models with one visible layer $V = \{v_i\}$ and three hidden layers $H = \{h^i\}$. Moreover, v_i reflects the index of the visible unit that pass data x_i through it, h_i represents the index of a hidden unit that contains the transformed data, and b_i is the bias that shifts the values of activation of neurons in the corresponding hidden layer to a better fit a model to the data instead of centering the fitting line around zero.

2.3 Neural Network Models

Generative models based on NNs play a substantial role in learning representations of data. These models are probabilistic graphical models PGMs and can be divided into three categories. Models in the first category are called directed generative models as it is depicted in Fig. 2.28(h) 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, given the states of the hidden variables $P(V|H)$ ($P(v|h)$ for a pairwise visible and hidden units) [123, 159]. The second category consists of undirected models and it is shown in Fig. 2.28(i), in which joint probabilities $P(V, H; W)$ between the visible and hidden units are defined given the weights parameters W. Furthermore, NNs models are inherently probabilistically undirected or directed; however, different models may comprise both directed and undirected graph schemes as it appears in Fig. 2.28(j). Moreover, these models usually involve estimating the probability density function (pdf) of the approximated data ($\tilde{X}^m$, 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 [8].

From this perspective, three approaches 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 [124, 126]. Furthermore, either a product or a compo-

sition 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, $h_i = \sigma(w_i^\intercal v_i)$, and σ is the sigmoid activation. 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. [29], see Section 2.2. However, the combinations between the MfL, BSS, and NNs can bring interesting and obedient representations among the data to facilitate learning and reduce computational complexity.

In the following, we review three classes of models derived from the NNs family, namely, energy-based models (EBMs) in Section 2.3.1, autoencoder models (AE) in Section 2.3.2, and adversarial learning models (GANs) in Section 2.3.3.2.

2.3.1 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 [58] (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). Moreover, EBMs are built based on the Markov process employed to capture the probabilistic dependencies among a set of RVs. Also, the Markov process has a condition that enforces the independence between RVs, or neurons, in the same layer given the activation of neurons in the preceding or subsequent layers. Furthermore, in a Markov network, each node is parameterized by a unary potential function, ϕ, and it is also parameterized by a pairwise potential function, ψ, with another connected node [286]. Fig. 2.29 depicts an example of a pairwise Markov network of two variables.

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, which is 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 [125]. More specifically, the probabilities for the visible and hidden units in an EBM are assigned following the Boltzmann distribution:

$$P(v,\, h;\, W) = \frac{\exp^{-E(v,\, h;\, W)}}{Z} \tag{2.70}$$

where Z is the partition function (canonical ensemble) that is used to normalize the

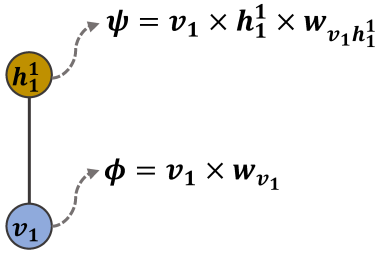


Figure 2.29: The graphical representation of a pairwise Markov network of one visible and hidden units, where ϕ reflects the unary potential energy in the visible node, w_{vi} is the node weight, and ψ represents the pairwise potential energy between v, h interaction.

approximated distribution over the possible values of the distribution variables among all possible configurations, and is gives as:

$$Z = \sum_{v_i, h_j} \exp^{-E(v_i, h_j; w)} \tag{2.71}$$

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 models can provide closed-form representations of the distributions of the training set and the hidden variables. In the following section, we highlight typical EBMs, including restricted Boltzmann machines (RBMs) [124] in Section 2.3.1.1, conditional restricted Boltzmann machines (CRBMs) [197, 201] in Section 2.3.1.2, deep belief networks (DBNs) [126] in Section 2.3.1.3, deep Boltzmann machines (DBMs) [252] in Section 2.3.1.4, and their relevant extensions.

2.3.1.1 Restricted Boltzmann Machines

RBMs [268][124] were among the earliest models for data reconstruction based on Boltzmann machines [6] 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 [87]. The connections are characterized by the weights W and the two biases c and b for the visible and hidden units, respectively. Fig. 2.30 illustrates the graphical representation of an RBM with two layers.

For binary or Bernoulli units, the configurations (v, h) between the hidden and

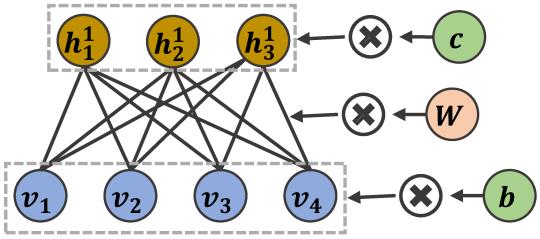


Figure 2.30: The graphical representation of an RBM model of 4 visible units and 3 hidden units, where two biases, b, c, are introduced for v and h, respectively.

visible units have the following energy function:

$$E(v,\, h;\, \theta) = -\sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} c_j h_j - \sum_{i,j} v_i h_j W_{i,j} \tag{2.72}$$

where $\theta = \{b, c, W\}$. 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 and can be factorized:

$$P(v|h) = \prod_{i \in \text{hidden}} P(v_i|h) \tag{2.73}$$

where, for a Bernoulli RBM, $P(v_i = 1|h)$ is given as:

$$P(v_i = 1|h) = \sigma(b_i + W_i h) \tag{2.74}$$

where σ is the sigmoid activation function. Moreover, this also applies to the hidden units:

$$P(h|v) = \prod_{i \in \text{visible}} P(h_i|v) \tag{2.75}$$

where, for a Bernoulli RBM, $P(h_j = 1|v)$ is given as:

$$P(h_j = 1|v) = \sigma(c_j + v^\intercal W_j) \tag{2.76}$$

These properties simplify the training algorithm because variables of the same type (in the same layer) can be sampled simultaneously. Moreover, the 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_i and h_j units in accordance with Eqn. (2.70).

Additionally, the reconstruction error in the RBM is estimated using the k-step contrastive divergence [124] to optimize the model parameters $\{W, b, c\}$ and maximize the log-likelihood probability. The contrastive divergence is used to minimize the KL divergence between the current obtained distribution, $P_0(v|W^*)$, and the equilibrium distribution, $P_\infty(x|W)$, which is obtained in the last stage of the model learning, or it

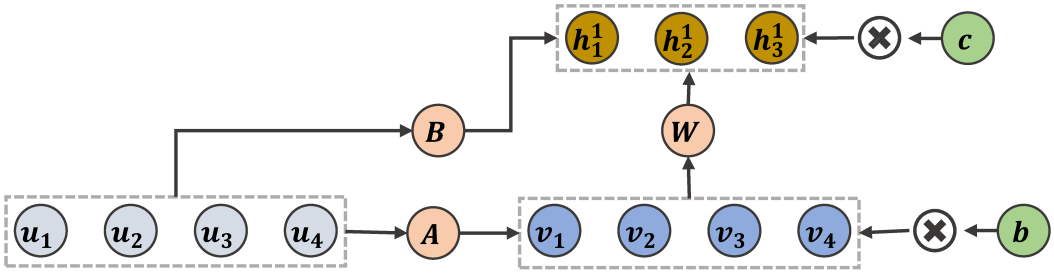


Figure 2.31: The graphical representation of an CRBM model of 4 visible and historical units, and 3 hidden units, where two biases, b, c, are introduced for v and h, respectively. Also, three different sets of weights, A, B, and W are used to learn the model.

can be sampled from the original data. 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 [125]. RBMs are used for various tasks, including multiclass classification, collaborative filtering, and information retrieval. They are also utilized as pretraining facilitators for autoencoder networks [128].

2.3.1.2 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 [27]. To this end, in [201], 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 $P(v|u)$.

$$\begin{aligned} E(v,\, u,\, h;\, A, B, W) = &- \sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} c_j h_j - \\ &\sum_{i,j} v_i h_j W_{i,j} - \sum_{i,k} v_i u_k A_{i,k} - \sum_{k,j} u_k h_j B_{k,j} \end{aligned} \tag{2.77}$$

where $\theta = \{A, B, W\}$, and the model contains an associated energy F:

$$F(v,\, u;\, A) = -\sum_{i \in \text{visible}} b_i v_i - \sum_{i,k} v_i u_k A_{i,k} \tag{2.78}$$

where during the learning, the pairs (u, v) are used as input (visible layers), while after training, only the u variables are known, and v has to be estimated from $P(v|u)$:

$$P(v|u) = \frac{\exp^{-F(v,u)}}{\sum_{v_i,u_k} \exp^{-F(v_i,u_k;w)}} \tag{2.79}$$

There are various extensions of CRBMs; among the most popular are Gaussian-Bernoulli restricted Boltzmann machines (GBRBMs) [59], 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 [27, 58]. Another CRBM model, proposed in [180], integrates a label layer instead of the historical unit layer u as the conditional layer. Moreover, in a gated restricted Boltzmann machine (GRBM) [197], 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 weights matrix W being a 3-way tensor among the input, hidden, and output layers. CRBMs are used in motion recognition, multi-class simultaneous labeling, and nonlinear time-sequence analysis; recently, interesting results have been achieved for gait recognition [61].

2.3.1.3 Deep Belief Networks

Deep learning has become increasingly popular in the ML community since the breakthrough work presented in 2006 by Hinton *et al.* [126], 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 failed to train most multilayer deep NNs [102]. 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 [88] when conceiving of the deep architecture as a sequential model. A DBN is a hybrid probabilistic model constructed by sequentially combining several simple blocks that are forced to extract different representations. The learning process described in [126] is performed employing greedy layer-wise 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. Fig. 2.32 depicts an example of DBNs, where we use the same network size that is shown in Fig. 2.28(j). Furthermore, the training of such a DBN starts from the generation

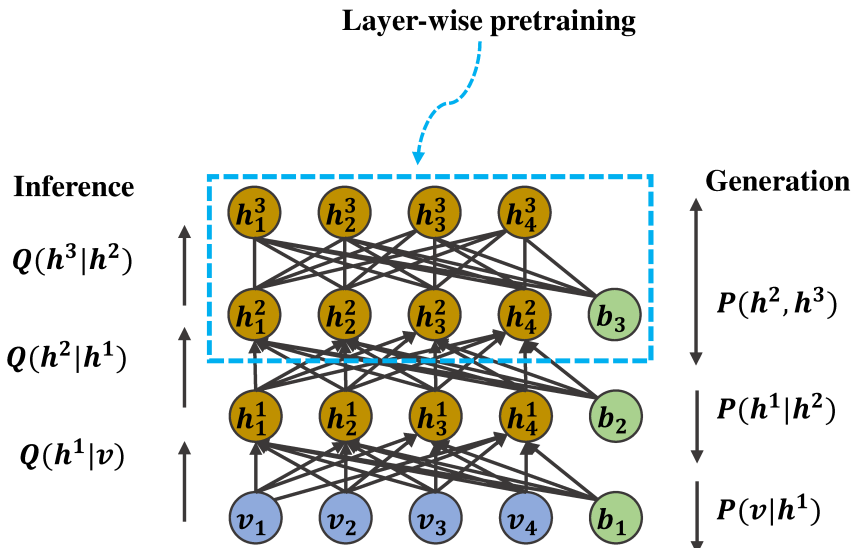


Figure 2.32: The graphical representation of a DBN model of 4 visible units and 3 hidden layers of the same size of the visible layer, where two different processes are utilized to learn the model including generation (right) and inference (left), and the layer-wise pretraining is performed by using an RBM.

of the posterior distribution for the preceding-down layers by using an RBM at the top, thereafter it uses the posterior distribution in the inference for model optimization (thus the training process exploits the complementary prior).

The training of a DBN is performed through a recursive procedure that starts with the training of an RBM. The weights matrix between the visible and hidden layers is then frozen, and its (transposed) replica is used to initialize the weights 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 joint distribution for the DBN model that is appeared in Fig. 2.32 is given as:

$$P(v, h^1, h^2, h^3) = P(h^2, h^3)P(h^1|h^2)P(v|h^1) \tag{2.80}$$

where the distribution is conditional in the directed-connected layers (v, h^1, h^2); however, the top two layers (h^2, h^3) are undirected-connected layers and their joint distribution is estimated first (top-down flow) according to the Boltzmann distribution as stated in Eqn. (2.72):

$$P(h^2,\, h^3;\, W) = \frac{\exp^{-E(h^2,h^3)}}{Z} \tag{2.81}$$

where Z is the partition function that gives the normalization scalar. The conditional distribution $P(h^1|h^2)$ is given as:

$$P(h^1|h^2) = \prod_j P(h_j^1|h^2) \tag{2.82}$$

and the conditional distribution $P(v|h^1)$ is given as:

$$P(v|h^1) = \prod_i P(v_i|h^1) \tag{2.83}$$

Eventually, the bottom-up flow approximates the posterior inference that is associated with a conditional distribution, Q, by using the variational mean-field inference:

$$P(h|v) \approx Q(h|v) = \prod_j Q(h_j|v) \tag{2.84}$$

where the optimization procedure maximizes the maximum posterior (MAP) for each hidden layer. Furthermore, after the pretraining, generation, and inference processes are completed, the model fine-tuning is introduced to measure how the model can fit data; such a fine-tuning can be a supervised or unsupervised procedure and includes discriminative fine-tuning by maximizing $\log P(Y|X)$ (X: data, Y: label), and generative fine-tuning by maximizing $\log P(Y, X)$. Finally, the complete training procedure can be found in [126, 128, 251].

Convolutional deep belief networks (CDBNs) are special types of convolutional neural networks (CNNs), 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 [173]. DBNs are employed in various application domains, including regression and classification on highly structured data, human motion analysis, and character recognition, and speech recognition [202]. Notably, DBN models are distinct from DBMs [252], of which we give a brief overview below.

2.3.1.4 Deep Boltzmann Machines

With the deep learning revolution, simpler RBM models have been combined to form DBM models [126, 252]. A DBM differs from a DBN, introduced in Section. 2.3.1.3, 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. 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. Additionally, DBMs use Markov chain Monte Carlo (MCMC) methods to estimate the gradient of the intractable partition function Z [251]. 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. Fig. 2.33 presents a DBM model composed of three RBMs, where we use the same structure proposed in Fig. 2.30 for each shallow RBM.

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.

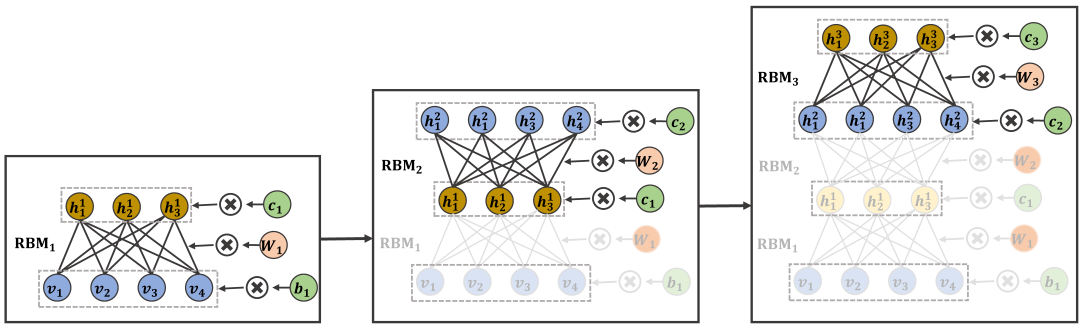


Figure 2.33: The graphical representation of a DBM model of three RBMs, where the model is initiated by an RBM (left) then it stacks all RBMs in the successive stages (right). The full model parameters, $\theta = \{b_1, W_1, c_1, W_2, c_2, W_3, c_3\}$, are considered in the learning and fine-tuning the model.

Moreover, the attention to duplicate visible units in the bottom-up, and top hidden units in the top-down passes alleviates the double-counting problem that is strongly present during training and inference [252]. The energy of the whole model that is shown in Fig. 2.33 is estimated according to Eqn. (2.72) as:

$$\begin{aligned} E(v,\, h^1,\, h^2,\, h^3;\, \theta) = &- \sum_{i \in \text{visible}} b_{1i} v_i - \sum_{j \in \text{hidden}} c_{1j} h_j^1 - - \sum_{k \in \text{hidden}} c_{2k} h_k^2 - \\ &- \sum_{l \in \text{hidden}} c_{3l} h_l^3 - \sum_{i,j} v_i h_j^1 W_{i,j} - \sum_{j,k} h_j^1 h_k^2 W_{j,k} - \sum_{k,l} h_k^2 h_l^3 W_{k,l} \end{aligned} \tag{2.85}$$

where $\theta = \{b_1, c_1, c_2, c_3, W_{i,j}, W_{j,k}, W_{k,l}\}$. Moreover, the joint distribution is driven according to Boltzmann distribution that is introduced in Eqn. (2.70) as:

$$P(v, h^1, h^2, h^3;\, W) = \frac{\exp^{-E(v,\, h^1,\, h^2,\, h^3;\, \theta)}}{Z_\theta} \tag{2.86}$$

where, for a Bernoulli DBM, $P(v_i = 1|h^1)$ is given as:

$$P(v_i = 1|h^1) = \sigma(b_1 + W_{i,j} h^1) \tag{2.87}$$

and $P(h_j^1 = 1|v, h^2)$ is given as:

$$P(h_j^1 = 1|v, h^2) = \sigma(c1 + vW_{i,j} + W_{j,k} h^2) \tag{2.88}$$

similarly, $P(h_k^2 = 1|v, h^1, h^3)$ is obtained as:

$$P(h_k^2 = 1|v, h^1, h^3) = \sigma(c2 + vW_{i,j} + W_{j,k} h^1 + W_{k,l} h^3) \tag{2.89}$$

finally, the conditional activation of the last layer, h^3, is estimated as:

$$P(h_l^3 = 1|h^2) = \sigma(c3 + W_{k,l} h^2) \tag{2.90}$$

where h_j^1 and h_k^2 are no longer independent given v_i, and hence $P(h^1|v)$ can not be factorized as $\prod_j P(h_j^1|v)$. Moreover, given h^1 the elements of the visible units, v_i are independent (as in an RBM); however, given h^2 and h^3 they are not independent.

Variational learning is utilized in DBMs to maximize the log-likelihood $P(v;W)$ and minimize the KL divergence between the approximated distribution $Q(h|v;\theta)$ and the exact posterior $P(h|v;\theta)$, utilizing a naïve mean-field method involving factorization of the approximated posterior [317] (see Eqn. (2.84)). Moreover, the maximum posterior, MAP, of a DBM is obtained by maximizing the conditional probability and learnable parameters θ:

$$h_{\text{MAP}} = \text{argmax}_h P(h|v,\theta) \tag{2.91}$$

and employing the same approximation for v inference:

$$v_{\text{MAP}} = \text{argmax}_x P(v|h,\theta) \tag{2.92}$$

DBMs are pretrained by using consecutive-separate RBMs, throughout fixing the values of the last layer of each RBM and considering it as input for the successive one. Thereafter, (after the pretraining) the whole model is trained and fine-tuned by discriminative learning, $P(v|y)$, or generative learning, $P(v,h)$:

$$P(v,y|\theta) = \Sigma_{h^1,h^2,h^3} P(v,y,h^1,h^2,h^3|\theta) \tag{2.93}$$

where y is the class label:

$$P(v,y,h^1,h^2,h^3|\theta) = \frac{\exp^{\mathrm{vW_{i,j}h^1+h^1W_{j,k}h^2+h^2W_{k,l}h^3+h^3W_{y}y}}}{Z} \tag{2.94}$$

considering that all biases are ignored in the model fine-tuning. Moreover, the optimization procedure maximizes the model likelihood that is inferred as follows:

$$P(v) = \Sigma_{h^1,h^2,h^3} P(v,h^1,h^2,h^3) \tag{2.95}$$

where the classification inference of a DBM requires to identify labels, y, for the data samples and is given as:

$$y_{\text{class}} = \text{argmax}_y \Sigma_{h^1,..,h^L} P(h^1,..,h^L,y|v) \tag{2.96}$$

where the MCMC sampling methods, e.g., Gibbs sampling, or the variational methods, e.g., variational naïve mean-field, are introduced to approximate $P(y|x)$.

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 [104], where multimodal DBM learning inspired by [276] is introduced. Finally, we refer the reader to [227, 251, 252] for further details.

2.3.2 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 [164], as EBMs do (see Section 2.3.1). 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 $z_i = f(x_i; \hat{\theta}_e)$, while the decoding stage produces a mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$, which reconstructs an approximation of the original data: $\tilde{x}_i = g(z_i; \hat{\theta}_d)$. The problem lies in finding the encoding/decoding parameters that minimize the reconstruction error:

$$\mathcal{L}_{\text{REC}_{\{\hat{\theta}_e, \hat{\theta}_d\}}} = \min \|X - (f \circ g)X\|_{\text{Er}} \tag{2.97}$$

where the reconstruction error E_r can be measured by different metrics including mean square error (MSE), Frobenius norm, reconstruction cross-entropy, or β- divergence [1] (see Section 1.3).

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 [18]. 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 weights matrix W_j and a bias b_j. In the following, we review the development of classical autoencoders [248], denoising autoencoders (DAEs) [303], contractive autoencoders (CAEs) [240], variational autoencoders (VAEs) [153], and their extensions.

2.3.2.1 Classical Autoencoders

One early type of AE, previously known as a circuit, was proposed in [248] as a shallow model consisting of 3 layers, i.e., input, output, and bottleneck (or hidden) layers, where the size of the bottleneck layer is restricted to $\log_2$ of the input layer size. Fig. 2.34 depicts an example of the early type of AE, where the architecture has been introduced in [248].

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, where the MSE reconstruction error is introduced for optimizing the model:

$$\mathbf{L}_{\text{REC}_{\{\hat{\theta}_e, \hat{\theta}_d\}}} = \frac{1}{2}\Sigma(X - (f \circ g)X)^2 \tag{2.98}$$

where $\{\hat{\theta}_e, \hat{\theta}_d\}$ are set of weights and biases for the encoder and decoder, respectively.

Shallow AEs learn representations of the original data based on two associated steps [130]: *(i) encoding*, in which the features of the data are encoded in the hidden layer using the encoding parameters $\theta = \{W_e, b_e\}$, and *(ii) decoding*, in which the encoded features are reconstructed using the decoding parameters $\theta_d = \{W_e^\intercal, b_d\}$, and utilizing the same weights used in the encoding stage. Stated otherwise, the encoder

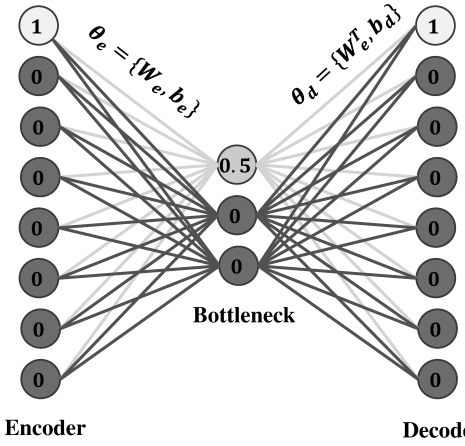


Figure 2.34: The graphical representation of the classic AE proposed in [248], where it encodes and decodes a binary string comprising 1 0 0 0 0 0 0 0.

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 [326]. These models have gained popularity due to their success in various tasks, including dimensionality reduction and discovering the underlying structures of data [128].

The difference between AEs 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 represent usable feature vectors. By contrast, in AEs, the process begins with direct mapping or encoding the inputs to their representations, generating parameterized feature vectors [29]. 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 [3]. Although the regularization of AEs can take different forms, the original form is a bottleneck, which is formed by restricting the hidden layer to have fewer units than the input and output layers. Also, the role of such a bottleneck in AEs can be viewed similarly to the role of PCA (see Section 2.1.1.1), i.e., reducing the dimensionality [257].

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 [209]. Thus, the sparse AE is optimized to minimize the sparsity loss and the reconstruction loss:

$$\mathcal{L}_{\text{SAE}_{\{\hat{\theta}_e,\,\hat{\theta}_d\}}} = \min[\mathbf{L}_{\text{REC}} + KL(\rho||\tilde{\rho})] \tag{2.99}$$

where $\mathbf{L}_{\text{REC}_{\{\hat{\theta}_e,\,\hat{\theta}_d\}}}$ is obtained according to Eqn. (2.97).

Furthermore, the effects of regularization on representations- consistency have led to the emergence of many versions of AEs models to achieve model generalization [33, 209]. Among the most popular models are DAEs [303], CAEs [240], and VAEs [153], all of which have been proposed as the results of various developments in regularization.

2.3.2.2 Denoising Autoencoders

DAE was proposed in [303] 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 [43]. Moreover, the corruption or noise can be practically implemented by adding a Gaussian noise for an image sample (or a matrix of data), x, such that:

$$\hat{x} = x + \epsilon \tag{2.100}$$

where $\epsilon \in \mathcal{N}(0, \sigma^2 I)$. Thereafter, the joint distribution between the original, corrupted (or noisy), and reconstructed samples, is drawn as:

$$P(X, \hat{X}, \tilde{X}) = P(X)P(\hat{X}|X)\delta_{f\theta_e}(\tilde{X}) \tag{2.101}$$

where the mass parameter $\delta_{f\theta_e(\hat{X})}(\tilde{X})$ enforces the decoding process from Z to $\tilde{X}$ to be deterministic; whereas if $\delta_{f\theta_e(\hat{X})} \neq (\tilde{X})$, then δ puts mass $= 0$. Accordingly, the joint distribution between X, $\hat{X}$, and $\tilde{X}$ is parameterized by the encoding/ decoding parameters θ_e, θ_d, which are optimized by using the reconstruction cross entropy $L_{\mathcal{H}}$:

$$\mathcal{L}_{\{\hat{\theta}_e,\,\hat{\theta}_d\}} = \min[(X, (f \circ g)X)]_{L_{\mathcal{H}}} \tag{2.102}$$

where:

$$L_{\mathcal{H}} = \mathcal{H}(\mathcal{B}_x||\mathcal{B}_{\tilde{x}}) = -\Sigma_{k=1}^{d}[x_k \log \tilde{x}_k + (1 - x_k) \log(1 - \tilde{x}_k)] \tag{2.103}$$

Additionally, the progression of deep learning based on stacking different shallow autoencoders, as discussed in [166], has allowed DAEs to achieve a deep form as it is shown in Fig 2.35; 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 [304]. Recently, DAEs have been employed for various

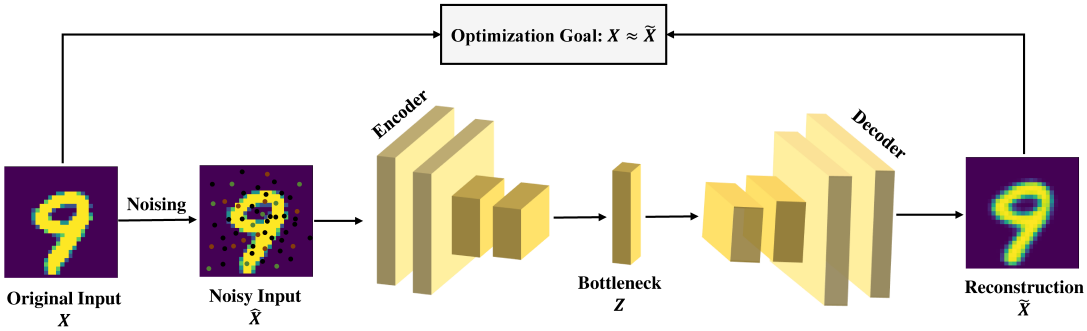


Figure 2.35: The graphical representation of a DAE with 4 layers and one level of dimensionality compression, i.e., reducing the encoding dimensionality by a stride of 2.

tasks, including machinery fault diagnosis, medical image processing, and collaborative filtering [337].

2.3.2.3 Contractive Autoencoders

In a CAE [240], 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 partial derivative is the Jacobian matrix, $\mathcal{J}$, which is used for gradient optimization. Moreover, the learning weights and their Jacobian are used in fitting the model to data. For a given data $X \in \mathbb{R}^D$, the Jacobian of the encoder activations is obtained by considering the partial derivative of the latent space (features space), Z, concerning X, where the Frobenius norm of the Jacobian is given as:

$$||\mathcal{J}(f(X))||_{\mathrm{F}} = \sqrt{\Sigma_{(i,j)}(\frac{\partial Z_j}{\partial(X_i)})^2} \tag{2.104}$$

where, for sake of computation, only the partial derivatives between Z and X are considered; however, the procedure is valid when considering the Jacobian of the weights of Z concerning the weights of X.

Furthermore, the Frobenius norm of the Jacobian of the encoder's activations, $||\mathcal{J}(f(X))||_{\mathrm{F}}^2$, 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. Fig. 2.36 shows the graphical representation of the CAE that uses the same encoding/decoding layers of the DAE depicted in Fig. 2.35.

CAE learning aims to find optimal set of parameters, θ_e, and θ_d, that minimize the error corresponding to the following objective function:

$$\mathcal{L}_{\{\hat{\theta}_e, \hat{\theta}_d\}} = \min[\|X - (f \circ g)X\|_{\mathrm{Er}} + \lambda||\mathcal{J}(f(X))||_{\mathrm{F}}^2] \tag{2.105}$$

where λ is a hyperparameter that is tuned according to the nature of data.

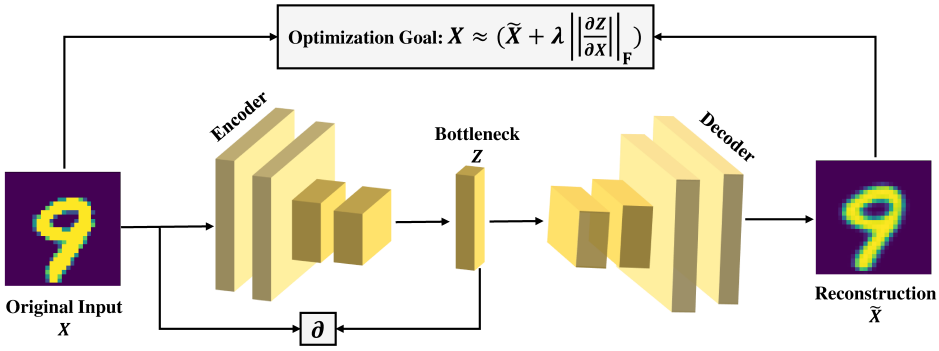


Figure 2.36: The graphical representation of the CAE that uses a similar architecture of Fig. 2.35, where ∂ represents the partial derivative operator between Z and X.

In other words, the difference loss between the reconstructed and original data is penalized and increased by the Frobenius norm of the Jacobian (between Z and X) of activation 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 [322].

2.3.2.4 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 [336]. The Bayes' rule is used to inferring the model according to a set of learning parameters, where it is given as:

$$P(X|Z) = \frac{P(Z|X)P(X)}{P(Z)} \tag{2.106}$$

where, $P(X)$ and $P(Z)$ are the priors for data X and latent space Z, respectively. Moreover, $P(Z|X)$ is the posterior probability that is intractable and usually is challenging to be approximated.

VAE uses a mean-field approximation to estimate $P(Z|X)$ by sampling a simpler Gaussian distribution, Q, thus the inference can be simplified with the Gaussian distribution, and the intractable posterior becomes tractable [153]. Throughout 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 (we use lowercase x and z for a single sample). Specifically, a VAE 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. Fig. 2.37 shows an example of a VAE that uses the same encoding architecture introduced in Fig. 2.35, where the latent code Z is sampled according to the preceding dense layers μ and σ, respectively.

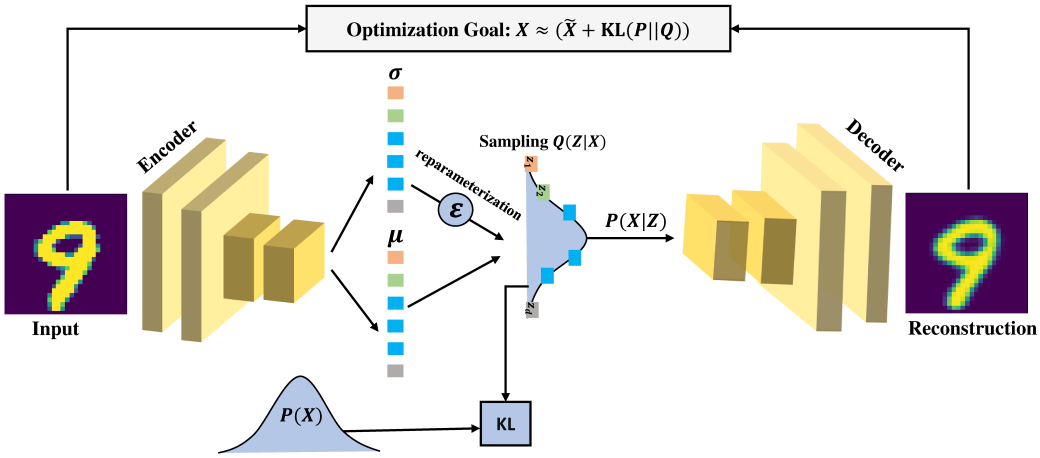


Figure 2.37: The graphical representation of the VAE that uses a similar architecture of Fig. 2.35, where the KL unit measures the KL divergence between the data distribution $P(X)$ and the sampled distribution $Q(Z|X)$, and ϵ reflect the noise unit to reparameterize Z.

As the first step, the prior distribution of the latent space $P(z)$ must be defined according to a unit Gaussian distribution (simply by copying the unit Gaussian distribution of the data manifold $P(x)$). Accordingly, the generated distribution $Q(z|x)$ and the prior distribution $P(z)$ can be compared using the KL divergence [239]. Moreover, adding noise to the approximated distributions by varying the standard deviations and training the AE for reconstruction following the true prior is called the reparameterization trick, in which several different new distributions are generated and compared with the prior for a better generalization. The noise can be appended to z according to the Gaussian corruption:

$$z = \mu + \sigma \odot \epsilon \tag{2.107}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. 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:

$$P(x) = \int P(x, z)\, \mathrm{d}z. \tag{2.108}$$

where, the ELBO is defined as the minimum bound on the evidence:

$$\mathrm{ELBO} = -\mathbb{E}_Q[P(x, z) - Q(z|x)] \tag{2.109}$$

and it can be utilized in the AE training process as:

$$\log P(x) = \mathrm{ELBO} + \mathrm{KL}(P(z)||Q(z|x)) \tag{2.110}$$

considering that $\mathrm{KL}(P(z)||Q(z|x))$ must be minimized and reach zero, whereas the ELBO must be maximized [336]. Here, VAE is optimizes depending on two losses where, the first loss is the latent loss assessed in terms of the KL divergence for each reparameterized approximated posterior $P(z|x)$ and the prior $P(z)$ and is used to judge the extent to which the latent variables of the reparameterized distribution 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 [159]. Practically, the VAE is trained according to the following objective function:

$$\mathcal{L}_{\mathrm{VAE}_\theta} = \min[\|X - (f \circ g)X\|_{\mathrm{Er}} + \mathrm{KL}(P||Q)] \tag{2.111}$$

where $\mathrm{VAE}_\theta = \{\hat{\theta}_e, \hat{\theta}_d, \hat{\mu}_X, \hat{\sigma}_X, \hat{\mu_Z}, \hat{\sigma_Z}\}$.

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 [22]. In a denoising VAE, as proposed in [144], noise is added during the 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 [48]. Furthermore, a posterior collapse VAE is another version that can optimize degenerate local minima during learning [118]; in addition, the beta-VAE model has been developed to generate interpretable factorized representations and facilitate disentangled learning [122].

2.3.2.5 Diversified Autoencoders

The emergence of regularized AE variants has led to a flourishing of AE frameworks for different applications. Recursive AEs (RAEs) were proposed in [270] and extended in [269] 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. Similar work to (RAEs) was proposed for the natural language processing by Bowman in [45] 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 [139, 195]. Another variant, which is specialized for Boolean applications, is called a Boolean AE (BAE) [19]. Finally, complex-valued AEs (CvAEs) were proposed in [20] to encode data from the complex domain, involving both real and imaginary data forms.

2.3.3 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 by mining the original data distribution. Such models are utilized for several tasks, including image/ video synthesis, dimensionality reduction, recognition, and visualization [105].

