PUSHING THE LIMITS OF TRADITIONAL UNSUPERVISED LEARNING

by

Eren Gultepe

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology

August 2018

Abstract

Unsupervised learning has important applications in extremely large data settings such as in medical, biological, social, and environmental data. Typically in these settings, copious amounts of data are collected, with the additional burden of high dimensionality and unavailability of class labels. Improving the performance and usability of unsupervised learning algorithms provides improved resource management and delivery of services to users. Although deep learning methods have become popular due to their success in the supervised learning problem of classification and unsupervised learning problems of feature extraction and cluster analysis, traditional machine learning methods can still provide state-of-the-art performance. In this thesis, a novel clustering framework that combines common clustering and feature extraction methods along with careful parameter selection is presented. This framework is able to achieve state-of-the-art clustering performance that is better than many deep learning-based methods on large benchmark and web-based text and image datasets. This pipeline incorporates deep learning-style feature extraction, but without the onerous hyper-parameter tuning procedure. Then two novel methods are provided for testing the significance and reliability of clusters, in which the null-hypothesis statistical distribution is formed either by: (1) a uniform distribution projected onto the principal components of the original data; or (2) a randomized, weighted adjacency matrix. Significance testing of clusters is important when the nature or underlying properties of the data are unknown, especially in large data settings or in nonstandard datasets. Since, a random sample of the population data could contain properties that are not representative of the whole population. Thus, providing a clustering result that is not typical of the population. Finally, given the success of traditional matrix factorization methods in the clustering pipeline, text

document classification using a new convolutional neural network architecture that leverages singular value decomposition was developed. This new model provided state-of-the-art document classification accuracy.

Keywords: Unsupervised feature learning, Cluster significance testing, Latent semantic analysis, Spectral clustering, Independent component analysis

Acknowledgements

I would like to thank my PhD advisor, Masoud Makrehchi for his patience, support, and invaluable vision throughout the duration of my study at UOIT. Also, I would like to thank my lab colleagues for providing help whenever I need it, and the faculty at UOIT for giving me the opportunity to complete my PhD in Computer Engineering.

Co-Authorship

In all cases, I (Eren Gultepe) was the main investigator for each of the co-authored studies.

- E. Gultepe and M. Makrehchi, "Predicting and grouping digitized paintings by style using unsupervised feature learning," *Journal of Cultural Heritage*, vol. 31, pp. 1323, 2018. doi:10.1016/j.culher.2017.11.008

- E. Gultepe, and M. Makrehchi, "Improving clustering performance using independent component analysis and unsupervised feature learning," *Human Centric Computing and Information Sciences*, 2018. doi:10.1186/s13673-018-0148-3

- E. Gultepe, M. Kamkarhaghighi, and M. Makrehchi, "Latent semantic analysis boosted convolutional neural networks for document classification," *5^{th} International Conference on Behavioral, Economic, and Socio-Cultural Computing 2018.*

Contents

List of Tables

List of Figures

Nomenclature

classification	Predicting the class of new observations based common patterns and class information
class	The label information of an observation
clustering	Grouping observations with common patterns without any class information
deep learning	Complex and varied combination of neural networks, often with many neurons and concatenated models
embedding	To convert a type of matrix into another form
features	Latent properties observed within data
matrix factorization	Decomposition of a matrix into a product of matrices, also known as matrix decomposition
feature extraction	Parsing out latent properties of a dataset
feature learning	Feature extraction without any predefined kernel
neural network	Algorithms purported to function like human neurons
null hypothesis	Claim that there is no relationship between two measurements
statistical significance	Unlikely occurrence of the null hypothesis, not a random event

Abbreviations

AGNews	Antonio Gulli's news
ACC	Accuracy
BSS	Blind source separation
BOW	Bag-of-words
CBOW	Continuous Bag-of-words
char	Character
CMU-PIE	Carnegie Mellon University Pose, Illumination, and Expression
CNN	Convolutional neural networks
COIL20	Columbia object image library with 20 objects
COIL100	Columbia object image library with 100 objects
DAE	Deep autoencoders
DBPedia	Database-pedia
dp	dropout
EEG	Electroencephalogram
FN	False negative
FP	False positive
GNMF	Graph regularized non-negative matrix factorization
ICA	Independent Components Analysis
ICA-SYM	ICA with symmetric Laplacian
IDF	Inverse document frequency
IMDB	Internet Movie Database
kNN	k-nearest neighbor
LSA	Latent semantic analysis

LSTM	long short-term memory
MNIST	Modified National Institute of Standards and Technology
MRI	Magnetic resonance imaging
NMI	Normalized mutual information
NMF	Non-negative matrix factorization
PCA	Principal components analysis
rand	random
RBM	Restricted Boltzmann Machine
ReLU	Rectified linear unit
REUTERS-10K	Reuters news service with 10000 documents
RICA	Reconstruction Independent Component Analysis
RGB	Red, green, blue
SFT	Sparse Filtering
SPC-RW	Spectral clustering with random walk Laplacian
SPC-SYM	Spectral clustering with symmetric Laplacian
SVD	Singular value decomposition
SVM	Support vector machine
TFIDF	Term frequencyinverse document frequency
TN	True negative
TP	True positive
UFL	Unsupervised feature learning
USPS	United States Postal Service
w2v	word2vec
wt	weight decay

List of Symbols

MATRICES	
X	a general variable
x	a scalar value
$\mathbf{X}$	a matrix
$\mathbf{x}_i$	the i^{th} column vector of $\mathbf{X}$
x_{ij}	the i^{th} and j^{th} element of $\mathbf{X}$
SETS	
$\mathcal{S}$	a set
$s \in \mathcal{S}$	s is a member of $\mathcal{S}$
$\{s_1, \ldots, s_n\}$	list of elements of a set S
$\mathbb{R}$	set of real numbers
$\mathbf{x} \oplus \mathbf{y}$	concatenation of $\mathbf{x}$ and $\mathbf{y}$
FUNCTIONS	
$f(x)$	value of the function f at x
$f: X \to Y$	function from X to Y
PROBABILITY	
$P(X)$	probability of X
$P(X, Y)$	probability of X and Y
$P(X\|Y)$	probability of X given Y

Chapter 1

Introduction

Big data is the collection of very large amounts of observations from various sources ranging from social media, such as video content from YouTube or Facebook, to vital signs and genomic sequences from patients in hospitals [1]. Machine learning is the preferred automated method of data analysis for big data applications. The goal of these methods is to detect the patterns or features within the data in order to facilitate predictions about data collected in the future. Typically, machine learning algorithms can be considered from the perspective of two types of analysis methods. When the labels of the data are available, the machine learning problem is considered a "supervised learning" problem and when data is unlabeled, it is considered an "unsupervised" learning" problem.

The first is the supervised learning setting or "learning with a teacher" [2], wherein the output response variables $Y = (Y_1, \ldots, Y_p)$ are predicted from the input variables $X = (X_1, \ldots, X_n)$. For instance, let each i^{th} training case be defined by the input vector $\mathbf{x}_i \in \mathbb{R}^n$, where n is the dimensionality of the vector and a single output response variable $y_i \in \mathbb{R}$. Then a supervised learning algorithm is trained on m observed pairs in a set defined by $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$. In the training of the

algorithm from the data $\mathbf{x}_i$, the algorithm or the "student" provides an estimate $\hat{y}_i$ of y_i. Next, the "teacher" provides an answer by using a loss function $L(y_i, \hat{y}_i)$, giving the student an idea as to how close he/she is to the correct answer.

If the response variable Y is either a categorical or nominal variable, then y_i can take on values $C \in \{1, \ldots, k\}$, such as sunny, windy, and cloudy. This is considered a classification or pattern recognition problem. If Y is a scalar value in $\mathbb{R}$, then the problem is considered as a regression analysis. In both cases, the problem of supervised learning can be considered as a conditional density estimation task of $P(Y|X) = P(X,Y)/P(X)$, where we assume (X, Y) are random variables represented by a joint probability density. The marginal density of $P(X)$ is of no interest since we are interested in the conditional probability. Usually Y is one dimensional, which simplifies the estimation of parameters that minimize the error for each $\mathbf{x}_i$.

The second type is the idea of unsupervised learning or to "learn without a teacher" [2]. This is more similar to how a human or animal may acquire knowledge and learn from its surroundings, without any human expert guiding them through the processes. Thus, it is easier to apply unsupervised learning algorithms than supervised learning because labeled data is not necessary. Moreover, the labeled data is usually difficult to acquire in real-world applications and typically does not provide enough information to solve the parameters of complex learning algorithms.

In unsupervised learning, for an input vector $\mathbf{x}_i \in \mathbb{R}^n$, there is a set m of observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ of a random variable $X = (X_1, \ldots, X_n)$, in which the task is to infer the properties of the probability density $P(X)$. In this case, there is no teacher providing the student or the learning algorithm the correct answer, therefore the algorithm needs to determine properties of $P(X)$ without any supervision. The problem of unsupervised learning is somewhat simpler than supervised learning since the only concern is $P(X)$ and not $P(Y|X)$. However, the properties of interest in

unsupervised learning algorithms are more complex than in supervised learning and the dimension of the observed data is usually much higher.

A common task in unsupervised learning is to organize data into groups, otherwise known as clusters. The goal is to separate observations in a dataset into groups that are more similar to each other within a cluster, than to those in other clusters. Then within these clusters, descriptive statistics and summary representations of the clusters can be obtained. Typically, cluster analysis requires the use of distance measures to assess the similarity of observations. The distance measure only uses the data X, unlike in supervised learning where Y can be used to assess prediction error. In clustering analysis, either the number of groups or clusters required will be known before the clustering step or the number of clusters can be estimated by determining the degree of difference between clusters. Furthermore, using the distance measures between the observations, a graph $G = \langle V, E \rangle$ represents the observations by the vertices V and the similarity between the vertices by the edges E. The purpose of discovering graph structure is to highlight new and interesting meaningful relationships that can also be used to group observations [1].

A very important and related task is the problem of cluster validity, in which the quality of the partitions obtained from the observed data is assessed [3]. The goal is to implement methods that provide a quantitative indicator of the resulting partitions. There are three types of approaches that can be used to implement cluster validity algorithms. The first is external criteria, in which the results of clustering are evaluated against a predefined partition structure, such as a known or expected grouping of the data. The second is internal criteria, where the clusters are evaluated according to a similarity matrix. The third is relative criteria, in which the segmentations provided by the different parameters of the same clustering algorithm are compared against each other.

Another related task in unsupervised learning is to discover the important or hidden features of the observed data, which is also otherwise known as feature extraction [1]. For instance, in a dataset consisting of images of faces, the extracted features may consistent of the angle of lighting, pose of the head, and the facial expression. As a result, any combination of these features can be used to construct the original data. In traditional machine learning, this is commonly accomplished by matrix factorization [2], where the original data matrix is decomposed into a linear combination of basis vectors and weights. Usually the basis vectors are chosen to be of lower dimensionality than the original data. This reduced representation is then used in other modeling algorithms, where the predictive and clustering capabilities of these algorithms are better than the original higher dimensional representation, since only the "essence" of the data is preserved [1].

A recent and popular application in unsupervised learning is to use algorithms termed neural networks, which are nonlinear generalizations of statistical models [1, 2]. The term neural network is derived from the idea that these models are similar to neurons in their processing of data [4]. In fact, a key capability of neural network models is that they can be stacked (connected) with each other, much like biological neurons, in which a signal from one neuron is passed onto the next. These stacked neural network models are called deep neural networks or "deep learning" algorithms when many layers of neural networks are stacked together [5]. Deep learning algorithms perform similar operations to matrix factorization, but the mapping of the features from the original data representation to the extracted representation is commonly performed using a nonlinear mapping function rather than a linear mapping function. In matrix factorizations, nonlinear mapping may be performed using predefined kernel functions. However, with neural networks, the type of nonlinear mapping is "learned" from the data rather than being defined a priori [6]. As a re-

sult, complex and difficult features can be captured with minimal domain knowledge [7].

Until recently, deep learning models had limited implementations due to the lack of computational power and numerical techniques necessary for solving the numerous model parameters [8, 9]. Ever since this limitation has been overcome, deep learning algorithms have become very popular. Particularly, for their ability learn features directly from the data rather than to design (handcraft) them [5]. Deep learning models can be used as unsupervised learning or supervised learning models, depending on the availability of class labels. In the supervised learning setting, deep learning has provided state-of-the-art classification performance in tasks like object recognition from images [10]. In unsupervised learning, deep learning models have shown top performance in extracting the hidden features for object, handwritten-digits, and face recognition from image data [11].

Nonetheless, the drawback of deep learning methods is that they tend to have many hyper-parameters (parameter that are set before the model is run), such as optimizer parameters, sparsity parameters, and number of features and layers. All of which can make deep learning models difficult to train, since hyper-parameters can severely effect performance [12]. Typically, choosing the correct hyper-parameters requires expertise and ad hoc selection [13, 14]. Despite the high performance of deep neural network methods, setting up deep learning models can be as difficult and complex as the handcrafted models which they were to replace [15]. Moreover, the high degree of complexity in implementing deep learning-based algorithms [12] may be a limiting factor of their application in non-computer science based research fields.

1.1 Overview of Contributions

Currently, there is great deal of attention to deep learning methods regarding their success in the supervised learning problem of classification and unsupervised learning problems of feature extraction and cluster analysis. Although warranted, traditional machine learning methods can still hold ground in terms of performance in suitable problem settings against deep learning methods.

First, we present a parsimonious and accessible clustering scheme that incorporates deep learning-style feature extraction, but without the complex hyper-parameter tuning procedure. Based on the empirical studies demonstrated in this thesis, we are able to the bridge gap between standard unsupervised learning techniques and the recently developed deep learning techniques. By carefully combining traditional techniques for clustering and feature extraction along with careful parameter selection, state-of-the-art performance that is better than many deep learning-based methods in cluster analysis and feature extraction can be achieved.

Then, we present two novel techniques for testing the significance of clusters in order to determine whether the input observations have been grouped into meaningful clusters by virtue of the extracted features and not because of some random occurrence of the underlying properties of the data. This is mainly accomplished by hypothesis testing against simulated data derived from the properties of the original. Significance testing provides a complementary assessment of cluster performance alongside effect size based performance metrics of clusters, such as accuracy.

Finally, given the success of employing carefully tuned traditional matrix factorization techniques in clustering, we present how to successfully use traditional matrix factorization techniques as a preprocessing step in deep learning classification models for text documents in order to achieve state-of-the-art accuracy. To leverage the capabilities of traditional matrix factorization techniques in document classification,

we developed a new convolutional neural network architecture that is better than .

1.2 Outline

The following is an outline of the remaining sections of the thesis:

- **Chapter 2 – Literature Review:** This chapter covers the literature and background methods that will help to understand the rationale of the three main experiments in the Methods section.
- **Chapter 3 – Methods:** In this chapter, the three novel methodological developments consisting of (1) the clustering pipeline, (2) cluster significance testing, and (3) matrix factorization boosted convolutional neural networks for text documents are reviewed.
- **Chapter 4 – Results:** Here the results obtained from the three main algorithms of this thesis are presented and compared against competing methods.
- **Chapter 5 – Discussion:** This chapter covers the methodological implications of the three main proposed algorithms in this thesis.
- **Chapter 6 – Conclusions:** Here the main findings of the thesis and future avenues for research are highlighted.

1.3 First Published Appearances

Below, the two published studies and one submitted using the methods and results presented in this study are summarized:

1) Clustering Pipeline Using Independent Component Analysis and Unsupervised Feature Learning. In our initial study, the pipeline for clustering data using deep learning-inspired feature extraction is tested with six benchmark datasets, ranging from images to text documents. Extracting features using neural networks and deep learning algorithms is referred to as unsupervised feature learning (UFL). However, to overcome the complexities of training deep networks, several studies had introduced deep learning-inspired feature learning algorithms. These algorithms use the same principles as deep learning (extracting features without any handcrafting), but do not have to use neural networks as their feature extraction algorithm. These features were then used in graph-based clustering techniques (such as standard spectral clustering algorithm). Specifically for this study, we used reconstruction Independent Component Analysis (RICA) and Sparse Filtering (SFT) as the feature learning algorithms. Both of which are not based on neural networks and yet retain the benefit of having only one hyper-parameter, the number of features to extract.RICA and SFT helped to achieve very high accuracies on benchmark data, besting many deep clustering models.

Furthermore, blind source separation (BSS) using Independent Component Analysis (ICA) with principal components analysis (PCA) feature extraction was also include in the clustering pipeline to provide a simple feature extraction baseline. Surprisingly, this combination had better accuracy than most deep learning-based clustering methods. ICA is a mathematical model resolving the "cocktail party problem" or un-mixing source signals into their separate components. Thus, ICA was also used with spectral clustering to improve the factorization of the eigenvectors, which performed better than most state-of-the-art deep clustering techniques. We combined ICA with another graph-based clustering technique, Graph Regularized Non-negative Matrix Factorization, with the same success as spectral clustering. Also, the study

demonstrated that combining ICA blind source separation with unsupervised feature learning had a cumulative effect on increasing clustering performance. Overall, the main findings of this study indicated that standard clustering techniques can achieve effective clustering performance without employing deep learning algorithms and their accompanying hyper-parameter tuning procedure. This study has been published as,

- E. Gultepe, and M. Makrehchi, "Improving clustering performance using independent component analysis and unsupervised feature learning," *Human Centric Computing and Information Sciences*, 2018. doi:10.1186/s13673-018-0148-3

2) Significance Testing of Clusters. In our second study, we used a simplified variant of the clustering pipeline presented above on a dataset of paintings, without any consideration about the styles labels of these paintings. These features were then used with a standard spectral clustering algorithm to solve the "grouping" problem of painting styles. This problem has received very little attention in digital humanities literature due to its difficulty of finding appropriate clusters and proper performance evaluation. To solve this problem we introduced to two novel techniques for significance testing of clusters. The significance and reliability of the painting style clusters were determined using two novel methods, in which the null-hypothesis statistical distribution is formed either by: (1) a uniform distribution projected onto the principal components of the original data; or (2) a randomized, weighted adjacency matrix. When there is uncertainty regarding the effect size of the clustering accuracy, significance testing can provide a complementary reliability assessment. This work has been published as,

- E. Gultepe and M. Makrehchi, "Predicting and grouping digitized paintings by style using unsupervised feature learning," *Journal of Cultural Heritage*, vol. 31, pp. 1323, 2018. doi:10.1016/j.culher.2017.11.008

3) Latent Semantic Analysis Boosted Convolutional Neural Networks. Using the clustering pipeline as an inspiration from our first study, we demonstrated that traditional matrix factorization techniques of text documents could be used to improve the accuracy of deep learning classification of documents. To showcase the efficacy of traditional matrix factorization, we developed a new deep learning architecture that takes advantage of the vector representation of words based on singular value decomposition (SVD), which is called latent semantic analysis (LSA). Using this new deep learning architecture, the LSA-based word vector representation was able to achieve higher accuracy than state-of-the-art deep learning-based word vector representation on three benchmark text document datasets. Furthermore, due to the parsimonious design of the deep learning model developed in this study, the model together with the SVD-based word vector can be used as a baseline classifier in text document classification, similar to popular linear models. Thus, this study shows the efficacy of traditional word vectors in deep neural network document classification. This work has been submitted as a full-paper to a conference as,

- E. Gultepe, M. Kamkarhaghighi, and M. Makrehchi, "Latent semantic analysis boosted convolutional neural networks for document classification," *5^{th} International Conference on Behavioral, Economic, and Socio-Cultural Computing 2018.*

Chapter 2

Literature Review

2.1 Background

Important concepts and methods that will help in the understanding of the experiments of the subsequent chapters are covered here.

2.1.1 Clustering Methods

Two key unsupervised learning techniques for clustering data are reviewed here.

K-means

K-means is one of the most widely used clustering due to its simple and quick implementation. Given training data with m observations, $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$, we seek to cluster each $\mathbf{x}_i \in \mathbb{R}^n$ observation, without using any label y_i information, into unique $C_i \in \{1, 2, \ldots k\}$ clusters. The K-means algorithm proceeds as follows [16]:

1. Initialize each cluster centroid $\{\mu_1, \mu_2, \ldots, \mu_k\}$ randomly, where the j^{th} centroid is $\mu_j \in \mathbb{R}^n$ and k is the number of clusters.

2. Repeat until convergence:

For all i, set

$$C_i := \underset{j}{\operatorname{argmin}} \left\| \mathbf{x}_i - \ \mu_j \right\|^2 .$$

For each j, set

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{I}\left\{C_i == j\right\} \mathbf{x}_i}{\sum_{i=1}^{m} \mathbf{I}\left\{C_i == j\right\}}.$$

The centroids are the estimates of the centers of the clusters and where $\mathbf{I}$ is the identity matrix. The algorithm works by assigning each observation $\mathbf{x}_i$ to its nearest centroid μ_j and then updating the centroids μ_j with a new mean of the observations. Thus K-means provides cluster labels from unlabeled data and can extract features from the centroids.

Spectral Clustering

Spectral clustering is a technique that is able to cluster non-spherical data better than K-means, because it takes into consideration the underlying geometrical structure of the data[16]. Here eigendecomposition is used to factorize or embed the graph Laplacian into a smaller k dimensional representation, where k is the number of classes. Then K-means is usually applied to reduced space to obtain the class memberships, $C_i \in \{1, 2, \ldots k\}$ of $\mathbf{x}_i \in \mathbb{R}^n$ for m observations.