Although EBMs (Section 2.3.1) and AEs (Section 2.3.2) 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 [153] and in their inability to leverage linear transformations. Conversely, adversarial learning models, which consist of two separate networks participating in the complete learning cycle, can overcome these challenges experienced by previous generative models based on NNs. The first network is called the generator, $\mathbf{G}$, and is able to generate new samples and noise. The second one is called the discriminator, $\mathbf{D}$, and is trained to distinguish between real and fake samples. In the following models, these two networks share a similar optimization:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathcal{V}(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{X \sim P_X(x)}[\log \mathbf{D}(x)] + \mathbb{E}_{Z \sim P_{Z(z)}}[\log(1 - \mathbf{D}(\mathbf{G}(z)))] \tag{2.112}$$

where, $\log \mathbf{D}(x)$ is the logarithm of the marginal probability that describes $\mathbf{D}$ 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 $\mathbf{D}$ for the generated data or distributions. The optimization function in Eqn. (2.112) is common among adversarial learning models and is built upon the concept of a min-max game for $\mathcal{V}(\mathbf{D}, \mathbf{G})$ to optimize the two opposing losses of $\mathbf{G}$ and $\mathbf{D}$. However, different variants of this optimization function can be found in the literature, which characterizes different models.

We will now describe the family of adversarial learning models, including standard GANs [105] in Section 2.3.3.1, adversarial autoencoders (AAEs) [193] in Section 2.3.3.2, conditional generative adversarial networks (CGANs) [200] in Section 2.3.3.3, and their relevant extensions.

2.3.3.1 Generative Adversarial Networks

GANs, which were introduced in [105], 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 $\mathbf{G}$, which generates noisy samples based on gradient optimization to supply the discriminator $\mathbf{D}$ with samples that are similar to previously misclassified samples; thus, $\mathbf{G}$ can be used as a tool to determine $\mathbf{D}$'s performance. The other block is the discriminator $\mathbf{D}$, which recognizes whether samples are drawn from the generator distribution $P_Z(z)$ or from the real data distribution $P_X(x)$ based on a probability measure. The model is trained through stochastic

backpropagation optimization and regularized via dropout [275]. Fig. 2.37 illustrates the graphical representation of a GAN model, where the model is introduced in the form of convolutional blocks; however, it can take other forms such as dense neurons (or fully connected layers) for specific datasets. A GAN is similar to a VAE [153] (see Section 2.37) in that it uses a similar propagation method for optimization; however, they differ in their optimization goals. A VAE performs optimization concerning the latent space, Z, whereas GAN performs optimization and differentiation based on the input space X.

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, $\mathbf{D}$, 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. (2.112). In Practice, $\mathbf{D}$ is trained to maximize the likelihood of the real data, and it is trained to assign the correct label for real and generated samples:

$$\mathcal{L}_{\mathbf{D}_{\hat{\theta}}} = \text{argmax}[\mathbb{E}_{X\sim P_X(x)}[\log \mathbf{D}(x)]] \tag{2.113}$$

whereas $\mathbf{G}$ is trained to minimize the following optimization objective:

$$\mathcal{L}_{\mathbf{G}_{\hat{\theta}}} = \text{argmin}[\mathbb{E}_{Z\sim P_{Z(z)}}[\log(1 - \mathbf{D}(\mathbf{G}(z)))] \tag{2.114}$$

considering that, if $\mathbf{G}$ provides poor samples (very noisy and far from x), then $\mathbf{D}$ distinguishes between real and generated sample with a high confidence level. Consequently, the loss that is presented in Eqn. (2.114) can be saturated. Alternatively, $\mathbf{G}$ is trained to maximize the following optimization objective:

$$\mathcal{L}_{\mathbf{G}_{\hat{\theta}}} = \text{argmax}[\mathbb{E}_{Z\sim P_Z(z)}[\log \mathbf{D}(\mathbf{G}(x))]] \tag{2.115}$$

Also, after the first pass is completed, the generator $\mathbf{G}$ samples the discriminator's distribution to adapt the variances of subsequent samples. Accordingly, $\mathbf{G}$ 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 $\mathbf{G}$:

$$P_Z(z) = P_X(x) \tag{2.116}$$

and the discriminator probability for both sides of Eqn. (2.112) is balanced:

$$\mathbf{D}(\mathbf{G}(z)) = 1/2 = \mathbf{D}(x) \tag{2.117}$$

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 non-convergence leading to instability [253]. Consequently, recent works have devoted attention to optimizing GAN learning. From the perspective of learning optimization, a bidirectional GAN was proposed in [77] considering the joint

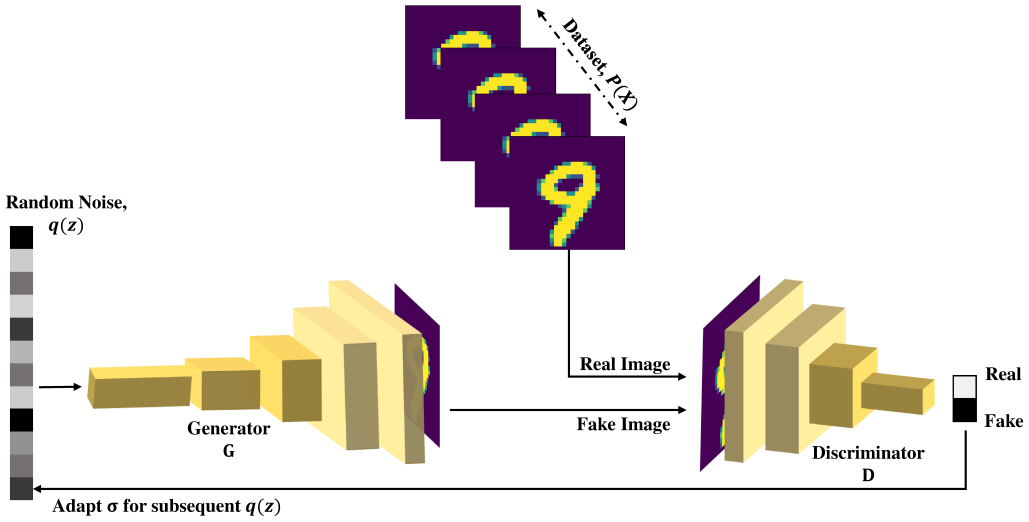


Figure 2.38: The graphical representation of a GAN model for image data, where $q(z)$ represents the noise distribution that is mapped to the real data distribution depending on the generation process, σ is the standard deviation of $q(z)$, "Real" and "Fake" are the discriminator score for real and generated data, respectively, and $P(X)$ represents the real data distribution which is utilized by the generator to adapt the generation of future samples by varying σ.

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 $Z = (z, h_z)$. Moreover, h_x is obtained by an encoder module $\mathbf{E}$ to map X to h_x; however, h_z is obtained from the output of $\mathbf{G}$. Thus, $\mathbf{D}$ can discriminate between X and Z based on the assigned joint distributions:

$$\min_{\mathbf{G},\mathbf{E}} \max_{\mathbf{D}} \mathcal{V}(\mathbf{D},\mathbf{E},\mathbf{G}) = \mathbb{E}_{X\sim P_X(x)} \underbrace{[\mathbb{E}_{h_x\sim \mathbf{E}(.|x)}[\log \mathbf{D}(h_x,x)]]}_{\log \mathbf{D}(x,h_x)} + \mathbb{E}_{Z\sim P_Z(z)} \underbrace{[\mathbb{E}_{h_z\sim z(.|z)}[\log(1-\mathbf{D}(h_z,z))]]}_{\log(1-\mathbf{D}(\mathbf{G}(h_z,z)} \tag{2.118}$$

To force $\mathbf{G}$ to produce samples that are close to the real boundary (the manifold of the real data), in a least-squares GAN, as introduced in [194], the sigmoid cross-entropy loss is replaced with a least square error, i.e., the log term is removed from the objective function, and the model is optimized in terms of two different objectives:

$$\min_{\mathbf{D}} \mathcal{V}(D) = \frac{1}{2}\mathbb{E}_{X\sim P_X(x)}[\mathbf{D}(x) - b]^2 + \frac{1}{2}\mathbb{E}_{Z\sim P_{Z(z)}}[\mathbf{D}(\mathbf{G}(z)) - a]^2 \tag{2.119}$$

$$\min_{\mathbf{G}} \mathcal{V}(G) = \frac{1}{2}\mathbb{E}_{Z\sim P_{Z(z)}}[\mathbf{D}(\mathbf{G}(z)) - c]^2 \tag{2.120}$$

where a and b are labels for the real and generated data, which introduce a special coding in a form of $a - b$ for D, and c reflects the level of that G wants D to believe about the generated data (in practice c is obtained as $c = b$, or $b - c = 1$).

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 (WGAN), proposed in [13] and improved in [111]. Moreover, in WGAN the discriminator does not actually classify examples, but it attempts to assign bigger output scores (or differences) for real examples than the generated ones. Accordingly, Eqn. 2.117 is not applicable as a threshold to reach the equilibrium, and the discriminator in WGAN is called "critic" that tries to maximize the difference between the real and generated data:

$$\mathcal{L}_{\mathbf{Critic}_{\hat{\theta}}} = \text{argmax}[\mathbb{E}_{X \sim P_X(x)}[\mathbf{D}(x)]] - [\mathbb{E}_{Z \sim P_{Z(z)}}[\mathbf{D}(\mathbf{G}(z))] \tag{2.121}$$

whereas G is trained to maximize the discriminator scores of its generated examples:

$$\mathcal{L}_{\mathbf{G}_{\hat{\theta}}} = \text{argmax}[Z \sim P_{Z(z)}[\mathbf{D}(\mathbf{G}(z)))]] \tag{2.122}$$

The boundary equilibrium GAN (BEGAN) proposed in [35] focuses on the error distribution rather than on directing matching the distributions of the real and generated samples. Also, in BEGAN the equilibrium is reached and controlled by a new hyperparameter $\gamma \in [0, 1]$:

$$\gamma = \frac{\mathbb{E}[\mathcal{L}(\mathbf{G}(z))]}{\mathbb{E}[\mathcal{L}(x)]} \tag{2.123}$$

where γ can control the diversity and quality of the generated data. Moreover, the BEGAN optimization objective is given as:

$$\begin{cases} \mathcal{L}_{\mathbf{D}_{\hat{\theta}}} = \mathcal{L}(x) - k_t.\mathcal{L}(\mathbf{G}(z_{\mathbf{D}})), & \text{for } \hat{\theta}_{\mathbf{D}} \\ \mathcal{L}_{\mathbf{G}_{\hat{\theta}}} = \mathcal{L}(\mathbf{G}(z_{\mathbf{G}})) & \text{for } \hat{\theta}_{\mathbf{G}} \\ k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L} - \mathcal{L}(\mathbf{G}(z_G))) & \text{for each trainings tep } t \end{cases} \tag{2.124}$$

where $k_t \in [0, 1]$.

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 [133].

2.3.3.2 Adversarial Autoencoders

VI is utilized in VAEs to approximate the intractable posterior [153]; similarly, inference can be performed using GANs to regularize AEs. AAEs were proposed in [193] 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 parameters θ_e to produce the latent distribution $q(z_i|x_i)$ and subsequently reconstructs the latent features in accordance with the posterior $P(x_i|z_i)$ and parameters θ_d. The

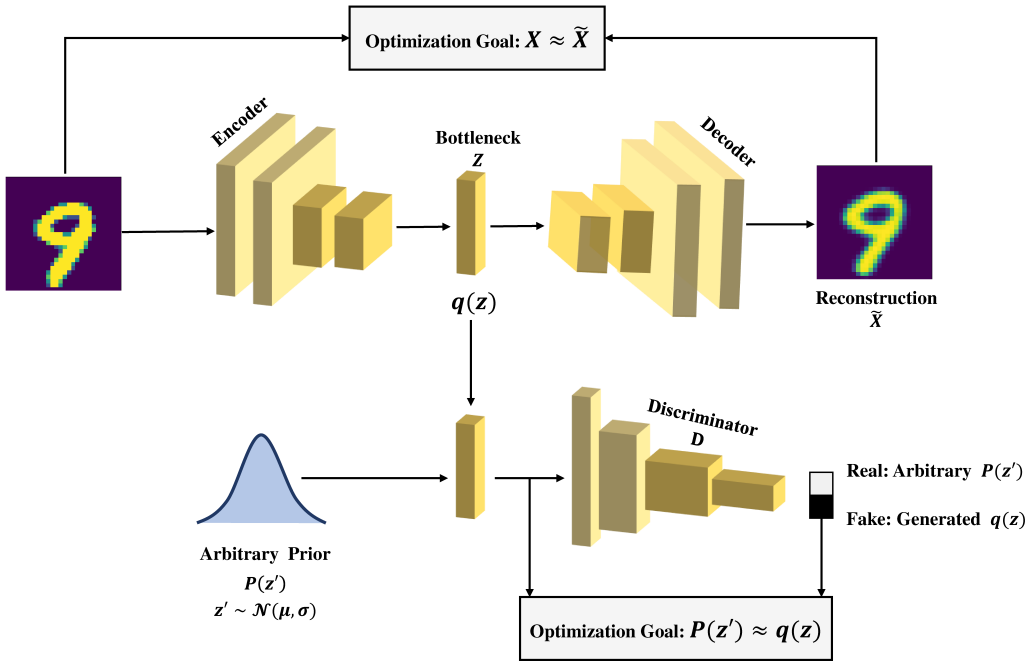


Figure 2.39: The graphical representation of an AAE model, where the "Encoder" module encodes data and acts as a generator **G**, and the discriminator is connected with both "Bottleneck" and the "Arbitrary Prior" modules.

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 arbitrary drawn prior $P(z')$, 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 $P(x)$.

Fig. 2.39 depicts the graphical representation of an AAE model, where the AE can be any model of which were introduced in Section 2.3.2. Moreover, in the figure, there are two optimization objectives, whereas the first objective is related to the AE reconstruction loss that is minimized by any metrics (e.g., MSE or reconstruction cross-entropy); the other optimization objective is the adversarial loss minimization in which the discriminator is enforced to distinguish between the samples from the arbitrary imposed prior, $P(z')$, and the samples from the bottleneck of the AE $q(z)$. The complete optimization objective of the AAE is given as:

$$\mathcal{L}_{\text{AAE}_{\hat{\theta}}} = \text{argmin}[\mathcal{L}_{\mathbf{G}} + \mathcal{L}_{\mathbf{D}} + \mathcal{L}_{\mathbf{AAE}}] \tag{2.125}$$

where $\mathcal{L}_{\text{G}}$ is the generator loss which is optimized as:

$$\mathcal{L}_{\mathbf{G}_{\hat{\theta}_{\text{G}}}} = \text{argmin}[-\frac{1}{m}\Sigma_{k=1}^{m} \log(\mathbf{D}(z))] \tag{2.126}$$

and $\mathcal{L}_{\text{D}}$ is the discriminator loss that is optimized as:

$$\mathcal{L}_{\mathbf{D}_{\hat{\theta}_{\text{D}}}} = \text{argmin}[-\frac{1}{m}\Sigma_{k=1}^{m} \log(\mathbf{D}(z')) + \log(1 - \mathbf{D}(\text{z}))] \tag{2.127}$$

where z' represents the sample that comes from the arbitrary imposed prior. Moreover, $\mathcal{L}_{\text{AE}}$ in Eqn. (2.125) is the loss that depends on the AE model (see Section 2.3.2).

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. (2.112). Conversely, the reconstruction loss obtained from $(P(x|z))$ has been replaced from the VAE with the adversarial loss that is taken from $\mathbf{D}$ of the GAN sub-module in [168]. Also, the authors have been combined the VAE and GAN models as a single model to guide the VAE by the learned representations at $\mathbf{D}$, i.e., the learned representations at $\mathbf{D}$ side are considered the basis for the VAE sub-module. The idea behind learning both VAE and GAN jointly in [168] 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 better data generations tasks, and better similarities measures in the data space. Consequently, the similarity metric can be taken from the $\mathbf{D}$ 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) [21], which consists of four sub-modules: *(i)* The Encoder that maps data to the latent space z, *(ii)* $\mathbf{G}$ to generate data by sampling from the learned distribution $P(x|z)$, *(iii)* $\mathbf{D}$ to distinguish between the real and generated samples, *(iv)* adversarial classification "categorizer" 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 pairwise feature matching and pixel-wise l_2 loss. The CVAE-GAN architecture followed in Zero-VAE-GAN model [91], 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 in 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, $\mathbf{L}_{\text{perc}}$, as a similarity metric (loss) for feature generation, by

employing an intermediate output between the discriminator and classification sub-modules, as:

$$\mathbf{L}_{\text{perc}} = ||f_D(x) - f_D(\tilde{x})||_2^2 + ||f_C(x) - f_C(\tilde{x})||_2^2 \tag{2.128}$$

where f_C and f_D are the activations of the last hidden layers of the categorizer and discriminator, respectively.

Finally, the literature includes many other works based on the integration of AEs with GANs; in [242], 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 [141, 144], Creswell *et al.* [69] 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, $q(z|\hat{x})$, with the predefined prior $q(z)$, the other involves matching the posterior of the reconstructed samples, $P(\tilde{x}|z)$, with the distribution of the arbitrary imposed prior $P(z')$.

2.3.3.3 Conditional 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 [201]. The CGAN model proposed in [200], 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 $\mathbf{D}$ and $\mathbf{G}$ to condition their learning as it is depicted in Fig. 2.40. In a CGAN, the class labels, $y_i \in Y$, are used for conditioning; therefore, such a model is classified as a supervised GAN that is optimized according to the following objective:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathcal{V}(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{X \sim P_X(x)}[\log \mathbf{D}(x|y)] + \mathbb{E}_{Z \sim P_{Z(z)}}[\log(1 - \mathbf{D}(\mathbf{G}(z|y)))] \tag{2.129}$$

Improvements to CGANs followed in [76] 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 that is considered an external linear image inverter (it produces new representations of images). At each level, the image is processed with the LAP operator to obtain a downsampled version of the image from the previous layer. Moreover, LAP also provide an upsampled version of the downsampled one, which is imposed as a label for the corresponding image, i.e., in the LAPGAN the label, y, is a new transformation of the image obtained from LAP by upsampling the downsampled image, and the model is trained in accordance to Eqn. (2.129). Accordingly, the noise can be reconstructed by the processed image versions and the real samples.

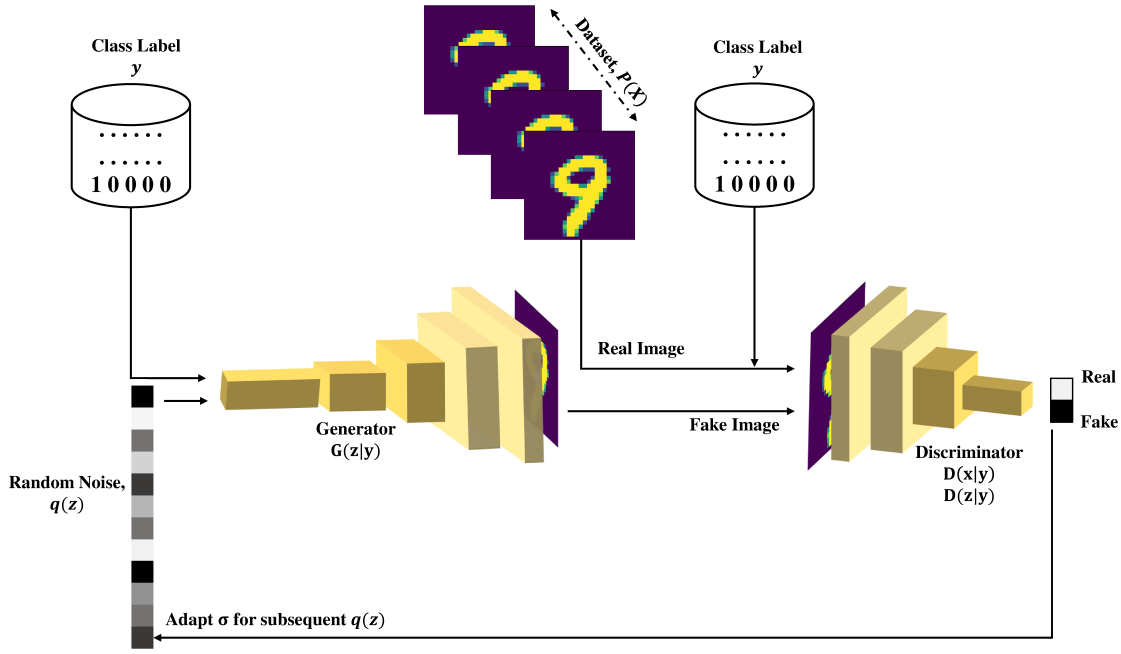


Figure 2.40: The graphical representation of a CGAN model, where the architecture is identical to the vanilla GAN model that is depicted in Fig. 2.38 except adding the class label module. Moreover, for each sample, the label y can be in a one-hot encoding vector, or it can be in another format as in semantic, metadata, or descriptive tags labels.

Further improvement of the CGAN followed in [145], 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 [145] the **G** has been built based on skip connection approach (or Residual learning [120]) and U-Net model [241], 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 [356] 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 full optimization objective function of the the Cycle-GAN is given as:

$$\mathcal{L}_{\text{Cycle-GAN}_{\hat{\theta}}} = \text{argmin}[\mathcal{L}_{\mathbf{G},\mathbf{D_Z}} + \mathcal{L}_{\mathbf{F},\mathbf{D_X}} + \lambda \mathcal{L}_{\mathbf{Cyc(G,F)}}] \tag{2.130}$$

where $\mathcal{L}_{\mathbf{G},\mathbf{D_Z}}$ and $\mathcal{L}_{\mathbf{F},\mathbf{D_X}}$ are optimized by using the min-max optimization objective

introduced in Eqn. (2.112); however, $\mathcal{L}_{\mathbf{Cyc}(\mathbf{G},\mathbf{F})}$ is optimized by utilizing the adversarial and L_1 norm losses:

$$\min_{\mathbf{G}} \max_{\mathbf{F}} \mathcal{V}(\mathbf{G},\mathbf{F}) = \mathbb{E}_{X\sim P_X(x)}[||\mathbf{F}(\mathbf{G}(x)-x)||_1] + \mathbb{E}_{Z\sim P_{Z(z)}}[||\mathbf{G}(\mathbf{F}(z)-z)||_1] \quad (2.131)$$

The Cycle-GAN losses have recently been utilized for image deraining in the DerainCycleGAN model, which was proposed in [312] as a model for single-image deraining (SID). The DerainCycleGAN 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 [314], 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.

Furthermore, the CGAN [200] and pix2pix [145] methodologies in conditioning the generation stage, by utilizing the residual learning [120] and Patch-GAN, have been followed in [185]. By introducing two GAN models that are termed as attribute-GAN and CA-GAN. Both models share similar goals in guiding $\mathbf{G}$ to the synthesis of images with specific attributes and different supervision conditions, by using multi-discriminators frameworks. The attribute-GAN comprises a generator $\mathbf{G}: \mathbb{R}^X \rightarrow \mathbb{R}^Z$, two discriminators including real/ fake $\mathbf{D}_c: \mathbb{R}^Z \rightarrow \{0,1\}$, and attribute $\mathbf{D}_{attri}: \mathbb{R}^Z \rightarrow \{1,.....,M\}$ or allocation $\mathbf{D}_{collc}: \mathbb{R}^X \times \mathbb{R}^Z \rightarrow \{0,1\}$ (one for attribute and one for allocation but not both are used). However, the CA-GAN contains a generator $\mathbf{G}: \mathbb{R}^{X+1} \rightarrow \mathbb{R}^Z$, three discriminators including categorical discriminator $\mathbf{D}_{cate}: \mathbb{R}^Z \rightarrow \{0,.....,c-1\}$, also similar attribute and collocation discriminators which are used in the attribute-GAN. Moreover, both GAN models in [185] 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 [307], 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 [83]; however, in pix2pixHD this approach is extended for the modeling of high-resolution images. The GAN loss that expressed in Eqn. (2.112) is improved in pix2pixHD by incorporating the least square error loss (LSE) to match the image features based on

the discriminator. Recently, the pix2pixHD has been further developed in [218] 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 were addressed also in [229], 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 [323].

The structure of the DCGAN has been exploited with the Resnet [120] and self-attention network architectures [338] in [92], 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 to employ the whole object to obtain a similarity score, instead, it exploits the advantage of the self-attention mechanism [92] 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 [229]. 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 terms of new metrics, e.g., the major deep perceptual similarity metrics (DeePSiM) proposed in [80] 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 concerning 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 [211], 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 [210] with different generator outputs by enforcing the additional diversity prior in the latent space.

2.3.4 Summary

The EBMs discussed in Section 2.3.1 including RBMs [6] in Section 2.3.1.1, DBNs [126] in Section 2.3.1.3 , and DBMs [252] in Section 2.3.1.4, use MCMC sampling

during the initial stages to sample the posterior parameters distribution, $P(Z|X)$ from the available data X. The core role of MCMC sampling is to estimate the posterior distribution when it is difficult to approximate directly; by sampling in each iteration in accordance with the acceptable probability and considering the statistics (μ and σ) over the densities, the distribution parameters that can describe the data can be obtained. The term "chain" refers to the idea that when the MCMC technique is applied, the results will be a deterministic sequence, and one can determine a future state from the previous state. We refer to [82], which describes how MCMC sampling can be used in practice. Consequently, after extensive sampling from the data to obtain the posterior distribution, the equilibrium distribution P_∞ (which identify the best model parameters, W and b, which fit the model to data) can be obtained and thus used for inference or in further estimation.

Moreover, the approximated distribution is paired with a contrastive divergence term, which is widely used to train EBMs. The high-dimensional data space is composed of a sequence of branch models called experts, which represent specific levels of the dimensions. The product of these experts can be represented by a feature vector of conditional probability, which constitutes the initial distribution $P_0(X|W^*)$, where W^* represents the conditional distribution parameters. Thus, the contrastive divergence is used to minimize the KL divergence between $P_0(X|W^*)$ and the equilibrium distribution $P_\infty(X|W)$. Then, after one iteration, the reconstruction process produces another distribution $P_1(X|W^1)$, and the computation minimizes the divergence between $P_1(X|W^1)$ and $P_\infty(X|W)$. The process continues to iterate in this way until the maximum likelihood parameters that may be close to equilibrium are estimated [124].

Various challenges accompany EBM learning, including the computation of the intractable partition function Z (see Eqn. (2.70)) that is used in estimating the joint probability between hidden and visible units and the greedy layer-by-layer pretraining of DBMs [252] and DBNs [6]. To overcome these challenges, the research community has been motivated to introduce additional models based on statistics and Bayesian networks. Among such models, the neural autoregressive distribution estimator (NADE) [167] model attempts to overcome the challenges in training RBMs by decomposing the observed prior distribution $P(v)$ via a factorized chain, where each state is conditioned on all previous states and the connections in each state v_i are correlated to speed up the approximations. Accordingly, the NADE model can be used in the case of missing data because it can reconstruct later components based on all previous states by arranging (ordering) the components over the visible vectors. NADE can be further improved by running it in k steps during learning instead of using a single inference step and conditioning the distribution using convolutional filtering; this method has been utilized for natural image generation based on the PixelCNN architecture [294].

Interested researchers will find that various AE models borrow a term from linear algebra, i.e., "overcomplete", which is used in reference to sparse AEs. If the data are encoded with losses and thereafter can still be reconstructed well with a minimum loss, then the representation is called overcomplete, i.e., the data are still complete after partial data removal. In practice, this is achieved by extracting several features

that are larger than the original data dimensions (as in SparseAE that enforces the bottleneck layer to be larger than the input and output layers of an AE), consequently enhancing the model regularity [240]. Moreover, the AE regularization term plays a major role in the decoding stage for achieving good generalizability. However, in some models, there is no decoding stage, although an encoding stage is included; such models typically resort to various expensive reconstruction approaches [29, 232]. To name one important model of this kind, the slow feature analysis (SFA) model, which aims to discover features that vary slowly over time, is used in image sequence applications and is able to measure temporal coherence [320]. Also, in MfL models (see Section 2.2), high-dimensional data are directly encoded employing k-neighborhood analysis.

The flourishing of AE models is attributed to the regularizer term, which is a component that is integrated into model optimization (objective function) to add special restrictions when carrying out learning scenarios [29]. Accordingly, the generalizability of the learned model can be enhanced for both the development and testing (or production) stages. In AE models, different forms of the regularizer have been considered, where the basic form comes from the AE's structure that has a bottleneck layer [130], which is obtained by restricting the size of the mapping (or middle) layer Z to be smaller than the input X or output $\tilde{X}$ layers to reduce the dimensionality and allow data to be compressed. Moreover, the bottleneck layer reflects the original AE purpose in compressing and mapping data samples $x_i \in \mathbb{R}^D$ through benefiting from both encoding $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$, $0 < d < D$ to obtained the mapped data $x_i = f(x_i; \hat{\theta_e})$, moreover, decoding $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$ stages to generate the original data as $\tilde{x}_i = g(z_i; \hat{\theta_d})$, by minimizing the reconstruction loss among data samples, see (Eqn. 2.97).

Another form of regularization is presented in the DAE [303] (see Section 2.3.2.2) and it can be implemented by adding a noise ϵ to the original data x thus obtaining the corrupted data $\hat{x}$, then encoding data samples $z_i = f(\hat{x}_i; \theta_e)$ followed by reconstructing the corrupted samples according to the original uncorrupted one $\tilde{x}_i = g(z_i; \theta_d)$. Additionally, the corruption or noise can be practically implemented by adding a Gaussian noise such that $\hat{x} = x + \epsilon$, $\epsilon \in \mathcal{N}(0, \sigma^2 I)$. The joint distribution between input sample x, corrupted one $\hat{x}$, and the reconstructed version $\tilde{x}$ is drawn by using a delta function $\delta_{f\theta_e(\hat{x})}(\tilde{x})$ (or a mass parameter) as $P(x, \hat{x}, \tilde{x}) = P(x)P(\hat{x}|x)\delta_{f\theta_e}(\tilde{x})$. Using $\delta_{f\theta_e(\hat{x})}$ enforces the decoding process from z to $\tilde{x}$ to be deterministic; whereas if $\delta_{f\theta_e(\hat{x})} \neq (\tilde{x})$, then δ puts mass $= 0$. Finally, the objective function is optimized considering the reconstruction cross-entropy $\mathcal{H}$ between the original data x and the reconstructed (after corruption) version in accordance with the Eqn. (2.103).

The deterministic decoding in the DAE that is obtained by $\delta_{f\theta_e}(\tilde{x})$ has been replaced by a contractive activations regularizer in the CAE model [240] (see Section 2.3.2.3). The CAE is considered another AE model that has been developed in terms of regularization, where it depends on adding penalty derived from the Jacobian $\mathcal{J}$ of the encoder activations (it can be also the weights) to the optimization objective function; thus letting the model decoder to learn the contractivity from the mapping space z to the original data space x. Practically, the regularization term can be implemented by considering the square of the Frobenius norm of the Jacobian $\mathcal{J}$ of the encoder acti-

vations, i.e, for sample $x_i \in \mathbb{R}^D$ $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$, the contraction regularizer is obtained by $||\mathcal{J}(\theta_e)||_F = \sqrt{\Sigma_{(ij)}(\frac{\partial f(x_i)}{\partial (x_i)})^2}$, and is integrated to the model learning in accordance with the Eqn. (2.105). Instead of the activations, the Jacobian of the weights of latent space z concerning the input weights can be considered a as contraction regularizer.

The Jacobian contraction is associated with what is called weights decay L_2 regularizer (the sum of the square of the weights matrix $\Sigma_{(ij)}{W_{ij}}^2$), where both are identical in the linear AEs that enforce the values of the weights to be small. Moreover, both DAE and CAE share a similar goal in adding a regularity to the learning process, but they are different twofold. *(i) The representations side*: in a DAE the representations are affected in both encoder and decoder sides by the amount of noise or corruption, and invariance considerations must be shared between both encoder and decoder sides, which leads to implicitly or indirectly induce the robustness of the representations. However, in a CAE the representations are affected only on the decoder side by the encoder activations or weights, thus leading to encourage the robustness explicitly among the representations. *(ii) The robustness method*: in a DAE the robustness of the representations is gained by explicit noise or corruption that is added to the data. However, in a CAE the robustness is gained by a tiny perturbation, which is added to the objective function from the Jacobian of the encoder weights.

Following the CAE in modifying and optimizing the objective function by novel regularization terms, a VAE [153] (see Section 2.3.2.4)) has been proposed based on a variational inference approach. In terms of structure, a VAE is similar to the DAE and CAE in containing the architecture to encoder and decoder components, see Fig. 2.37, Fig. 2.36, and Fig. 2.35. However, a VAE is different than the others in terms of learning methodology and using reconstruction metrics. Moreover, both DAE and VAE utilize the corruption and noise, ϵ, when carrying out the learning; wherein the DAE the noise is explicitly added to the input data and the reconstruction is carried out regarding the original clean data; but in the VAE, a tiny perturbation is added by varying the standard deviation, σ, of the approximated (or sampled) distribution in the latent space $Q(z|x)$. Furthermore, In the VAE, two losses are optimized to obtain a regularized trained model, where the first one is termed as a latent loss and is measured by the KL divergence for each reparameterized approximated posterior $q(z|x)$ and the prior $P(z)$: such loss 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 model accuracy in terms of the reconstruction loss as the average of the negative expectation of the log-likelihood.

The motivation behind developing VAEs and GANs (see Sections 2.3.2.4 and 2.3.3) is to enable the utilization of amortized variational inference (AVI), in which the aim is to indirectly estimate the variational parameters of the distributions of data samples. By imposing a parameterized function and passing the data through it, the past inference is exploited to support future inference, which is implemented by factorizing the approximated posterior distribution over the local variational parameters [336]. Accordingly, mapping the data or observations to the latent space to approximate the posterior distribution from the latent variables leads to better generalizability and plays

the role of regularization. We have found that in works that aim to optimize amortized inference learning, among the prevalent methods, InfoVAE, proposed in [353], uses the maximum mean discrepancy (MMD) as a metric for the divergence between distributions, thus alleviating the problem of uninformative latent code encountered in ordinary VAEs.

Amortized maximum a posteriori (MAP) inference was employed in [271] to estimate a loss function using an NN architecture to produce affine transformations of the original images to establish a focal subspace $q(z)$. Instead of measuring the similarity between the imposed prior distribution $P(z)$ and the posterior distribution $P(z|x)$ for each image, amortized inference assists in generating a function that is generally applicable for a set of images. Accordingly, high-resolution and less blurry images can be generated by matching the objective function to both distributions until it approximates the MAP distribution, resulting in the highest entropy concerning the distribution of the focal subspace $q(z)$. Moreover, inspired by [304], GAN models were stacked on top of each other in [150] to divide the whole image generation problem into a set of progressive targets; in this strategy, intermediate manifold representations of the data are utilized to achieve stability in generating high-resolution images.

The ladder variational AE (LVAE) was introduced in [272] to facilitate shared learning between the encoder and decoder, thus leveraging joint training of the generative and inference parameters (or the decoder and encoder parameters). Comparable to the LVAE, [77] suggested a similar principle for GAN learning based on forcing the discriminator to distinguish between the joint distribution of the real data and their latent parameters and that of the generated samples and their variables. In this regard, researchers being newly introduced to the field may doubt whether GANs are truly distinct from VAE models. Indeed, GANs and VAEs belong to the same family because they both fall under the umbrella of unsupervised generative learning models. However, VAEs cannot generate a new image or new data; they can only reconstruct the input data, for which they depend on minimizing the error between the reconstructed and real data and leveraging the reparameterization trick. On the other hand, GANs can generate new data samples by learning a complete model for mapping a given noise prior to the real data distribution, therein utilizing the discriminator to guide the generator to adapt its distribution. Furthermore, in VAEs, the inference is mandatory on the recognition (encoder) side, whereas in GANs, inference does not occur on the generator side. For more details on VI, the lower bound on the log-probability or log-likelihood, mean-field approximations, and the KL divergence, we refer to [41, 336] because these references introduce these terminologies in an ideal manner.