For data $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$, with k clusters the spectral clustering algorithm using the symmetric normalized graph Laplacian proceeds as follows [17]:

1. Construct the similarity graph $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{m \times m}$. First form the similarity matrix $\mathbf{S} = (s_{ij}) \in \mathbb{R}^{m \times m}$, using the Gaussian similarity function, $s\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp(-\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 / 2\sigma^2)$ controlled by the scaling factor σ. Then, using the k-nearest neighbor (kNN) criterion, keep only the k-nearest neighbors of each $\mathbf{x}_i$ observation in the similarity matrix [17], thus giving the weighted adjacency matrix or the similarity graph $\mathbf{W}$.

2. Define the normalized graph Laplacian according to $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ [18], where $\mathbf{D} = (d_{ij}) \in \mathbb{R}^{m \times m}$ is the diagonal degree matrix with diagonal elements set to $d_{ii} = \sum_{i=1}^{m} w_{ij}$ and off-diagonal elements are set to 0. $\mathbf{I}$ is the identity matrix.

3. Compute the first k eigenvectors of $\mathbf{L}$ and concatenate the eigenvectors column-wise to form $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \mathbb{R}^{m \times k}$.

4. Define $\mathbf{Y} = (y_{ij}) \in \mathbb{R}^{m \times k}$ as the row normalized matrix of $\mathbf{U}$ where $y_{ij} = u_{ij} / \left(\sum_{l=1}^{k} {u_{il}}^2 \right)^{1/2}$.

5. Apply K-means clustering to the rows of $\mathbf{Y}$ to obtain clusters $C_1, \ldots, C_k$.

2.1.2 Classification

For some of the experiments presented in this thesis, the features extracted from matrix factorization techniques, deep learning, or deep learning-inspired methods are used to classify data in a supervised learning setting. Typically, the features extracted using these algorithms handled the complex relationships of the features within the data, thus a simple linear classifier would suffice to provide good accuracy.

Logistic Regression

A binary logistic regression function is defined as [1]

$$p\left(y|\mathbf{x}, \mathbf{w}\right) = \text{Ber}\left(y|\text{sigm}\left(\mathbf{w}^T\mathbf{x}\right)\right)$$

where Ber() is the Bernoulli distribution providing binary class labels $y \in \{0, 1\}$, $\mathbf{x} \in \mathbb{R}^n$ is an observation from the dataset, and $\mathbf{w} \in \mathbb{R}^n$are the weights or parameters

of the model. The sigmoid function is defined as

$$\text{sigm}(t) = \frac{1}{1+\exp(-t)}$$

for some random variable $t \in \mathbb{R}$. The logistic binary classification problem can be easily extended to a multiclass problem. This is defined as multinomial logistic regression or otherwise known as the softmax function. The softmax function is commonly used in supervised learning models of deep neural networks to provide the class memberships. The multinomial logistic regression for class labels $y \in \{1, 2, \ldots, k\}$, a random observation $\mathbf{x} \in \mathbb{R}^n$, and model weights $\mathbf{W} = (\mathbf{w_1}, \ldots, \mathbf{w_k}) \in \mathbb{R}^{n \times k}$, can be defined as [1]

$$p\left(y = i|\mathbf{x}, \mathbf{W}\right) = \frac{\exp\left(\mathbf{w}_i^T\mathbf{x}\right)}{\sum_{j=1}^{k}\exp\left(\mathbf{w}_j^T\mathbf{x}\right)}$$

where $i = 1, \ldots, k$. The formula can be solved by obtaining the maximum likelihood estimates of the parameters using Newton's method.

Support Vector Machines

A support vector machine (SVM) finds a linear separating hyperplane with the maximal margin between two different classes of data [19]. Given $\mathbf{x}_i \in \mathbb{R}^n$ and $\text{y}_i \in \{1, -1\}$, for m training pairs $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, SVM with hinge loss [20] is used to perform classification using the following definition

$$\min_{\mathbf{W}} \sum_{i=1}^{m} \max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right)\right) \quad + \lambda\|\mathbf{w}\|^2$$

where $\mathbf{w} \in \mathbb{R}^n$ are the model parameters, b is the boundary between two classes, and λ is a regularization parameter. To extend the binary classification to a multiclass problem, the one-vs-rest strategy [20] is followed, wherein one class is defined as

positive and all other classes are negative. This strategy is followed for each unique class in the dataset and the maximum score for $\mathbf{x}_i$ pertaining to a class $C_i = \{1, \ldots, k\}$ is used to identify the class label.

2.1.3 Matrix Factorization Techniques

Some of the key matrix factorization techniques are reviewed here. These techniques have many applications such as feature extraction and embedding data into lower dimensional space.

Singular Value Decomposition

For a matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$, with m observations, n-dimensional features, and rank $r \leq \min(m, n)$, the singular value decomposition (SVD) can be formed as $\mathbf{X} = \mathbf{USV}^T$. The matrix $\mathbf{V} = (v_{ij}) \in \mathbb{R}^{n \times n}$ of right singular vectors, represents the extracted features. The matrix $\mathbf{U} = (u_{ij}) \in \mathbb{R}^{m \times m}$ is the left singular vectors. The diagonal matrix $\mathbf{S} = (s_{ij}) \in \mathbb{R}^{m \times n}$ is the matrix of ordered singular values in decreasing order. A low- rank approximation $\mathbf{X}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^{\mathrm{T}}$ can be obtained using the k largest singular values along with the corresponding singular vectors.

Principal components analysis (PCA) feature extraction can be formed by the multiplication $\mathbf{U}_k \times \mathbf{S}_k$, which yields the transformed observations onto the extracted feature space [2]. If PCA is performed prior to classification, where there is training and testing data, the test data can be projected onto the training feature space by the multiplication $\mathbf{X}^{(test)} \times \mathbf{V}_k^{(train)}$.

Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is matrix decomposition technique in which the data and its components are assumed to be non-negative [21]. This assumption

has been shown to be useful in face recognition [22] and document clustering [23], due to the non-negative constraint that forces an additive model of the decomposed components [22]. NMF is performed by decomposing a matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$ into non-negative matrices $\mathbf{U} = (u_{il}) \in \mathbb{R}^{m \times k}$ and $\mathbf{V} = (v_{jl}) \in \mathbb{R}^{n \times k}$ where $k \leq \max(m, n)$ and $x_{ij},\ u_{il},\ v_{jl}\ \geq 0$. Then the matrix $\mathbf{X}$ is approximated by

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^T.$$

The matrices $\mathbf{U}$ and $\mathbf{V}$ are determined by maximizing the log-likelihood function for the "divergence" between two matrices [24]

$$L(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij} \log\left(\mathbf{U}\mathbf{V}^T\right)_{ij} - \left(\mathbf{U}\mathbf{V}^T\right)_{ij} \right)$$

where x_{ij} has a Poisson distribution with mean $\left(\mathbf{U}\mathbf{V}^T\right)_{ij}$ since the data is positive. In NMF, typically $k \ll m$ and $k \ll n$, which results in a compressed version of the original data $\mathbf{X}$. Thus, the approximation can be described as a linear combination of the column vectors of $\mathbf{U}$ weighted by the components of $\mathbf{V}$

$$\mathbf{x}_i \approx \sum_{l=1}^{k} \mathbf{u}_l v_{il}.$$

Independent Component Analysis

Independent component analysis (ICA) is a solution to the problem of blind source separation (BSS). ICA is similar to PCA in that represents the data using new basis functions. However, its rational for doing so is different. A canonical example used to describe ICA is the "cocktail party problem", in which the objective is separate out the voices from n individuals in a room that were recorded by n randomly placed microphones in a room. The problem here is that the randomly placed microphones

measures a different mixture of voices. In this case ICA is used to unmix the source signals, which are the voices. Formally ICA is defined as

$$\mathbf{x}_i = \mathbf{A}\mathbf{s}_i$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the mixing matrix, $\mathbf{x}_i \in \mathbb{R}^n$ are the mixed signal observations, and $\mathbf{s}_i \in \mathbb{R}^n$ are the signal sources. The goal is to find a mixing matrix that maximizes non-Gaussianity and minimizes the mutual information among the sources [25]. Then, the unmixing matrix $\mathbf{W} = \mathbf{A}^{-1}$ can be applied to recover $\mathbf{s}_i = \mathbf{W}\mathbf{x}_i$, or in this case the voices of each individual.

2.1.4 Neural Networks and Deep Learning

The learning models discussed so far follow a two-layer architecture [1]. In unsupervised learning, this is observed as the features being extracted from the original data using matrix factorizations. In supervised learning, this is seen as the mapping of the observed data to labels. A class of models that can extend the two-layer specification are the neural network models. Neural networks, in their simplest form, are based on a single neuron $g(z)$ that can be stacked or connected with as many other neurons as needed. Some common functions that are used to define a neuron are the identity, sigmoid, or rectified linear unit (ReLU) functions, which are shown below

$$g(z) = \frac{1}{1 + \exp(-z)} \qquad \text{(sigmoid)}$$

$$g(z) = \max(0, z) \qquad \text{(ReLU)}$$

$$g(z) = z \qquad \text{(indentity)}$$

for some random variable $z \in \mathbb{R}$. ReLu and sigmoid are both nonlinear functions. Thus a function $g(z)$ is used to calculate the "activation" of a neuron, which is usually denoted by a vector $\mathbf{a}$ in the hidden layers. The simplest form of a neural network consists of an input layer of data, a hidden layer of the features learned by the model, and output layer.

Depending on the specification of the output layer, a neural network can be used to perform supervised learning for regression or classification and unsupervised learning for feature extraction. A three-layer neural network (input, hidden, and output layers) for performing classification on m observations using logistic regression is shown for an n-dimensional input vector $\mathbf{x} \in \mathbb{R}^n$ and binary label $y \in \{1, -1\}$

$$
\begin{aligned}
\mathbf{a}^{[0]} &= \mathbf{x} && \text{(indentity)} \\
\mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]} \\
\mathbf{a}^{[1]} &= \frac{1}{1 + \exp(-\mathbf{z}^{[1]})} && \text{(sigmoid)} \\
z^{[2]} &= \mathbf{W}^{[2]}\mathbf{a}^{[1]} + b^{[2]} \\
a^{[2]} &= z^{[2]} && \text{(indentity)} \\
y &= a^{[2]}
\end{aligned}
$$

where the layer numbers are indicated by "$[i]$" superscript. The input layer is denoted by the "[0]" superscript, the hidden layer by "[1]", and output layer by "[2]". The model weights/parameters are $\mathbf{W}^{[1]} \in \mathbb{R}^{k \times n}$ for the hidden layer and $\mathbf{W}^{[2]} \in \mathbb{R}^{1 \times k}$ for the output layer. The bias parameters are denoted by $\mathbf{b}^{[1]} \in \mathbb{R}^{1 \times k}$ and $b^{[2]} \in \mathbb{R}$. In the weight layers, k is the number hidden neurons, which are used to learn the mapping from the n input features to a reduced feature space representation of k dimensions. The model classifies the input $\mathbf{x}$ based on this reduced feature representation. The model and bias parameters are trained with gradient descent using cross entropy as

the loss function on the output layer values [26].

Using a similar architecture to the logistic regression network, feature extraction or "feature learning" using neural networks can be performed. Except in this case, the output label layer is replaced with an estimate of the input features, which makes the model an unsupervised learning algorithm. This type of neural network is called an autoencoder [5] and is shown in Figure 2.1. An example architecture of an autoencoder for a n-dimensional random observation $\mathbf{x} \in \mathbb{R}^n$ reduce to k features is shown

$$
\begin{aligned}
\mathbf{a}^{[0]} &= \mathbf{x} && \text{(indentity)} \\
\mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{a}^{[0]} + \mathbf{b}^{[1]} \\
\mathbf{a}^{[1]} &= \frac{1}{1+\exp(-\mathbf{z}^{[1]})} && \text{(sigmoid)} \\
\mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \\
\mathbf{a}^{[2]} &= \mathbf{z}^{[2]} && \text{(indentity)} \\
\widehat{\mathbf{x}} &= \mathbf{a}^{[2]}
\end{aligned}
$$

where $\widehat{\mathbf{x}} \in \mathbb{R}^n$ is an estimate of $\mathbf{x}$, $\mathbf{W}^{[1]} \in \mathbb{R}^{k\times n}$, $\mathbf{W}^{[2]} \in \mathbb{R}^{n\times k}$, $\mathbf{b}^{[1]} \in \mathbb{R}^k$, and $\mathbf{b}^{[2]} \in \mathbb{R}^n$. In the autoencoder, the activation vector $\mathbf{a}^{[1]} \in \mathbb{R}^k$ is the input vector $\mathbf{x}$ transformed on to the k extracted features of $\mathbf{W}^{[1]}$. For a data matrix with m observations, $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m\times n}$, the data would be transformed as $\mathbf{X}_{new} = (x_{ij}) \in \mathbb{R}^{m\times k}$, similar to the matrix factorization techniques discussed previously. In the autoencoder, if in place of the sigmoid activation function, the identity function was used, then the autoencoder would become a PCA model. Neural network are able to perform "feature learning" due to the nonlinear "activation functions" $g(z)$. For images, many of these "learned features" appear like Gabor filters, which until the application autoencoders to images, needed to be handcrafted.

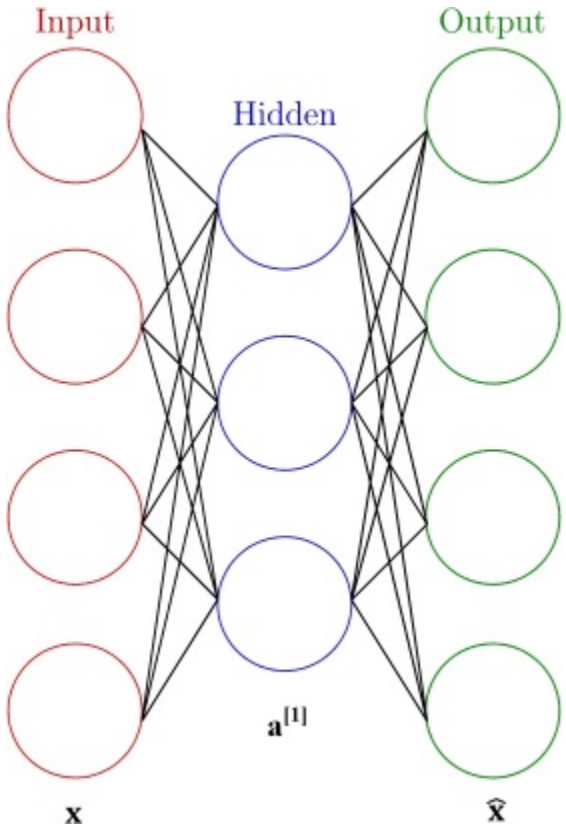

Figure 2.1: Autoencoder neural network. A three-layer neural network that encodes the input features $\mathbf{x}$ into a hidden layer of features $\mathbf{a}^{[2]}$) by matching the reconstruction $\widehat{\mathbf{x}}$.

With current computational resources, the number of hidden layers can be increased greatly, hence the name deep learning for neural networks with many hidden layers and features. Current models in vision research can have at least 19 layers [27].

2.1.5 Unsupervised Feature Learning

Here we describe a general pipeline for learning features (feature extraction) from images without using any label information from the data. This framework, called unsupervised feature learning (UFL) [12, 15, 28], is able to extract high quality features using a wide range of feature extraction algorithms, ranging from traditional matrix factorization techniques like PCA, ICA, and NMF to neural network tech-

niques like autoencoders, and even using the centroids from K-means. UFL showed that using careful image preprocessing procedures, equivalent performance to features extracted from supervised deep learning models is possible [12, 15, 28].

The procedure for performing UFL on the input data $\mathbf{X} \in \mathbb{R}^{m \times n}$, where m is the number of observations and n is the number of pixels in an image, is given below (for details please refer to [12, 15, 28]):

1. Extract l random small patches of size $N = \ w \cdot w \cdot d$, where w is the receptive field size in pixels and d is the dimension of either a grayscale (1 channel) or RGB (3 channels) color image. The input patches for feature learning is a set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_l\}$, where $\mathbf{p}_i \in \mathbb{R}^N$.

2. Apply normalization to each $\mathbf{p}_i$ by mean subtraction and dividing by the standard deviation of the pixels, which is followed by pre-whitening (decorrelation of neighboring image pixels).

3. Apply the feature learning algorithm, which can be any unsupervised learning algorithm that provides the mapping $f : \ \mathbb{R}^N \rightarrow \mathbb{R}^K$, where K is the number of extracted features.

4. Extract mapped features for the original input images or test set images by performing matrix convolution with the learned feature mapping $f : \ \mathbb{R}^N \rightarrow \mathbb{R}^K$.

5. Pool the features over quadrants to reduce the number of features of the mapped data.

Once the features are extracted and pooled, clustering or classification can be performed using any other algorithm.

2.1.6 Performance Evaluation

For all datasets, we have the ground-truth label information at our disposal, thus standard metrics can be used to assess classification and clustering performance. The following metrics are used in this thesis:

$$\text{F} - \text{score} = 2 \cdot P \cdot R/(P + R)$$

$$\text{Accuracy} = (TP + TN)/(TP + FP + TN + FN)$$

where $P = TP/(TP + FP)$ is the precision and $R = TP/(TP + FN)$ is the recall (also known as sensitivity). In the above definitions, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

For clustering problems, since the algorithms are blind to the class labels, combinatorial optimization is used to determine the lowest cost matching of the clustered labels and the ground-truth labels. We also use the normalized mutual information (NMI) metric for evaluating clustering performance. NMI measures the dependence of clusters on each other and is independent of the label permutations of the clusters. It is defined as [23]

$$\text{NMI}(C, C^{'}) = \frac{\text{MI}\left(C, C^{'}\right)}{\max\left(H\left(C\right), H\left(C^{'}\right)\right)}$$

where C and $C^{'}$ are two sets of cluster, $\text{MI}\left(C, C^{'}\right)$ is the mutual information between them, and $H\left(C\right)$ and $H\left(C^{'}\right)$ are the individual cluster entropies.

2.2 Relevant Literature

Related work and important studies related to each of the three main algorithms of this thesis are reviewed below.

2.2.1 Clustering Pipeline using ICA and UFL

Grouping observed data into cohesive clusters without any prior label information is an important task. Especially, in the era of big-data, in which very large and complex amounts of data from various platforms are collected, such as image content from Facebook or vital signs and genomic sequences measured from patients in hospitals [1]. Often, these data are not labeled and a significant undertaking is typically required (usually by individuals with domain knowledge). Even in simple tasks, such as labeling images or video data can require thousands of hours [29, 30]. Therefore, using the unsupervised learning technique of cluster analysis can aide in the process of providing labels to observed data [2].

Classical clustering algorithms used for analysis are K-means [31], Gaussian Mixture Models [20], and hierarchical clustering [2], all of which are based on using a distance measure to assess the similarity of observations. The choice for distance measure is typically data dependent. For instance, in image data, the similarity between pixels can be represented by the Euclidean distance, where as in text documents cosine distance matrix is typically used [22]. Moreover, appropriate feature representation of the observations is even more critical in order to obtain correct clusters of the data [32], since improved features provide a better representative similarity matrix.

Early approaches for learning the appropriate feature space in clustering algorithms implemented deep autoencoders (DAEs) [5]. Song et al. [33] used DAEs to

directly learn the data representations and cluster centers. Huang et al. [34] employed a DAE with locality and sparsity preserving constraints, which is followed by a K-means to obtain the cluster memberships. A more recent and popular approach by Xie et al. [32] learned the feature space and cluster membership directly using a stacked denoising autoencoder [35]. Following Xie et al. [32], there have been many studies proposing deep clustering algorithms to learn the feature space and cluster membership simultaneously using some form of an autoencoder [13, 36, 37]. A departure from the autoencoder framework was demonstrated by Yang et al. [38], who used recurrent and convolutional neural networks with agglomerative (hierarchical) clustering.

Another class of clustering algorithms, called spectral clustering [18, 39], is based on embedding the graph structure of the data through eigendecomposition (also known as spectral decomposition) of the Laplacian matrix [17]. Spectral clustering usually performs better than K-means and the aforementioned classical algorithms due to its ability to cluster non-spherical data [2]. A key issue in spectral clustering is to solve the multiclass clustering problem. This is accomplished by representing the graph Laplacian in terms of k eigenvectors, k being the number classes [40]. Then, either K-means clustering [18], exhaustive search [39], or discretization [41] is applied to this lower dimensional representation of the Laplacian to determine the final cluster memberships. Recently, autoencoders have been applied on the Laplacian to obtain the spectral embedding provided by the eigenvectors [42]. Another approach has been to use a deep learning network that directly maps the input data into the lower dimensional eigenvector representation, which is then followed by a simple clustering algorithm [43].

The drawback of deep learning methods is that they tend to have many hyperparameters, such as learning rates, momentum, sparsity parameters, and number of

features and layers. All of which can make deep learning models difficult to train, since hyper-parameters can severely effect performance [12]. Typically, choosing the correct hyper-parameters requires expertise and ad hoc selection [13, 14]. However, the high degree of complexity in implementing deep learning-based algorithms [12] may be a limiting factor of their application in non-computer science based research fields. To have real-world applicability, clustering applications need to have as few hyper-parameters as possible [13].