Recently, AE regularization has been remarkably boosted through the integration between AE and GAN in the same model (see Section 2.3.3.2), thus benefiting from the AE and adversarial losses to generalize a learned model among many different tasks. Such a combination is termed AAE, which has been proposed for dimensionality reduction, data visualization, clustering, and recognition [193]. Furthermore, AAE has been extended for high-resolution image generation and embedded learning with high-level visual features in [168]; it also has been utilized in a conditional fine-details

image generation in [21]. Moreover, the AAE has been utilized for zero-shot learning applications in [91], and it was used for image-to-image translation applications in [356]. Lately, it has been exploited in learning constant curvature of latent manifolds among data in [106, 107], and it has been utilized for spectral clustering with graph neural networks (GNN) (including AE and VAE) in [36, 266].

As witnessed from recent works, the training of very deep models (generative or discriminative, supervised or unsupervised) faces various challenges, including overfitting, gradient vanishing, and performance degradation. Such challenges impede the ability to learn powerful representations, thus hindering model generalization for real-life evaluations (i.e., the deployment of ML models for applications in daily life). Accordingly, various approaches have been introduced in the literature to overcome the aforementioned challenges [29, 229, 251, 333, 354], including *(i)* unsupervised greedy layerwise pretraining, *(ii)* intelligent weights initialization as in using orthogonal weights W, i.e., $W^T W = I$ [140], *(iii)* minibatch normalization and advanced pooling methods [36], *(iv)* dropout by deactivating some neurons in accordance with a predefined probability [275], *(v)* knowledge or domain transfer [257], and *(vi)* skip connections (residual learning). Residual learning by employing skip connections (as exemplified by the ResNet architecture) was introduced in [120] to overcome the model performance degradation issue in terms of reconstruction; in this approach, a deep model is divided into different blocks, which are gradually trained by adding layers that apply an identity mapping, $f(X)$, to copy early layers that extract shallow forms of the data, X, as it is depicted in Fig. 2.41 that presents the learning of residual blocks.

Instead of using each block of layers to fit a desired transformation or mapping, ResNet aims to explicitly allow the layers to fit a residual mapping by fusing with a copy of the original data (the input data to a residual block) in the form $f(X) + X$, where $f(X)$ is the output from convolution mapping and X is the data that feeds the residual block. To improve the learning process for residual blocks or stacks, various works have recently proposed modifications to the model structure and different learning procedures for the residual blocks. One notable work is the identity mapping and residual block analysis presented in [121], which addressed various architectural designs and considerations, e.g., considering dropout between blocks and using bottleneck information to overcome diminishing feature reuse (when only some blocks learn useful representations while others do not [277]) in deep ResNets. The concept of stochastic depth in deep ResNets was considered in [138] to overcome the problem of diminishing feature reuse by applying a special form of dropout between residual blocks, which is carried out in accordance with scalar weights assigned to each block. Moreover, the widening of deep residual networks can counteract the undesirable effects of deepening them [333]; hence, a novel approach for overcoming diminishing feature reuse is to increase the width of the convolutional filters used.

The above approaches for improving deep residual models have been integrated into modern architectures to improve their representation power and achieve large-scale generalization. The major models include the faster region-based convolutional neural network (Faster R-CNN) model, which was introduced for real-time object detection

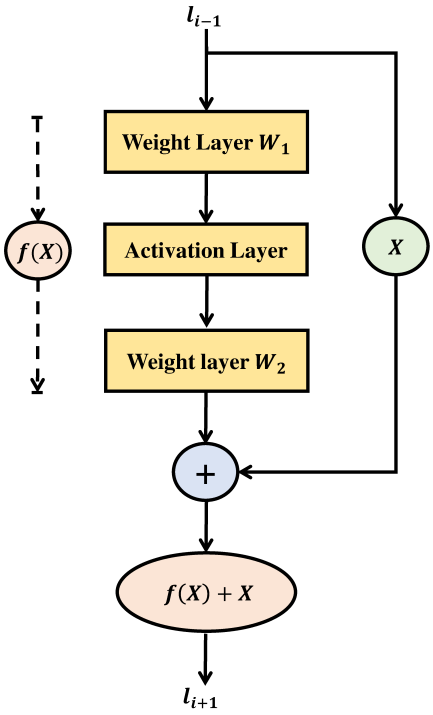


Figure 2.41: Residual block learning used in ResNet [120].

[237]; a large-scale, high-resolution image classification network called DenseNet [137]; the single shot multibox detector (SSD) for object detection [187]; an attention model for machine translation [300]; models for image segmentation [54, 119]; other models for large-scale, high-resolution image classification, including the Inception and Xception models [282]; the you only look once (YOLO) model for multitask object recognition [234]; a GAN-based unpaired image-to-image translation model [356]; GAN-based models for single-image deraining and rain generation [312, 314]; a multistream hybrid deraining model for single images [313]; and models for multiscale and high-resolution natural image learning [76, 83, 307], and multimodal image synthesis [218]. Moreover, these improvements have been utilized in state-of-the-art models for text recognition based on intensive and dense feature flows [346, 347, 348]. For further information, we refer the reader to [227] for additional application domains of recent deep models.

3

Research Challenges and Open Issues of UGL Models

Moving forward and considering the open issues and research directions, in the following, we categorize the challenges and room for improvement in three branches according to the literature analysis, as well as our belief of how can we coexist with the big data revolution. In the first branch, we will discuss how the BSS models (reviewed in Section 2.1) affect the explainable and interpretable ML models. In the second branch, we discuss how the MfL (recapped in Section 2.2) models can be used for dimensionality reduction, their effects on the learning process, and how they can act as an agent in noisy, missing, and corrupted data recreation. Finally, we end with open challenges for neural network models (highlighted in Section 2.3) on achieving reliability factors, and the generalizability to real-life data.

3.1 Explainable Artificial Intelligence and Blind Source Separation

Currently, multidimensional datasets, or high-order tensors, are abundant and can be amassed from different fields, including *(i)* high-frequency data from Industry 4.0, *(ii)* real-time process variations in finance and supply chain management, *(iii)* social network activities and interactions, *(iv)* large-scale engineering experiments and instrumental measurements, and *(v)* medical data and IoT flow. Such datasets require a conscious effort to be analyzed and interpret the causal relationships among the entries. Currently, supervised deep learning is used to analyze such datasets; however, this is challenging due to various factors. This requires prior knowledge "labeling", which results in high learning costs. Adversarial attacks that have recently come to light have also hindered the recognition performance, as in Deepfool proposed in [203]. Finally, there are effectiveness limitations due to the inability to interpret the results and explain the decisions [7].

On the other hand, UL models can reasonably interpret and explain the results of

various real-time applications, deliver clear outcomes to end-users, and overcome the challenges presented by growing dataset sizes. Currently, there is a strong trend toward explainable ML and artificial intelligence, XAI, principles to ensure that the decisions and actions of models can be explained and to provide a level of understanding of how they will behave in the future. Interpretable results can be obtained by developing or modifying unsupervised ML models that can meet the challenges of the big data era while having the ability to disentangle the representative factors that describe causality. Thus, understanding the causality relations among the relevant parameters and factors offers a better way to understand model decisions, and such analysis is usually accompanied by real evidence of what exactly drives a certain situation and on what basis a decision has been made. Unfortunately, however, there are currently no explicit ML models that can be fully and correctly explained while also producing reliable results for real-time applications in the big data context.

Accordingly, we provided a brief recap of the unsupervised representation learning methods that belong to the BSS family (Section 2.1), which can decompose data and determine the factors that are linearly independent by identifying which factors are derived from the other factors. Each of the BSS methods has its potential; however, the methods that are based on restrictions when decomposing the datasets can offer the best results, where that restriction is considered as a conditional prior. Thus, there is the ability to utilize that prior to reconstructing the data for delivering certain results while interpreting the factors' causality. For instance, in the BSS methods, NMF (Section 2.1.2.2) decomposes the data into two matrices, on which it imposes strict additivity of the positive data components, which are parts of the original data. Exploiting such a technique during data decomposition leads to different tensorial factors that are forced to be linear and positive, also by which the different levels of the hidden structures of the datasets can be extracted and interpreted in an explainable manner. Moreover, SVD (Section 2.1.2.1) with the nuclear norm, $\Sigma_i \sigma_i$, regularization utilized to win the Netflix challenge ($1M Prize) [24] was able to generate a series of linear models; it was employed in the Netflix challenge to improve the accuracy by 10% over the original recommender algorithm. SVD restricts the values to be semi-definite positive (eigenvalues are positives or zero) and is considered as a method that will continue to be utilized in building real-time explainable models; it is also especially capable of giving the dataset's rank simply. Additionally, what is important for high-order decomposition is to combine factorization methods to improve the accuracy and reduce the computational complexity. Recently, many projects have been launched to build interpretable ML models, including explainable artificial intelligence (XAI)[1] and BIGMATH[2], which target heterogeneous real-time dataflows. In the following, we discuss the research directions toward explainable ML models, for various tasks:

- Investigating deeply optimization algorithms (by modifying old algorithms and proposing new algorithms) specialized in dataset decomposition and reconstruction and also proposing algorithms to identify the optimal rank of the datasets

[1] https://www.darpa.mil/program/explainable-artificial-intelligence
[2] http://itn-bigmath.unimi.it/about/

for describing which features can be linearly separable. The decomposition is thus carried out based on that optimal rank, and the interpretability and model learning can be carried out according to the reduced data employing lower rank [1].

- Building models that can interpret themselves by disentangling the related factors from the data, understanding the causality interactions, identifying how each factor can affect the other factors, and considering the logic and reasoning among the factors. Furthermore, reliable models that are able to mimic real-time data flows should be constructed.
- Extracting new rich representations utilizing combinations of BSS models to identify the separable features flexibly, as in tensor decomposition integration with the SVD, R-SVD, G-SVD, NMF, and TD methods. Moreover, new algorithms and error metrics should be developed to restrict the data decompositions in a standard manner.
- Introducing algorithms that can address noisy and attenuated data, therein being achieved by integrating different data exploration tools (multiple feature detectors and cross representations), depending on the multi-scale decompositions and engagement with ML methods that are able to build the general joint mapping space for the data.
- Online learning from dynamical or temporal data as in the NLP applications needs more investigations, specifically, for topic modeling in different domains including financial, commercial, medical or pharmaceutical, and bank-data; in these domains data and topics are changing from time to time according to different rules, thus adaptive and interpretable topic modeling models must be considered in the future work [236].
- Document understanding and topic modeling are branches of the NLP, and they are also characterized by the prior probabilities and the way we infer the model when they are modeled by PBMs (see Section 2.1.3); however, different considerations must be addressed in future works including, besides the hierarchical model learning, cross representations (shared representations) for a topic when it belongs to more than one document or a document with different sub-topics. Also, selecting the optimal models' parameters such as the prior, and inference methodologies must be boosted by research to overcome computational complexity [151].
- Modern deep learning-based models (see Section 2.3) for topic modeling and word embedding face challenges in embedding out-of-vocabulary words, i.e., scaling up the model for different languages by benefiting from different pre-trained models, and other challenges arising from sparse and noisy data must be addressed [198].

Currently, the available data are often inherently heterogeneous and may include defects such as missing data, messy data, or noisy and unlabeled data, thereby hindering

the development of reliable and explainable models. The conscientious development of suitable UGL models can allow these issues to be overcome due to the potential to capture better representations [227]. Future research regarding these issues is expected to follow two directions. The first direction is to build explainable models that offer self-interpretability and an understanding of causality; the second direction is to advance the capabilities of cross-modality structure representations to complementarily exploit the latent variables in heterogeneous data. Recently, attempts have been made to adapt the learning strategies for deep learning networks to follow the cross-modality approach. To this end, the matrix capsules (capsule networks) introduced in [249] attempt to represent each feature point with respect to several poses (different activations for each neuron to represent each data point from multiple views); thus, each piece of a feature is represented by a 4×4 matrix to overcome the limitations of CNNs in addressing messy data. Ultimately, applying BSS methods in combination with modern ML architectures will facilitate advancement beyond black-box implementations toward white-box modeling. Accordingly, interpretable, self-configurable, adaptable, multimodal, and trustworthy models are foreseen as the next generation of ML models.

3.2 Manifold Learning for Data Visualization and Unsupervised Pretraining

The MfL methods highlighted in Section 2.2, including both global (Section 2.2.1) and local (Section 2.2.2) techniques, play a major role in discovering the hidden structures in massive datasets by unfolding the underlying manifolds describing the intrinsic structures of the data in a lower-dimensional space. Moreover, such methods have been developed from different perspectives on analyzing the geometrical structure of a manifold, e.g., *(i)* the spatial distance view [244], in which the manifold geometry is estimated based on optimization of the distances between data points and their neighbors in both the high- and low-dimensional spaces; *(ii)* the probabilistic view [30], in which an attempt is made to analyze the manifold based on probabilistic distributions and to optimize the similarity between densities; *(iii)* the clustering view [23], in which each related data point is accommodated in a shared region, as described in the context of the graph cut technique in Section 2.2.2.2; and *(iv)* the hybrid view [296], in which the distance, probabilistic density, and clustering views are all considered in different stages, as in t-SNE, which is described in Section 2.2.2.4. From these perspectives, MfL methods can be introduced into ML models to facilitate UL by constructing joint mapping data (similar data will have a similar joint mapping ϕ); such spaces can eliminate redundancies, offer a consistent basis for learning, and represent focal spaces for messy data, in which learning or data reconstruction can be implemented to achieve a reduction in complexity and an increase in normality. Consequently, even if new data suffer from attenuation, they can be recreated and visualized from the manifold joint space. Areas for future improvement are listed as follows:

- Combining the BSS methods and MfL methods is required to investigate it deeply

in future works; such a combination leads to affordable learning from massive datasets, and it reduces the dimensionality among data samples that leads to lessening the computational complexity of the proposed models [2].

- Creating manifold agents or tools that can reconstruct attenuated data and reshape messy data when included as major components of an analysis system.
- Overcoming the current lack of cross-modality manifold representations. At present, it is important to build multiple agents that share multiple representations of the same data; as an analogy, one can think of such agents as the different parts of an octopus's brain (an octopus has 9 brains, each of which addresses a different task based on different representations). Such investigations can offer improved consistency for complex applications and supply practitioners with multiple views of very large-scale data.
- Developing new optimization approaches for recovering and reconstructing the original data from the joint mapping space of the manifold, considering that the joint space containing the main factors or point cloud may become corrupted or noisy under any circumstances.
- The recent manifold learning models have been introduced in the context of graph neural networks and GANs with a higher level of accuracies in terms of clustering, dimensionality reduction, visualization, and recognition [107, 160, 230, 265]. However, many challenges and open issues still need more investigations to mimic real-life conditions, including learning scenarios when the manifold contains holes; considerations for temporal data that is characterized by dynamical manifolds (stationary, point cloud, or k-neighbors points changing over time) as in the physical, brain, biological, and sensor-networks data. Lastly, boundary bias and vulnerability to noise issues must be addressed in future works [290].
- Constructing measures for relating manifold agents to online or real-time data, which could be achieved by modeling and considering the quality of each embedding (every subspace or reconstructed data point), while also considering the challenges mentioned in Section 3.1.

Recently, agent learning has received attention in the contexts of reinforcement learning (RL) [149] and GANs [105] which introduced in Section 2.3.3. In RL, an agent is conceived of as a component that performs an action on its environment after interpreting it via outsourcing. By contrast, a GAN consists of two agents: the discriminator and the generator. Thus, MfL can support ML models through integration as an additional agent that can transform data into a finer space, which can serve as a reference space for corrupted observations. Combining MfL methods with other models increases the chance that a broader space can be discovered from the hidden structures buried in the data and enables the determination of more compact representations [2]; thus, the number of free parameters (learning parameters, such as weights and biases) can be reduced, leading to lower computational complexity. Currently, GANs and VAEs are the

main focus of attention in the research community. Joint mapping approximations have been applied in various works to improve GAN and VAE learning; however, MfL has not been fully utilized. Additionally, such models have achieved competitive results in generating data; however, they still require further investigation [150, 194]. GAN models still have difficulty generating natural images at small scales or high resolutions due to instability [13] and data manifold complexity; however, integrating manifold learning methods into GANs, or any generative models, can help to generate image features that are reliable and have better representations. Furthermore, improving MfL for big data exploration offers the ability to forcibly characterize the noisy and messy data with a joint scale; consequently, the variability of new data can be overcome.

In the following subsection, we discuss in detail the opportunities for the improvement of NN models, including GANs, regarding the quality of representation learning.

3.3 Neural Networks and Reliable Models

In contrast to supervised deep learning, which has been extensively investigated and addressed in many works, few investigations have addressed the integration of UL with similar modern architectures. However, innovative ML models are needed to solve real-time problems utilizing rich unsupervised representations [354]. The goal is to improve methodologies for data exploration by introducing trustworthy representations and following design principles similar to those of recent learning architectures [105, 153] (see Section 2.3.2.4, and 2.3.3.1). From this perspective, one of the approaches used to improve data representations in learning scenarios is called sparsity regularization [232]. This approach is applied in AE learning by penalizing the activations of the hidden units to be near 0, or by restricting that the additive biases should be negative [29]. Nevertheless, a comprehensive analysis of sparse penalization has yet to be performed, and the advantages and disadvantages of each based technique have not been systematically identified, although different works have recently taken a short step in that direction [340]. Thus, a comprehensive analysis must be performed alongside empirical investigation to compare all possible ways to apply penalization concerning sparsity while also considering diverse probabilistic density metrics.

Recently, unsupervised representation learning has greatly shifted toward using variational Bayesian inference to improve the regularization performance. Although many models developed under this dependency are VAE extensions [153] (see Section 2.3.2.4), many open issues still need to be addressed. We noticed many tools to calculate the similarity of two probabilistic distributions beyond the KL divergence, such as the α, β, f, and J-S divergence [63], which are usually used in optimizing the objective function. However, numerous such similarity measurements did not fully perceive the variational distribution learning; moreover, there is no work in the literature comparing the superiority of each method. Accordingly, additional investigations of the distribution similarity metrics comparing and introducing them in the context of additive noise and outliers will facilitate future researchers in picking the best-trusted metric. Also, introducing additional tools and methods from the Bayesian statistics field, such as

uncertainty quantification to be utilized in unsupervised learning, will open the door to additional progress, especially for model physical interpretation and explanation, where such statistics are inherent in data representations in unsupervised techniques. Recent works that have targeted the uncertainty in deep learning proposed in [90] are based on dropout; however, more investigations are required, including those into causality interpretation, outliers, and other types of regularization.

The current trend in learning representations also concerns the field of quantum mechanics (QM), which can explain how processes interact in the real world. Hence, the tendency to build reliable models has been facilitated by the application of complex domain analysis for reducing the computational cost and for interpreting the deep architecture interactions [292]. Notwithstanding, attention has been recently paid to supervised learning models that can obtain very competitive results: such models are utilized to optimize the learning parameters in RNNs [70], to optimize the recurrence matrix in the unitary recurrent neural network (uRNN) [319], in neuron tuning and synchrony [235], and in image classification by CNNs [225], among others. Thus, the analysis of real-time applications, including Industry 4.0, IoT, sequential modeling, self-driving automobiles, speech and image, and sensor networks, based on unsupervised learning utilizing rich representations of the complex domain is expected to play a role in the future. Unfortunately, the literature evidences that little attention is given to utilizing complex values for unsupervised learning, and it's in its infancy. However, recent works have taken a small step following this trend by introducing the theoretical foundation for complex-valued AEs [20]. The following is a list of the facts and issues that strengthen future considerations for complex domain representations:

- Most recent works performed evaluations based on private datasets [170] utilizing matrix operations to obtain the imaginary values, therein lacking any generalization to real-time or real-world applications. Thus, real and complex-valued datasets have yet to be generated.
- The interactions between the real and imaginary parts have yet to be considered. How can one affect the other? The rich properties, including addition, multiplication, and division, are marginalized; however, they can bring about different advancements for UL representations.
- The exploration of optimization algorithms in the complex domain remains imperfect and is required for developing theoretical and practical foundations.
- In recent works proposed for complex domain representation, stacks of differential equations dominate such researches; future works must be delivered in a simple, technical, understandable, and comprehensive manner to stimulate the community to investigate in such a domain.

In addition to complex-valued representations, Boolean and quantum logic (QL) representations are considered as additional tools that can assist in understanding process reasoning and in considering various levels of logical abstraction. Additionally,

such interpretations can be utilized in logic-based applications such as cryptography, telecommunications, IoT technology, and cell biology, for which such representations have not been a focus in recent works, although the relevant theoretical foundations were proposed in [19] to facilitate an understanding of the concept of BAEs. Nevertheless, intelligent environments generally have a Boolean nature, therefore, we expect that in forthcoming studies, unsupervised Boolean representation learning and reasoning will be of interest.

State-of-the-art unsupervised representation learning models based on deep generative modeling are also known as GANs (see 2.3.3.1). GANs are used for many different tasks, including reconstruction, image synthesis, and visualization as well as nefarious tasks such as security spoofing. Adversarial learning based on the min-max concept offers an enormous degree of flexibility in interrepresentations that can be utilized in transition modeling (changing from one state to another) and for building a level of understanding of how a given noise prior is mapped to the original data. Consequently, transition spaces can be processed to extract latent factors, which can assist in generating new data based on reduced factors. At present, GANs and VAEs continue to be extensively studied, although most works dedicated to specific tasks neglect the learning of the data generation process (or reconstruction in the context of VAEs) itself when attempting to achieve superior performance [145].

Many recent works on building GANs have failed to offer clear insight into how generators and discriminators work logically or which representations best enable rapid progress toward equilibrium. Specifically, when a model consists of multiscale representations, it is important to visualize what the GAN is able to learn and to determine the conditions necessary to achieve balanced learning. Moreover, the tendency of recent works to neglect generalizability has caused the proposed models to be scattered; accordingly, in future works, it will be important to consider the construction of generalized models that can be adapted for different situations while also proving their ability to yield reliable evaluations [35, 229]. Additionally, robust frameworks for real-time processing are lacking; although different architectures are available, it will be important to build competitive frameworks that utilize data augmentation and data factorization and consider missing data for effective use in real-world applications. Below, we summarize the current opportunities for improvement according to our literature analysis:

- The online explainability of the VAEs and GANs behaviors rather than building an offline visual explainability (as in the heat mapping or class activation map (CAM) [355]) is lacking in the literature; such an explainability offers the ability to get an insight into the convergence of the model parameters, and it can improve the level of accuracy and generalizability.

- The building of a joint mapping generation for generalization purposes that acts as a manifold generation agent must be investigated (see Section 2.2). Different works have attempted to impose the prior in the latent space; however, their prior did not generalize well [211].

- Generating the micro-scale geometrics remains in its infancy and requires more effort to modify the Euclidean measurements in shifting to the geodesic analysis. Thus, the model can be optimized based on the geodesic distance rather than the Euclidean distance, and future works can use simple geometric perception rather than classical deep representations.

- LAPGANs [76] are a promising approach that considers multi-scale and multi-resolution transforms in the generating stage; however, modern multi-scale techniques, such as contourlets and curvelets [84], have not been discussed recently. Extending such models with multi-scale techniques helps to achieve stability and confirms the potential of such models.

- The utilization of the error distribution matching instead of directly matching the data was recently proposed in [35]. In addition, margin adaptation was addressed in [306] to enhance the learning; however, both methods were not fully investigated and need to be confirmed.

- Modifying CGANs (see Section 2.3.3.3) by designing a new objective function to generate any stochastic object, by exploiting the full conditional distribution entropy, has not been investigated [200].

- With the explosion in the number of VAE and GAN improvements and extensions, it is important to build general guidelines and understandable ISO metrics, when evaluating the results [150].

- The aim of SAASLN [92] is to measure the relevance or similarity between the objects belong to each category or concept in an explainable manner, but measuring how tight is the object and proposing a novel method to connect object still an open research direction.

- Similarity-based learning GAN models have been introduced to measure the quality of the generated data and can scale up generative models for large-scale and complex data. However, such models are in their infancy stages and more investigations are required in terms of explainable metrics and complex data structures [92, 168].

- Combining GANs and VAEs into the same model faces challenges in vanishing gradient because different images that are taken from the VAE output become blurry. Such an issue can be fixed by replacing the cross-entropy loss from the generator side with l_2 distance or utilizing mean feature matching and perceptual reconstruction loss [91]. However, it is required to be confirmed when the data are scarce, in case of unbalanced datasets, or when generating out-of-distribution data (from a different domain).

- Multi-discriminator frameworks have been recently proposed to enhance data generation, but no work or confirmation has been proposed to show the model

complexity (computational or learning) when adding further discriminator modules, and to which extent the model will be able to generate images with more delicate qualities. For instance, both attribute-GANs and CA-GAN [185] are considered recent models for cross-domain learning and clothes fitting; however, such models consider extra discriminators in their architectures, but no guarantee has been given that adding extra discriminators will reach stability at the generator side. Moreover, the proposed models in [185] have been evaluated in the clothes (fashion) dataset, but additional evaluations for other life applications such as cosmetics, shoes, accessories, and jewelry fitting are required.

- Graph-based learning GAN models have been introduced recently to learn from a graph data and Riemannian manifold; however, the experimental investigations among different works did not introduce performance analysis for a real-life graph data including engineering or technical drawing maps from architecture and mechanical fields, geographical data, and other medical applications as in electroencephalography data [36, 106, 107, 266].

- An impressive challenge that requires deep study by the research community is: How can we generate (or learn) high-resolution data or images when the original samples are corrupted [111]?

The demand for rigorous, standardized UGL learning methods is strong because such methods will lead to reproducible research and offer the ability to standardize models in a sophisticated manner, thereby accelerating evaluations [12]. Generalized models can be used in different real-world domains; in particular, 3D applications are of daily relevance in various areas, and learning suitable representations for related domains is consequently an important task that can be facilitated using adversarial learning. For this purpose, 3D GANs [323] are promising models whose generation ability has already been demonstrated; however, a confirmation of the reliability of recent models is missing and should be achieved through clear evaluation metrics and learning strategies. For instance, Wu *et al.* [323] proposed an adaptive learning strategy for 3D GANs in which the discriminator is updated if its accuracy was no higher than 80% on the previous batch; this strategy may lead to instability and poor generalizability to different applications because the discriminator and generator are not updated simultaneously.

Finally, it is also necessary to consider scenarios in which GANs are employed by malicious actors aiming to attack security systems, e.g., in cyber or biometric attacks, in which the GANs learn to mimic identities for spoofing purposes. A deep network attack procedure was recently proposed in [280] for hindering the ability of deep learning models to recognize images based on adversarial learning by adding very small perturbations (at the single-pixel level) in a manner that depends on differential evolution optimization. Currently, there are no clear countermeasures against soft adversarial perturbations, and deep models are unable to address such attacks. Hence, it is important to build unified rules to protect critical systems from such spoofing attacks.

3.4 Visual XAI Attention Mapping and Deep Neural Networks

Nowadays, DL models are improved by uncovering the underlying parameters that perform the functional operation of models' layers; such functions are utilized to perform specific data mapping at each layer including convolutional, dense, or rectifying functions. To show how models learn data, the visual explainability mapping methods (XAI attention) are specially studied, due to their ability in visualizing the intrinsic structure of the learned parameters. Moreover, XAI attention methods translate the decision and learning convergence to the models' developers in an ordered manner, where such methods can be generalized regardless of if the data lies in the Euclidean space (unstructured images or videos), manifold and graph space (unstructured graph), verbal space (text and speech), or if the data lies in a combined space (multimedia).

The visual saliency mapping (SM) approach has been introduced in [267], where it introduces visual-offline explainability through gradient heat mapping (HM), i.e., it estimates the gradient between the input, x, and output, y, and it considers the result as a visual explainability. For each class c_i, the method differentiates the output, y, concerning input, x, as $\frac{\partial y_{c_i}}{\partial x}$. Thereafter, the differentiation result is passed to a RELU activation function to eliminate the effect of negative values, and thus keeping the positive values that reflect the maximum positive rate of change, ∂, as follows:

$$\mathrm{HM}_{\mathrm{SM}} = \mathrm{RELU}(\frac{\partial y_{c_i}}{\partial x}) \tag{3.1}$$

Moreover, the method produces a heat map for each input considering the norm of the gradient; by taking into account the norm over the gradient of the input,x variables, concerning the output,y, reflects their importance:

$$\mathrm{HM}_{\mathrm{SM}} = \mathrm{RELU}||(\frac{\partial y_{c_i}}{\partial x})|| \tag{3.2}$$

Fig. 3.1 shows the ability of the gradient saliency mapping method to produce a visual explanation when considering for the image classification task.

As it is noticed from Fig. 3.1 , the visual explanation of the saliency mapping method represents noise rather than transformed signals, and the budding attention does not reflect a real explanation concerning the input image. To overcome such challenges, the class activation mapping (CAM) method has been introduced in [355], where it is built based on the parameters' importance through introducing weighting factors α. Moreover, CAM restricts the penultimate layer (the layer before the output softmax layer) to be convolutional, and it estimates a weighting factor, α_k, for each filter channel, $f_k \in \mathbb{R}^{(u \times v)}$, from that layer as follows:

$$\alpha_k = \frac{1}{Z} \Sigma_i \Sigma_j f_k \tag{3.3}$$

where α_k is the global average pooling of the parameters constituting the convolutional channel k.

Figure 3.1: Visual SM attentions in the image classification task, where the top images are the input, and the bottom (black and white) are the result taken from the gradient saliency mapping [267].

Accordingly, CAM offers the visual explainability through the summation over the filters channels' parameters constituting the penultimate layer that is scaled by α_k as follows:

$$\text{HM}_{\text{CAM}} = \text{RELU}(\Sigma_k \alpha_k f_k) \tag{3.4}$$

Furthermore, the CAM method achieved a better visual explanation than the gradient saliency mapping approach in the literature, where both methods have been compared in the context of image recognition applications; however, both methods do not work properly in the very deep neural networks when the penultimate layer is a dense layer, i.e., not convolutional. Hence, a novel method termed gradient-weighted class activation mapping (Grad-CAM) has been introduced in [259]. Moreover, Grad-CAM relaxes the restriction of the penultimate layer by computing the gradient between the output layer, y, and the last convolutional layer to estimate the weighting factors, α_k, which are obtained through the global average pooling among the grading concerning each filter channel k, i.e., the number of α_k is a function of the depth of the convolutional layer:

$$\alpha_k = \frac{1}{Z} \Sigma_i \Sigma_j \frac{\partial y_c}{\partial f_k} \tag{3.5}$$

where ∂ represents the gradient which measure the rate of change between the output and the last convolutional layer. The final XAI attention map is obtained as follows:

$$\text{HM}_{\text{Grad}-\text{CAM}} = \text{RELU}(\Sigma_k \alpha_k f_k) \tag{3.6}$$

The Grad-CAM method is employed in many different image and text recognition applications including image captioning, and visual question answering tasks; notwithstanding, it suffers from evidence collapse issue that is caused by the inter-class data correlation, i.e., it shows similar XAI attentions for close classes even they are different. Accordingly, the excitation backpropagation (EB) method has been introduced in [339]

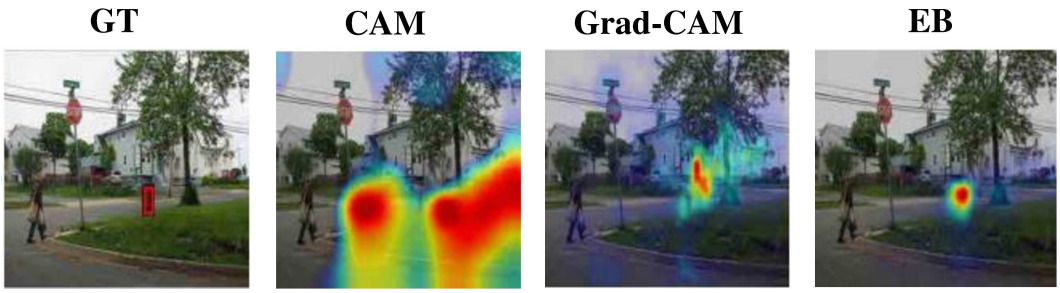


Figure 3.2: Visual XAI attention mapping of a fire hydrant image detection, where GT represents the ground truth input image, and CAM, Grad-CAM, and EB have been introduced in [355], [259], and [339], respectively.

based on factorizing the joint probability of the consequential neurons. Moreover, EB builds XAI attention maps by defining a relative inference between pairwise neurons, i, j, as follows:

$$P(i) = P(i|j)P(j) \tag{3.7}$$

where i is a neuron lies in the layer L_{l-1} and j is a neuron in the layer L_l. Also, the joint distribution is factorized as follows:

$$P(i|j) = \begin{cases} Z_i i w_{i,j} & \text{if} w_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where Z_i is the partition function that average the probability among all possible configurations:

$$Z_i = \frac{1}{(\Sigma_{j:w_{i,j} \geq 0}) a_i w_{i,j}} \tag{3.9}$$

Fig. 3.2 shows the ability of the CAM, Grad-CAM, and EM methods to draw an offline XAI attentions for image recognition task. As it is shown in Fig. 3.2, the EM method outperforms all other related methods on drawing the normalized XAI attention maps based on the localized region of interest (ROI) areas. Moreover, all the above methods offer offline XAI attentions that are represented by the heat maps and based on weighting the activations or gradients by global average pooling of the convolutional filter blocks. The following is a list of open issues that for future improvements:

- Introducing cross XAI approaches has not been noticed in the related works, where further investigations are required to extract multiple explainability from the model's representations as in offering visual, contextual, and function explainability in the training and testing stages.
- Proposing techniques for visual-online and contextual explainability is missing and has not been covered by the literature, where such methods can show the

temporal behavior of each model. Such online explainability can help in improving the overall model's accuracy.

- Considering XAI losses has not been mentioned in the literature, where such losses assist in modifying the current optimization algorithms used to learn the deep NN models. That is by constructing hybrid optimization losses functions, thus the model can be generalized for large-scale data or applications.

4

UGL Models for dimensionality reduction and XAI

This chapter presents the proposed investigations and the experimental analysis of the recent UGL models including NMF (reviewed in Section 2.1.2.2), t-SNE (reviewed in Section 2.2.2.4), AE (reviewed in Section 2.3.2.1), and VAE (reviewed in Section 2.3.2.4). Such methods are utilized in dimensionality reduction, visualization, and the visual explainability of DL models. Also, the proposed investigations of the selected models can be employed individually to carry out one ML task as in employing β-NMF for dimensionality reduction; alternatively, the investigations can be combined to reduce the computational complexity and accelerate learning from compact representations as in employing t-SNE on the factorized β-NMF subspaces to visualize the manifold of data. The structure of the following chapter as follows.

Section 4.1 introduces our proposed nonnegative rank approximation that is imposed, as a prior, to identify the dimensions of the compact (reduced) subspaces to carry out the β-NMF factorization. Based on algebraic transformations among data matrices, the proposed rank can be approximated and it is generalized from small to large-sized images. Thereafter, β-NMF and the proposed rank method are combined with a shallow AE model to learn from limited available data in Section 4.2; such data comprise fewer samples for the training than the testing. Consequently, combining β-NMF and AE can be utilized for a better generalization during the testing (or production) stage, it also reduces the learning complexity among data classes because of the compact subspaces that are obtained from the β-NMF. Moreover, such a combination can be used in data clustering and manifold visualization applications.

Section 4.3 presents a novel method to explain the VAE models based on gradient-derivative attention that exploits the first derivative of the gradient. Specifically, the derivative of gradient between the latent space layer, Z, and the encoder's layers L_{e_i}, is utilized to build an explainable attention map that shows how the VAE behaves according to different inputs. We evaluate the utilization of such visual attention mapping in two different ML applications including one-class anomaly detection in Subsection 4.3.4.2, and for the explainable autonomous driving systems in Section 4.4.

4.1 β-NMF Rank Truncation for Unsupervised Dimensionality Reduction

In Chapter 2, the UGL methods have been highlighted which are a class of ML models that aims to learn representations, and it also reduces the dimensionality among data without any predefined labels or with limited levels of assumptions. Among all UGL methods, the NMF (Section 2.1.2.2) factorizes the data into two subspaces of nonnegative components. Moreover, the NMF enforces the nonnegativity, sparsity, and part-based analysis; thus, the dimensionality of data can be reduced, and also the representations can be interpreted and explained for the XAI applications. However, one of the main issues when using the NMF is to impose the factorization rank, r, to identify the dimensionality of the subspaces, where the rank is usually unknown in advance and termed as the non-negative rank that is employed as a prior to carrying out the factorization.