In this study, the aim is to provide a parsimonious and accessible clustering processing scheme that incorporates deep learning-style feature extraction, but without the complex hyper-parameter tuning procedure. The goal is to bridge the gap between deep learning-based clustering methods and widely available standard clustering techniques. This is accomplished by using two procedures. First, we improve the clustering accuracy of standard clustering algorithms by applying Independent Component Analysis (ICA) [25] blind source separation (BSS) after the initial matrix factorization step in principal component analysis (PCA) and graph-based clustering algorithms. Second, we improve the features used for constructing the distance matrix in graph-based clustering techniques by performing feature extraction using deep learning-inspired feature learning techniques. Prior to any clustering algorithm, we implement the unsupervised feature learning (UFL) algorithms of ICA with reconstruction cost (RICA) [44] and sparse filtering (SFT) [45], both of which have only one tunable hyper-parameter – the number features [45].

By implementing these two procedures we demonstrate that effective clustering performance that is on par with more complex deep learning clustering models can be achieved. Thus, the clustering methodologies provided herein are designed to be simple to train and implement in different data applications.

2.2.2 Significance Testing of Clusters

Finding meaningful patterns in high dimensional and complex data in scientific, medical, and social media data is one of the key challenges in machine learning. This is particularly challenging when the number of observations or samples of the data are low relative to the number of features of the data [46]. An example of this problem is in functional neuroimaging studies, where the goal is to discover a pattern of brain activity measured using technology such as electroencephalogram (EEG) or magnetic resonance imaging (MRI) that is indicative of brain function processing specific tasks. Typically in these studies the number of subjects is low ($10 - 20$ subjects per study) in comparison to the high dimensionality of the time-series data (thousands of time points), which results in a high degree of variance [46]. Furthermore, the sample size of the study is very low even compared to the general population. Thus, being able to determine the reliability in the accuracy of the discovered pattern would have implications in terms of the generalizability of the finding.

The reliability can be assessed using hypothesis testing, which would assign a significance level to the discovered pattern in the data. This has been study extensively with respect to classification problems, in which the null hypothesis assumes that due to some random occurrence of the data there is correlation between the class labels and the data [46, 47, 48]. For clustering analysis, the null hypothesis assumes that the clustering structure is due to some random occurrence of the underlying properties of the data [3]. However, there is a lack of studies in testing for the significance of clusters, where the focus has been mainly to find number of clusters inherent to a dataset [49]. For gene expression microarray data, there have been a number studies analyzing the stability of phylogenetic trees and hierarchical clustering using bootstrapping to determine confidence intervals [50, 51, 52]. Confidence intervals for clusters differs from significance testing in that the confidence intervals provides a range of "possible"

cluster structures from which would representative of the population data.

To perform the significance testing for clustering, the null hypothesis needs to be formed from randomly generated data. However, this is not a trivial problem as the random data needs to preserve some of the underlying information regarding the original data distribution. In nonlinear dynamical systems, this is known as generating surrogate data [53, 54], where the goal is to perform a hypothesis test to determine whether the system significantly different from a linear system. For generating random data in nonlinear systems, many of the random data generations implementations take advantage of the underlying data structure by keeping constant one or two these properties (e.g., signal amplitude) while randomizing the remaining properties (e.g., signal phase) [53, 55, 54].

Following the same principal, to create the null hypothesis for significance testing clusters, we implement two novel random distributions formed either by: (1) a uniform distribution projected onto the principal components of the original data; or (2) a randomized, weighted adjacency matrix. Thus, in this thesis we provide a two new methods for determining whether the cluster structure obtained is a real and not due to a random occurrence in the data. For instance it may be possible to achieve high clustering accuracy, but this may be due to the randomness of the data. Also the converse problem may be the case where, a cluster exhibits low accuracy, yet it is not due to chance – i.e., it is significant effect, just small in magnitude.

The two novel distributions for hypothesis testing are applied to the "grouping problem of paintings" [56], in which a naïve person is unable to cluster paintings with similar styles without expert knowledge of the art field. The grouping problem is due to the naïve person's inability to provide the necessary prior style information during the training phase of deep neural networks such as CNNs [57]. In our work, we overcame this limitation of CNNs in feature learning and artistic style clustering

by using unsupervised feature learning.

2.2.3 LSA Boosted Convolutional Neural Networks

Automated text classification is a classic problem where the objective is to assign a set of categories to documents. The studies in text classification vary from developing a sophisticated document feature representation methodology [58, 59] to implementing the best possible classifiers that use simple document representations [60]. A common approach in text classification is the bag-of-word as ngrams representation, where documents are represented with a vector of words that appear in each document [61]. Although, ngrams are very simple to generate, however, the main challenge in such a presentation is that the resulting vector is very large and sparse. The sparsity and semantically understanding text documents are the major challenges in text categorization [62, 63]. In fact, data sparsity is encountered in ngrams due to the length of generated text and the use of different words with the same meaning.

Recent studies [64] have used 1-dimensional convolutional layers (1D-CNNs) directly with one-hot vectors representation of bag-of-words data. Typically this is not well-suited for convolution networks, wherein dense data is preferred. To mitigate this issue, an embedding layer to densify the one-hot vectors is preferred prior to the convolutional layers [65, 66, 67, 68]. An embedding layer takes a large vocabulary and projects the full representation into smaller dimensional space [65]. Furthermore, in sentiment analysis [69, 70], CNNs with an embedding layer are similar to long short-term memory (LSTM) models [71] that have an embedding layer. However, LSTMs are known to be difficult to train since they are sensitive to hyper-parameters such as batch size, hidden dimensions, and etc. [72].

Recently, pretrained word vectors obtained from an unsupervised neural network language model (word2vec [73]) have been successfully used as trainable weights into

the embedding layer, prior to the convolutional layers [65]. The word2vec (w2v) word vectors were trained on 100 billion words of Google News. The goal in [65] was similar to [74], which showed that for image classification, pretrained features from a different domain can be fine-tuned to domain specific-tasks. However, if the domain of the document classification task is very different from the pretrained vectors, classification accuracy may not reach its full potential, since the purpose of unsupervised pretraining is to provide relevant features that can improve accuracy [75]. There are limitations for training word2vec models from natively. The first is that it requires a large datasets for training. For satisfactory performance, a minimum of 10 million words should be in the training corpora [76]. For smaller datasets, LSA word vectors were found to provide better performance in semantic similarity tasks [76]. The second is that there are a number of hyper-parameters that need to be chosen before successful model training can be implemented [77].

Thus, the goal of this study is to develop simple CNN-based baseline for text classification using locally trained word vectors. We use locally trained word vectors from an LSA model, which is easily obtained by SVD on a unigram transformed and term frequency–inverse document frequency (TFIDF) weighted data. Our model has the advantage of learning the words embeddings natively, without the onerous hyper-parameter search and extensive training time. Thus, in the following sections we highlight how to efficiently setup the LSA-based convolution networks and demonstrate its effectiveness against baseline linear classifiers and pretrained w2v-based CNN models.

Chapter 3

Methods

The methods developed are given first and then the related methodological advancement and issues are provided.

3.1 Clustering Pipeline using ICA and UFL

An overview of the clustering pipeline implementing unsupervised feature learning and ICA blind source separation is provided in Figure 3.1. The clustering pipeline consists of four key components: (1) feature extraction, (2) graph construction, (3) graph embedding, and (4) K-means clustering.

3.1.1 Benchmark Data

In order to test the generalizability of the proposed clustering methodology, six benchmark datasets are used for the experiments; five image datasets and one text dataset (see Table 3.1 for a summary of datasets). Each dataset is briefly described below:

1. **COIL20:** grayscale images for object recognition dataset containing 20 objects positioned at 72 different angles [78].

2. **COIL100:** RGB images of 100 objects at 72 different poses [102]. The images were downsampled to 32 × 32 pixels from the original 128 × 128 pixels to facilitate analysis for unsupervised feature learning [12].

3. **CMU-PIE:** grayscale images of 68 human faces with 4 different poses [79].

4. **USPS:** grayscale images of handwritten digits (0 – 9) from the USPS postal service [80].

5. **MNIST:** grayscales images of handwritten digits (0 – 9) obtained from NIST [81].

6. **REUTERS-10K:** A Reuters news service dataset containing text documents in English that is used for topic recognition. According to the processing scheme of Xie et al. [32], four topics are extracted: corporate/industrial, government/social, markets, and economics. As standard in analyzing text documents, the term frequency–inverse document frequency (shortened as TFIDF) [61] feature matrix was computed, using the 2000 most frequent words. The TFIDF matrix represents how important words are in a document. Following Xie et al. [32], a subsample of 10,000 observations was used, due to memory limitations in constructing the graphs.

Dataset	COIL20	COIL100	CMU-PIE	USPS	MNIST	REUTERS-10K
# Observations	1,440	7,200	2,856	9298	70,000	10,000
# Classes	20	100	68	10	10	4
Dimensions	32×32×1	32×32×3	32×32×1	16×16×1	28×28×1	2,000
Type	Image, pixel	Image, pixel	Image, pixel	Image, pixel	Image, pixel	Text, TFIDF
Task	Object Rec.	Object Rec.	Face Rec.	Digit Rec.	Digit Rec.	Topic Rec.

Table 3.1: Description of datasets used in experiments.

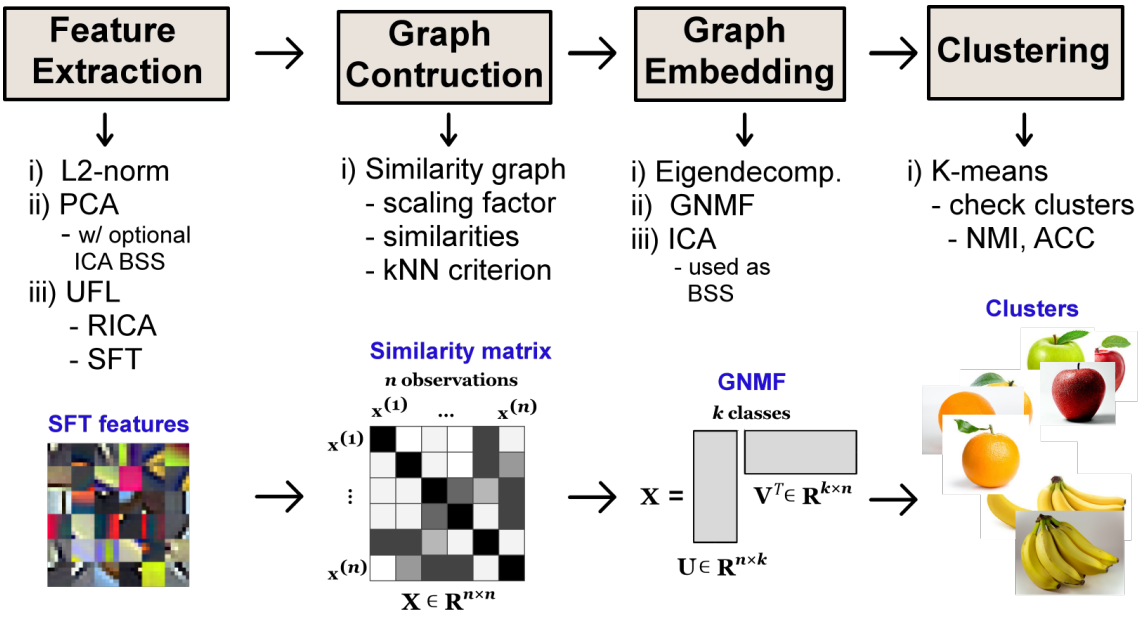

Figure 3.1: Pipeline for processing. Each of the components contains the options available for implementation. The simplest processing pipeline to obtain clustering results consists of a L_2-normalization on the data, followed by K-means clustering. The processing stream with the most components would consist: (1) L_2-normalization followed by UFL using either RICA or SFT; (2) similarity graph construction; (3) GNMF or spectral decomposition followed by ICA blind source separation; and (4) K-means clustering.

3.1.2 Feature Extraction and Unsupervised Feature Learning

For all image datasets, L_2-normalization (each observation input feature vector is transformed to have a unit norm) was performed on the image pixel intensities (each observation input feature vector is transformed to have a unit norm). L_2-normalization has been empirically shown to improve clustering performance [38]. The L_2-normalization was not performed directly on the raw input data of the text document dataset because for text documents, the TFIDF matrix [61] was constructed. Nonetheless, each observation still has unit norm [22]. At this stage, the normalized image and text data can be used as input into the clustering algorithms.

Following normalization, the feature extraction component diverges into two sep-

arate stages. The first stage is performed by principal components analysis (PCA), which is a linear method of feature extraction and data compression [82]. PCA is performed on the normalized datasets by using singular value decomposition (SVD). SVD has been shown to be successful in topic recognition for text documents (Latent Semantic Indexing [83]) and face recognition in images (using eigendecomposition in "eigenfaces" [84]). SVD provides a good baseline for comparison to more complex feature extraction algorithms [22, 85]. For simplicity, the number of principal components extracted is set to the number of classes in each dataset. The goal is to embed the signal that is unique to each class into a vector, as in spectral clustering [17]. The principal components obtained from PCA are then used in K-means clustering as in Cai et al. [22]. In Cai et al. [22], the number of principal components is not limited to the number of classes, as in our processing, but the full dimensionality of the original input is preserved.

After PCA is applied, blind source separation (BSS) can be applied to the extracted components. BSS is the problem of resolving the mixed signal sources, without knowing the nature of the mixture [86, 87]. ICA is mathematical model of BSS, where the mixed signals are separated into their original sources. This is accomplished by determining the mixing matrix that the maximizes non-Gaussianity and minimizes the mutual information [25]. ICA has been shown to successfully separate source signals into biological meaningful signals in electroencephalography (EEG) [88, 89] and functional magnetic resonance imaging (fMRI) data [90, 91]. Furthermore, ICA has been directly applied data without PCA feature extraction to successfully cluster genomic data [92]. By applying ICA to the components obtained by PCA, the goal is provide more salient embedding vectors to the K-means clustering algorithm. In this study, the popular FastICA algorithm is used for the application of ICA [93].

The second stage of our feature extraction, which is separate from the first stage,

uses unsupervised feature learning (UFL). UFL is a general term that is used to describe deep neural network-type feature extraction methods that do not use labels during model training. Examples of UFL are the deep learning methods of Restricted Boltzmann Machine (RBM) and autoencoders neural networks [5]. Many of the deep clustering methods use autoencoders in the core of their models [13, 32, 33, 34, 36, 37].

A disadvantage of deep neural networks such as autoencoders is their parameter complexity during training [12]. To overcome this complexity, RICA and SFT algorithms have been used as substitutes for the feature learning algorithm. These algorithms have greatly reduced the number of parameters needed to train an effective model, thus decreasing training time [44, 45]. Both algorithms have only one hyper-parameter, which is the number of features or hidden nodes that need to be learned for proper feature extraction. A summary of RICA [44] and SFT [45] is given below, where $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{q \times N}$ is the input data (with q as the of # observations and s as the # of features in a random image patch) and $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{K \times N}$ is the weight matrix (with K as the # of learned features and N as the # of input features):

1. **Reconstruction ICA:** RICA relaxes the orthonormal constraint of ICA to learn an over-complete feature representation. Over-complete features are more robust to noise, sparse, and may capture the underlying structure of the data better [94]. The RICA algorithm minimizes $\mathbf{W}$ using the following objective function, $\frac{1}{2}\left\| \mathbf{W}^T\mathbf{W}\mathbf{x} - \mathbf{x} \right\|_2^2 + \lambda \left\| \mathbf{W}\mathbf{x} \right\|$, where λ is a sparsity parameter.

2. **Sparse Filtering:** SFT directly learns the k^{th} feature f_k of the data by L_1-minimization, without ever modeling the data itself. The feature f_k for the i^{th} input $\mathbf{x}_i$ is defined by the soft absolute function $f_k = \sqrt{\left(\mathbf{w}_k^T\mathbf{x}_i\right)^2 + \gamma}$ where $\gamma = 10^{-8}$. The feature f_k is L_2-normalized first by the rows and then the columns. Then L_1-minimization is performed following the normalization steps to ensure equally distributed sparse feature activation.

The second stage of UFL feature extraction is only applied to the image datasets. This is due to the 2D image patch extraction process that is required to implement RICA and SFT [15, 44, 45]. A detailed discussion about how features are extracted and convolved to represent a new feature space can be found in [12, 15, 28].

In summary, there are three types of feature extraction streams for the input data: (1) L_2-normalization of the input features, (2) using the L_2-normalized data, PCA feature extraction with the added option of ICA BSS, and (3) unsupervised feature learning with RICA or SFT using the L_2-normalized features. At this stage, K-means clustering can be directly applied to each of the three types of feature representations, without using any of the graph-based clustering techniques described in the subsequent subsection.

3.1.3 Graph Embedding and Clustering

After the feature representation of the input data is finalized, the graph embedding procedure can be performed on the new representations of the data. This is a two-step process in which the similarity graph is first constructed and then followed by a matrix factorization step. Once the matrix factorization is obtained, clustering can be performed.

We briefly review the process for constructing the similarity graph (for a detailed review see [17]), as used in this study. First, using the Gaussian similarity function, $s(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$, all pairwise similarities among the observations $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$ (where m is the # of observations and n is the # of input features) are computed, controlled by the scaling factor σ. Using the similarities $\mathbf{S} = (s_{ij}) \in \mathbb{R}^{m \times m}$ (where m is the # of observations) and the k-nearest neighbor (kNN) criterion, the similarity graph, otherwise known as the weighted adjacency matrix $\mathbf{W}$, is constructed [17]. Essentially, the kNN criterion simplifies the similarity

matrix to a sparse matrix where each observation is connected to only its k-nearest neighbors and all other entries are set to 0.

To perform clustering using the similarity graph $\mathbf{W}$, it needs to be represented by k eigenvectors or basis vectors by way of matrix factorization [22]. One method to accomplish this is through the standard spectral clustering algorithm [17]. In this study, the two popular spectral clustering algorithms, both of which require the construction of the normalized graph Laplacian, are used. The unnormalized graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = (d_{ij}) \in \mathbb{R}^{m \times m}$ (where m is the # of observations) is the diagonal degree matrix with diagonal elements set to $d_{ii} = \sum_{i=1}^{m} w_{ij}$ and all other entries set to 0. The first version of spectral clustering used, is the method formulated by Ng et al. [18] and called SPC-SYM in our study, normalizes the graph Laplacian according to the following equation, $\mathbf{L}_{sym} = \mathbf{D^{-1/2}LD^{-1/2}}$, where $\mathbf{L}_{sym}$ is a symmetric matrix. The second formulation by Shi and Malik [39], called SPC-RW subsequently, performs the normalization by $\mathbf{L}_{rw} = \mathbf{D^{-1}L}$, where $\mathbf{L}_{rw}$ is a related to a random walk matrix [17]. Both algorithms obtain embedding of the similarity graph by using the k eigenvectors (where k is the number of classes) computed from the eigendecomposition of their normalized graph Laplacians. SPC-SYM and SPC-RW both conclude with K-means clustering on the obtained eigenvectors [17].

The second method used in our study to obtain embedding of the similarity graph $\mathbf{W}$, is the Graph Regularized Non-negative Matrix Factorization (GNMF) algorithm [22]. GNMF extends non-negative matrix factorization (NMF) by including the geometrical information of the input data during the minimization phase of the NMF algorithm [22]. In NMF, the data and the decomposed components are assumed to be non-negative [21], whereas in SVD the components may take on negative values. GNMF incorporates the unnormalized Laplacian $\mathbf{L}$ while solving for the NMF basis vectors of the input data [22]. Thus, GNMF embeds the graph structure into k basis

vectors. To cluster with the GNMF, K-means clustering is performed on the k basis vectors. For comparison purposes and as in Cai et al. [22], we also provide the cluster memberships obtained from NMF that is performed directly on the data feature matrix.

On each of the aforementioned graph embedding algorithms, we apply ICA BSS immediately after the k eigenvectors (from SPC-SYM and SPC-RW) and basis vectors (from GNMF) are obtained. Then, K-means is applied to obtain the cluster memberships. We also apply ICA BSS directly on the symmetric Laplacian $\mathbf{L}_{sym}$ matrix to compare to ICA BSS performed on the pre-embedded vectors obtained from SPC-SYM, SPC-RW, and GNMF. This method is called ICA-SYM, for which we also apply K-means clustering on the k independent components.

Overall, graph embedding is obtained from four different methods, (1) SPC-SYM, (2) SPC-RW, (3) GNMF, and (4) ICA-SYM. Since the end goal of each method is to obtain clustering using K-means, we also refer to the four methods as clustering methods rather than specific embedding methods.

3.1.4 Experimental setup

As addressed in Dizaji et al. [13], successful clustering algorithms need to have a few hyper-parameters for wide applicability in real-world situations. Dizaji et al. [13] in their deep learning-based clustering algorithm, limited their hyper-parameter space to a fixed set of values, i.e., they did not perform a parameter search. Following the same principle, we set our clustering processing pipeline to the following hyper-parameter values:

1. **PCA:** The number of features is set to the number of classes.