To help solve the above issue, this section proposes a novel method for the non-negative rank, r, approximation. Furthermore, we generalize our method among many different image scales, where the results on different image datasets confirm the validity of our proposed approach.

4.1.1 Introduction to Nonnegative Factorization

XAI is considered an emerging field in ML and data analysis to interpret how the decisions of ML applications are made, and to capture the underlying structure of data [1, 112]. The early stages of any XAI model include data reduction and disentangling the interpretable representations, *i.e.*, the explainable features (or transformations weights W) that reflect parts of the data [100].

To disentangle the representations and reduce data dimensionality, UGL methods have been specially studied, due to their advantage of working on limited levels of prior assumptions and labels. In particular, the UGL methods used in this regard include PCA, NMF, SVD, TD, and AEs [5, 65, 94, 301], see Chapter 2. Among all UGL methods, the NMF introduced in Section 2.1.2.2 decomposes the data into low dimensional subspaces, where the first one contains the bases of the latent features and is named as the latent space, Z, the other space hides the coefficients that reconstruct the data and is called the mixing space, A, [171]. The capability of the NMF lies in the fact that it factorizes the data into non-negative components (rows or columns of subspaces Z and A), which usually in low dimensional form and represent portions of the original data itself. Moreover, the other advantage of the NMF is that it is an inherently sparse method for feature representations, thus it offers sparse components that are isolated and represent the original data objects [7].

The NMF can be integrated into the XAI models to improve their explainability and interpretability, as it is easy to relate the hidden (or latent) representations to the original data itself throughout reconstructing data after factorization. Also, it reduces the dimensionality and the computational time required to disentangle the interpretable representations among the data. However, one of the main issues when using the NMF

is to identify the factorization rank, r, among the data, which is usually unknown in advance and can be imposed as a prior to carrying out the factorization [165].

To help solve this problem, we propose a novel rank approximation method for the NMF in this section, where our contribution is twofold: *(I)* introducing an effective rank truncation method based on algebraic operators and a physical conceive, thus the dimensionality among the data and the processing time are reduced according to the approximated rank; *(II)* generalizing the rank truncation from small-size data samples to large-size ones, thus achieving adaptability at different scales. The rest of this section is organized as follows. Section 4.1.2 highlights the β-NMF. Section 4.1.3 outlines the NMF rank analysis. The experimental results and related works comparison will be reported in Section 4.1.4. Section 4.1.4.4 summarizes the conclusions and remarks of our proposed method.

4.1.2 β-NMF Factorization

The NMF is a linear dimensionality reduction method that belongs to BSS models reviewed in Section 2.1, due to its ability to learn part-based representations in an unsupervised way [309]. Also, it is used in image recognition and reconstruction applications: for a given image $X \in \mathbb{R}^{m\times n}$, it decomposes the data matrix in accordance to Eqn. (2.18) as:

$$X = ZA + R \tag{4.1}$$

where $Z \in \mathbb{R}^{m\times r}$ represents the bases of the latent subspace, $A \in \mathbb{R}^{r\times n}$ contains the mixing subspace, R is the residual, and r is the factorization rank. The NMF represents the data as a product to two subspaces, where the objective function of the optimization procedure minimizes the residual, R, by measuring the mismatch between the original data, X, and the reconstructed ones, $\tilde{X}$.

The most widely used class of distributional optimization objective is termed as β-divergence which comprising the Itakura-Saito (IS) when $\beta = 0$, Kulback Leibler (KL) when $\beta = 1$, and Frobenius norm when $\beta = 2$ [192]. The role of such objective functions is to minimize the residual, R, by quantifying and minimizing the distance between the original data and the reconstruction of the two factored subspaces, *i.e.*, Z and A [99]. The β-divergence between two matrix elements (or image pixels) is given as:

$$d_\beta(x,\tilde{x}) = \begin{cases} \frac{x}{\tilde{x}} - \log\frac{x}{\tilde{x}} - 1, & \beta = 0 \\ x\log\frac{x}{\tilde{x}} - x + \tilde{x}, & \beta = 1 \\ \frac{1}{\beta}(\beta-1)(\tilde{x}^\beta + (\beta-1)\tilde{x}^\beta - \beta x\tilde{x}^\beta), & \text{otherwise} \end{cases} \tag{4.2}$$

where d_β is the divergence, x represents the original data point (or pixel), $\tilde{x}$ is the reconstructed pixel after applying the factorization or learning. When extending the notation from pixel or data point to matrix (whole image), the β-divergence general-

ization is given as:

$$d_\beta(X, \tilde{X}) = \sum_{(i,j)} d_\beta(X_{(i,j)}, (Z_r A_r)_{(i,j)}) \tag{4.3}$$

where d_β is the divergence, X is the original image (or data matrix), $\tilde{X} = Z_r A_r$, Z_r and A_r are the bases of the latent space and the coefficients of the mixing space, respectively, and resulting by the factorization using rank r. To achieve the minimum divergence, the matrix update procedure [89] is followed as:

$$Z \leftarrow Z \odot \frac{([ZA]^{\odot\beta-2} \odot X)A^\intercal}{[ZA]^{\odot\beta-1}A^\intercal} \tag{4.4}$$

$$A \leftarrow A \odot \frac{Z^\intercal([ZA]^{\odot\beta-2} \odot X)}{Z^\intercal[ZA]^{\odot\beta-1}} \tag{4.5}$$

where $^\intercal$ is the matrix transpose and $\odot$ denotes the element-wise multiplication.

The most widely used approaches for the NMF initializations, i.e., identifying the dimensions and values of Z and A, are SVD-NMF, non-negative double SVD, and non-negative SVD with low-rank correction [15]. These methods share a similar procedure to carry out the NMF initialization and identifying the dimensions of both Z and A spaces; by updating the matrices Z and A based on the rank increment at each iteration until a given level of performance is reached. Other approaches, based on the rank adaptation from the lowest to the highest or full rank, are time-consuming since they depend on a trial and error procedure [98].

To avoid the cost given by the iterative search, a commonly used practical approach (the rule of thumb) keeps the singular values that contribute to $90 : 99\%$ of the total energy sum and impose their number as the rank; however, such an approach suffers from the instability due to fixing the bounds of the singular values [231]. The recent method proposed in [274] is based on the minimum description length (MDL). The MDL method depends on finding a possible way to encode the data with high precision and low decoding error, where it does not approximate the rank directly from the data itself; it selects the suitable rank that reflects the minimum MDL among a list of all imposed ranks, thus it can be seen as a kind of trial and error method. Moreover, the MDL method assumes that the data samples are already factorized with all available ranks and the subspaces Z and A which correspond to each rank are kept, then it converts the subspaces to distributions and utilizes the Shannon information content [66] among the distributions to measure the minimum MDL that reflect the best rank.

To obtain the factorization rank automatically, the following section introduces an innovative rank truncation method based on the combinations between the matrix trace, nuclear, and the Frobenius-norm. Such a combination able to preserve suitable bounds of the singular values that reflect the optimal rank, also it obtains the rank directly from the data itself to achieve stability and reduce the computational cost.

4.1.3 Proposed Rank Truncation Methodology

Advocated by the fact that the NMF requires to impose the factorization rank, r, before carrying out the factorization, *i.e.*, the size of the data columns and row spaces that identify the dimensions of the NMF subspaces, also the rank must be accurate and able to preserve an acceptable level of the reconstruction accuracy [15]. Practically, running the empirical rank approximations for the NMF factorization among all available ranks is considered a computational burden, especially when the data comes in a multi-way form as in the RGB images and when its size is relatively big [7]. For instance, to identify the optimal rank for an image with dimensions of 256×256 empirically, we need to run 256 iterations from the lower rank ($r = 1$) to the full rank ($r = 256$). Thus, to reach the appropriate rank without alternating all possible ranks, we propose an automatic rank truncation procedure, which can be useful for many different image datasets and based on the linear transformations among the data.

Both the trace and nuclear norm have been utilized to produce a very low-rank solution theoretically [219, 233]; however, we extend the theory to the practice by proposing a suitable rank approximation for ML datasets. The trace of the data matrix (the data matrix is an image and denoted as X in all parts of this work) is considered a useful linear transformation, and it gives the derivative of the determinant $\det|X|$ that offers the volume equipped by the column space to approximate the data rank [134]. Mathematically, the trace is the sum of all diagonal elements of a square matrix (or sum of the eigenvalues), where the physical meaning of the trace is the constructions of the Hamiltonians (the total energy) of the quantum system that associated with a finite set of the energy eigenvalues [115]. The trace of the $n \times n$ square matrix X is given as:

$$\mathrm{Tr}(X) = \sum_{i=1}^{n} x_{(i,i)}, \qquad i = 1, , , n \tag{4.6}$$

In the same orientation, the nuclear norm $||\cdot||_*$ is considered a substantial tool in the field of multivariate statistics and dimensionality reduction, whereas it used recently for deep learning optimization as a convex replacement of the dimensional rank [233]. The nuclear norm reflects the importance of the singular values that constitute the rank variation among the data. The nuclear norm can be obtained by summing all singular values, which can be retrieved using the SVD decomposition reviewed in Section 2.1.2.1. Moreover, the singular values' matrix is constituted in a diagonal form and reflects the importance of the eigenvalues and data points when reconstructing the data. Factually, summing all singular values is equivalent to adding up the absolute values of the diagonal elements (*i.e.* the L_1 norm) of a diagonal matrix, where the rank optimization problem is tuned to find a sparse vector of singular values in the affine subspace σ [233]. The nuclear norm of a given data matrix X is obtained from the SVD as:

$$X \approx \sum_{i=1}^{r} U_i \sigma_i V_i^{\intercal}, \quad \text{where} \quad ||X||_* = \sum \sigma_i \tag{4.7}$$

where U_i and V_i are orthonormal matrices called the left and right singular vectors,

respectively, and they represent the eigenvectors of $XX^\intercal$ and $X^\intercal X$, respectively. Additionally, σ_i is a diagonal matrix that contains the singular values $\{\sigma_i \,|\, i = 1, \ldots, n\}$ of the matrix X and are sorted in decreasing order.

In our proposed approach, we combine the Frobenius norm [222] due to its capability to the transformations that are unitary invariant, where it penalizes (or expands) the rank value. Thus, the bound of the singular values are adapted to an appropriate level reaching the optimal rank. The Frobenius norm for a given matrix $X \in \mathbb{R}^{m \times n}$ can be obtained by the square root of the sum of the squares of the matrix elements as follows:

$$||X||_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{(i,j)}^2} \tag{4.8}$$

The nuclear norm and the trace of the data are considered algebraic tools, which help to identify a truncation limit to remove the unwanted portions of data and only keep the ones that reflect the rank r. The rank truncation limit is a threshold, ϵ, at which performance can be saturated and the data can be reduced [110, 134]. Dividing the trace of eigenvalues by the nuclear norm (as in Eqn. (4.9)) is equivalent to spreading out the total energy of eigenvalues among different separate states of singular values with different capacities, where the highest singular values absorb the highest energies and the lowest values only get very low energies [115].

Practically, the trace of the square matrix is equal to the sum of its eigenvalues[1], and there is an ability to compute the trace directly from the data matrix without the need to diagonalize the data to obtain the trace of its eigenvalues matrix. However, if the data matrix in a non-square form or if the majority of values is zeros in its diagonal, i.e., sparse, it must diagonalize the matrix first (*i.e.,* covert it to sets of eigenvalues and eigenvectors), then summing of its eigenvalues as a trace.

The proposed truncation threshold can identify suitable bounds of singular values that reflect the rank; by truncating the singular values that absorb the minimum energies, and only keeping the highest ones that conclude the essential and appropriate features among the data. Our methodology is divided into four steps as follows:

- For a given image (or data matrix) $X \in \mathbb{R}^{n \times n}$, calculate the $\mathrm{Tr}(X)$ and $||X||_{\mathrm{F}}$.
- Calculate the SVD and obtain $||X||_*$.
- Identify the truncation threshold, ϵ, as:

$$\epsilon = \frac{\sqrt{\mathrm{Tr}(X)}}{||X||_* + ||X||_{\mathrm{F}}} \tag{4.9}$$

- The rank is obtained by counting the singular values (taken from SVD) up to ϵ.

[1] *http : //people.math.harvard.edu/ ∼ knill/teaching/math*19*b*_2011*/handouts/math*19*b*_2011.*pdf*

Accordingly, the rank can be expressed as the total number of the singular values that lie up to the truncation threshold ϵ.

To measure the performance of our proposed method, we will employ both MNIST digits and MNIST fashion datasets [75, 324], which share similar small image dimensions. Thus, there is a possibility to build a common threshold to obtain the nonnegative rank, r, as it appears in the Eqn. (4.9). Moreover, we found that it is important to inject the Frobenius norm to the threshold denominator as a penalty factor and expand the minimum bound of singular values, σ_r, into an appropriate limit that reflects the optimal rank.

However, for the datasets which are relatively larger than the MNIST in terms of image size, the trace and the nuclear norm can be fixed without injecting the Frobenius norm; since the MNIST datasets contain a lot of zero elements in the images which affect the bounds of the singular values. Following the same footprint in approximating the rank for the MNIST datasets, we can build similar rank truncation based on the trace and the unclear norm for the color and large-sized images. The rank truncation threshold for such type of images that hides a lot of information due to their size and dimensionality can be designed as:

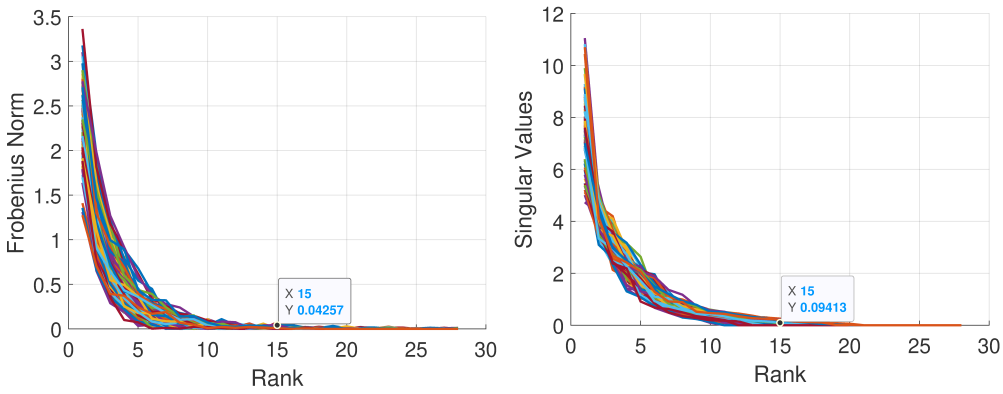
$$\epsilon = \sqrt{\frac{\mathrm{Tr}(X)}{||X||_*}} \tag{4.10}$$

4.1.4 Experimental Results

4.1.4.1 β-NMF Rank Truncation for Small Size Images

The MNIST digits dataset comprises 70000 handwritten images divided into 10 classes, representing digits from $0-9$. The kin new dataset to the MNIST digits is called the MNIST fashion with the same number of samples and classes. The size of each image in the MNIST datasets is similar, and each image dimension is 28×28 pixel which forms the data columns and the rows spaces. The MNIST datasets are ideal for analyzing the small-size images, due to the limited size of the images. Also, our used computational platform to approximate the NMF rank includes Intel CPU Core $i7-4800MQ$ processor with 8GB DDR3 RAM, and all experiments are implemented using MATLAB R2019b on windows 10 operating system.

To extract the non-negative features for the MNIST datasets, i.e., the representations of Z and A subspaces, the factorization rank, r, among the images must be imposed. The rank identifies the size and dimensions of features and mixing subspaces. Traditionally, the rank can be approximated by alternating the rank from the lowest ($r = 1$) to the full rank-size, (r=28), where the rank is adapted and estimated as a function of the image size. We used the β-NMF introduced in [99] with $\beta = 1$ to show how the traditional rank affects the image reconstruction. Moreover, the Frobenius norm as a function of image rank is utilized in the following Fig. 4.1(a) to depict how the original images and the reconstructed ones are similar to each other, where the minimum values of Frobenius norm reflect better reconstruction results after carrying out



(a) (b)

Figure 4.1: (Fig. 4.1(a)) The Frobenius norm of 100 images of the third class as a function of rank r, (Fig. 4.1(b)) the corresponding singular values of the same images in Fig. 4.1(a). It is possible to observe that the Frobenius norm decays around $r = 15$.

the factorization. Furthermore, to show the relationship between the rank truncation and the distribution of singular values of the same data used in Fig. 4.1(a), we plot the singular values and the rank (from $r = 1 : 28$) for each image as it can be illustrated in Fig. 4.1(b). For sake of computation, we plot only the Frobenius norm and singular values distributions for the first 100 images of the third class from the MNIST digits, where the idea is generalized among all samples and classes.

As can be noticed from the Fig. 4.1(a) and Fig. 4.1(b), when the rank is very low at $r = 1 : 5$, the Frobenius norm is very high; conversely, the norm is saturated in the pool of ranks greater than 15. The rule of thumb [231] to extract the rank from the data can be carried out by retaining $90 - 99\%$ of the singular values that contribute to the sum of total energies and utilizing their number as the NMF rank. However, according to Fig. 4.2 using the rule of thumb reflects the rank $r = 10$, also at that rank the averaged Frobenius norm between the original images and the reconstructed ones for the same samples used in Fig. 4.1(a) and Fig. 4.1(b) is equal to 0.0209. Furthermore, our method expands the bounds of the singular values to enhance the performance and gives the rank $r = 15$, also the mean reconstruction loss obtained by the Frobenius norm enhanced and reached to 0.0049 when comparing the useful rule of thumb [231].

As it evident from Table 4.1 that the performance based on the SSIM metric is greater than 0.9950 among all classes in both MNIST datasets, which confirms the validity of our proposed rank truncation method.

Another used approach in the practice, by keeping 90% of the singular values that contribute to the nuclear norm introduced in Eqn. (4.7), where it gives the average rank for the images used in Fig. 4.1 equal to $r = 10$. Also, at that rank the Frobenius norm still not saturated, *i.e.*, the reconstruction loss requires to increase the rank to be minimized. From this angle, it is important to build a suitable rank truncation threshold, ϵ, to be utilized for automatic non-negative rank approximation, thus the

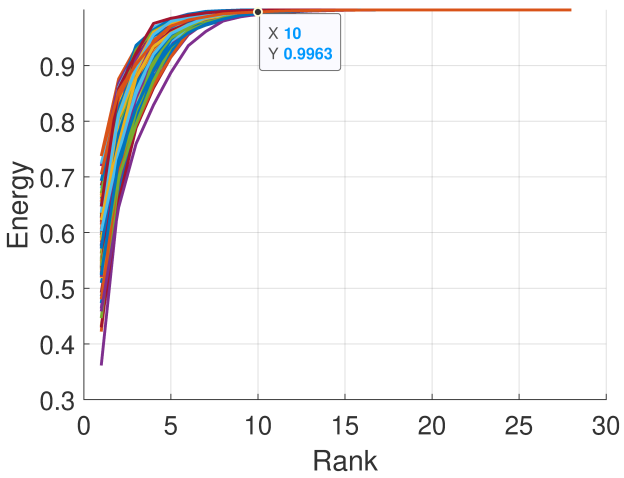


Figure 4.2: The distribution of the accumulated energies of the singular values. It is apparent from the figure that the highest $r = 10$ singular values contribute to the 99% of their total energy.

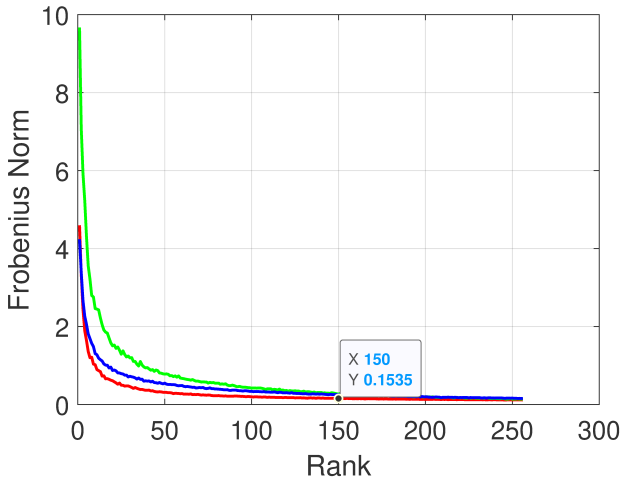
Table 4.1: The Rank and performance analysis for the MNIST datasets, where the ClassID represents the label of each class, Samples is the total number of data samples belong to each class, Rank is the approximated nonnegative rank, Fnorm measures the Frobenius norm that evaluates the reconstruction loss after the factorization, and SSIM reflects the structural similarity index that measures the quality of images after reconstructing the factorized subspaces.

MNIST Digits					MNIST Fashion				
ClassID	**#Sample**	**Rank**	**Fnorm**	**SSIM**	**ClassID**	**#Sample**	**Rank**	**Fnorm**	**SSIM**
0	5923	16	0.0107	0.9962	**T-shirt/top**	6001	17	0.0048	0.9971
1	6742	10	0.0029	0.9992	**1Trouser**	6001	11	0.0001	0.9997
2	6742	15	0.0074	0.9974	**Pullover**	6001	16	0.0027	0.9974
3	6131	15	0.0065	0.9975	**Dress**	6001	14	0.0019	0.9987
4	5842	14	0.0071	0.9976	**Coat**	6001	17	0.0031	0.9969
5	5421	14	0.0097	0.9977	**Sandal**	6001	15	0.0096	0.9953
6	5918	16	0.0068	0.9982	**Shirt**	6001	18	0.0056	0.9950
7	6265	13	0.0090	0.9971	**Sneaker**	6001	14	0.0025	0.9984
8	5841	15	0.0083	0.9987	**Bag**	6001	17	0.0036	0.9988
9	5949	14	0.0056	0.9986	**Ankle boot**	6000	19	0.0020	0.9991

rank can be estimated and imposed easily with minimum reconstruction loss as it is proposed in Eqn. (4.9).

Table 4.1 shows the average rank for each class extracted from the MNIST datasets using our threshold introduced in Eqn. (4.9), using only the average rank of 1000 random images from each class to be generalized among the whole images within each class. Moreover, the table presents the performance analysis of all datasets samples (not just 1000 samples) when using the generalized rank, utilizing the Frobenius norm and

(a)



(b)

Figure 4.3: Approximating the rank for Lena's image, where Fig. 4.3(a) depicts Lena image, and Fig. 4.3(b) shows the Frobenius norm as a function of the rank. The red, green, blue curves in Fig. 4.3(b) represent the Frobenius norm as a function of rank for R-, G-, B-domain, respectively.

the SSIM metrics to compare the original images and the reconstructed ones after the factorization. For more details about the SSIM index we refer to [310] and Section 1.3.

4.1.4.2 β-NMF Rank Truncation for Large Size Images

In this section, we select miscellaneous standard color images of size 256×256 and 512×512, which were published by the Computer Vision Group of the University of Granada[1]. Also, the resolution of such images is very high, and approximating their rank for the NMF factorization in their RGB domain is considered substantial [310]; in the view of fact that a lot of information is vulnerable to be lost during images transformation to the grayscale or in resizing their dimensions. Accordingly, it is important to approximate the rank at R, G, and B-domain separately to keep an appropriate level of the information at each domain, then extracting the rank for each domain. Fig. 4.3(b) highlights the Frobenius norm of the Lena image of size 256×256 and its reconstructed version after applying the β-NMF with $\beta = 1$. Accordingly, the rank is identified following the traditionally used approach, by varying the rank from the lowest rank ($r = 1$) to the highest rank ($r = 256$).

As it can be noticed from the Fig. 4.3 that the highest Frobenius difference between the original and the reconstructed images can be noticed at the first 50 ranks, and the performance is improved (by reducing the Frobenius norm) as a function of rank around $r = 130 : 170$. The proposed truncation threshold in Eqn. (4.10) gives $r = 141$ for R domain, $r = 165$ for G domain, and $r = 193$ for B domain for Lena image, which

[1] http://decsai.ugr.es/cvg/index2.php

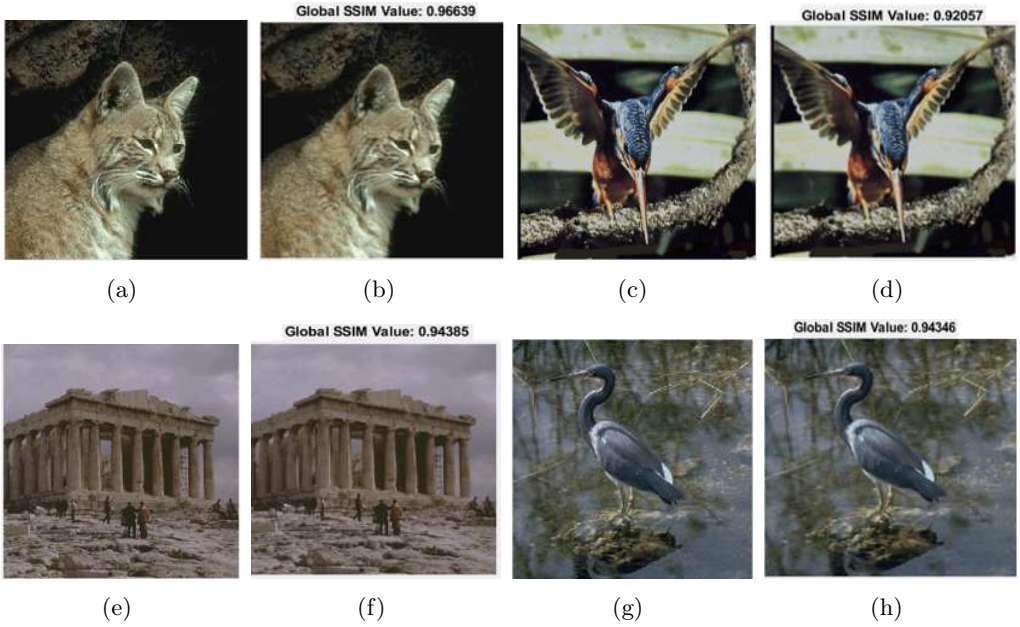


Figure 4.4: Generalizing the rank truncation threshold for natural images with a size of 512×512 for each, where the first and third columns contain the original images and the second and fourth columns represent the corresponding reconstructed ones. The SSIM index is equal to 0.9663, 0.9205, 0.9438 and 0.9434 for (b), (d), (f), and (h), respectively.

is applicable with the rank plot in Fig. 4.3(b). Moreover, the SSIM index is equal to 0.9880 according to our truncation threshold, where it is lower than the full rank (when $r = 256$) by 0.0060. Thus, the proposed rank threshold in Eqn. (4.10) is able to preserve a higher level of the original structure (for large-size images) after the NMF factorization, by retaining the appropriate bounds of σ_i that reflect the rank.

The application domain is extended to a new series of natural images with a dimension of 512×512 for each one in Fig. 4.4. Throughout utilizing the same rank truncation threshold proposed in Eqn. (4.10), where the images are factorized in each RGB domain and they are reconstructed from their reduced spaces. To measure the reconstruction loss after the factorization, we fixed the evaluation indicator to the SSIM index which is commonly used to compare the quality of large-sized images in terms of the structure and the local mapping.

4.1.4.3 Related Works Comparison

The rank selection stage is considered the first step to carry out the NMF factorization, where it initializes the dimensions of the latent subspace, Z, and the mixing subspace A. Moreover, the rank selection stage is usually tricky, and there are limited practical approaches available to identify the rank r. For that aim, we proposed a novel method to identify the NMF rank automatically, and we compare our proposed method with the common practical approaches.

Table 4.2: Related methods comparisons of the same images used in Fig. 4.1, where the same computational settings that are reported in Section 4.1.4.1 is considered. Moreover, this table compares the related methods concerning the estimated rank "Rank", the total computational time required to factorize the images "CPU Time", and the quality of the reconstructed images after the factorization is measured by using "SSIM" index.

Comparison	**Indicators**		
Method	Rank	CPU Time	SSIM
Trial and Error [98]	1 : 28	NA	0.4732 : 0.9995
99% of Energy of Singular Values [231]	10	5.259 s	0.9866
90% of Truncated Singular Values [98]	9	4.781 s	0.9839
MDL [274]	**13**	16.903 min	0.9943
Our Method	**15**	**3.873** s	**0.9955**

Table 4.2 reports the performance evaluation of our proposed method employing the same images used in Fig. 4.1. Moreover, we compare our results with respect to *(i)* trial and error method by adapting the rank from 1 to full image size [98], *(ii)* retaining 99% of the energy of singular values contribute to the total sum [231], *(iii)* the truncated SVD with keeping 90% of the whole singular values [98], and *(iv)* the MDL [274]. Furthermore, the table shows the average execution time of the CUP used in our machine for each method to obtain the rank. Also, we evaluate the stability when reconstructing the images; respecting that, the stability is application dependent, and we measured the SSIM among the reconstructed and original images, thus the method that gives the highest SSIM is considered the most stable one.

As can be noticed from Table 4.2, that our method achieves the minimum execution time and high reconstruction accuracy based on the SSIM, especially when comparing our result to the recent method (MDL) [274]. Although the MDL achieves similar results to our method, it requires time greater than our method by 262 times to obtain the rank of the set of images used in Fig. 4.1. Also, the trial and error method that is used in the practice is considered time-consuming and requires increasing the rank by 1 at each factorization round. Moreover, the mean CPU time needed to factorize each MNIST image using β-NMF is 0.1200 s, provided that the factorization rank is imposed. For the trial and error method, we found that the factorization time when adapting all ranks is equal to 0.1200 s $\times$ 28, and if the image size is relatively large such method is considered time-wasting. Additionally, the truncated singular values [98] however it achieved the lower SSIM, but it requires an average factorization time greater than our method by 0.908 s for the same 100 images.

Instead, our proposed method reduces the required processing time to obtain r and carry out the factorization by automatically derive the rank from the data itself, and our performance analysis shows superior results in terms of preserving a similar amount of the structural information when using all possible ranks.

4.1.4.4 Summary

The NMF is a linear factorization technique that enforces non-negativity constraints among the factorized subspaces, which leads to precisely additive positive components to the factorized subspaces that are parts of the original data. Thus, the NMF offers interpretability when it is integrated into the XAI models, due it its ability to relate the factorized subspaces to the original data. The NMF reduces the dimensionality but it requires identifying and impose the non-negative rank, r, before carrying out the factorization, that usually unknown in advance. To solve this problem, this section proposed a novel rank truncation method that reflects the rank by counting the singular values which lie on certain bounds and consider their number as the rank. The rank is obtained directly according to the truncation threshold, ϵ, instead of iteratively adapting the rank. Also, it evaluated using β-NMF for the small-sized images of size 28×28 and generalized for the large-sized-images of size 512×512.

Our proposed method can approximate the NMF rank with a lower computational time concerning methods in the literature, at the same time obtaining a higher similarity to the original images. The proposed approach can therefore be used to efficiently map the data to reduced subspaces, with limited information loss. Due to its advantages, it could be used to optimize the learning process of deep ML architectures, by using a compact data representation that can reduce the computational times required to perform the training. Furthermore, our approach could prove beneficial when dealing with devices with limited computational resources (e.g., mobile CPUs or FPGAs), or in methodologies requiring real-time learning, e.g., online learning.

The following Section 4.2 introduces a novel way to improve the generalizability of the ML models through combining β-NMF and AE learning (see Section 2.3.2), where such a combination can be exploited in cases when only limited training samples are available to fit the models, i.e., test ML models with more testing samples than the training ones. Moreover, the following section presents a novel method to reduce the computational complexity of unfolding the manifold of data based on t-SNE that reviewed in Section 2.2.2.4, by unfolding the intrinsic structure of the factorized data (low-dimensional) that preserves similar characteristics to the original (high-dimensional) version of data.

4.2 β-NMF Based AE for Limited Data Learning and Unsupervised Image Reconstruction

UGL models gain popularity due to their abilities to learn with limited levels of predefined labels and prior assumptions, and they are also able to reduce the noise, redundancy, and visualize the manifold among data samples. However, generalizing the UGL models for different applications including image generation, compression, encoding, and recognition faces different challenges due to limited available data for learning, diversity of classes, and complex data dimensions.

This section introduces a partial learning procedure to learn from limited available data by utilizing β-NMF introduced in Section 4.1.2, which maps the data into two complementary subspaces constituting generalized driven priors among the data. Moreover, this section employs a dual-shallow AE (see Section 2.3.2) to learn the subspaces separately or jointly for image reconstruction, and to reduce the computational complexity of data manifold visualization tasks. Our proposed model shows superior performance to the literary works, especially when learning the model with a small amount of data and generalizing it for large-scale unseen data.

4.2.1 Introduction to Limited Data Learning

UGL models reviewed in Chapter 2 have the abilities to operate with limited levels of prior assumptions to perform dimensionality reduction, manifold visualization, and other representations learning tasks [2, 3, 65]. Among all UGL models, NMF is the only one that decomposes the data into two non-negative subspaces: the first one is termed as a latent space, Z, that hides the latent features, and the other is named as a mixing space, A, that contains the reconstruction coefficients [171]. The NMF subspaces constitute rooted priors to learn data because they hide the positive features, sparse, and part-based representations that represent the original data [64].

Recently, there is a demand to generalize ML models for real-life applications, where the data and computational resources are limited or scarce to carry out learning [1, 49, 114]. In most cases, the learning procedure in recent works uses more samples in the training stage than the testing: around 70% of the data are used for the training and the other for the validation and testing. Furthermore, different works feed ML models by data without preprocessing and considering learning the relevant representation, thus the models show more bias and overfitting [29].

To overcome the above issues, this section proposes a novel method to learn with limited available data (fewer samples for training than the testing), utilizing the β-NMF factorization introduced in Section 4.1; due to its ability in providing a driven prior among the image data and lead to generalizing the learned model for large-scale unseen data, i.e., out-of-distribution generalization. The β-NMF maps the data into two positive and sparse subspaces that constitute parts of the original data (rooted representations), thus the representations learning can be facilitated when building shallow ML models to learn among the data [192].

Since any NMF method requires imposing a factorization rank, r, that reflects the original data space dimensions [15] to be learned, our proposed method utilized the same rank truncation procedure introduced in Section 4.1.3 to obtain the nonnegative rank for β-NMF. Moreover, a shallow AE will be employed to learn the factorized subspaces partially (partial AE) or jointly using Dual-shallow AE. The rest of this section is organized as follows. Section 4.2.2 highlights the AE learning. Section 4.2.3 introduces the proposed AE learning methodology. The experimental results will be given in Section 4.2.4. Section 4.2.5 reports the remarks and conclusions.

4.2.2 AEs Learning

AEs reviewed in Section 2.3.2 are UGL models that can be formed in shallow or deep architectures, which are utilized in ML to perform dimensionality reduction, recognition, and generation tasks [29]. They also share a similar goal in capturing the hidden structure among the data, by reconstructing through the samples and benefiting from the advantages of the encoding and decoding stages [18]. For a given image (or data sample) $X \in \mathbb{R}^D$ (D is the input image dimensions), the encoding stage provides a mapping $f : \mathbb{R}^D \to \mathbb{R}^d$, $0 < d < D$ (d is the bottleneck layer dimensions), to corresponding encoded data $Z = f(X; \theta_e)$, while the decoding stage provides a decoding mapping $g : \mathbb{R}^d \to \mathbb{R}^D$, which reconstructs an approximation of the input data: $\tilde{X} = g(Z; \theta_d)$. Commonly, f and g can be composed of several encoding decoding stages (deep architecture) with a high degree of symmetry, in which the mappings of the j^{th} stage are parameterized by a weight matrix W_j and bias b_j. The objective function that minimizes the reconstruction error is given as:

$$\mathcal{L}_{\text{REC}_{\{\hat{\theta}_e, \hat{\theta}_d\}}} = \text{argmin}||X - (f \circ g)X||_{\text{Er}} \tag{4.11}$$

where $\hat{\theta_e} = \{W_e, b_e\}$ contains the encoder's weights and biases, $\hat{\theta_d} = \{W_d, b_d\}$ comprises the decoder's weights and biases, and $\circ$ is the Hadamard product of two matrices that gives element-wise commutative product $(\theta_e \circ \theta_d) = (\theta_d \circ \theta_e)$. Such product gives a realization (in terms of data matrix) that the decoding weights are similar to the transpose of the ones that are used for encoding, or trained to be similar, *i.e.*, $\theta_d \approx \theta_e{}^\intercal$. The reconstruction loss, E_r, can be measured by different metrics including Mean Square Error (MSE), Frobenius norm, β- divergence, and a recently utilized one for image applications is the Structure Similarity Index (SSIM) [2].