2. **ICA BSS:** The number of source signals is set to the number of classes.

3. **UFL:** The number of features is 256 for both RICA and SFT. This was determined by the source code examples from [44, 45].

4. **kNN graph:** The nearest neighbor value k is set to 5, as in Cai et al. [22]. According to [17], the σ scaling factor is set to the mean of value of all of the k^{th} nearest neighbors from the similarity matrix. On the large datasets such as MNIST, ordering the similarity matrix to obtain the 5^{th} nearest neighbor for each observation is very time consuming. To avoid this issue, a smaller set of observations from the data is used to calculate σ. The smaller sample size is estimated using the method of estimating population proportions from a smaller sample [95].

5. **Spectral clustering:** Two types of normalized graph Laplacians were used, the symmetric Laplacian and random walk Laplacian[17].

6. **GNMF and NMF:** The default parameters as provided in the code available from Cai et al.[24] were used.

7. **K-means algorithm:** 200 repetitions of the algorithm was used in order to obtain stable clusters.

In summary, the clustering methods used to achieve the aims of this study are:

1. **Without graph embedding:** K-means, PCA with K-means, and NMF with K-means.

2. **With graph embedding:** SPC-SYM, SPC-RW, ICA-SYM, and GNMF – all of which used K-means as a final step to obtain the clusters.

3.1.5 Performance Evaluation

To evaluate the clusters, the standard protocol and performance metrics provided as given in other studies [13, 32, 96] are used. The number of clusters in all algorithms is provided by the number of unique ground-truth labels in each dataset. The first metric used to assess clustering performance is the normalized mutual information (NMI), which measures the dependence of two labels of the same data [23]. NMI is independent of the label permutations of the clusters and its values range from 0 (completely independent) and 1 (completely identical). The second measure, unsupervised clustering accuracy (ACC), is the common accuracy metric computed for the best matching permutation between clustered labels and ground-truth labels, provided by the Hungarian algorithm [97]. Implementation details about the two metrics can be found in Xu et al. [23].

3.2 Significance Testing of Clusters

3.2.1 Painting Data and Preprocessing

Unlike the previous study, here we used a nonstandard (not a benchmark) dataset to gauge the ability of the novel random distributions to create a null hypothesis for significance testing. A nonstandard dataset was used since the underlying properties, such as the latent features, are not well clearly known. Thus, when relying on the groups output from a clustering algorithm on such a dataset, a reliability analysis should be performed.

Particularly, for this study, we use the images from the digitized painting gallery database from abcgallery.com to extract paintings from the following eight styles: Art Nouveau, Baroque, Expressionism, Impressionism, Realism, Romanticism, Renaissance, and Post-Impressionism (Figure 3.2). For Realism, we refer to the artistic

movement, which started after the French revolution and rejected the exotic themes of the Romantic style [98]. These eight styles were chosen since they provide satisfactory coverage of the stylistic development of paintings in recent history. Each style category consisted of 847 randomly chosen paintings, providing 6,776 paintings in total.

Typically, for unsupervised feature learning and deep neural network learning, images are considered on the scale of 32×32 pixels [12, 57, 99, 100], due to hardware and computation complexity. This scaling is applicable to grayscale or color images with RGB channels. Therefore, to facilitate the use of common unsupervised feature learning and deep learning algorithms, each painting was downsampled to 64×49 pixel dimensions using a bicubic interpolation. The average ratio between the width and height was $\sim$1.3 among the paintings, which was determined from the mean of the histogram of the paintings dimensions. If a painting was taller than it was wide, it was rotated and then scaled accordingly.

Once the data is scaled, the type of feature extraction that will be performed on the images is chosen. For a comparison of the efficacy of the two random distribution algorithms for significance testing of clustering, three types of features were used for analysis in this study. The first case is when no feature extraction is performed, which means the raw pixels of the images are used as-is. In the second case, PCA feature extraction is performed on the raw pixels of the scaled images. Finally, in the third case, UFL feature extraction is performed, similar to the previous study. However, the difference here is that, instead of using RICA or SFT as the feature extraction algorithms, K-means is used [12]. Normally K-means is used to obtained cluster labels, however, if the centroids (the mean of a cluster) are used, then K-means can be used as feature extracting algorithm [12].

For PCA, 100, 300, 500, and 1000 features were extracted to see the effect of

varying the number of features. For UFL with K-means, 500, 1000, 2000, and 3000 features were extracted.

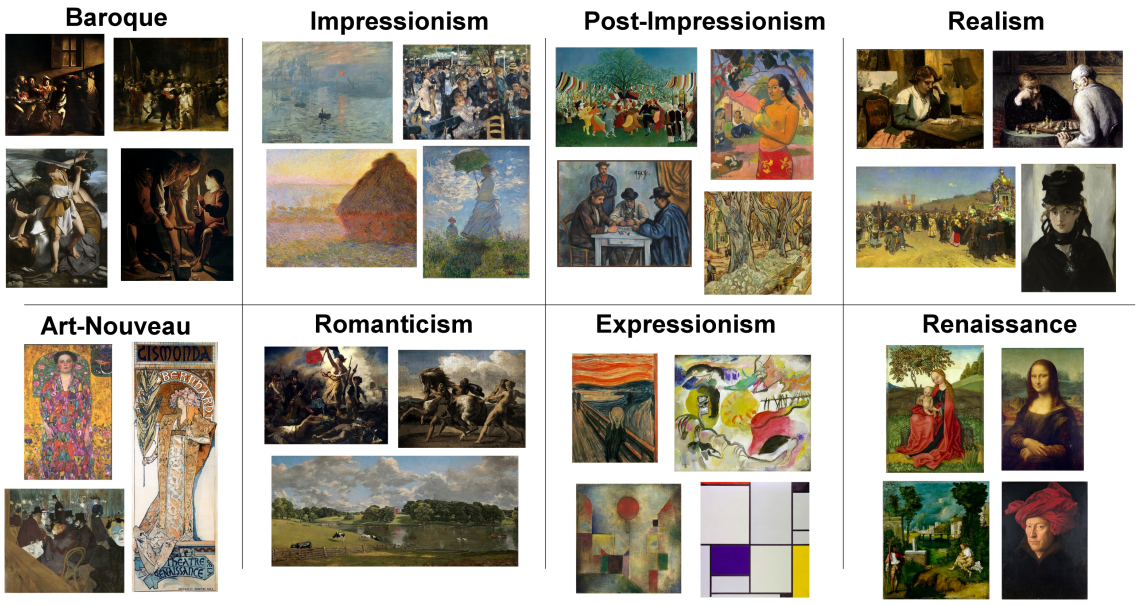

Figure 3.2: Types of painting styles. Representative paintings of each style type employed in this study are displayed.

3.2.2 Clustering

After applying one of the three feature processing steps (raw pixels, PCA, or UFL), painting styles were grouped using spectral clustering. For this study, we use the normalized spectral clustering of Ng, Jordan and Weiss [18]. The style labeling of the final grouping of the paintings provided by the spectral clustering is blind to the actual painting style labels, hence the correspondence between painting style labels and clusters must first be established. To overcome the issue of cluster label assignment [101], the Hungarian method [97], a combinatorial optimization of the assignment problem, was used to determine the lowest cost matching of the spectral clustering labels with the painting style labels.

3.2.3 Performance Evaluation

The outcome of interest is the positive (target) class, therefore it is preferable to use metrics such as precision (the ratio of clustered paintings that actually belong to the true style) and recall (also known as sensitivity; the ratio of paintings in the true style category that have been selected together). The F-score was used as the metric of interest as it is an unbiased combination (harmonic mean) of the precision and recall measures. The best-performing painting style or feature extraction method was defined by having the highest F-score value. Furthermore, to summarize the performance for a feature selection method, we calculated the average F-score of the eight styles to create the macro-averaged F-score (F-mac). Sometimes, to compare our results to other studies where the F-scores are not reported, we use the accuracy measure.

3.2.4 Hypothesis Testing

The basic goal in determining the significance of clusters is to assess whether the clustering algorithm has indeed found groupings of data due to similar patterns and not due to some random structure in the observed data. A significant clustering would reject the null hypothesis that the clustering structure is random – i.e., the discovered clusters are due to chance occurrence [3].

To perform hypothesis testing, first the appropriate test statistic $t \in \mathcal{T}$ (such as clustering accuracy, F-score, or even NMI) and significance level α (typically 0.05) needs to be chosen. Given training data with m observations, $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{m \times n}$, suppose for each $\mathbf{x}_i$ the ground truth (correct) k clusters $C_i \in \{1, 2, \ldots k\}$ are known. The hypothesis test procedure is as follows:

- Repeat r times with $l = 1 \ldots r$:

  - Generate a random distribution $\mathbf{Z} = (z_{ij}) \in \mathbb{R}^{m \times n}$ that preserves some distribution property from the original data $\mathbf{X}$.
  - Perform the clustering algorithm on $\mathbf{Z}$ to obtain $C_i^{(l)} \in \{1, 2, \ldots k\}$
  - Compute the test statistic $t^{(l)} = \mathcal{T}\left((C_1^{(l)}, C_1),\ \ldots,\ (C_m^{(l)}, C_m)\right)$

- Construct the empirical cumulative distribution function (ECDF) that estimates the underlying cumulative distribution of $\mathcal{T}$

$$\hat{P}(X \geq x) = \frac{1}{r} \sum_{l=1}^{r} I(t_l \geq t)$$

  where the indicator function I is defined by

$$I(t_l \geq t) = \begin{cases} 1 & \text{when } t_l \geq t \\ 0 & \text{when } t_l < t. \end{cases}$$

- Determine the test statistic t_0 with clustering on $\mathbf{X}$ and calculate the p-value $\hat{p}_0 = \hat{P}(t_0)$. If $(1 - \hat{p}_0) \leq \alpha$ then the null hypothesis is rejected.

Usually, the number of iterations r is set between 50 – 100. In summary, to calculate $\hat{p}_0$, count the number of t less than t_0 and divide by r. If $(1 - \hat{p}_0) \leq \alpha$, then the null hypothesis that cluster structure is due to chance is rejected. An example ECDF used for hypothesis testing is shown in Figure 3.3. In our experiments r is set to 50, and the test F-score is used as the test statistic $\mathcal{T}$.