The main challenge in AE learning lies in finding and generalizing both encoding and decoding parameters (θ_e and θ_d), which minimize the reconstruction loss [164]. Especially, when only limited data available for learning, or when the computational resources restrict learning from big datasets (as in high order tensor data) [114]. The β-NMF helps the AE to learn reduced sparse, non-negative, and part-based representations from both Z and A subspaces (see Section 4.1.2), while feeding the AE with the original data enforces the AE to learn the data with noises, redundancy, and costly due to the original dimensions of data. We will show that our approach achieves a minimum reconstruction loss by comparing the related works; specifically, when generalizing the

trained model for large-scale unseen samples. Also, we will show how our proposed method allows capturing the rich representations that retain the fidelity among the factorized and encoded data to the manifold of the original data.

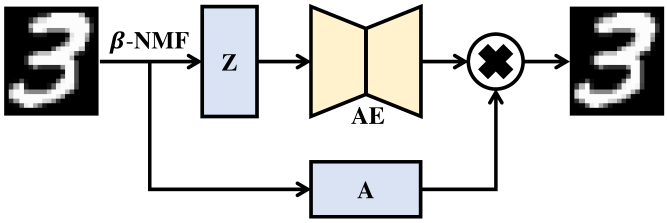
4.2.3 Proposed AE Learning Methodology

The NMF factorization has been utilized in the literature to build AE models to achieve a large-scale generalization. The Non-negative Sparse AE (NNSAE) proposed in [175] for online learning, enforces the weights of the AE hidden layer to be positive by using an asymmetric regularization and logistic activation function among neurons. Moreover, the NNSAE approach has been followed in [135] to build the Non-negative Constrained AE (NCAE) model for image reconstruction and classification. The recent work proposed the AE with a simplified Random Neural Network (AERNN) [329], where a training rule similar to the NMF used and able to update the model's weights in a non-negative way. However, such models do not exploit the NMF as a data-driven method to disentangle rooted representations and conscious prior among the data. Our proposed methodology is characterized from the others by decomposing the data in an unsupervised way (as what is introduced in Section 4.1) to be in a volume where the noise and redundancy are removed, thereafter uses the AE to learn the representations. Thus, the learned model is generalized for large-scale unseen data.

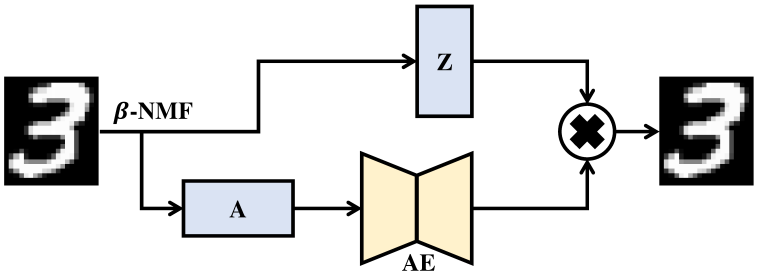
The concept of consciousness prior has been proposed for NLP applications, to combine different priors for disentangling abstract factors to be learned in further stages [28]. Such priors are seen as a bottleneck from which the extracted factors or representations have to proceed for further processing, and lead to generalization improvement for ML and AEs. Following similar footprints of [28], we employ the β-NMF to derive consciousness priors among the data, followed by learning each β-NMF subspace using a shallow AE for each. The subspace Z offers a prior among the data's hidden structure, while the A space contains a prior of the reconstruction coefficients.

To show the performance of our proposed work, we employ both MNIST digits and MNIST fashion datasets [75, 324]. Each comprises 60k images for training and 10k for the testing stage, divided into 10 classes with image size of 28×28. However, to challenge the generalization ability of our model to large unseen data, only 10k images will be trained, thus mimicking the limited available data scenario. Moreover, the testing performance will be measured on the other dataset samples, i.e., 60k. Two steps conclude the proposed methodology as follows:

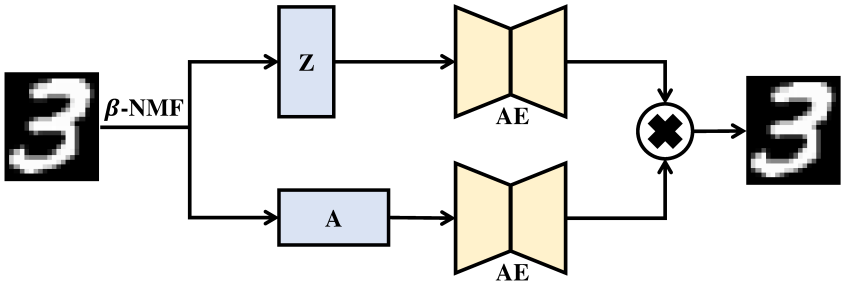
- The initial stage includes image factorization to extract the β-NMF subspaces (Z, A), utilizing the rank truncation threshold, ϵ, and setting $\beta = 1$. The same decomposition steps presented in Section 4.1 are followed, and the rank is estimated using a subset of the training set (1k images from each class) from each dataset, to be generalized among the whole data samples. The first and third classes from both datasets have been employed to measure the rank threshold robustness, then the Frobenius norm has been applied to measure the factorization loss acquired by the β-NMF, *i.e.*, the difference between the factorized data



(a) Z-AE



(b) A-AE



(c) Dual-AE

Figure 4.5: The three proposed AE learning schemes, Fig. 4.5(a) Z-AE in the top, Fig. 4.5(b) A-AE in the middle, and Fig. 4.5(c) Dual-AE in the bottom.

and the original one. We obtained $r = 16$ and $r = 15$ (for the first and third class, respectively) from the MNIST digits, and $r = 16$ and $r = 17$ (for the first and third class, respectively) from the MNIST fashion dataset. The averaged factorization loss among all testing samples (5k images from each class) did not exceed 0.017. Similarly, the rank is approximated among all classes in both data sets, where the generalized rank (taken from ϵ) $r = 16$ for both data sets due to their image size similarity.

- The second stage is dedicated to the AE learning among the β-NMF subspaces, separately by a shallow AE (Z-AE or A-AE), or jointly by a dual-shallow AE (Dual-AE) as depicted in Fig 4.5. Each AE for either separate or joint learning shares the same number of layers: one layer for each encoder, decoder, and

Table 4.3: The reconstruction performance of the Z-AE model, where SSIM refers to the structural similarity index that measures the quality of the reconstructed images, and MSE is the mean square error that measures the AE reconstruction loss.

Z-AE	Training		Testing	
Dataset	SSIM	MSE	SSIM	MSE
MNIST Digits	0.935	0.005	0.923	0.005
MNIST Fashion	0.760	0.021	0.759	0.022

bottleneck layer. The bottleneck's size still an open problem in the AE learning, thus we followed the [245] approach to identify the required number of neurons, where a discussion about the size of the bottleneck of the different data sets given therein. The method in [245] is implemented by halving the feature vector dimensions and imposing the number as a bottleneck size. In our experiments, the feature dimensions are $28 \times r = 448$, but we expand the bottleneck size for the W-AE (see Fig. 1) to 250 neurons due to its sparsity nature, and we reduce it to 200 neurons to the H-AE (196 neurons proposed in [245]). Because of the class complexity in MNIST fashion, we expand the size to 400 and 300 for the W-AE and H-AE, respectively. Finally, all AE experiments are fixed under 2000 epochs, saturated liner encoder and decoder transfer function, $l_2 = 0.0001$ and sparsity regularizer with coefficient $= 0.01$.

4.2.4 Experimental Results

To evaluate the performance of the proposed learning method, this section considers three scenarios indicated as *(i)* Z-AE and *(ii)* A-AE when only Z and A subspaces are learned, respectively, and *(iii)* Dual-AE when both the subspaces are jointly learned to reconstruct the data, see Fig. 4.5. We employ both MNIST datasets to compare our method with the literature in terms of MSE error introduced in Eqn. (2.98), which is commonly used in shallow AE learning. Besides, to show the ability of our method to preserve the original data structure, we also use the SSIM index. For more details about the SSIM, we refer to [310] and Section 1.

4.2.4.1 Z AE:

The latent subspace, Z, hides sparse, non-negative, and part-based representations that are obtained from the β-NMF and learned by the AE to reconstruct the data. The mixing subspace, A, contains sparse values that require a wider bottleneck layer than A-AE ($250, 400$ neurons for the MNIST digits and MNIST fashion, respectively). We measured to which extent that Z subspace can be learned separately, while the A subspace is used for the reconstruction (see Fig. 4.5(a)). Moreover, the learning complexity has been reduced from $O(m \times n)$ to $O(m \times r)$ where m is the row space

Table 4.4: The reconstruction performance of the A-AE model, where SSIM refers to the structural similarity index that measures the quality of the reconstructed images, and MSE is the mean square error that measures the AE reconstruction loss.

A-AE	**Training**		**Testing**	
Dataset	SSIM	MSE	SSIM	MSE
MNIST Digits	0.986	0.004	0.961	0.005
MNIST Fashion	0.888	0.006	0.872	0.007

Table 4.5: The reconstruction performance of the Dual-AE model, where SSIM refers to the structural similarity index that measures the quality of the reconstructed images, and MSE is the mean square error that measures the AE reconstruction loss.

Dual-AE	**Training**		**Testing**	
Dataset	SSIM	MSE	SSIM	MSE
MNIST Digits	0.910	0.009	0.900	0.010
MNIST Fashion	0.778	0.017	0.767	0.018

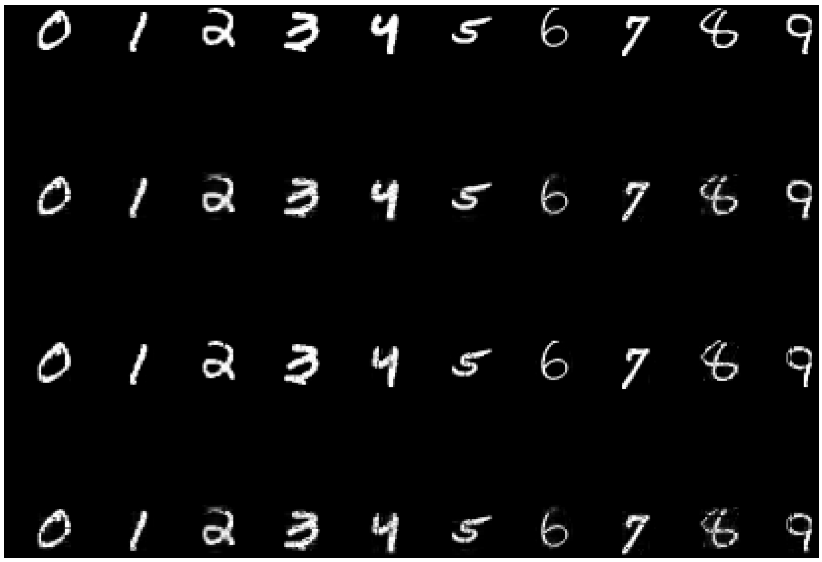
dimension, r is the data rank, and n is the column space dimension. Table 4.3 shows the reconstruction performance of the Z-AE model.

4.2.4.2 A AE:

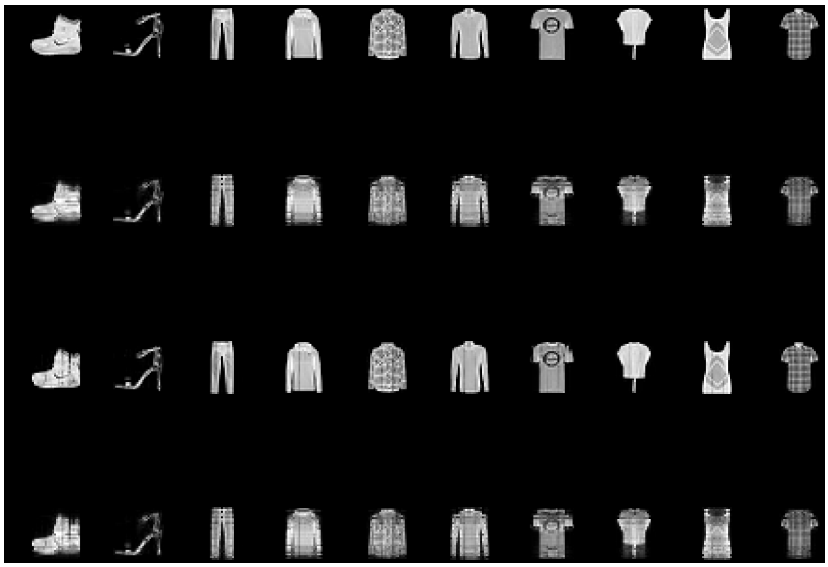
The mixing space, A, is considered a combiner to reconstruct the original data, where it is multiplied by Z for the reconstruction purpose. Also, the coefficients of A subspace are less sparse than Z, thus we reduce the number of neurons in the AE bottleneck layer to 200 and 350 for the MNIST digits and fashion, respectively. As in learning Z subspace, we measured to which extent that A subspace can be learned while keeping Z space as an identity for the sake of reconstruction (Fig. 4.5(b)). The learning complexity is carried out in $O(r \times n)$, instead of learning at $O(m \times n)$. Table 4.4 shows the reconstruction performance of the A-AE based on the used indicators in Table 4.3.

4.2.4.3 Dual AE:

In this scenario, both Z and A subspaces are learned separately to reconstruct the data jointly utilizing a dual AE: the model is fully automated and avoids keeping Z or A as identities (as in the Z-AE or A-AE) for the sake of data reconstruction. Also, the learning complexity for the Dual-AE scenario is $O(m \times r + r \times n)$. Accordingly, the learning process has been facilitated and implemented among different machines; we carried out the learning for the Dual-AE in two separated machines with a core i7-CPU at each one, where the learning complexity was $O(m \times r)$ and $O(r \times m)$ for the first and second CPU, respectively. Table 3 highlights the reconstruction performance of the Dual-AE model.



(a) MNIST Digits Reconstruction



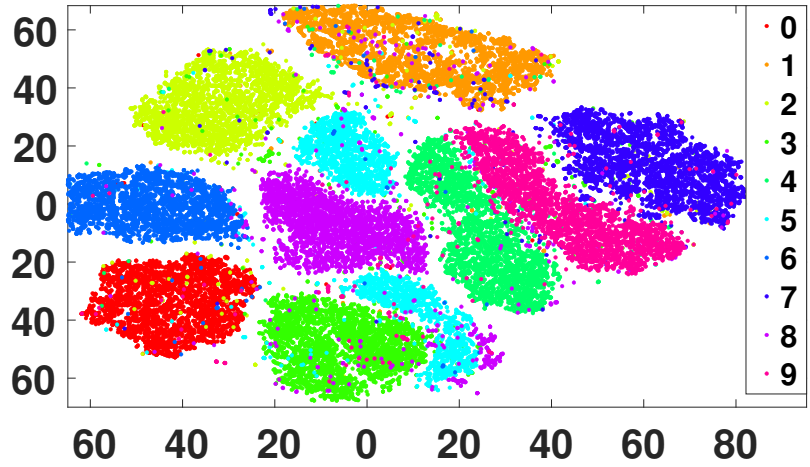
(b) MNIST Fashion Reconstruction

Figure 4.6: Visual comparison of the testing samples reconstruction from both datasets, where the first row of Fig. 4.6(a) and Fig. 4.6(b) represents the Ground-Truth samples, the second row of Fig. 4.6(a) and Fig. 4.6(b) represents the Z-AE reconstruction, the third row of Fig. 4.6(a) and Fig. 4.6(b) represents the A-AE reconstruction, the forth row of Fig. 4.6(a) and Fig. 4.6(b) represents the reconstruction of dual AE.

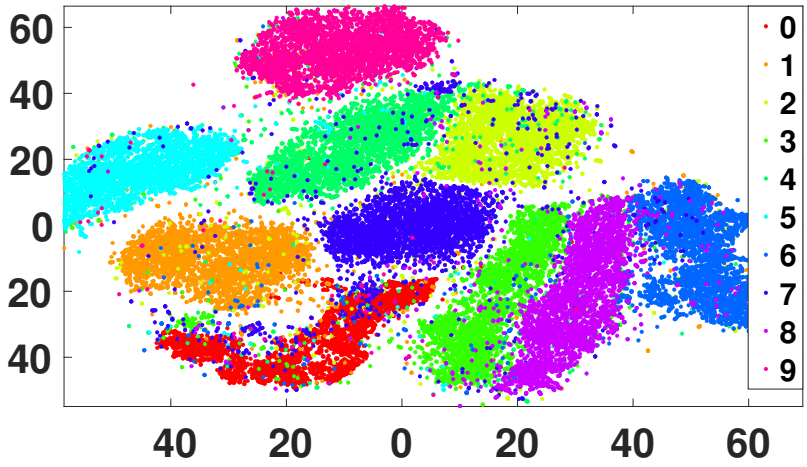
To show the reconstruction ability of the Z-AE, A-AE, and Dual-AE, Fig. 4.6 depicts the reconstruction differences between each AE model with respect to the ground truth data. As it can be noticed from Fig. 4.6 and the above tables, that the A-AE outperforms the other AEs in terms of data reconstruction.

Eventually, to show the fidelity of the factorized and the encoded data to the original one, in terms of clustering and preserving the inter-classes variations, Fig. 4.7 shows the t-SNE (see Section 2.2.2.4) embedding of the original MNIST digits dataset, the factorized subspace Z of the β-NMF, and the latent (bottleneck) space of the Z-AE model.

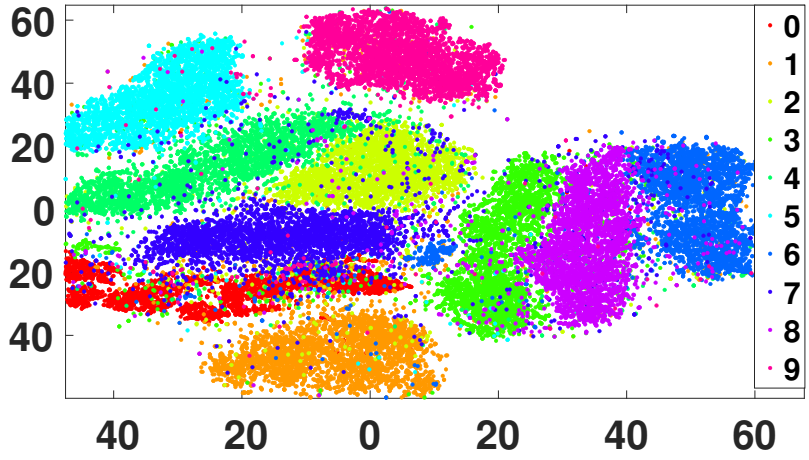
As it can be concluded from Fig. 4.7, that the factorized subspace Z and the AE's latent space maintain the discriminating features to be clustered, and both show the same fidelity to the original dataset; preserving the clustering properties and avoiding classes shuffling (*i.e.*, realizing an invariant transform). Finally, the computational complexity of the t-SNE is reduced from $O(DN^2)$ (Fig. 4.7(a)) where D is the dimensions of each image (for the MNIST samples $D = 28 \times 28$) and N is the number of samples in the data set to $O(d_r N^2)$ (Fig. 4.7(b)) where d_r is the Z space dimensions $d_r = 28 \times 16$, see Section 4.2.3.



(a) Original Data Embedding



(b) Z Subspace Embedding



(c) AE Latent Space Embedding

Figure 4.7: The t-SNE embedding of the original MNIST digits dataset (Fig. 4.7(a)), Z subspace of the factorized dataset (Fig. 4.7(b)), the latent (bottleneck) space of the Z-AE (Fig. 4.8(c)). Moreover, 40000 samples have been employed for each sub-fig.

Table 4.6: Comparison with recent methods in the literature, including deep unsupervised learning methods [245, 246, 254, 329].

Method	Training MSE		Testing MSE	
	MNIST		MNIST	
	Digits	Fashion	Digits	Fashion
SAE-RBM [284]	0.823	NA	NA	NA
NNSAE [175]	0.012	NA	0.015	NA
Fold-AE in [305]	0.178	NA	5.929	NA
Group Sparse-AE [254]	1.10	1.10	NA	NA
AE-SNN [245]	0.110	0.150	0.122	0.178
Structuring-AE [246]	0.025	0.014	NA	NA
NNAE-sRRNN [329]	0.024	NA	NA	NA
Our proposed W-AE	**0.005**	**0.021**	**0.005**	**0.022**
Our proposed H-AE	**0.004**	**0.006**	**0.005**	**0.007**
Our proposed Dual-AE	**0.009**	**0.017**	**0.010**	**0.018**

4.2.4.4 Recent Works Comparison

The performance comparison of the basic recent deep UL models based on the NMF and AE learning is reported in Table 4.6. It also comprises: stacked AE with restricted boltzmann machine (SAE-RBM) [284], non-negative sparse AE (NNSAE) [175], fold-AE [305], group sparse AE (GSAE) [254], AE spiking neural networks (AE-SNN) [245], structuring AE (SAE) [246], and non-negative AE with simplified random neural net-

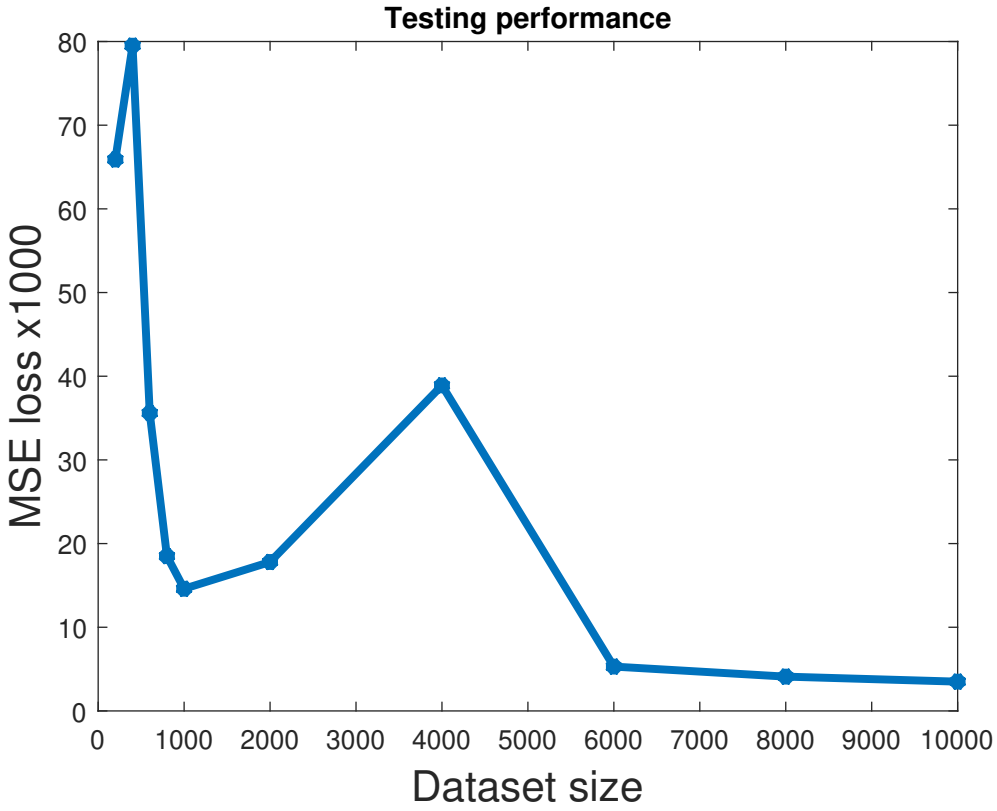


Figure 4.8: The Testing loss performance as a function of dataset size.

work (NNAE-sRNN) [329]. Every work except [175] and [305] employed 60k training samples and 10k testing to carry out the learning and performance evaluation. In [175] only 10k training samples were used and 50k considered as testing samples, and in [305] only a mini dataset of 200 images were used. We followed [175] and employed 10k samples for the training stage and the other for the testing. As it can be reported from Table 4.6, our proposed AE learning method obtained the best results in both datasets concerning the image reconstruction metric (MSE) and recent UL Models.

To show a fair comparison, we adjusted the size of the dataset in the range of $[200 : 10000]$ considering 70% of the whole size as a training set and the other for the testing set, thus demonstrating how the testing loss can be saturated in Fig. 4.8. As it can be noticed from Fig. 4.8, the testing loss shows a minimum around a dataset size of 1000, but it is saturated around a data size of $8000 - 10000$, which meets the specifications of our training size (see Section 4.2.3). Finally, besides obtaining the minimum reconstruction loss, our method outperforms all other UL models in terms of the limited data training and generalization abilities.

4.2.5 Summary

We proposed an approach to improve the generalizability of ML models based on unsupervised data factorization and an encoding scheme, where the proposed method can be utilized in image reconstruction, and manifold visualization. We used the β-NMF to reduce the data dimensionality and obtain both latent and mixing spaces Z and A, respectively. We trained a shallow AE at each subspace and used a dual-shallow AE to learn from both subspaces jointly.

The performance analysis shows that our proposed work achieves the minimum reconstruction loss respecting the relevant literary works, especially when learning with limited available data (a small set for training but large for testing). Moreover, our proposed method can reduce the required computational complexity to visualize the intrinsic structure of the data manifold. Furthermore, our method retains the same clustering characteristics among the original data, the factorized version, and the encoding of the factorized data.

The following Section 4.3 presents a novel method to explain an AE model is termed VAE, which has been reviewed in Section 2.3.2.4 and is built based on the variational inference to enhance the generalizability of the learned representations. Also, the following section shows the advantage of the second-order partial derivative to capture the curvatures of the neural activations, which are aggregated to build a visual attention map. Furthermore, the proposed model will be employed in the application of one-class anomaly detection, which is considered a branch of ML and deals with a special form of limited data. Additionally, in one-class anomaly detection, the model training is limited to the normal version of data; however, both normal and anomaly data are considered in the production and testing stages.

4.3 Second Order Gradient ($Grad_2$) Attention for Explainable Autoncoders

In Section 2.3.2, AEs have been reviewed which are a class of UGL models characterized by their large-scale generalizability. AEs are utilized to perform many different tasks including dimensionality reduction, visualize the decision, and learn data with limited levels with prior assumptions. Among all AEs, the VAE model (highlighted in Section 2.3.2.4) is a UGL model widely utilized in many different applications, where it uses variational inference (VI) to approximate the posterior distribution of large datasets.

In this section, we propose a novel XAI method to explain the behavior of the VAE based on the second-order (2^{nd}) derivative of the latent space concerning the encoding layers, which reflects the amount of acceleration required from the encoding to decoding space. The proposed model is termed as $Grad_2$VAE, and it can capture the local curvatures of the learned representations to visually explain the model's behavior. Besides the VAE explanation, we employ our proposed method for anomaly detection task, where our model outperforms the recent deep UGL models when generalizing it for large-scale anomaly data.

4.3.1 Introduction to VAEs

XAI models are associated with UGL models to explain the behavior of the learned representations, throughout performing dimensionality reduction and learn data with limited labels or prior assumptions. AEs are a class of UGL methods, which can reduce dimensionality, visualize and generate data, and perform other ML tasks such as object recognition [1, 2]. Many different types of AEs have been introduced recently and they are characterized by a regularization term; such a term enforces AEs to learn with an additional penalty to capture different representations for a better generalization [29].

Deep AEs encompass many different encoding and decoding stages, where at each stage diverse layers with associated parameters ($\theta = \{W, B\}$, W and B are weights and biases, respectively) are employed to perform a specific mapping (convolution, deconvolution, dense multiplication, etc.), by utilizing several sets of representations to capture the neurons' responses [18]. Moreover, for each setting among parameters (after each learning iteration or epoch), the gradient is approximated between the input and output by using the first-order partial derivative to optimally fit the model to data. AEs comprise classic [248] (Section 2.3.2.1), denoising [303] (Section 2.3.2.2), contractive [240] (Section 2.3.2.3), sparse [209] (Section 2.3.2.1), VAE [153] (Section 2.3.2.4), and they can also be integrated with other UGL models, for example, when combining GANs with VAE [193] (Section 2.3.3.2).

Among all AEs, the VAE is regularized by VI [336] to optimize the posterior distribution for large datasets, and it outperforms the others AEs in terms of large-scale generalization (when the testing data is larger than the training set). The VAE is utilized in many different fields including object detection, image reconstruction and recognition, compression sensing, and other deep learning tasks [153]. However, ex-

plaining VAEs did not receive an appropriate interest in the literature, where explaining such a UGL model is considered essential to understanding the behaviors of neurons when new data (normal or abnormal) is generated or reconstructed [247, 255]. Thus, different works have been proposed for explaining DL models (including UGL models) through supplementary inputs to carry out specific tasks. However, such works did not explain the behaviors of the models themselves, and they explain the models through offline attention mapping (after learning) on the penultimate layer of the trained model [196, 259, 285, 339, 355].

The first explainable VAE has been introduced in [188], where it generates an attention map by reduplicating the last layer of the encoder, thereafter scaling it up by the global average pooling of the gradient of the latent space concerning that layer. Such attention is seen similar to explaining discriminating models [335] that utilize a copy of the penultimate layer after training the model, then scaling that layer by different scalars, α_i, derived from the global average pooling as in the class activation map (CAM) [355]. Alternatively, different models scale the copied layer by the gradient between that layer and the output layer, such as the gradient weighted class activation mapping (Grad-CAM) [259]. Similar to Grad-CAM, the proposed work in [188] scales the penultimate layer at the encoder side by the gradient between the bottleneck layer, Z, concerning the copied layer. The drawback of such an attention map lies in unfair scaling, i.e., related and unrelated features in the channels of the filters can be scaled with the same factor. Furthermore, in recent works, attention maps are obtained after training the models, neglecting to integrate the attention loss during the learning process to enhance the models' generalizability.

To help explain VAEs, this section proposes Grad_2VAE, a novel XAI model by means of visual attention mapping. Moreover, the Grad_2VAE utilizes the (2^{nd}) derivative between the latent and 1^{st} encoder's layers to obtain the 1^{st} derivative of the gradient, which captures the curvatures of neurons responses that are aggregated to show how the VAE learns data without additional scaling. Also, our attention map is learnable through a novel loss that can be integrated into the VAE optimization to enforce the model to reconstruct an attention map analogous to the input data. Therefore, our contribution is twofold: *(i)* introducing a novel method to explain the VAEs using (2^{nd}) partial derivative between the latent space and the encoding module, and *(ii)* expanding our method to accelerate VAE learning (reduced epochs) and one-class anomaly detection. The rest of this section is organized as follows. Section 4.3.2 highlights the VAE learning and the 2^{nd} derivative interpretation. Section 4.3.3 describes the Grad_2VAE. The experimental results will be given in Section 4.3.4. The conclusion and remarks will be reported in Section 4.3.5.

4.3.2 VAE Learning and Second Order Derivative Interpretation

This section highlights the learning methodology of the vanilla VAE model, which is build based on VI and has been reviewed in Section 2.3.2.4. Moreover, this section introduces our proposed interpretation methodology, where it utilizes the (2^{nd}) partial derivatives to capture the local curvatures among the learned representations.

4.3.2.1 VAE Learning

Similarly to any AE model, the VAE contains two main modules: *(i)* the encoding module (inference side) that is employed to map data, $X = \{x_i \,|\, x_i \in \mathbb{R}^D, i = 1, \ldots, N\}$, D is the original dimensionality, to a latent space, $Z = f(X) = \{z_i = f(x_i) \in \mathbb{R}^d, | \, i = 1, \ldots, M\}$; such a module reduces dimensionality where $0 < d < D$, and it is used to infer the model likelihood $P(X|\theta)$; *(ii)* the decoding module (generation side) that is utilized to generate or reconstruct the original data, $\tilde{X}$, from the latent space Z [2, 3, 128, 130]. For a given data $X \in \mathbb{R}^D$, the encoding module creates a mapping $f : \mathbb{R}^D \rightarrow \mathbb{R}^d$, while the decoding module creates an inverse mapping $g : \mathbb{R}^d \rightarrow \mathbb{R}^D$, which generates an approximation of the original data $\tilde{X} = g(Z; \hat{\theta}_d)$ [2]. The AEs are regulated to find the parameters $(\hat{\theta}_e, \hat{\theta}_d)$ that achieve a better generalization [29], and to obtain the minimum reconstruction loss:

$$\mathcal{L}_{\text{REC}_{\{\hat{\theta}_e, \hat{\theta}_d\}}} = \min \|X - (f \circ g)X\|^2_{\text{Er}} \tag{4.12}$$

where the reconstruction error E_r can be measured by different metrics including mean square error (MSE), Frobenius norm, reconstruction cross-entropy, or β- divergence [1] (see Section 2.3.2 and Section 1).

Among all AE models, VAE is regulated by the VI, and it is optimized based on two different losses that are minimized simultaneously [153]. VI method is one of the Bayesian techniques, which can be utilized to estimate an intractable posterior, $P(Z|X)$, over a big dataset using a simpler variational distribution to obtain the solution to an optimization problem [336], i.e., the VI approximates probability distribution through optimization. By considering the encoder module output, the approximate posterior distribution $Q(Z|X)$ is estimated, which parameterizes the shape of the latent distribution according to the original input data X. Moreover, optimizing $Q(Z|X)$ characterizes the VAE, where it enforces the latent space distribution to follow a unit Gaussian distribution with a certain mean μ (which reflects the center of the Gaussian), and a standard deviation σ (which reflects the Gaussian shape).

Initially, the prior distribution of latent space $P(Z)$ is drawn (simply by copying the unit Gaussian distribution of the data manifold $P(X)$). Thereafter, the approximated distribution $Q(Z|X)$ and the prior $P(Z)$ are compared using the KL divergence [239]. The KL divergence (see Section 1.3 and Eqn. (1.6)) is defined as $\text{KL}(P\|Q) = \Sigma_x P(x) \log \frac{P(x)}{Q(x)}$, which is always positive and tends to zero if and only if P and Q are almost equal. Furthermore, appending noise to $Q(Z|X)$ throughout varying σ by a small value ϵ, and then enforcing the AE to reconstruct the data following the true (not varied one) Gaussian $P(Z)$ is called the reparameterization trick; such a trick generates several different distributions (similarly to duplicate the training data with fusion) that are optimized and compared with prior distribution by the KL divergence, consequently the model can be better generalized for a large-scale testing stage [153]. Finally, the vanilla VAE is optimized as stated by in Eqn. (2.111) to minimize the reconstruction loss according to Eqn. (4.12), and it is also optimized to minimize the latent loss between $Q(Z|X)$ and $P(Z)$ using $\text{KL}(P\|Q)$, which measures to which extent

the reparameterized latent distribution can follow a unit Gaussian:

$$\mathcal{L}_{\text{VAE}_\theta} = \min[\|X - (f \circ g)X\|_{\text{Er}} + \text{KL}(P||Q)] \tag{4.13}$$

where $\text{VAE}_\theta = \{\hat{\theta}_e, \hat{\theta}_d, \hat{\mu}_X, \hat{\sigma}_X, \hat{\mu_Z}, \hat{\sigma_Z}\}$.

4.3.2.2 The second Order (2^{nd}) Derivative Interpretation

The first-order (1^{st}) partial derivative between input and output neurons reflects the 1^{st} gradient, which measures the instantaneous rate of change (velocity or speed), ∂, among model parameters, θ, that are employed to optimally fit ML model [152, 226]. Moreover, if the gradient sign is negative, then it is decreasing (velocity is reduced); however, if the gradient sign is positive, then it is increasing (velocity is accelerated).

For a VAE with an encoding layer L_{e1} and a latent layer Z, the 1^{st} gradient of Z with respect to L_{e1} is computed by carrying out the partial derivative of each neuron z_i as $\frac{\partial z_i}{\partial L_{e1}}$; considering that, if an additional layer L_{e2} acts between L_{e1} and Z, then the chain rule is introduced [9]:

$$\frac{\partial z_i}{\partial L_{e1}} = \frac{\partial z_i}{\partial L_{e2}} \frac{\partial L_{e2}}{\partial L_{e1}} \tag{4.14}$$

The final result of derivations gives all possible rates of changes, which are required to update θ laying between L_{e1} and Z. Because the rate of change, ∂, of a neuron's response (activation) is changing during a period of learning time (over several epochs), thus capturing such variations draws an attention map that gives an insight into how the neurons respond among different inputs (analogous to derive the acceleration for a neuron from the velocity [334]), and it is achieved by considering the derivative of the gradient, i.e., 2^{nd} derivative $\frac{\partial^2 z_i}{\partial L_{e1}^2}$ [85].

Graphically, a neuron response that is modeled by a non-linear ReLU function [208] is given according to Fig. 4.9: the 1^{st} gradient is the slope at a point in the graph (blue curve), whereas the 1^{st} derivative of gradient explains how the slope is changing over time (the red and green points). As it is noticed from Fig. 4.9, the gradient of a neuron response can be steady at a period of the learning time, i.e., the 2^{nd} derivative around the green points is ≈ 0; however, it can change at a different period of time, i.e., the derivative around the red points is $>$ or < 0. Accordingly, utilizing the 2^{nd} derivative which measures how the 1^{st} gradient of neurons responses is changing (as in deriving the acceleration from speed), is able to capture the curvatures of representations and the temporal behavior of neurons when learn data. Moreover, such a strategy is valid for any activation type and can be represented by a learnable attention map, which aggregates all 2^{nd} partial derivative to explain how the latent neurons of Z are activated to the local curves and edges.

Our Grad$_2$VAE employs the 2^{nd} gradient, i.e., the (2^{nd}) partial derivative, between the latent space layer, Z, concerning the first encoding layer to explain the learned representations by the VAE. Also, it exploits such explanations in the application of

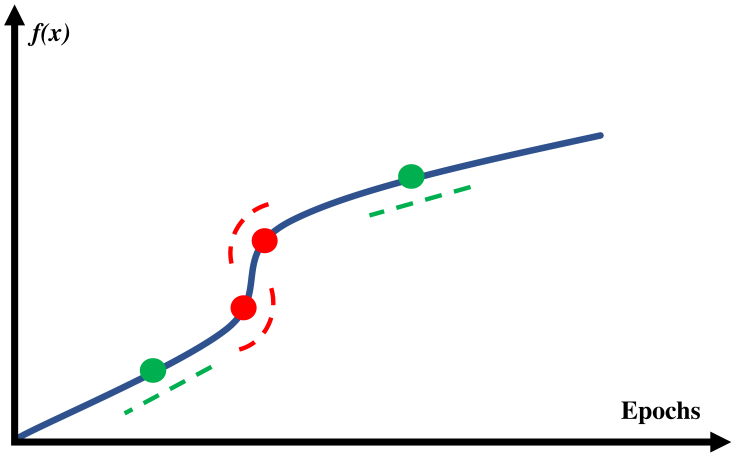


Figure 4.9: The graphical representation of a neuron activation (or response) over time, where the number of epochs reflects the time index. Moreover, the numerical values of the activation are increasing or decreasing (around the red dashed lines), also can be steady during a specific period of learning time (around the green dashed lines).

one-class anomaly detection [247]. Furthermore, we will show how such a strategy is able to accelerate the convergence among the learning parameters θ.

4.3.3 Grad$_2$VAE

Fig. 4.10 shows our proposed Grad$_2$VAE, where it comprises an encoder, decoder, and attention modules. Both encoder and decoder contain one stage of down-sampling (convolution with a stride of 2) and up-sampling (de-convolution with a stride of 2), respectively. Moreover, the size of the first two layers of the encoder and the last two layers of the decoder, are fixed to uniform the dimensionality. Consequently, the obtained attention by the 2$^{\text{nd}}$ derivative can be fused with the $L_{d_{n-1}}$ layer (d_n is the total number of the decoder's layers); such a fusion is seen as a form of residual learning [120], which enforces the Grad$_2$VAE to learn the residual of mapping between the encoder and decoder by utilizing the gradient attention mapping. Accordingly, besides the explainability of the Grad$_2$VAE, it also boosts the reconstruction of data by utilizing the curvatures of representations that are combined with the decoder. Therefore, the Grad$_2$VAE optimizes two losses by using Adam [152] as:

$$\mathcal{L}_{\text{Grad}_2\text{VAE}} = \min[\mathcal{L}_{\text{VAE}} + \|X - \theta_{\text{grad}}(Z, L_{e1})\|^2_{\text{Er}}] \tag{4.15}$$

where the first loss, $\mathcal{L}_{\text{VAE}}$, is taken from the vanilla VAE [153] that is depicted at Eqn. (4.13), and the second loss is the reconstruction loss between the input data and the aggregated attention map that is obtained from the attention module (see Fig. 4.10). Moreover, θ_{grad} represents the 2$^{\text{nd}}$ derivative between each latent neuron, z_i, with respect to L_{e1}, i.e., for each z_i there is a corresponding tensor of size of the

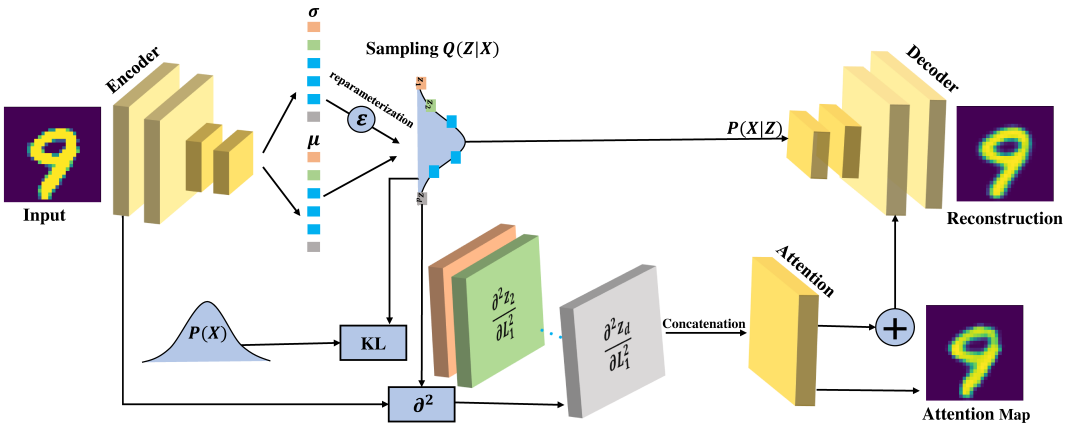


Figure 4.10: The graphical representation of the Grad$_2$VAE model, where it comprises an encoding module (left), a decoding module (right), and an attention module (bottom-middle). The size of the first two layers of the encoders is fixed and is equal to the input size (similarly in the last two decoding layers), i.e., $28 \times 28 \times 16$ for height, width, and filters depth, respectively. Moreover, the model's architecture can be customized to capture the explainability of more depth layers at different scales, considering re-scaling the attention size for sake of optimization. For each neuron in the latent space, Z, a tensor of the size of the intended layer to produce the visual explainability is defined (as the gray tensor), thus allocating all partial derivatives to build an attention map utilizing the local curvatures among the representations. Finally, all attention tensors are concatenated and contracted by using a convolutional layer with a depth typical to the depth of the input layer.

L_{e1} to allocate all derivatives. Additionally, the derivative of the gradient of Z can be implemented concerning all other encoder's layers (as in considering L_{e3}); however, considering more depth layers requires re-scaling the dimensionality, which increases the computational time, and leads to the loss of global characteristics of data and global representations.

4.3.4 Experimental Results

To show the performance of our Grad$_2$VAE, we employ both MNIST digits and MNIST fashion datasets [75, 324]. Each comprises 60k images for training and 10k for testing, divided in 10 classes with image size of 28×28. Moreover, all quantitative analysis experiments are implemented with a batch size of 128, and 100 epochs with a starting learning rate (η) of 0.001. Thereafter, η is reduced after each 50 epochs by a factor of 10^{-2} to search and fine-tune the Grad$_2$VAE parameters, i.e., θ_{VAE}, and θ_{grad}.

4.3.4.1 Grad$_2$VAE Explainability

The explainability of the Grad$_2$VAE model lies in the attention module, which aggregates the derivatives of gradients, and it reflects the curvatures among representations that are obtained at the neurons response level. For each neuron in the latent space, z_i,

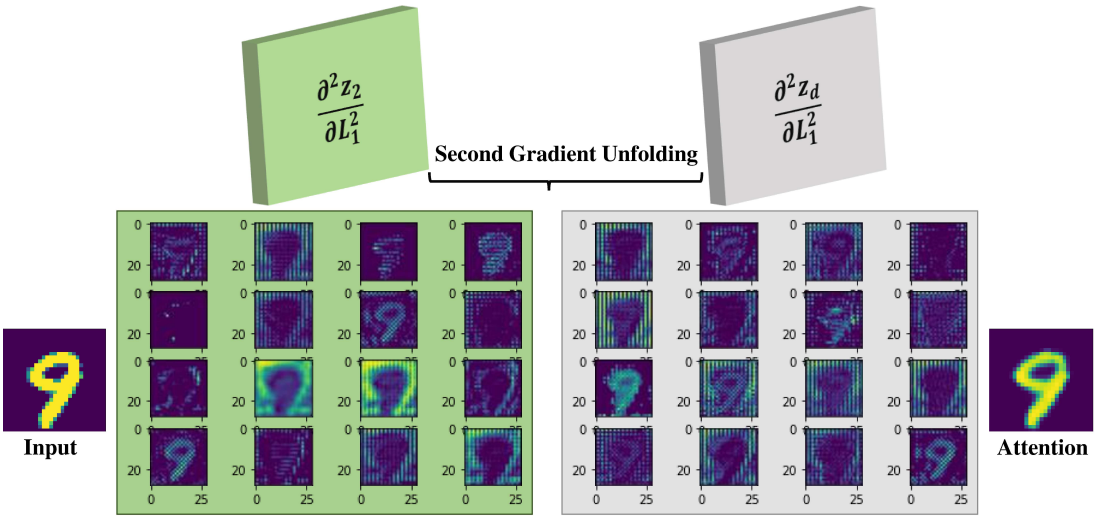


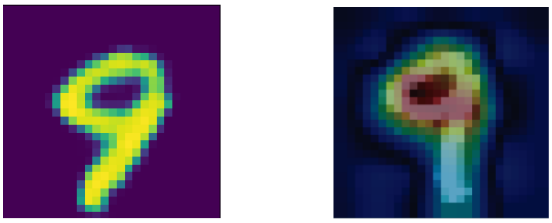
Figure 4.11: The unfolding of the tensor that holds all second-order partial derivatives (Grade_2) of z_2 (green tensor) and z_{16} (gray tensor) concerning L_{e1}, respectively.

the corresponding tensor of 2^{nd} partial derivative is produced and it represents the local curvatures (attention) of neuron activation, where the number of tensors is a function of the latent space dimensions. Thereafter, all tensors can be aggregated by different matrix methods [188] including the addition, mean, convolution, etc.

Fig. 4.11 shows the 2^{nd} partial derivative attentions of the second and last neurons of the latent space, Z, as a function of the depth of the convolutional filters. The Grad_2VAE has been trained to show the explainability for the 10^{th} class of the MNIST digits, by considering 16 neurons for Z, and a depth of 16 filters for each layer in the encoding and decoding modules. Also, Fig. 4.12 depicts a comparison between the aggregated attention of the Grad_2VAE (based on convolutional aggregation) and the attention proposed in [188] (based on the mean aggregation). As it can be noticed from the figure, considering the 2^{nd} partial derivative (a derivative of gradient) offers an explainable attention map that retains all possible curvatures of the representations. Moreover, integrating the attention reconstruction loss in the whole model optimization (see Eqn. (4.15)) leads to a better visual explainability map than offline attention as in [188]; such integration preserves the full spectrum of the characteristics among the data features that are obtained through optimizing the global representations.

4.3.4.2 Grad_2VAE in One Class Anomaly Detection

Anomaly detection (AD) is a branch of ML that characterizes data samples that are misrepresented from what is normal or predicted [255]. One-class AD is referred to as a learning approach, in which only normal data is considered at the training stage, where the ML model learns to classify or reconstruct the normal data only [247]; however, at the testing stage, all data samples that are falling out of the normal class distribution of the trained data (abnormal data and unseen classes) are considered, and the learned ML



(a) The learned attention of Grad$_2$VAE.

(b) The offline attention [188].

Figure 4.12: Grad$_2$VAE learnable attention (Fig. 4.12(a)) vs. the offline attention proposed in [188] that depend on scaling of the penultimate encoding layer by the gradient of Z respecting that layer (Fig. 4.12(b)) when learning the 10^{th} class from MNIST digits.

model must be able to distinguish between normal and anomalies samples concerning a high level of accuracy and confidence.

In the Grad$_2$VAE, the decoder is guided by the curvature of representations that are obtained from the encoder and attention modules, thus the reconstruction process is accelerated and the reconstruction loss is optimized. Accordingly, the Grad$_2$VAE is directly applied to the AD application due to its reconstruction ability, where besides reconstructing the original data, it also reconstructs an attention map of the same input data distribution that gives an insight about learned representations at an early stage, i.e., it detects the anomaly data at an early stage before decoding the encoded data.

In the following, we employ the Grad$_2$VAE in the one-class AD, where we benchmark our model on the MNIST digits and MNIST fashion datasets. Moreover, the average area under the receiver operator characteristic curve (AUC-ROC) index will be considered to show the performance, where we report the qualitative and quantitative comparisons to the recent works.

4.3.4.2.1 Qualitative Analysis Comparison In this section, we visually compare our work with [188] (the only related work in the literature) based on the MNIST dataset. For this analysis, we consider 600 epochs to learn the whole model. Moreover, we used the same experimental setup that has been reported in Section 4.3.3 and Section 4.3.4, i.e., 4 layers for each of the encoder and decoder, the size of the first two layers of the encoder, and the last two layers of the decoder is equal to the input size to reconstruct the output and attention maps, the depth of the convolutional filters is equal to 16, and the model is trained by ADAM optimizer [152] (see Section 4.3.4).

Furthermore, we have followed [188] to train our model considering one class from the training set as a normal class, subsequently test the learned model with all other data classes from the testing set, i.e., out of the training data distribution. Hence, our model must be able to distinguish and produce visual attentions for all testing classes avoiding bias to the previously learned normal classes.

Fig. 4.13 shows the attention maps comparison between our model and [188] when

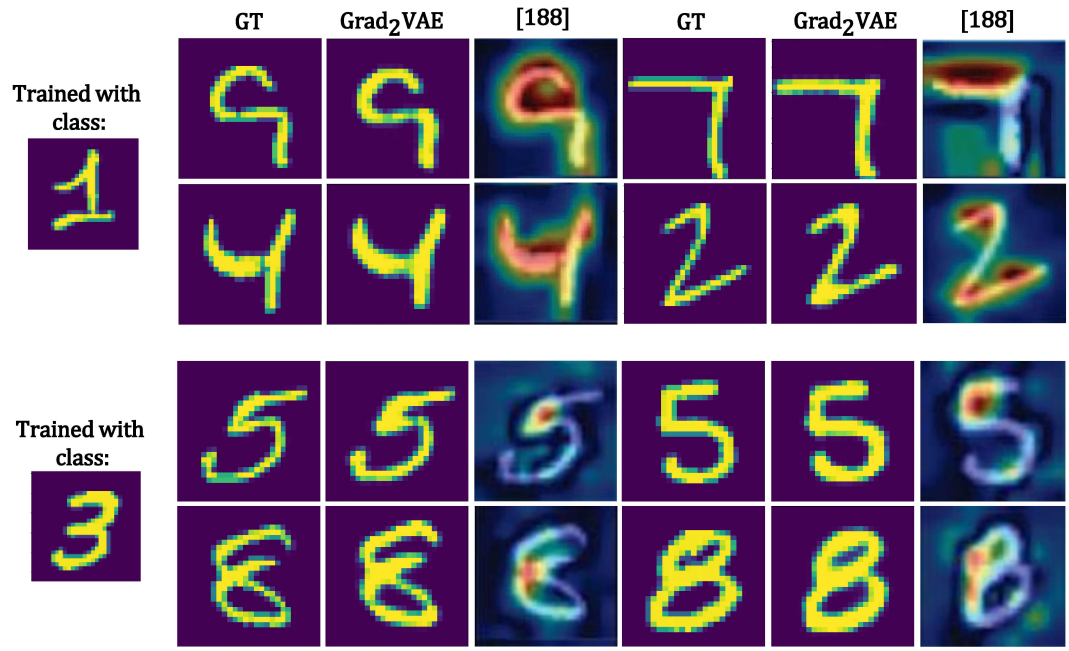


Figure 4.13: The qualitative analysis comparison between our Grad$_2$VAE that produces learnable attention maps (online attention) and the offline attention introduced in [188], where GT represents the ground truth samples. Moreover, the images that appear in the first two rows of the figure have been tested when only the second class of the MNIST digits has been trained, also in the bottom two rows of the figure, the model has been trained considering only the fourth class and tested with all other images from all classes. Also, the attention maps that are produced by Grad$_2$VAE are considered the reconstruction of attention maps, which are optimized through the model learning; however, the attention maps produced by [188] reflect the local heat mapping after learning the model.

both are trained with the 2$^{\text{nd}}$ and 4$^{\text{th}}$ classes from the dataset separately, then considering data from other classes (out of the trained data distribution) as testing samples.

As it can be noticed from Fig. 4.13, our model produces visual attention maps through reconstructing the 2$^{\text{nd}}$ partial derivative that reflects the curvature of neurons responses. Moreover, our proposed model also considers reconstructing the full spectrum attention as part of the optimization objective that minimizes the attention loss, instead of considered the penultimate encoding layer after learning the model and highlighting portions of images through heat mapping. Consequently, our proposed attention mapping method completely retains the curvatures among representations of the input data regardless that they are normal (seen) or anomalies (unseen from other classes). Specifically, our proposed model is able to visually explain the differences between the normal and anomalies samples, avoiding further preprocessing such as scaling attentions maps by the gradient as in [188] to partially detect and explain the anomalies.

Table 4.7: The AUC-ROC metric comparison with recent deep learning models, where the first column shows the utilized datasets, the second column gives the data classes related to each dataset, and all other columns contain the results of the employed models to evaluate the performance. Moreover, each row in the table contains the results when training the model with the class of data that is labeled by the value of the normal class column and lies in that row, subsequently test the model with the testing data from all classes.

Dataset	Normal-class	CAE OCSVM	Deep SVDD	Inception CAENN	$Grad_2$VAE
Digits	0	95.40	99.10	98.70	97.62
	1	97.40	99.70	99.70	96.81
	2	77.60	95.40	96.70	98.84
	3	88.60	95.10	95.20	98.53
	4	83.60	95.90	95.00	97.98
	5	71.30	92.10	95.20	98.70
	6	90.10	98.50	98.30	98.54
	7	87.20	96.20	97.00	98.59
	8	86.50	95.70	96.20	98.09
	9	87.30	97.70	97.00	98.62
	Average	86.50	96.60	96.90	**98.23**
Fashion	T-shirt	88.00	98.80	92.40	96.54
	Trouser	97.30	99.77	98.80	95.88
	Pullover	85.50	93.50	90.00	96.59
	Dress	90.00	94.90	95.00	96.29
	Coat	88.50	95.10	92.00	96.49
	Sandal	87.20	90.40	93.40	96.39
	Shirt	78.80	98.00	85.50	96.65
	Sneaker	97.70	96.00	98.60	96.05
	Bag	85.80	95.40	95.10	96.58
	Boot	98.00	97.60	97.70	96.42
	Average	89.70	95.90	93.90	**96.37**

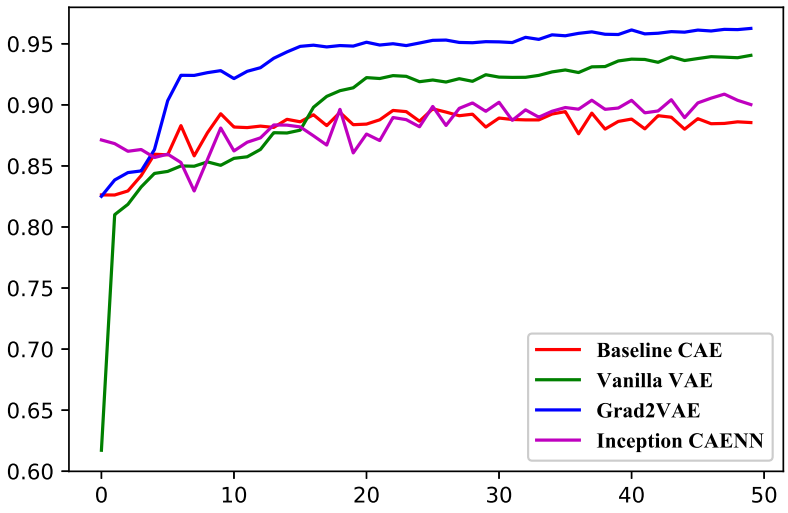
4.3.4.2.2 Quantitative Analysis Comparison In this section, we compare our model with the recent deep UL models dedicated for one class anomaly detection, where we consider the convolutional AE that is built based on the one-class support vector machine (CAE OCSVM), Deep support vector data description (Deep SVDD), and inception-like convolutional AE termed as InceptionCAE NN-QED [247, 255, 283]. Moreover, we followed [255] to train 10 models for each normal class, thus reporting the average AUC-ROC over 10 $Grad_2$VAEs. Additionally, we used 16 and 32 neurons as the bottleneck size for the MNIST digits and MNIST fashion datasets, respectively.

Our results are reported in Table 4.7, considering the same experimental protocol used in Section 4.3.4.2.1 except that only 100 epochs are employed to learn the model; due to the acceleration in the learning ability that the $Grad_2$VAE possess, which is obtained by the residual fusion between the attention map and the reconstruction output.

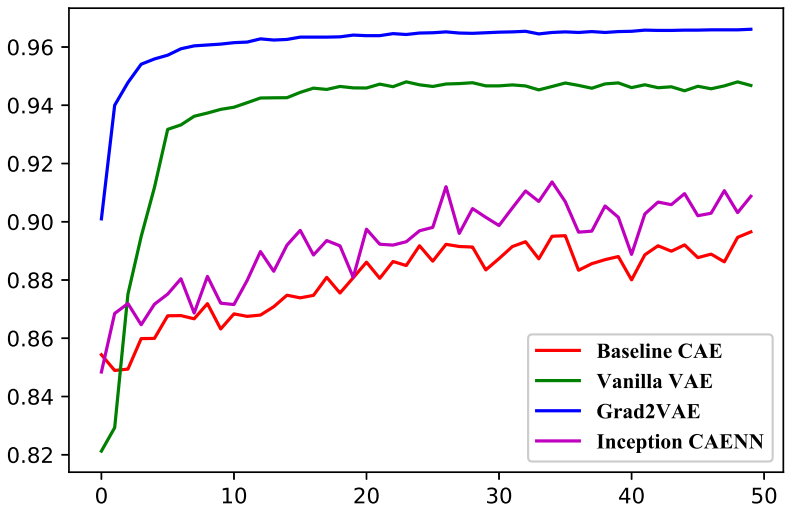
As it can be noticed from Table 4.7, the $Grad_2$VAE outperforms the other deep models under reduced epochs, where all other models have been trained considering 150 epochs [255]. Moreover, the $Grad_2$VAE does not show any bias to a class against all other classes, e.g., the Deep SVDD and Inception-CAENN models learn the 2^{nd} class from the fashion dataset perfectly; however, they show minimum accuracies for the 6^{th} and 7^{th} classes, respectively. Finally, the $Grad_2$VAE model shows a better mean standard deviation (mstd) among the averaged results of 10 models (or 10 runs), where our maximum mstd for both datasets did not exceed 0.117, whereas it reached 3.8, 3.9 for the Deep SVDD and Inception-CAENN models, respectively [255].

To compare the learning complexity and performance convergence of our proposed $Grad_2$VAE model with the related models in the literature, we consider the baseline convolution AE (Baseline CAE), vanilla VAE, and Inception CAENN models [3, 255]. Moreover, under the same experimental setup reported in Section 4.3.4, the Baseline CAE learns 266 K, vanilla VAE learns 3.254 M, and Inception CAENN learns $334,597$ K parameters to complete one training epoch. Additionally, our proposed $Grad_2$VAE learns 3.380 M parameters as a consequence of the online attention mapping module and the 2^{nd} order partial derivative parameters.

Fig. 4.14 depicts the learning acceleration (convergence) ability over the learning epochs of our proposed $Grad_2$VAE model considering the related models in the literature. As it is noticed from the figure, our proposed $Grad_2$VAE shows the highest learning convergence ability at an early stage of the learning period, i.e., our proposed model is able to search and find the suitable learning parameters, $(\theta = \{W, B\})$, at an initial interval of the learning epochs, which accelerate the model learning and prevent the model from overfitting. That is by considering the XAI attention maps, which are fused with the penultimate layer to compensate for the loss caused by encoding and decoding operations. Finally, our model learns a number of parameters that are equal to 12.7 x, 1.03 x, 10.10 x of the Baseline CAE, vanilla VAE, and Inception CAENN models, respectively. However, it converges to an accuracy of 0.9500 for both datasets considering the half number of the learning epochs required to learn the vanilla VAE, and the other remaining models did not reach such accuracy in the first 50 epochs. Accordingly, our proposed model converges to the optimal set of learning parameters under a limited number of learning epochs, and it also outperforms all related models in the literature considering both datasets.



(a) The performance over epochs for the MNIST digits dataset.



(b) The performance over epochs for the MNIST fashion dataset.

Figure 4.14: AUC metric over the first 50 epochs for MNIST Digits (Fig. 4.14(a)) and MNIST fashion (Fig. 4.14(b)), where the red curves represent the performance plot of the Baseline CAE, the green curves show the performance plot of the vanilla VAE, the magenta curves reflect the performance plot of the Inception CAENN, and the blue curves represent the performance of our proposed $Grad_2$VAE.

4.3.5 Summary

We proposed an explainable VAE model termed (Grad_2VAE) to be utilized for XAI, image reconstruction, generation, object detection, and anomaly detection applications. We used the 2^{nd} partial derivative of the neuron activation (or responses) between the latent space, Z, and the 1^{st} encoding layer to capture the curvatures of the representations at an early stage. Our proposed model can be expanded for different data types and scales, it also accelerates the learning process by boosting the whole reconstruction process through the residual fusion.

Furthermore, we employed our proposed model to explain the learned representations through a learnable attention mapping, where it shows a better visual explainability than the related works based on offline attention mapping. Furthermore, we generalized our proposed model in the one-class anomaly detection. Our model outperforms all related deep models in both qualitative and quantitative analysis.

In the following Section 4.4, we will expand the utilization of the second-order gradient (or Grad_2) attention mapping for autonomous driving applications, where we will propose a novel semantic image segmentation method based on our gradient attention mapping. Specifically, utilizing the VAE model for explainable semantic image segmentation in the application of autonomous driving systems will be highlighted, throughout benefiting from the state-of-the-art datasets and adapting the architecture of our Grad_2VAE model to meet the application of explainable semantic image segmentation.

4.4 Multiscale Variational Attention for Explainable Autonomous Driving Systems

Explainable autonomous driving systems (EADS) are emerging recently as a combination of XAI and vehicular automation (VA). EADS explains events, ambient environments, and engine operations of an autonomous driving vehicular, and it also delivers explainable results in an orderly manner. Explainable semantic segmentation (ESS) plays an essential role in building EADS, where it offers visual attention that helps the drivers to be aware of the ambient objects irrespective if they are roads, pedestrians, animals, or other objects. Moreover, ESS assists the ML model developers to realize the behaviors of the learned models before distributing the models for testing, where it builds a level of confidence to understand each stage (or layer) of ML and DL models.

In this section, we propose the first ESS model for EADS based on the VAE, and it uses the multiscale second-order derivatives between the latent space, Z, and the encoder layers to capture the curvatures of the neurons' responses. Our model is termed as Mgrad$_2$VAE, and it is considered an extension of the Grad$_2$VAE model proposed in Section 4.3 for EADS. Furthermore, our proposed Mgrad$_2$VAE is bench-marked on the SYNTHIA and A2D2 datasets, where it outperforms the recent models in terms of image segmentation metrics.

4.4.1 Introduction to EADS and VAEs

The rapid advancement of AI and ML has lead to the development of AI-powered autonomous systems, which can sense, learn, decide and interact for many different applications including computer vision, NLP, robotics, autonomous driving, and other fields[3, 4, 109]. Moreover, AI-powered autonomous systems are build based on DL models comprising CNNs, AEs, GANs, and Bayesian models [14] (see Section 1 and Section 2.3). However, the effectiveness of many recent models and systems are limited due to the scarcity of explainability; such an explainability translates the actions and decisions of the learned models to users who operate and develop them. XAI is a branch of AI that aims to explain the behaviors of the ML models [113].

An autonomous driving system (ADS) is referred to any vehicle that can sense the surrounded environment without human control, or with a limited level of supervision. ADS is also able to control engines, visualize objects, detect abnormal actions, drive vehicles, and activate breaks [311]. AI and ML influence ADS by automatically processing data, offering instantaneous recommendations, and recognizing objects; such objects include pedestrians, trees, bicyclers, and other moving and static objects [53, 81]. Explainable autonomous driving systems (EADS) combine XAI and ADS to enhance the vehicular automation (VA), throughout interpreting sensory data, mentoring vehicles behaviors, and semantically segmenting the ambient objects [4, 113]. In this regard, the explainable semantic segmentation (ESS) is a branch of the ML in which each pixel of the segmented object holds a semantic meaning, and can be integrated into the EADS to improve the explainability of the detected objects, and to offer roads conditions conclusion to the drivers [4, 95].

XAI-powered models are associated with UGL to visualize the hidden structure of data and learn with limited levels of prior assumptions [1, 3]. AEs are a class of UGL methods that are able to generate and visualize data, reduce dimensionality, and perform other ML tasks such as object recognition [2]. AEs comprise classic, de-noising, contractive, sparse, variational-AE (VAE) (see Section 2.3.2). Moreover, the success of AEs architectures led to the flourishing of different supervised AEs for structured prediction, i.e., semantic segmentation, such as Seg-net, U-net, and others [29, 199, 241]. Among all AEs, VAE is regulated by the variational inference (VI) to optimize the posterior distribution of large datasets, which leads to a better generalization. The VAEs have been utilized in the ADS in the absence of XAI, where it has been used in the steering control [10], pedestrian prediction in [224], trajectory simulation in [55], and anomaly detection for ADS [279].

The first work towards explaining the VAE behavior is proposed in [188] and highlighted in Section 4.3.1, where it generates visual attention to show how the encoder side behaves. Moreover, the proposed attention map is built by copying the last layer of the encoder, thereafter it scales each feature point in the filter channels by a global average pooling of the gradient of the latent space concerning that layer. Actually, the drawback of such attention lies in the unfair scaling, i.e., both related and unrelated feature points are scaled with the same factor. Moreover, such an explainability is built in an offline learning mode (after carrying out model learning), where it avoids employing the attention mapping in the optimization objective to improve the model performance and offer a better visual explainability. On the other hand, the first work that has been attempted to build the attention of the CNN for ADS is described in [53], where the visual attention is built by averaging the activations of 100 images; such attention hides the effects of the high and low activations, i.e., approximated attention, and is not stable for time-series segmentation.

To fill the gap of explaining VAEs in the EADS applications, we propose Mgrad$_2$VAE, a novel ESS model for EADS applications. Moreover, the Mgrad$_2$VAE utilizes the multiscale second-order derivative between the latent space, Z, and each encoder layer, which captures the curvatures of neurons' activations to build online multiscale explainable attentions without unfair scaling or averaging the final attention. Therefore, our contribution is twofold: *(i)* introducing a novel ESS model for EADS applications, by using the unsupervised VI and a supervised convolutional AE, and *(ii)* proposing a novel multiscale gradient attention mapping scheme for ESS to improve EADS applications using the second-order derivative operator. The rest of this section is organized as follows. Section 4.4.2 recaps the explainability methodology that presented in Section 4.3.2.2 for ESS, and it describes our proposed Mgrad$_2$VAE model. Section 4.4.3 gives the experimental results. The conclusion and remarks will be reported in Section 4.4.5.

4.4.2 Grad$_2$ Attention in Semantic Segmentation and Mgrad$_2$VAE

Deep semantic segmentation models comprise many different encoding and decoding blocks to map data from the domain of the original image to the corresponding masks

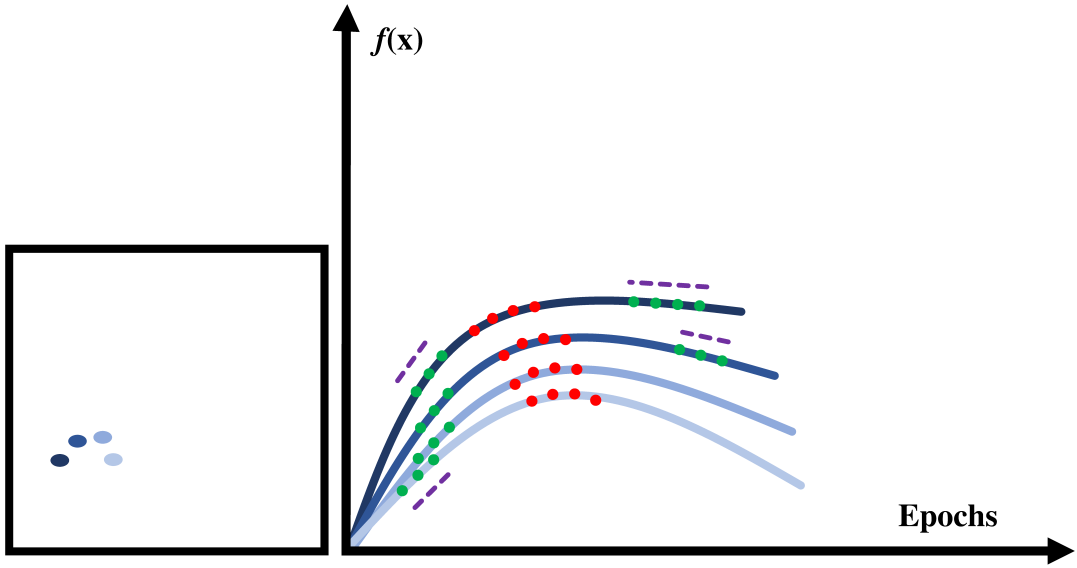


Figure 4.15: Neurons activations and gradient over learning epochs. Four pixels (on the left side) appear with different intensities and the corresponding activations over learning time (on the right side). Moreover, the magenta dashed lines represent the slope line (to measure the gradient) to the activations curves. The green points show that the gradients of activations are steady; however, the gradients of neuron activations are changing around the green points.

[199, 241]. Moreover, neurons with different parameters ($\theta = \{W, B\}$, W and B are weights and biases, respectively) are employed to optimally fit models. Moreover, at each learning epoch, the gradient that measures the instantaneous rate of change among the model parameters is measured, by utilizing the first-order partial derivative, ∂, between each pixel in the segmented mask concerning the input image [152]. Following the same methodology presented in Section 4.3.2.2, the variation in the rate of change, ∂, for each associated neuron with the segmentation mask (or model output) can be captured by the derivative of the gradient, i.e., the second-order partial derivative, between the input and output. Accordingly, the variation of gradient reflects the curvature of the learned representations and neuron activations, which are aggregated and learned to build explainable visual attention maps.

Visually, four pixels of an image with their associated neurons activations are illustrated in Fig. 4.15, where the activations are given according to the non-linear ReLU functions [208] (the method is valid for other types of activations, and it is valid for any neuron in the depth layers). The 1^{st} gradient is the slope (magenta dashed lines) at any point in the curves (blue curves), where the derivative of the gradient interprets how the curves are varied during a time (the red and green points). As it is observed from Fig. 4.15, the gradient of activations can be stationary during a period of the learning time, i.e., the 2^{nd} derivative around the green points is ≈ 0; however, it can vary at

a different period of time, i.e., the 2^{nd} derivative around the red points is $>$ or $<$ 0. Accordingly, utilizing the 2^{nd} derivative which measures how the 1^{st} gradient of the activations of the neurons are changing, is able to capture the temporal behaviors of the neurons (as in deriving the acceleration from speed) which reflects the curvatures of learned representations[4].

Due to the VI, the latent space Z hides many different representations that are generated to regularize the VAE (see Section 4.3.2.1); such representations assist in building ESS attention utilizing the behavior of the neurons' activations. To build a visual attention map, our Mgrad$_2$VAE aggregates all multiscale second-order derivatives (2^{nd} gradient) of the activations of the latent layer, Z, concerning each encoding layer, which represents a different scale of dimensionality. For a better visual explanation, our proposed attention map is enforced to follow the original mask distribution, by minimizing the reconstruction and KL losses between the reconstructed mask, attention map, and original mask, simultaneously. Consequently, our proposed Mgrad$_2$VAE reconstructs learnable attention maps in an online fashion, i.e. during learning time, unlike the offline attention maps that are obtained from the penultimate layer after carrying out model learning [53, 188].

4.4.2.1 Mgrad$_2$VAE

Fig. 4.16 shows our proposed Mgrad$_2$VAE which is built based on modifying the architecture and optimization objective of the Grad$_2$VAE (see Section 4.3.2.1 and Section 4.3.3). Moreover, Mgrad$_2$VAE encompasses encoder, decoder, and attention modules. The encoder and the decoder jointly include three stages of down-scaling (convolutional neurons with a stride of 2 to reduce the dimensionality, i.e., re-scaling) and up-scaling (de-convolutional neurons with a stride of 2), respectively. Furthermore, the Mgrad$_2$VAE visually explains the learned representations by reconstructing online attention maps utilizing the 2^{nd} gradient attention at each encoding scale, i.e., for each encoder's layer there will be a corresponding visual attention map that reflects explainability at that layer, and each attention is enforced to follow the mask distribution to help to contract the representations to the mapped mask.

Moreover, for each layer, the dimensionality of the tensor that holds all partial derivatives of the gradient is re-scaled for sake of optimization to match the mask size ("Stride" in Fig. 4.16). Thereafter, all attention maps are aggregated and fused with the $L_{d_{n-1}}$ layer (d_n is the total number of the decoder's layers); such a combination is considered as a novel form of the residual learning [120], which enforces the Mgrad$_2$VAE to learn the residual of mapping between the images and masks by using the 2^{nd} gradient attention. Consequently, besides the explainability of the Mgrad$_2$VAE, it also assists in mask reconstruction by employing the curvatures of activations that are fused to the decoder. Accordingly, the Mgrad$_2$VAE optimizes two losses by using Adam optimizer [152] as:

$$\mathbf{L}_{\text{Mgrad}_2\text{VAE}} = \min[\mathbf{L}_{\text{VAE}} + \|X - \theta_{\text{Mgrad}}(Z, L_{e_i})\|^2_{\text{Er}}] \tag{4.16}$$

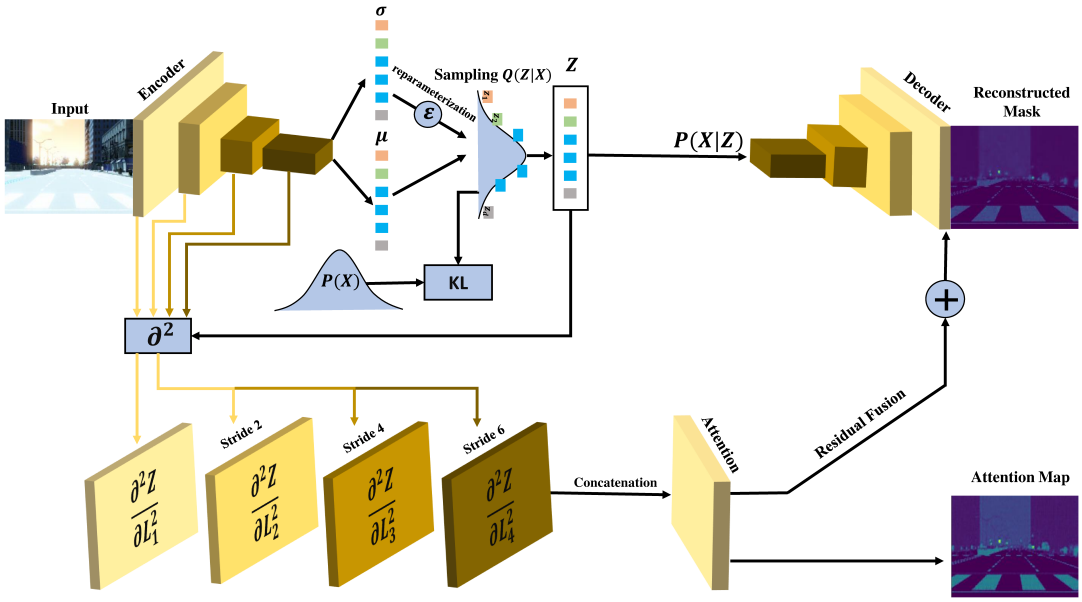


Figure 4.16: The graphical representation of the Mgrad$_2$VAE model, where it encompasses encoder (left side), decoder (right side), and attention modules (in the bottom). The size of the input layer is equal to the image dimensionality and its depth of 8 filters. Moreover, the dimensionality (except the depth) is reduced three times at the encoder side (convolution with a stride of 2 and double filters' depth); however, the size of each layer is re-scaled up at the decoder side. At each encoding a tensor is defined to store all second-order derivatives and it is re-scaled up to match the dimensionality of the target "mask".

where the first loss is obtained from the vanilla VAE [153] that has been introduced in Section 4.3.2.1 and Eqn. (4.13), and the second loss is the reconstruction loss between the original mask and the aggregated attention at the attention module (see Fig. 4.16). Furthermore, θ_{Mgrad} reflects the 2^{nd} derivative parameters between the latent space, Z, concerning all encoder layers L_{e_i}, i.e., for each layer, there will be a corresponding tensor of the size of that layer to allocate all partial derivatives, and the final tensor holds the multiscale-aggregated attention. Additionally, the model is trained to minimize the loss between each mapped image and its corresponding segmentation mask, and it also optimizes the loss between each attention map that is obtained at a different scale with the same mask; such an optimization enforces all encoder layers to contract to the same data, and it compensates the encoding loss that is raised from down-scaling the dimensionality in the depth layers.

4.4.3 Experimental Results

To show the performance of our proposed Mgrad$_2$VAE, we used a collection of SYNTHIA [26] and A2D2 [97] datasets, which contain high-resolution image samples for ADS.

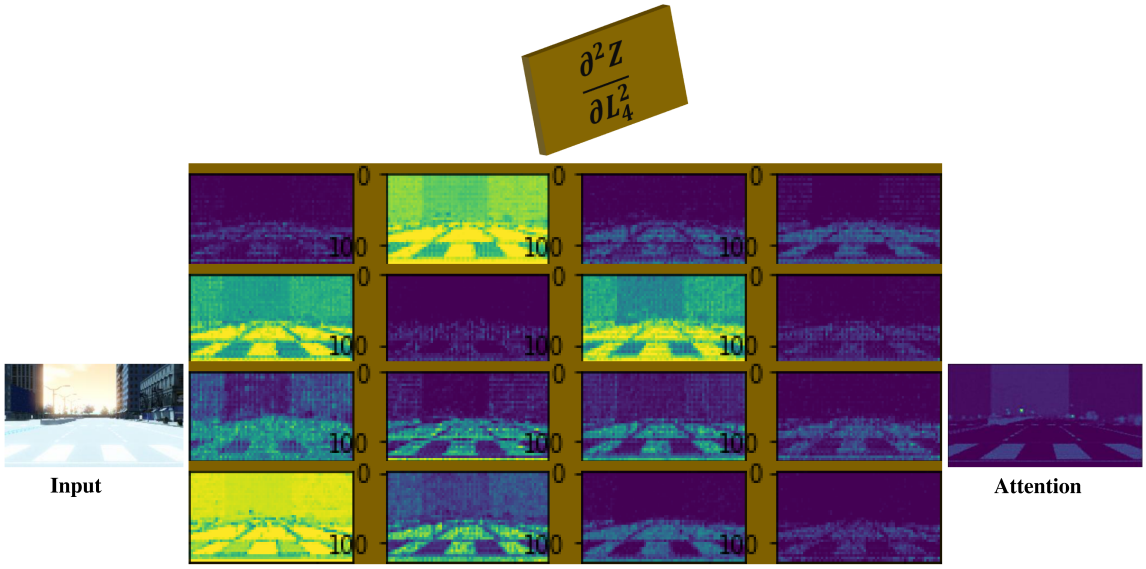


Figure 4.17: Unfolding of the tensor that holds all second-order partial derivatives (Grad_2) of the latent space, Z, concerning the last encoding layer.

Specifically, 5600 samples are categorized to the corresponding semantic classes that have been employed. Moreover, the dataset partition to the training and testing subsets complies with 75 : 25 protocol, i.e., 75% and 25% of the original data size are the training and testing subsets, respectively. For the sake of computation, in the qualitative analysis, the Mgrad_2VAE considers an input layer of the size of $128 \times 256 \times 3$, where the output layer of the size of $128 \times 256 \times 1$. For all experimental works, we consider a minibatch size of 16, and 600 epochs with a learning rate $\eta = 0.001$, where the η is decreased every 100 epoch by a factor of 10^{-2} to search and fine-tune the Mgrad_2VAE parameters, i.e., θ_{VAE} and θ_{Mgrad}.

4.4.3.1 Qualitative Analysis Comparison

Our Mgrad_2VAE visually explains the learned representations at the neurons activations level through reconstructing attention maps at different encoding scale, where it considers the 1^{st} derivative of the gradient (i.e., the 2^{nd} order derivative of neurons activations) between the latent space, Z, and the encoder layers. For each encoding layer (with a different scale), our proposed model produces a tensor to allocate all second-order partial derivatives, and the final attention map can be obtained by concatenating and aggregating all corresponding tensors by using a convolutional layer (see Section 4.3.4.1). Fig. 4.17 shows the corresponding tensor unfolding (of an image from SYNTHIA dataset) of the learned attention that is obtained from the last encoding layer, L_{e_4}, which represents the last encoding scale as a function of 16 filters depth.

Furthermore, Fig. 4.18 depicts the final aggregated attention of all encoding layers,

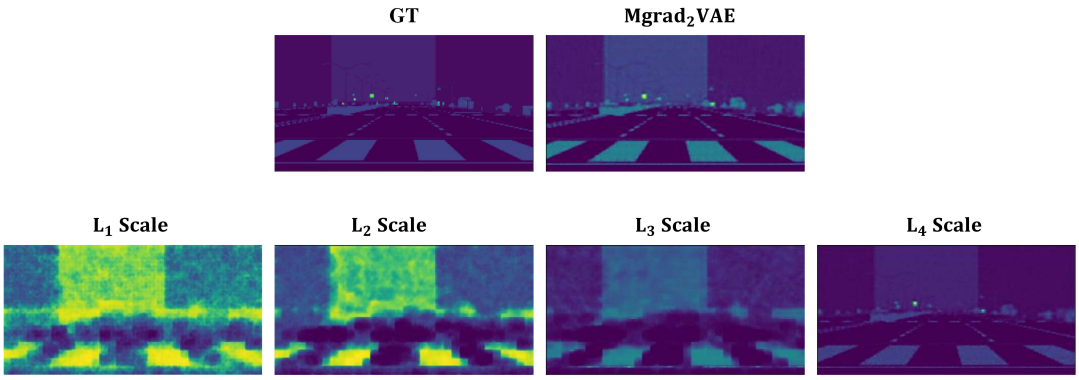


Figure 4.18: Multiscale attention of our proposed Mgrad$_2$VAE (top right), where GT represents the ground truth mask (top left). Each encoding layer is associated with an attention map in the bottom, where four encodin scale are considered in our model with four attention are depicted.

Table 4.8: SSIM index of the reconstructed masks and attention maps concerning GT masks, where all selected samples (5600) from both datasets are employed to report the reconstruction quality of the mapped masks and attentions.

SSIM Index	SYNTHIA	A2D2
Reconstructed masks	97.57%	60.38%
Attention maps	96.47%	55.71%

where it shows how our model can visually explain the global characteristics of the learned representations at an early stage (L_{e_1}). Moreover, it is also able to show the local characteristics among representations that are captured from the fine-grained features in the depth layers (L_{e_4}).

Fig. 4.19 shows different examples from the testing set, ground-truth (GT) masks, reconstructed masks, and the attention maps obtained by our Mgrad$_2$VAE. As it can be noticed from Fig. 4.18 and Fig. 4.19, all attention maps which are obtained from our Mgrad$_2$VAE are contracted to the ground truth mask distribution (target domain), and they jointly utilize the multiscale attention mapping (attention at each layer) to build a complimentary map for a better visual explainability.

To assess the semantic structure qualitatively, we employ the SSIM index [310] for both datasets. Moreover, we report the semantic structure similarities between the ground-truth masks, the corresponding reconstructed masks, and the attention maps in Table 4.8. As it can be noticed from Table 4.8, the Mgrad$_2$VAE produces an attention map (for each sample) that preserves a similar SSIM index for the reconstructed mask by the decoder, which confirms our methodology and reflects the high quality of the produced learnable attentions.

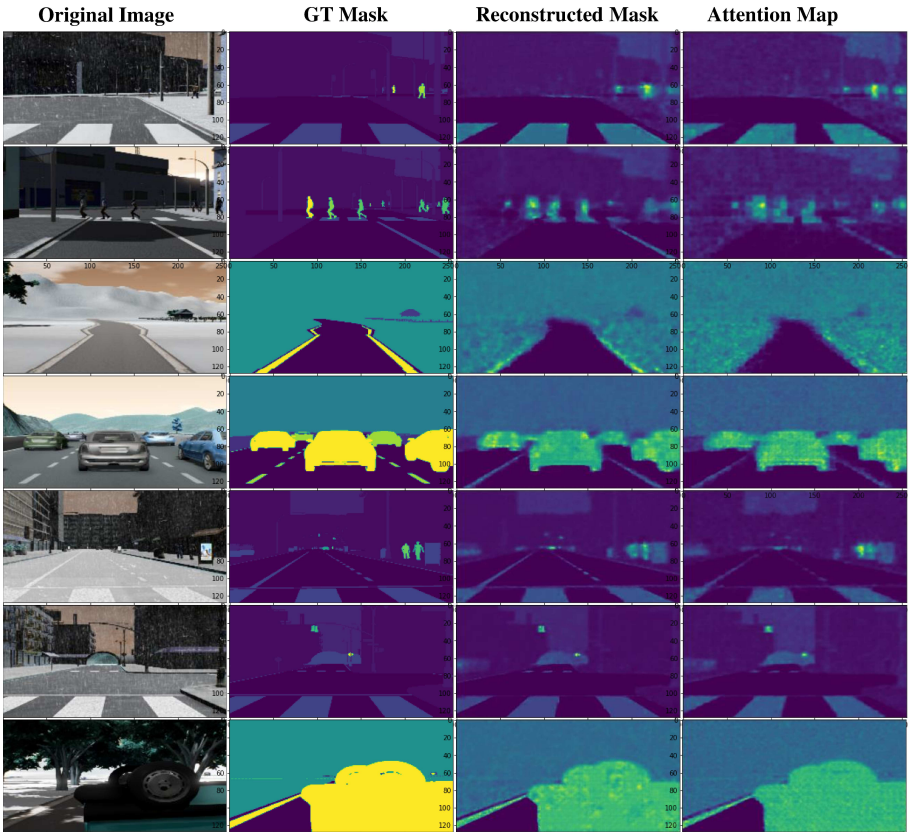


Figure 4.19: Examples of the reconstructed masks and attention maps for the testing set samples, where the original images, GT masks, reconstructed masks, and the Mgrad$_2$VAE attention maps are illustrated from left to right, respectively.

4.4.3.2 Quantitative Analysis Comparison

Table 4.9 reports the pixel-wise predictive performance of our proposed Mgrad$_2$VAE, where we consider AUC-ROC index (as in Section 4.3.4.2.2) that reflects an aggregated measure of each pixel classification (in the reconstructed masks or attention maps) accuracy. Moreover, we consider the same experimental setup that is reported in section 4.4.3. For the sake of numerical stability, the depth of the output layers at the decoder and attention modules have been adapted from $128 \times 256 \times 1$ to $128 \times 256 \times 3$.

As it can be observed from Table 4.9, our proposed model offers high performance at the pixel-level classification for both the reconstructed masks and attention maps. Moreover, our attention mapping method outperforms the reconstruction obtained from the decoder side in terms of pixel-level classification in the SYNTHIA dataset.

Table 4.9: AUC-ROC metric of the reconstructed masks and the attention maps.

AUC-ROC	SYNTHIA	A2D2
Reconstructed masks	81.50%	95.44%
Attention maps	83.20%	95.36%

Table 4.10: AUC-ROC comparison with recent deep semantic segmentation models.

AUC-ROC	SYNTHIA	A2D2
Deep VAE [153]	79.60%	94.05%
Xception [60]	67.43%	95.19%
Our Mgrad$_2$VAE reconstruction	81.50%	95.44%
Our Mgrad$_2$VAE attention	83.20%	95.36%

4.4.4 Recent Work Comparison

In this section, we compare our proposed Mgrad$_2$VAE model with the recent deep learning models, where we consider the deep VAE [153], and the Xception model [60] that has been built based on the U-net architecture [241] and trained on ImageNet dataset [161]. Moreover, we summarize the AUC-ROC metric between the GT masks and the reconstructed masks among all models in Table 4.10.

As it can be seen from Table 4.10, our proposed Mgrad$_2$VAE model outperforms all other models in reconstructing masks and attention maps. Moreover, although the reconstruction module of our model is typical to the Deep VAE [153], the reconstruction performance of the Mgrad$_2$VAE is better than [153] by 1.90% and 1.39% for the SYNTHIA and A2D2 datasets, respectively, because of the residual fusion between the decoder and attention modules of our model.

4.4.5 Summary

We proposed an explainable VAE model termed as the (Mgrad$_2$VAE) to be utilized for ESS and EADS applications. Our model uses the multiscale second-order gradient (Grad$_2$) attention at each encoding layer, unlike the Grad$_2$VAE model proposed in Section 4.3 and produces attention maps concerning only one encoding layer. Our proposed Mgrad$_2$VAE offers attention maps utilizing the second-order partial derivative at each layer to capture both global and local behaviors of the learned representations, throughout aggregating the curvatures of neurons' activations and then learn the final attention map to contract to the target domain.

Furthermore, we employed our proposed model to visually explain the behaviors of the latent space of the VAE, concerning the application of ESS in EADS by employing state-of-the-art semantic image segmentation datasets. Our proposed model outperforms all related deep semantic image segmentation models based on the pixel-wise predictive performance obtained by the AUC-ROC metric.

5

Conclusion and Future Works

5.1 Conclusion

The objectives of this thesis were to propose innovative methods which integrate UGL and XAI approaches to improve the generalizability and explainability of ML models by considering dimensionality reduction and visual explanation. That is by enabling data factorization and the visual attention mapping of the learned representations in modern ML applications such as anomaly detection and autonomous driving systems.

Accordingly, the first contribution was to review the development of UGL methods utilized for data analysis, which are employed for representation learning and XAI applications. Moreover, three families of UGL models including BSS, MfL, and NNs which aim to capture, learn, visualize the manifold, and extract the representations from data were considered. Also, the investigated methods in this thesis were chosen to satisfy the demand of modern data trends in which the data size is exploding, and their dimensions are high-order tensors. We aimed to help interested researchers be aware of the development and recent research directions of the UGL models, also how their role can affect the next generation of ML models and XAI applications. From this perspective "the current state of research", it is concluded that real-time data processing has not received appropriate attention, where the ML model can be learned to be generalized for large-scale testing and production stages. Also, the explainability of recent UGL models used for modern ML applications is still challenging and more investigation is required, specifically, when utilizing the online explainability (at each learning epoch) to improve the general model performance.

Consequently, the second contribution of this thesis was to improve the generalizability of ML models, in which more samples were considered in the testing stage than the training to simulate a reliable production stage. Initially, an innovative NMF rank truncation method was proposed for unsupervised image dimensionality reduction. Moreover, β-NMF was utilized to perform the factorization based on the proposed rank threshold, which was able to identify the appropriate dimensions of the reduced subspaces (W, H) that retained less noise and high information. Another novel method was introduced based on the combination between β-NMF and a dual autoencoder

model for unsupervised hierarchical dimensionality reduction; such a combination was able to draw conciseness prior to being learned, and it was also able to extract rooted representations among data that enable more generalization.

Additionally, the proposed methods were evaluated to learn with limited available data and computational resources, in which the dimensionality was reduced, and the representations were learned hierarchically to meet the limit of the available resources, i.e., data and computational environment. The proposed methods achieved superior results concerning the-state-of-the art models and the publicly available datasets, respecting the image reconstruction metrics, processing time, availability of data, and computational complexity.

A further contribution in this thesis was to improve the visual explainability of UGL models which are utilized in modern ML applications as in the application of anomaly detection. Specifically, this thesis proposed the Grad_2VAE model, an XAI model which concerns the visual explainability of the autoencoder models, especially VAE. By considering the second-order partial derivative of the bottleneck layer of the VAE respecting the encoding layers, the proposed XAI method was able to offer visual attention maps; such maps were obtained by aggregating all partial derivatives, thereafter, learning the proposed model to minimize the loss between the attention maps and the input data to obtain the full spectrum maps (irrespective of the input data). The proposed XAI method was evaluated in the application of one-class anomaly detection, and it outperformed all deep learning modes that were dedicated to the specified task.

Moreover, the proposed visual attention mapping method was utilized in other application domains, where it was used for semantic image segmentation for autonomous driving systems. Therefore, a novel XAI model termed Mgrad_2VAE was proposed in this thesis for multi-scale semantic image segmentation introduced in the context of autonomous driving. Also, the proposed model was considered an extension of the Grad_2VAE, where it built a separated attention map for each encoding layer to visualize the learned representations at different scales. Concerning the state-of-the-art models and datasets, the proposed model obtained higher accuracy by considering the image segmentation and image quality metrics.

5.2 Future Works

Factually most of the actions, events, and phenomena that are considered sources of data in the real world act in an unsupervised fashion. Additionally, such data sources adapt hierarchically from an initial state (compact space) to an advanced state (scattered space with more details), which motivates building ML models based on UGL modeling. That is due to the ability of UGL models to reduce dimensionality and facilitate learning among the representations. Accordingly, this thesis introduces innovative methods based on UGL modeling for dimensionality reduction and modern XAI applications, where the obtained results confirm the potential of the proposed methods to meet the reliability and achieve superior accuracy. However, many different research issues are still open and challenging, where they must be considered in future works.

In the following, the current research directions and open issues are discussed:

- Learning existing models based on modern data considerations is still challenging, and building UGL models for real-time data streaming is still missing. Moreover, from the analyzed literature, an important discrepancy can be found: whereas the size of datasets is currently exploding in the big data era, the majority of recent works on practical methods (supervised and unsupervised) 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 in the existing works.

- Learning from high-order tensorial datasets (or higher-dimensional datasets) will receive further investigations, where it can be initiated by developing conscientious approaches for decomposing and disentangling the main representative factors (or exploratory factors) from such datasets. Consequently, accelerate learning among representations can be obtained through carrying out the learning procedure among the extracted factors (or the compact space).

- Developing countermeasures to deal with learning in cases of data incompleteness and online streaming will be considered in future works. Besides the definition of suitable evaluation metrics for evaluating the learned (or extracted) representations and overall model performance in a standardized manner, thus enabling reproducibility and reducing model complexity.

- Data factorization, or decomposition, shows its ability to reduce the dimensionality and to extract rooted representations among data. Specifically, NMF is considered a promising method that retains the non-negative factors and coefficients in its decomposed subspaces, hence NMF can extract factors and representations which are part of the original data and can be interpreted easily; however, the realization of the ability of NMF to accelerate and explain deep learning models is still missing. In this regard, NMF is able to boost the learning process of deep learning models by reducing the dimensionality and utilizing its subspaces as features for classification or regression tasks, e.g., accelerating learning from high-dimensional datasets and reducing the computational complexity can be achieved by utilizing autoencoder models to encode W, or H, to compress the weights, i.e. partial learning. On the other hand, NMF is able to visualize the representations through factorizing the learned representations at different layers, which reflect the explainability at different levels of abstraction spaces.

- The proposed method for limited data learning (Section 4.2) has been proposed for 2-D images, where the proposed method has shown its ability to reduce the computational complexity and obtain the highest performance in terms of image reconstruction. Notwithstanding, generalizing the proposed method for large-scale and high depth images (such as RGB and hyperspectral images) is still challenging. In this context, such a challenge can be overcome by building high-order models based on n-way factorization and autoencoding methods, where

n-models are jointly learned according to the depth of used images (the depth of each image identify the number of models).

- The development of visual attention mapping has led to the flourishing of the XAI models, which attempted to explain the learned representations depending on the neurons' activation level. Among all models, Grad_2VAE model (Section 4.3) is a novel XAI model that depends on the second-order partial derivatives between the latent space and the encoding layers, and it is able to provide online explainability for the VAE during the learning time (during each learning epochs); such an XAI method can be utilized to build a visual attention mapping and assists in optimizing the model losses. Nevertheless, the proposed method will be confirmed for other UGL methods such as GANs and modern DL nets, it also will be confirmed for other ML application domains.

- Mgrad_2VAE is a novel XAI model (Section 4.4) that has been introduced in the context of autonomous driving systems, where the main task of that model is to perform semantic image segmentation. Also, the proposed model has shown superior results concerning the state-of-the-art semantic image segmentation models; however, it has been evaluated in synthetic and real image data which have been built in controlled environments neglecting the rough conditions and environments. In future works, the investigation of XAI models in harsh environments and rough weather conditions will be targeted, where the ambient environment includes rain, snow, dust, fog, etc. Also, to improve the explainability in autonomous driving systems, future works will consider exploring how to communicate such an explainability from the model to the vehicle's engine or from vehicular to vehicular.

- Expanding the application of Mgrad_2VAE and other XAI models to modern ADS tasks will be considered in future works. Specifically, further investigations are still required on the application of XAI models in devise reliability and functional safety, privacy concerns and attack detection, vehicles powertrains design, driver replacement, vision, and other tasks. Finally, the considerations to standardize works and XAI models in the context of ADS will be highlighted in future works, throughout introducing dedicated evaluation metrics and protocols.

References

[1] M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti. On approximating the non-negative rank: Applications to image reduction. In *Proc. of CIVEMSA*, 2020. 2, 3, 4, 5, 12, 73, 98, 111, 123, 134, 136, 148

[2] M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti. Unsupervised learning from limited available data by β-nmf and dual autoencoder. In *Proc. of ICIP*, 2020. 2, 3, 4, 5, 41, 44, 100, 123, 124, 134, 136, 148

[3] M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti. A survey on unsupervised generative models for exploratory data analysis and representation learning. *Acm computing surveys (csur)*, 2021. 2, 3, 4, 16, 58, 74, 123, 136, 144, 147, 148

[4] M. Abukmeil, A. Genovese, V. Piuri, F. Rundo, and F. Scotti. Towards explainable semantic segmentation for autonomous driving systems by multi-scale variational attention. In *Proc. of ICAS*, 2021. 4, 5, 147, 150

[5] M. A. M. Abukmeil, H. Elaydi, and M. Alhanjouri. Palmprint recognition via bandlet, ridgelet, wavelet and neural network. *Journal of Computer Sciences and Applications*, 3(2):23–28, 2015. 5, 111

[6] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985. 5, 65, 89, 90

[7] B. Alexandrov, V. V. Vesselinov, and H. N. Djidjev. Non-negative tensor factorization for robust exploratory big-data analytics. Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2018. 5, 96, 111, 114

[8] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020. 3, 63

[9] W. F. Ames. *Numerical methods for partial differential equations*. 2014. 137

[10] A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus. Variational ae for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *Proc. of IROS*, 2018. 148

[11] A. O. Andrade, S. Nasuto, P. Kyberd, and C. M. Sweeney-Reed. Generative topographic mapping applied to clustering and visualization of motor unit action potentials. *Biosystems*, 82(3):273–284, 2005. 5, 40

[12] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *Proc. of ICLR*, 2017. 105

[13] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. Proc. of ICML, pages 214–223, 2017. 83, 101

[14] A. B. Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. 147

[15] S. M. Atif, S. Qazi, and N. Gillis. Improved SVD-based initialization for nonnegative matrix factorization using low-rank correction. *Pattern Recognition Letters*, 122:53–59, 2019. 113, 114, 124

[16] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina. Exploiting manifold geometry in hyperspectral imagery. *IEEE Trans. on Geoscience and Remote Sensing*, 43(3):441–454, 2005. 49

[17] M. Balasubramanian and E. L. Schwartz. The isomap algorithm and topological stability. *Science*, 295, 2002. 5

[18] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proc. of the 2011 Int. Conf. on Unsupervised and Transfer Learning workshop*, 2012. 3, 73, 124, 134

[19] P. Baldi. Boolean autoencoders and hypercube clustering complexity. *Designs, Codes and Cryptography*, 65(3):383–403, 2012. 79, 103

[20] P. Baldi and Z. Lu. Complex-valued autoencoders. *Neural Networks*, 33:136–147, 2012. 79, 102

[21] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. CVAE-GAN: fine-grained image generation through asymmetric training. In *Proc. of ICCV*, pages 2745–2754, 2017. 85, 94

[22] G. Barello, A. Charles, and J. Pillow. Sparse-coding variational auto-encoders. *bioRxiv*, page 399246, 2018. 79

[23] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 51, 53, 99

[24] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM Sigkdd Explorations*, 9(2):75–79, 2007. 5, 97

[25] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Trans. on Signal Processing*, 45(2):434–444, 1997. 16

[26] J. Z. Bengar, A. Gonzalez-Garcia, G. Villalonga, B. Raducanu, H. H. Aghdam, M. Mozerov, A. M. Lopez, and J. van de Weijer. Temporal coherence for active learning in videos. In *Proc. of ICCVW*, 2019. 151

[27] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2009. 67, 68

[28] Y. Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017. 125

[29] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. 3, 4, 6, 58, 64, 74, 91, 94, 101, 123, 124, 134, 136, 148

[30] Y. Bengio, H. Larochelle, and P. Vincent. Non-local manifold parzen windows. In *Proc. of NIPS*, pages 115–122, 2006. 50, 99

[31] Y. Bengio and M. Monperrus. Non-local manifold tangent learning. In *Proc. of NIPS*, pages 129–136, 2005. 46, 49

[32] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *Proc. of NIPS*, pages 177–184, 2004. 58

[33] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *Proc. of NIPS*, pages 899–907, 2013. 75

[34] R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017. 62

[35] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 83, 103, 104

[36] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. 2020. 3, 94, 105

[37] C. M. Biship. Pattern recognition and machine learning (information science and statistics), 2007. 11

[38] C. M. Bishop, M. Svensén, and C. K. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998. 36, 39

[39] C. M. Bishop, M. Svensén, and C. K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21(1-3):203–224, 1998. 40, 44

[40] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010. 34

[41] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. 35, 93

[42] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, January 2003. 5, 32, 34

[43] C. Boncelet. Image noise models. In *The Essential Guide to Image Processing*, pages 143–167. Elsevier, 2009. 75

[44] W. M. Boothby. *An introduction to differentiable manifolds and Riemannian geometry*, volume 120. Academic press, 1986. 45, 57

[45] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *Proc. of CoNLL*, pages 10–21, 2016. 79

[46] M. Brand. Charting a manifold. In *Proc. of (NIPS*, pages 985–992, 2003. 50

[47] R. Brualdi and H. Ryser. *Combinatorial Matrix Theory*. Cambridge University Press, 1991. 54

[48] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2016. 79

[49] Y. Cai, A. Genovese, V. Piuri, F. Scotti, and M. Siegel. Iot-based architectures for sensing and local data processing in ambient intelligence: Research and industrial trends. In *Proc. of I2MTC*, pages 1–6, 2019. 123

[50] L. Cao. Data science: a comprehensive overview. *ACM Computing Surveys*, 50(3):43, 2017. 2, 7

[51] L. Cao and C. Zhang. Domain driven data mining. In *Data Mining and Knowledge Discovery Technologies*, pages 196–223. IGI Global, 2008. 2, 5

[52] L. Cayton. Algorithms for manifold learning. Technical Report CS2008-0923, University of California, San Diego, CA, 2005. 58

[53] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proc. of ECCV*, 2015. 147, 148, 150

[54] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017. 95

[55] X. Chen, J. Xu, R. Zhou, W. Chen, J. Fang, and C. Liu. Trajvae: A variational autoencoder model for trajectory generation. *Neurocomputing*, 428:332–339, 2021. 148

[56] Y. Chen, H. Zhang, J. Wu, X. Wang, R. Liu, and M. Lin. Modeling emerging, evolving and fading topics using dynamic soft orthogonal nmf with sparse representation. In *Proc. of ICDM*, pages 61–70, 2015. 43

[57] S.-S. Cheng, H.-C. Fu, and H.-M. Wang. Model-based clustering by probabilistic self-organizing maps. *IEEE Trans. on Neural Networks*, 20(5):805–826, 2009. 36, 39

[58] K. H. Cho, A. Ilin, and T. Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. In *Proc. of ICANN*, pages 10–17, 2011. 64, 68

[59] K. H. Cho, T. Raiko, and A. Ilin. Gaussian-bernoulli deep boltzmann machine. In *Proc. of IJCNN*, pages 1–7, 2013. 68

[60] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. of CVPR*, 2017. 155

[61] E. Chong and F. C. Park. Movement prediction for a lower limb exoskeleton using a conditional restricted boltzmann machine. *Robotica*, 35(11):2177–2200, 2017. 5, 68

[62] A. Cichocki. Tensor networks for dimensionality reduction, big data and deep learning. In *Advances in Data Analysis with Computational Intelligence Methods*, pages 3–49. Springer, 2018. 2, 5

[63] A. Cichocki and S.-i. Amari. Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010. 11, 101

[64] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009. 2, 5, 41, 123

[65] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proc. of AISTATS*, pages 215–223, 2011. 3, 111, 123

[66] R. H. Coding. Information theory. *Prentice Hall*, 1986. 113

[67] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994. 18, 20

[68] M. A. Cox and T. F. Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008. 47

[69] A. Creswell and A. A. Bharath. Denoising adversarial autoencoders. *IEEE Trans. on Neural Networks and Learning Systems*, pages 1–17, 2018. 86

REFERENCES

[70] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves. Associative long short-term memory. In *Proc. of ICML*, pages 1986–1994, 2016. 102

[71] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. 24

[72] B. L. R. De Moor and G. H. Golub. The restricted singular value decomposition: properties and applications. *SIAM Journal on Matrix Analysis and Applications*, 12(3):401–425, 1991. 25

[73] D. de Ridder and V. Franc. *Robust manifold learning*. Czech Technical University, 2003. 52

[74] P. DeLellis, G. Polverino, G. Ustuner, N. Abaid, S. Macrì, E. M. Bollt, and M. Porfiri. Collective behaviour across animal species. *Scientific Reports*, 4:3723, 2014. 5

[75] L. Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 58, 116, 125, 139

[76] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *Proc. of NIPS*, pages 1486–1494, 2015. 5, 86, 95, 104

[77] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *Proc. of ICLR*, 2017. 81, 93

[78] D. L. Donoho and C. Grimes. When does geodesic distance recover the true hidden parametrization of families of articulated images? In *Proc. of ESANN*, pages 199–204, 2002. 48

[79] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. of the National Academy of Sciences*, 100(10):5591–5596, 2003. 54

[80] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Proc. of NIPS*, pages 658–666, 2016. 89

[81] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. of robot learning*, 2017. 147

[82] A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001. 90

[83] I. P. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. In *Proc. of ICLR*, 2017. 5, 88, 95

[84] H. Elaydi, M. A. M. Abukmeil, and M. Alhanjouri. Palmprint recognition using multiscale transform, linear discriminate analysis, and neural network. *Science Journal of Circuits, Systems and Signal Processing*, 2(5):112–118, 2013. 104

[85] K. Fan, Z. Wang, J. Beck, J. Kwok, and K. A. Heller. Fast second order stochastic backpropagation for variational inference. In *Proc. of NIPS*, 2015. 137

[86] A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. A brief survey of automatic methods for author name disambiguation. *ACM Sigmod Record*, 41(2):15–26, 2012. 34

[87] A. Fischer and C. Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47(1), 2014. 65

[88] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995. 68

[89] C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, 21(3):793–830, 2009. 113

[90] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of ICML*, pages 1050–1059, 2016. 102

[91] R. Gao, X. Hou, J. Qin, J. Chen, L. Liu, F. Zhu, Z. Zhang, and L. Shao. Zero-VAE-GAN: Generating unseen features for generalized and transductive zero-shot learning. *IEEE Trans. on Image Processing*, 29:3665–3680, 2020. 85, 94, 104

[92] X. Gao, Z. Zhang, T. Mu, X. Zhang, C. Cui, and M. Wang. Self-attention driven adversarial similarity learning network. *Pattern Recognition*, page 107331, 2020. 89, 104

[93] A. E. Gawkea, J. Kacprzyk, L. Rutkowski, and G. G. Yen. *Advances in data analysis with computational intelligence methods: dedicated to Professor Jacek Żurada*, volume 738. Springer, 2017. 2

[94] A. Genovese, V. Piuri, K. N. Plataniotis, and F. Scotti. PalmNet: Gabor-PCA convolutional networks for touchless palmprint recognition. *IEEE Trans. on Information Forensics and Security*, 14(12):3160–3174, 2019. 5, 111

[95] A. Genovese, V. Piuri, F. Rundo, F. Scotti, and C. Spampinato. Pedestrian/cyclist distance estimation from a single rgb image: A cnn-based semantic segmentation approach. In *Proc. of Industrial Technology (ICIT 2021)*, 2021. 5, 147

[96] A. Gepperth and B. Pfülb. A rigorous link between self-organizing maps and gaussian mixture models. In *Proc. of ICANN*, 2020. 36

[97] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Muhlegg, S. Dorn, et al. A2d2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*, 2020. 151

[98] N. Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12(257):257–291, 2014. 113, 121

[99] N. Gillis, L. T. K. Hien, V. Leplat, and V. Y. F. Tan. Distributionally robust and multi-objective nonnegative matrix factorization. *arXiv preprint arXiv:1901.10757*, 2019. 112, 116

[100] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proc. of DSAA*, pages 80–89, 2018. 111

[101] R. Giryes, G. Sapiro, and A. M. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Trans. on Signal Processing*, 64(13):3444–3457, 2016. 62

[102] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*, pages 249–256, 2010. 68

[103] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Proc. of NIPS*, pages 513–520, 2005. 50

[104] I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio. Multi-prediction deep boltzmann machines. In *Proc. of NIPS*, pages 548–556, 2013. 72

[105] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. of NIPS*, pages 2672–2680, 2014. 5, 62, 80, 100, 101

[106] D. Grattarola, L. Livi, and C. Alippi. Adversarial autoencoders with constant-curvature latent manifolds. *Applied Soft Computing*, 81:105511, 2019. 94, 105

[107] D. Grattarola, D. Zambon, L. Livi, and C. Alippi. Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. *IEEE Trans. on Neural Networks and Learning Systems*, 31(6):1856–1869, 2019. 5, 62, 94, 100, 105

[108] T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum, and D. M. Blei. Hierarchical topic models and the nested chinese restaurant process. In *Proc. of NIPS*, pages 17–24, 2004. 5, 32, 34

[109] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. 147

[110] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Proc. of CVPR*, pages 2862–2869, 2014. 115

[111] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Proc. of NIPS*, pages 5767–5777, 2017. 83, 105

[112] D. Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017. 111

[113] D. Gunning and D. Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, 2019. 147

[114] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *Proc. of ICML*, pages 1737–1746, 2015. 123, 124

[115] R. Haag. *Local quantum physics: Fields, particles, algebras*. Springer Science & Business Media, 2012. 114, 115

[116] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. 1970. 3, 28

[117] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989. 54

[118] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019. 79

[119] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of CVPR*, pages 2961–2969, 2017. 95

[120] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016. xiv, 4, 87, 88, 89, 94, 95, 138, 150

[121] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. of ECCV*, pages 630–645, 2016. 94

[122] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proc. of ICLR*, 2017. 5, 79

[123] 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. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012. 63

[124] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. 63, 65, 66, 90

[125] G. E. Hinton. A practical guide to training restricted boltzmann machines. pages 599–619. 2012. 4, 64, 67

[126] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 63, 65, 68, 70, 89

[127] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Proc. of NIPS*, pages 857–864, 2002. 55

[128] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 4, 5, 67, 70, 74, 136

[129] G. E. Hinton, T. J. Sejnowski, and T. A. Poggio. *Unsupervised learning: foundations of neural computation*. MIT, 1999. 3

[130] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Proc. of NIPS*, pages 3–10, 1994. 73, 91, 136

[131] S. Hochreiter and J. Schmidhuber. Low-complexity coding and decoding. In *Theoretical Aspects of Neural Computation*, pages 297–306, 1997. 5, 22

[132] H. Hong. Multimodal discovering and fusion for semantics multimedia analysis. In *Proc. of ALPIT*, pages 155–158, 2007. 5

[133] Y. Hong, U. Hwang, J. Yoo, and S. Yoon. How generative adversarial networks and their variants work: An overview. *ACM Computing Surveys*, 52(1):1–43, 2019. 4, 7, 83

[134] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012. 114, 115

[135] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *IEEE Trans. on Neural Networks and Learning Systems*, 27(12), 2015. 125

[136] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5:1457–1469, November 2004. 43

[137] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proc. of CVPR*, pages 4700–4708, 2017. 95

[138] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Proc. of ECCV*, pages 646–661, 2016. 94

[139] H. Huang, X. Hu, Y. Zhao, M. Makkie, Q. Dong, S. Zhao, L. Guo, and T. Liu. Modeling task fMRI data via deep convolutional autoencoder. *IEEE Trans. on Medical Imaging*, 37(7):1551–1561, 2018. 79

[140] L. Huang, X. Liu, B. Lang, A. Yu, Y. Wang, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proc. of AAAI*, 2018. 4, 94

REFERENCES

[141] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. In *Proc. of CVPR*, pages 5077–5086, 2017. 86

[142] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004. 5, 21, 22, 40

[143] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000. 3, 20

[144] D. I. J. Im, S. Ahn, R. Memisevic, and Y. Bengio. Denoising criterion for variational auto-encoding framework. In *Proc. of AAAI*, 2017. 79, 86

[145] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. of CVPR*, pages 1125–1134, 2017. 87, 88, 103

[146] E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003. 30

[147] I. Jolliffe. *Principal component analysis*. Springer, 2011. 19

[148] J. Jost and J. Jost. *Riemannian geometry and geometric analysis*, volume 42005. 2008. 53

[149] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 100

[150] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of ICLR*, 2018. 93, 101, 104

[151] P. Kherwa and P. Bansal. Topic modeling: A comprehensive review. *EAI Endorsed Trans. on Scalable Information Systems*, 7(24), 2020. 98

[152] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of ICML*, 2014. 137, 138, 141, 149, 150

[153] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proc. of ICLR*, 2014. 5, 73, 75, 77, 80, 81, 83, 92, 101, 134, 136, 138, 151, 155

[154] T. N. Kipf and M. Welling. Variational graph auto-encoders. In *Proc. of NeurIPS)*, 2016. 62

[155] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Trans. on Automatic Control*, 25(2):164–176, 1980. 24

[156] T. Kohonen. The self-organizing map. *Proc. of the IEEE*, 78(9):1464–1480, 1990. 38

[157] T. Kohonen. Essentials of the self-organizing map. *Neural networks*, 37:52–65, 2013. 39

[158] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 2, 5, 24, 28

[159] D. Koller, N. Friedman, and F. Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 3, 30, 63, 79

[160] E. O. Korman. Autoencoding topology. *arXiv preprint arXiv:1803.00156*, 2018. 62, 100

[161] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*, pages 1097–1105, 2012. 2, 155

[162] J. B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964. 46

[163] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997. 10, 17, 22

[164] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014. 2, 3, 73, 124

[165] A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling. Initializations for the nonnegative matrix factorization. In *Proc. of ACM SIGKDD*, pages 23–26, 2006. 112

[166] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, January 2009. 75

[167] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Proc. of AISTATS*, pages 29–37, 2011. 90

[168] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proc. of ICML*, page 1558–1566, 2016. 85, 93, 104

[169] M. H. C. Law and A. K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(3):377–391, March 2006. 3, 5, 49

[170] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998. 102

[171] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 3, 27, 111, 123

[172] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*, pages 556–562, 2001. 27

[173] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. of ICML*, pages 609–616, 2009. 3, 5, 70

[174] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007. 58

[175] A. Lemme, R. F. Reinhart, and J. J. Steil. Efficient online learning of a non-negative sparse autoencoder. In *Proc. of ESANN*, 2010. 125, 131, 132

[176] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Computing Surveys*, 50(6):1–45, 2017. 2, 3

[177] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proc. of ICML*, pages 577–584, 2006. 35

[178] W.-J. Li, D.-Y. Yeung, and Z. Zhang. Probabilistic relational PCA. In *Proc. of NIPS*, pages 1123–1131, 2009. 20

[179] X. Li, S. Lin, S. Yan, and D. Xu. Discriminant locally linear embedding with high-order tensor data. *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):342–352, 2008. 5

[180] X. Li, F. Zhao, and Y. Guo. Conditional Restricted Boltzmann Machines for Multi-label Learning with Incomplete Labels. In *Proc. of AISTATS*, pages 635–643, 2015. 68

[181] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. on Information theory*, 37(1):145–151, 1991. 11

[182] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. Adversarial ranking for language generation. In *Proc. of NIPS*, pages 3155–3165, 2017. 5

[183] S. Lipovetsky. PCA and SVD with nonnegative loadings. *Pattern Recognition*, 42(1):68–76, 2009. 5, 20, 43

[184] L. Liu, L. Tang, L. He, W. Zhou, and S. Yao. An overview of hierarchical topic modeling. In *Proc. of IHMSC*, pages 391–394, 2016. 34, 35

[185] L. Liu, H. Zhang, X. Xu, Z. Zhang, and S. Yan. Collocating clothes with generative adversarial networks cosupervised by categories and attributes: a multidiscriminator framework. *IEEE Trans. on Neural Networks and Learning Systems*, 2019. 88, 105

[186] R. Liu, X. Wang, D. Wang, Y. Zuo, H. Zhang, and X. Zheng. Topic splitting: a hierarchical topic model based on non-negative matrix factorization. *Journal of Systems Science and Systems Engineering*, 27(4):479–496, 2018. 44

[187] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proc. of ECCV*, pages 21–37, 2016. 95

[188] W. Liu, R. Li, M. Zheng, S. Karanam, Z. Wu, B. Bhanu, R. J. Radke, and O. Camps. Towards visually explaining variational autoencoders. In *Proc. of CVPR*, 2020. xv, 4, 5, 135, 140, 141, 142, 148, 150

[189] M. Loey, F. Smarandache, and N. E. M Khalifa. Within the lack of chest covid-19 x-ray dataset: a novel detection model based on gan and deep transfer learning. *Symmetry*, 12(4):651, 2020. 5

[190] E. López-Rubio. Probabilistic self-organizing maps for continuous data. *IEEE Trans. on Neural Networks*, 21(10):1543–1554, 2010. 36, 49

[191] L. Maaten. Learning a parametric embedding by preserving local structure. In *Proc. of AISTATS*, pages 384–391, 2009. 57

[192] P. Magron and T. Virtanen. Towards complex nonnegative matrix factorization with the beta-divergence. In *Proc. of IWAENC*, pages 156–160, 2018. 112, 123

[193] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 80, 83, 93, 134

[194] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proc. of ICCV*, pages 2794–2802, 2017. 82, 101

[195] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proc. of ICANN*, pages 52–59, 2011. 79

[196] Y. A. Mejjati, C. Richardt, J. Tompkin, D. Cosker, and K. I. Kim. Unsupervised attention-guided image-to-image translation. In *Proc. of NIPS*, 2018. 135

[197] R. Memisevic and G. Hinton. Unsupervised learning of image transformations. In *Proc. of (CVPR*, pages 1–8, 2007. 5, 65, 68

[198] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119, 2013. 98

[199] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *arXiv preprint arXiv:2001.05566*, 2020. 148, 149

[200] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 5, 80, 86, 88, 104

[201] V. Mnih, H. Larochelle, and G. E. Hinton. Conditional restricted boltzmann machines for structured output prediction. In *Proc. of UAI*, pages 514–522, 2011. 65, 67, 86

[202] A.-r. Mohamed, G. E. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Processing*, 20(1):14–22, 2012. 5, 70

[203] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. of CVPR*, pages 2574–2582, 2016. 96

[204] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 10

[205] B. Muthen. Latent variable mixture modeling. *New developments and techniques in structural equation modeling*, 2:1–33, 2001. 5, 31

[206] B. Muthén. Latent variable analysis. *The Sage handbook of quantitative methodology for the social sciences*, 345(368):106–109, 2004. 31

[207] I. Nabney. *NETLAB: algorithms for pattern recognition*. Springer Science & Business Media, 2002. 37

[208] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICML*, 2010. 137, 149

[209] A. Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011. 75, 134

[210] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proc. of CVPR*, pages 4467–4477, 2017. 89

[211] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proc. of NIPS*, pages 3387–3395, 2016. 89, 103

[212] V.-A. Nguyen, J. L. Ying, and P. Resnik. Lexical and hierarchical topic regression. In *Proc. of NIPS*, pages 1106–1114, 2013. 34

[213] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Proc. of NIPS*, pages 6338–6347, 2017. 62

[214] G. C. Nutakki, O. Nasraoui, B. Abdollahi, M. Badami, and W. Sun. Distributed LDA-based topic modeling and topic agglomeration in a latent space. In *Proc. of WWW*, pages 17–24, 2014. 44

[215] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994. 24, 25, 27

[216] C. C. Paige and M. A. Saunders. Towards a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, 18(3):398–405, 1981. 25

[217] P. Pajunen, A. Hyvärinen, and J. Karhunen. Nonlinear blind source separation by self-organizing maps. In *Proc. of NIP*, 1996. 44

REFERENCES

[218] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. of CVPR*, pages 2337–2346, 2019. 5, 89, 95

[219] P. A. Parrilo and S. Khatri. On cone-invariant linear matrix inequalities. *IEEE Trans. on Automatic Control*, 45(8):1558–1563, 2000. 114

[220] K. Pearson. 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):559–572, 1901. 5, 18

[221] D. Peel and G. J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and computing*, 10(4):339–348, 2000. 5, 36, 52

[222] X. Peng, C. Lu, Z. Yi, and H. Tang. Connections between nuclear-norm and frobenius-norm-based representations. *IEEE Trans. on Neural Networks and Learning Systems*, 29(1):218–224, 2016. 11, 115

[223] Y. Petinot, K. McKeown, and K. Thadani. A hierarchical model of web summaries. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 670–675, June 2011. 34

[224] A. Poibrenski, M. Klusch, I. Vozniak, and C. Muller. M2p3: multimodal multi-pedestrian path prediction by self-driving cars with egocentric vision. In *Proc. of ACM Symposium on Applied Computing*, 2020. 148

[225] C.-A. Popa. Complex-valued convolutional neural networks for real-valued image classification. In *Proc. of IJCNN*, pages 816–822, 2017. 102

[226] G. Pospiech, M. Michelini, and B.-S. Eylon. *Mathematics in physics education*. Springer, 2019. 137

[227] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys*, 51(5):92, 2018. 4, 7, 72, 95, 99

[228] S. Pouyanfar, Y. Yang, S.-C. Chen, M.-L. Shyu, and S. Iyengar. Multimedia big data analytics: A survey. *ACM Computing Surveys*, 51(1):1–34, 2018. 2

[229] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 3, 89, 94, 103

[230] S. Raghavendra, T. Kipf, M. Welling, J. J. H. Pedersen, J. Petersen, and M. de Bruijne. Extraction of airways using graph neural networks. In *Proc. of MIDL*, 2018. 62, 100

[231] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011. 113, 117, 121

[232] M. Ranzato, Y.-L. Boureau, and Y. L. Cun. Sparse feature learning for deep belief networks. In *Proc. of NIPS*, pages 1185–1192, 2008. 91, 101

[233] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010. 114

[234] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Proc. of CVPR*, pages 7263–7271, 2017. 95

[235] D. P. Reichert and T. Serre. Neuronal synchrony in complex-valued deep networks. *CoRR*, abs/1312.6115, 2014. 102

[236] M. Reisenbichler and T. Reutterer. Topic modeling in marketing: recent advances and research opportunities. *Journal of Business Economics*, 89(3):327–356, 2019. 98

[237] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. of NIPS*, pages 91–99, 2015. 95

[238] D. A. Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009. 37

[239] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. of ICML*, 2014. 78, 136

[240] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proc. of ICML*, pages 833–840, 2011. 73, 75, 76, 91, 134

[241] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of MICCAI*, pages 234–241, 2015. 87, 148, 149, 155

[242] M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017. 86

[243] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345, 1999. 36, 37

[244] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 5, 51, 99

[245] D. Roy, P. Panda, and K. Roy. Synthesizing images from spatio-temporal representations using spike-based backpropagation. *Frontiers in Neuroscience*, 13:621, 2019. xix, 127, 131

[246] M. Rudolph, B. Wandt, and B. Rosenhahn. Structuring autoencoders. In *Proc. of ICCV*, pages 0–0, 2019. xix, 131

[247] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Muller, and M. Kloft. Deep one-class classification. In *Proc. of ICML*, 2018. 135, 138, 140, 143

[248] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. xiii, 5, 73, 74, 134

[249] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Proc. of NIPS*, pages 3856–3866, 2017. 99

[250] N. Saeed, H. Nam, M. I. U. Haq, and D. B. Muhammad Saqib. A survey on multidimensional scaling. *ACM Computing Surveys*, 51(3):1–25, 2018. 58

[251] R. Salakhutdinov. Learning deep generative models. *Review of Statistics and its Application*, 2:361–385, 2015. 3, 70, 72, 94

[252] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Proc. of AISTATS*, pages 448–455, 2009. 5, 65, 70, 71, 72, 89, 90

[253] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Proc. of NIPS*, pages 2234–2242, 2016. 81

[254] A. Sankaran, M. Vatsa, R. Singh, and A. Majumdar. Group sparse autoencoder. *Image and Vision Computing*, 60:64–74, 2017. xix, 131

[255] N. Sarafijanovic-Djukic and J. Davis. Fast distance-based anomaly detection in images using an inception-like autoencoder. In *Proc. of DS*, 2019. 135, 140, 143, 144

[256] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, Jun 2003. 3

[257] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. 4, 6, 74, 94

[258] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Proc. of ICANN*, pages 583–588, 1997. 5, 20

[259] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of CVPR*, 2017. xiv, 107, 108, 135

[260] B. Shaw and T. Jebara. Minimum volume embedding. In *Proc. of AISTATS*, pages 460–467, 2007. 55

[261] B. Shaw and T. Jebara. Structure preserving embedding. In *Proc. of ICML*, pages 937–944, 2009. 55

[262] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 54

[263] L. Shu, B. Long, and W. Meng. A latent topic model for complete entity resolution. In *Proc. of ICDE*, pages 880–891, 2009. 34

[264] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Proc. of NIPS*, pages 721–728, 2003. 49

[265] M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Proc. of ICANN*, pages 412–422, 2018. 62, 100

[266] M. Simonovsky and N. Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, editors, *Proc. of ICANN)*, pages 412–422, 2018. 94, 105

[267] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. of ICLR*, 2014. xiv, 106, 107

[268] P. Smolensky. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, pages 194–281. 1986. 65

[269] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. of NIPS*, pages 801–809, 2011. 5, 79

[270] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*, pages 151–161, 2011. 79

[271] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP inference for image super-resolution. In *Proc. of ICLR*, 2017. 93

[272] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Proc. of NIPS*, pages 3738–3746, 2016. 93

[273] R. Souvenir and R. Pless. Manifold clustering. In *Proc. of ICCV*, pages 648–653 Vol. 1, 2005. 5, 49

[274] S. Squires, A. Prügel-Bennett, and M. Niranjan. Rank selection in nonnegative matrix factorization using minimum description length. *Neural computation*, 29(8):2164–2176, 2017. 113, 121

[275] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 4, 81, 94

[276] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Proc. of NIPS*, pages 2222–2230, 2012. 5, 72

[277] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 94

[278] J. Steinberger and K. Ježek. Text summarization and singular value decomposition. In *Proc. of ADVIS*, pages 245–254, 2004. 43

[279] A. Stocco, M. Weiss, M. Calzana, and P. Tonella. Misbehaviour prediction for autonomous driving systems. In *Proc. of ICSE*, 2020. 148

[280] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. on Evolutionary Computation*, 23(5):828–841, 2019. 105

[281] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM review*, 48(4):681–699, 2006. 3, 54

[282] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. of AAAI*, 2017. 95

[283] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. of CVPR*, 2015. 143

[284] C. C. Tan and C. Eswaran. Reconstruction and recognition of face and digit images using autoencoders. *Neural Computing and Applications*, 19(7):1069–1079, 2010. 131

[285] C. Tang, N. Srivastava, and R. R. Salakhutdinov. Learning generative models with visual attention. In *Proc. of NIPS*, 2014. 135

[286] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. of ICML*, page 102, 2004. 64

[287] A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: an initial study on 2D projections of large multidimensional data. In *Proc. of AVI*, pages 49–56, 2010. 57

[288] Y. W. Teh and S. T. Roweis. Automatic alignment of local representations. In *Proc. of NIPS*, pages 865–872, 2003. 52

[289] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 5, 46, 48, 54

REFERENCES

[290] D. Ting and M. I. Jordan. Manifold learning via manifold deflation. *arXiv preprint arXiv:2007.03315*, 2020. 100

[291] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999. 5, 20

[292] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal. Deep complex networks. In *Proc. of ICLR*, 2018. 102

[293] L. R. Tucker et al. The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119, 1964. 28

[294] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Proc. of NIPS*, pages 4790–4798, 2016. 90

[295] L. Van der Maaten. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. 57

[296] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November 2008. xii, 4, 45, 51, 55, 58, 60, 99

[297] L. Van der Maaten and G. Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–55, 2012. 5, 57

[298] L. Van der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative review. *Technical Report TiCC-TR 2009-005, Tilburg University,*, 2009. 3, 4, 5, 46, 51, 58

[299] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996. 54

[300] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of NIPS*, pages 5998–6008, 2017. 95

[301] V. V. Vesselinov, M. K. Mudunuru, S. Karra, D. O'Malley, and B. S. Alexandrov. Unsupervised machine learning based on non-negative tensor factorization for analysis of reactive-transport site data simulations. 111

[302] P. Vincent and Y. Bengio. Manifold parzen windows. In *Proc. of NIPS*, pages 849–856, 2003. 50

[303] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*, pages 1096–1103, 2008. 73, 75, 91, 134

[304] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising. *Journal of Machine Learning Research*, 11:3371–3408, 2010. 3, 75, 93

[305] J. Wang, H. He, and D. V. Prokhorov. A folded neural network autoencoder for dimensionality reduction. *Procedia Computer Science*, 13:120–127, 2012. 131, 132

[306] R. Wang, A. Cully, H. J. Chang, and Y. Demiris. Magan: Margin adaptation for generative adversarial networks. *arXiv preprint arXiv:1704.03817*, 2017. 104

[307] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc. of CVPR*, pages 8798–8807, 2018. 5, 88, 95

[308] Y. Wang and Y. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Trans. on Knowledge and Data Engineering*, 25(6):1336–1353, June 2013. 27

[309] Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Trans. on Knowledge and Data Engineering*, 25(6):1336–1353, 2012. 112

[310] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004. 12, 119, 127, 153

[311] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi. Towards a viable autonomous driving research platform. In *Proc of Intelligent Vehicles Symposium (IV)*, 2013. 147

[312] Y. Wei, Z. Zhang, J. Fan, Y. Wang, S. Yan, and M. Wang. DerainCycleGAN: An attention-guided unsupervised benchmark for single image deraining and rainmaking. *arXiv preprint arXiv:1912.07015*, 2019. 5, 88, 95

[313] Y. Wei, Z. Zhang, H. Zhang, R. Hong, and M. Wang. A coarse-to-fine multi-stream hybrid deraining network for single image deraining. In *Proc. of ICDM*, pages 628–637, 2019. 95

[314] Y. Wei, Z. Zhang, H. Zhang, J. Qin, and M. Zhao. Semi-DerainGAN: A new semi-supervised single image deraining network. *arXiv preprint arXiv:2001.08388*, 2020. 88, 95

[315] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int. Journal of Computer Vision*, 70(1):77–90, Oct 2006. 5, 51, 54

[316] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proc. of ICML*, page 106, 2004. 55

[317] M. Welling and G. E. Hinton. A new learning algorithm for mean field boltzmann machines. In *Proc. of ICANN*, pages 351–357, 2002. 72

[318] C. K. Williams. On a connection between kernel PCA and metric MDS. *Machine Learning*, 46(1-3):11–19, 2002. 46

[319] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas. Full-capacity unitary recurrent neural networks. In *Proc. of NIPS*, pages 4880–4888, 2016. 102

[320] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002. 91

[321] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 5

[322] E. Q. Wu, G.-R. Zhou, L.-M. Zhu, C.-F. Wei, H. Ren, and R. S. Sheng. Rotated sphere haar wavelet and deep contractive auto-encoder network with fuzzy gaussian SVM for pilot's pupil. *IEEE Trans. on Cybernetics*, 2019. 5, 77

[323] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proc. of NIPS*, pages 82–90, 2016. 5, 89, 105

[324] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 116, 125, 139

[325] F. Xiong, O. I. Camps, and M. Sznaier. Low order dynamics embedding for high dimensional time series. In *Proc. of ICCV*, pages 2368–2374, 2011. 5

[326] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi. Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE Trans. on Medical Imaging*, 35(1):119–130, 2016. 5, 74

[327] W. Xu, X. Jiang, X. Hu, and G. Li. Visualization of genetic disease-phenotype similarities by multiple maps t-SNE with laplacian regularization. *BMC Medical Genomics*, 7(2):S1, 2014. 5

[328] M.-H. Yang. Face recognition using extended isomap. In *Proc. of ICIP*, 2002. 5

[329] Y. Yin and E. Gelenbe. Non-negative autoencoder with simplified random neural network. In *Proc. of IJCNN*, pages 1–6, July 2019. xix, 125, 131, 132

[330] C. H. Yu. Exploratory data analysis. *Methods*, 2:131–160, 1977. 2

[331] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *Proc. of ICML*, page 1215–1222, 2010. 51

[332] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Proc. of NIPS*, pages 2223–2231, 2009. 50

[333] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 94

[334] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 137

[335] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional nets. In *Proc. of ECCV*, 2014. 135

[336] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in variational inference. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 41(08):2008–2026, August 2019. 77, 79, 92, 93, 134, 136

[337] C. Zhang, T. Li, Z. Ren, Z. Hu, and Y. Ji. Taxonomy-aware collaborative denoising autoencoder for personalized recommendation. *Applied Intelligence*, 49(6):1–18, June 2019. 5, 76

[338] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *Proc. of ICML*, pages 7354–7363, 2019. 89

[339] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. *Int. Journal of Computer Vision*, 126(10):1084–1102, 2018. xiv, 107, 108, 135

[340] L. Zhang and Y. Lu. Comparison of auto-encoders with different sparsity regularizers. In *Proc. of IJCNN*, pages 1–5. 101

[341] Y. Zhang, Z. Zhang, S. Li, J. Qin, G. Liu, M. Wang, and S. Yan. Unsupervised nonnegative adaptive feature extraction for data representation. *IEEE Trans. on Knowledge and Data Engineering*, 31(12):2423–2440, 2018. 52

[342] Y. Zhang, Z. Zhang, J. Qin, L. Zhang, B. Li, and F. Li. Semi-supervised local multi-manifold isomap by linear embedding for feature extraction. *Pattern Recognition*, 76:662–678, 2018. 49

[343] Y. Zhang, Z. Zhang, Z. Zhang, M. Zhao, L. Zhang, Z. Zha, and M. Wang. Deep self-representative concept factorization network for representation learning. In *Proc. of ICDM*, page 361, 2020. 5, 27

[344] Z. Zhang, T. W. Chow, and M. Zhao. M-isomap: Orthogonal constrained marginal isomap for nonlinear dimensionality reduction. *IEEE Trans. on Cybernetics*, 43(1):180–191, 2012. 5, 49

[345] Z. Zhang, F. Li, M. Zhao, L. Zhang, and S. Yan. Robust neighborhood preserving projection by nuclear/l2, 1-norm regularization for image feature extraction. *IEEE Trans. on Image Processing*, 26(4):1607–1622, 2017. 5, 58

[346] Z. Zhang, Z. Tang, Y. Wang, J. Qin, H. Zhang, and S. Yan. Fast dense residual network: Enhancing global dense feature flow for text recognition. *arXiv preprint arXiv:2001.09021*, 2020. 95

[347] Z. Zhang, Z. Tang, Y. Wang, Z. Zhang, S. Yan, and M. Wang. Fast DenseNet: Towards efficient and accurate text recognition with fast dense networks. *arXiv preprint arXiv:1912.07016*, 2019. 95

[348] Z. Zhang, Z. Tang, Z. Zhang, Y. Wang, J. Qin, and M. Wang. Fully-convolutional intensive feature flow neural network for text recognition. *arXiv preprint arXiv:1912.06446*, 2019. 95

[349] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004. 54

[350] Z. Zhang, Y. Zhang, S. Li, G. Liu, D. Zeng, S. Yan, and M. Wang. Flexible auto-weighted local-coordinate concept factorization: A robust framework for clustering. *IEEE Trans. on Knowledge and Data Engineering*, 2019. 5, 27

[351] Z. Zhang, Y. Zhang, G. Liu, J. Tang, S. Yan, and M. Wang. Joint label prediction based semi-supervised adaptive concept factorization for robust data representation. *IEEE Trans. on Knowledge and Data Engineering*, (5):952–970, 2019. 27

[352] Z. Zhang, Y. Zhang, L. Zhang, and S. Yan. A survey on concept factorization: From shallow to deep representation learning. *arXiv preprint arXiv:2007.15840*, 2020. 27

[353] S. Zhao, J. Song, and S. Ermon. InfoVAE: Balancing learning and inference in variational autoencoders. In *Proc. of AAAI*, pages 5885–5892, 2019. 93

[354] G. Zhong, L.-N. Wang, X. Ling, and J. Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016. 2, 6, 94, 101

[355] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proc. of CVPR*, pages 2921–2929, 2016. xiv, 103, 106, 108, 135

[356] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of ICCV*, pages 2223–2232, 2017. 5, 87, 94, 95

[357] M. Zibulevsky and B. A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001. 16

Appendix A

Publications

1. *"A survey of unsupervised generative models for exploratory data analysis and representation learning"*

   M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti
   ACM Computing Surveys (CSUR), Jul 9, 2021, pp. 1-40.

   ABSTRACT: 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.

2. *"Towards explainable semantic segmentation for autonomous driving systems by multi-scale variational attention"*

   M. Abukmeil, A. Genovese, V. Piuri, F. Rundo, and F. Scotti
   in Proc. of the 1st IEEE Int. Conf. on Autonomous Systems (ICAS 2021), Mon-

treal, Canada, August 11-13, 2021, pp. 1-5.

ABSTRACT: Explainable autonomous driving systems (EADS) are emerging recently as a combination of explainable artificial intelligence (XAI) and vehicular automation (VA). EADS explains events, ambient environments, and engine operations of an autonomous driving vehicular, and it also delivers explainable results in an orderly manner. Explainable semantic segmentation (ESS) plays an essential role in building EADS, where it offers visual attention that helps the drivers to be aware of the ambient objects irrespective if they are roads, pedestrians, animals, or other objects. In this paper, we propose the first ESS model for EADS based on the variational autoencoder (VAE), and it uses the multiscale second-order derivatives between the latent space and the encoder layers to capture the curvatures of the neurons' responses. Our model is termed as Mgrad_2VAE and is bench-marked on the SYNTHIA and A2D2 datasets, where it outperforms the recent models in terms of image segmentation metrics.

3. *"Unsupervised learning from limited available data by β-NMF and dual autoencoder"*

   M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti
   in Proc. of the 27^{th} IEEE Int. Conf. on Image Processing (ICIP 2020), Abu Dhabi, UAE, October 25-28, 2020, pp. 81-85.

   ABSTRACT: Unsupervised Learning (UL) models are a class of Machine Learning (ML) which concerns with reducing dimensionality, data factorization, disentangling and learning the representations among the data. The UL models gain their popularity due to their abilities to learn without any predefined label, and they are able to reduce the noise and redundancy among the data samples. However, generalizing the UL models for different applications including image generation, compression, encoding, and recognition faces different challenges due to limited available data for learning, diversity, and complex dimensions. To overcome such challenges, we propose a partial learning procedure by utilizing the β-Non Negative Matrix Factorization (β-NMF), which maps the data into two complementary subspaces constituting generalized driven priors among the data. Moreover, we employ a dual-shallow Autoencoder (AE) to learn the subspaces separately or jointly for image reconstruction and visualization tasks, where our model performance shows superior results to the literary works when learning the model with a small amount of data and generalizing it for large-scale unseen data.

4. *"On approximating the non-negative rank: Applications to unsupervised image reduction"*

   M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti

in Proc. of the 2020 IEEE Int. Conf. on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA 2020), Tunis, Tunisia, June 22-24, 2020, pp. 1-6.

ABSTRACT: Unsupervised Learning (UL) methods are a class of machine learning which aims to disentangle the representations and reduce the dimensionality among the data without any predefined labels. Among all UL methods, the Non-negative Matrix Factorization (NMF) factorizes the data into two subspaces of non-negative components. Moreover, the NMF enforces the non-negativity, sparsity, and part-based analysis, thus the representations can be interpreted and explained for the Explainable Artificial Intelligence (XAI) applications. However, one of the main issues when using the NMF is to impose the factorization rank r to identify the dimensionality of the subspaces, where the rank is usually unknown in advance and known as the non-negative rank that is used as a prior to carrying out the factorization. Accordingly, we propose a novel method for the non-negative rank r approximation to help solving this problem, and we generalize our method among different image scales. Where, the results on different image data sets confirm the validity of our approach.

A.1 Other Publications

1. *"Experimental results on palmvein-based personal recognition by multi-snapshot fusion of textural features"*

M. Abukmeil, and GL Marcialis
International Journal of Biometrics (IJBM), 2020 (in press).

ABSTRACT: In this paper, we investigate multiple snapshot fusion of textural features for palmvein recognition including identification and verification. Although the literature proposed several approaches for palmvein recognition, palmvein performance is still affected by identification and verification errors. As well-known, palmveins are usually described by line-based methods which enhance the vein flow. This is claimed to be unique from person to person. However, palmvein images are also characterized by the texture that can be pointed out by textural features, which relies on recent and efficient handcrafted algorithms such as Local Binary Patterns, Local Phase Quantization, Local Tera Pattern, Local directional Pattern, and Binarized Statistical Image Features (LBP, LPQ, LTP, LDP, and BSIF, respectively), among others. Finally, they can be easily managed at feature-level fusion, when more than one sample can be acquired for recognition. Therefore, multi-snapshot fusion can be adopted for exploiting these features' complementarity. Our goal in this paper is to show that this is confirmed for palmvein recognition, thus allowing us to achieve very high recognition rates on a well-known benchmark dataset.

2. *"Grad$_2$ VAE: An Explainable Variational Autoencoder Model Based on Attention Preserving Curvatures of Representations"*

M. Abukmeil, S. Ferrari, A. Genovese, V. Piuri, and F. Scotti
submitted to an international conference, 2022.

ABSTRACT: Unsupervised learning (UL) is a class of machine learning (ML) that learns data, reduces dimensionality, and visualizes the decisions without labels. Among UL models, variational autoencoder (VAE) is considered a UL model that is regulated by variational inference to approximate the posterior distribution of large datasets. In this paper, we propose a novel explainable artificial intelligence (XAI) method to explain the VAE behavior based on the first-order derivative of the latent space gradient, which reflects the amount of acceleration required from encoding to decoding space. Our model is termed as Grad$_2$VAE and it is able to capture the local curvatures of the representations to explain the model's behavior. Besides the VAE explanation, we employ our method for anomaly detection, where our model outperforms the recent UL deep models when generalizing it for large-scale anomaly data.