3.2.5 Random Distributions

Two novel random data generation schemes were implemented in this study for generating the null hypothesis. In the first method, the reference distribution is generated

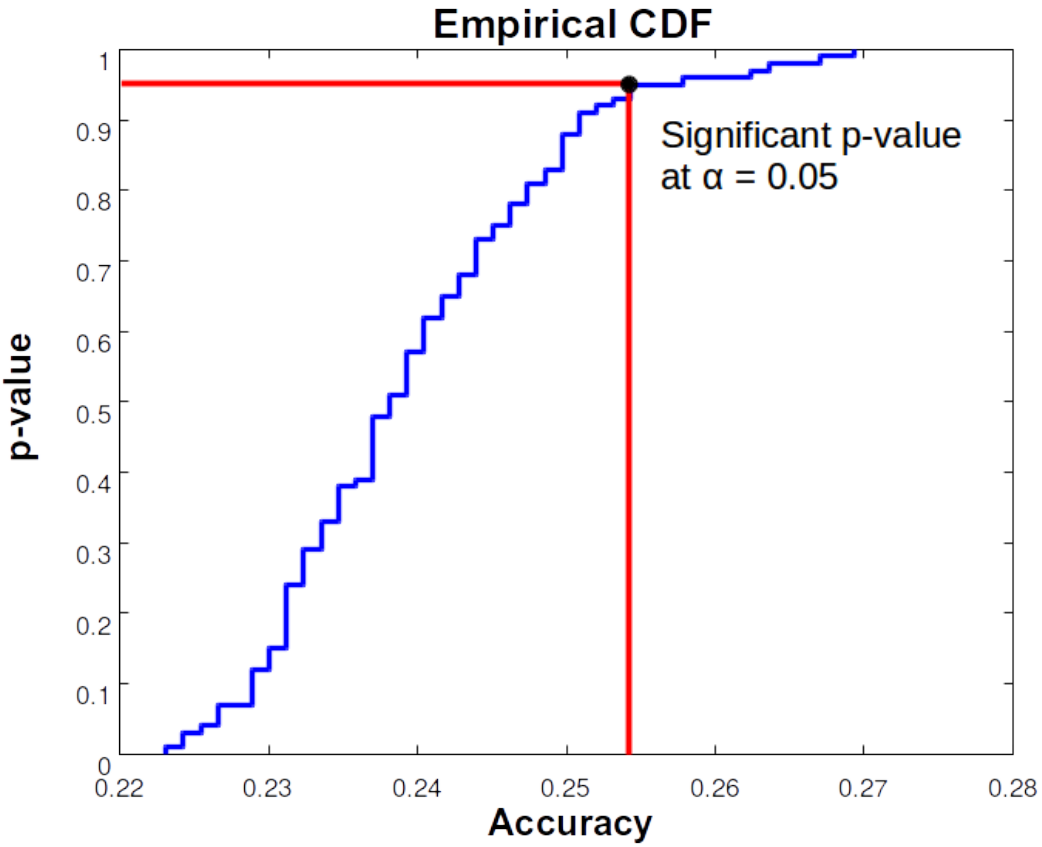

Figure 3.3: Empirical cumulative distribution function. If the p-value of the test statistic, which in this example is clustering accuracy, is higher than 0.254, then the clustering structure can be considered statistically significant.

by projecting a uniform distribution over the principal components of the data, which has the benefit of preserving the shape and local structure of the data distribution [49]. The procedure for generating this distribution is as follows:

- For the data $\mathbf{X} \in \mathbb{R}^{m \times n}$, remove the mean from each column and perform SVD to obtain the principal components $\mathbf{V}$ (right singular vectors) of the data.
- Then project $\mathbf{X}$ onto $\mathbf{V}$ to obtain $\mathbf{X}' = \mathbf{XV}$.
- From $\mathbf{X}' \in \mathbb{R}^{m \times n}$, generate a new $\mathbf{Z}' \in \mathbb{R}^{m \times n}$ where each $\mathbf{z}'_i \in \mathbb{R}^n$ is defined by the continuous uniform distribution on the interval $[\min(\mathbf{x}'_i), \max(\mathbf{x}'_i)]$.

- Project $\mathbf{Z}'$ into the original feature space of $\mathbf{X}$ by performing $\mathbf{Z} = \mathbf{Z}'\mathbf{V}^T$.

This first random distribution was performed prior to inputting data into the spectral clustering algorithm.

In the second method, new data was generated by randomizing the weighted adjacency matrix, which has the effect of preserving the magnitude of similarity among the observations by assigning similarities to random pairs of data. This randomization procedure is as follow:

- From the spectral clustering algorithm, select the upper triangle portion of the weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$, excluding the diagonal entries, to form the upper triangular matrix $\mathbf{U}$.
- Randomize the upper triangle portion of $\mathbf{U}$, first along the rows then by the columns, excluding the diagonal.
- Then form $\mathbf{Z} \in \mathbb{R}^{m \times m}$ by $\mathbf{Z} = \mathbf{U} + \mathbf{U}^T$.

This random distribution method is implemented immediately after the first step of the spectral clustering algorithm – i.e., right after the weighted adjacency matrix is formed.

3.3 LSA Boosted Convolutional Neural Networks

In the following, we describe the implementation of the baseline linear classifier, w2v-based CNN, and LSA-based CNN models. Document datasets for all models used the same preprocessing steps.

3.3.1 Text Document Data

The models were evaluated on three benchmark datasets (IMDB, AGNews, DBPedia). These three datasets were chosen because they were shown have to high classification accuracies using linear classifiers on ngram transformed and TFIDF weighted data [102, 103]. The last two datasets are from [102]. The statistics of the datasets and the size of training and test data are presented in Table 3.2. All datasets contain balanced distribution of classes. The information regarding each of the datasets are described as follows:

1. **IMDB Movie reviews:** This dataset contains 25,000 training and 25,000 test samples, wherein the objective is to predict if a given review has either negative or positive sentiment [104].

2. **AGNews:** Antonio Gulli's news (AGnews) article corpus contains 496,835 articles from more than 2000 different sources [105]. Only the four largest classes from this corpus, where each document is represented by the title and description fields are used. For each class, 30,000 training and 1900 testing samples are used.

3. **DBPedia:** DBpedia is ontology dataset extracted from Wikipedia [106]. It contains 14 non-overlapping classes. Each class contains 40,000 training and 5,000 testing samples. Each document is represented by the title and abstract of each Wikipedia article.

For each review dataset we remove punctuation, hypertext, stopwords, and convert to lowercase.

Review Dataset	Classes	Train	Test
IMDB Movie	2	25,000	25,000
AG's News	4	120,000	7,600
DBPedia	14	560,000	70,000

Table 3.2: Summary of the text document datasets.

3.3.2 Linear Classifier with ngram and TFIDF

To construct the TFIDF weighted {1,2,3}-ngram text representation, first a bag-of-words (BOW) transformation is performed. The BOW representation is constructed by using the frequency count of each word (unigram), bigram, and trigram. Then for the TFIDF weighting [107], the frequency counts (term-frequency) is divided by the inverse document frequency (IDF). The IDF is the log of the total number of samples divided by the number of samples in the data. The {1,2,3}-ngram size is limited to the top 30K most frequent terms, similar to [64]. TFIDF representation, was shown to generally perform better than the BOW representation [102]. Finally, the TFIDF weighted {1,2,3}-ngram representation was used with multinomial logistic regression with the L_2 regularization parameter $\lambda = 1$ from the LIBLINEAR package [60].

3.3.3 CNN Model

CNNs are feed-forward neural networks, where the features generated by the neural networks layers are convolved with each other until a final classification is applied. Typically, the features are extracted from small 2D patches of an image. For instance, for a 32×32 grayscale image, 5×5 regions are extracted from which the features are learned. However, for text documents the 2D patches are simplified down to 1D patches.

The core of the LSA-based CNN and w2v-based CNN [65] is a 1D-CNN model [67]. A sentence can be defined by a sequence of n concatenated k-dimensional word

vectors, $\mathbf{x}_i \in \mathbb{R}^k$,

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus ... \oplus \mathbf{x}_n.$$

The convolution of words, selected by a window of length h, with a filter $\mathbf{w} \in \mathbb{R}^{h \times k}$ produces the feature mapping c_i given by the formula below,

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b),$$

where $b \in \mathbb{R}$ is the bias term and f is a nonlinear function such as the sigmoid function. Applying the filter c_i to all sentences selected by the window h provides the feature mapping $\mathbf{c} \in \mathbb{R}^{n-h+1}$, where

$$\boldsymbol{c} = [c_1, c_2, ..., c_{n-h+1}].$$

Then max-over-time pooling is applied to all the features, where the maximum activation value for the filter is obtained. This convolutional process is applied to all filters for a given window of words. The last step in the CNN model is a softmax layer, which is used to obtain the output class of the text document. A schematic workflow of a general word vector based CNN is shown in Figure 3.4. In the subsections below, the details regarding the number of filters, window sizes, and word vectors is provided.

3.3.4 Embedding Layer

In this study, three types of word vectors are used to initialize the embedding matrix. First, as a baseline, random word vectors from $\mathbf{W}_{rand} \in \mathbb{R}^{d \times v}$ are used, which are initialized by the uniform distribution within the range [-0.05, 0.05]. Second, the LSA word vectors are formed by using SVD on a unigram transformed and TFIDF

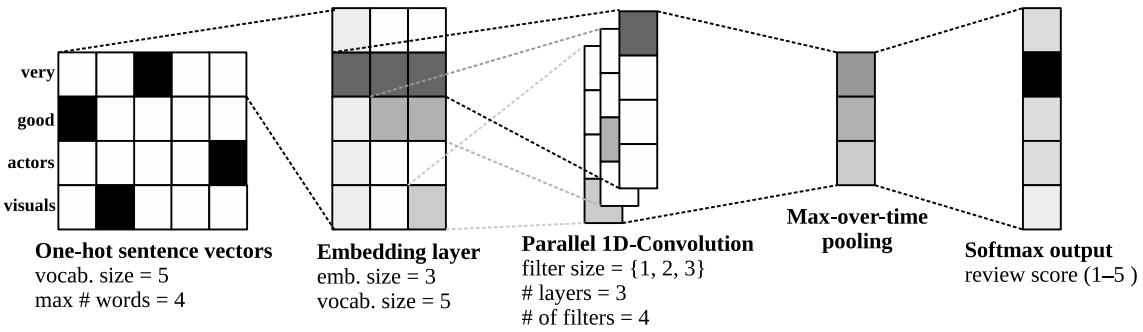

Figure 3.4: Schematic of the proposed model. In this example, a single sentence of a preprocessed movie review is input into a model where there is a vocabulary size of 5 in the dictionary. The word vector embedding layer reduces the dimensionality of the dictionary to 3. Then 3 parallel 1D-convolutions are applied, using 3 different filter sizes. on which L_2 reguralization is performed. Max-over-time pooling is applied, which takes the single best feature per feature map. Finally, the review is classified with a score from 1 to 5.

weighted data. Specifically, using only the top d singular values, SVD is performed on $\mathbf{X} \in \mathbb{R}^{v \times n}$, where v is the size of the vocabulary and n is the number of documents, giving

$$\mathbf{X}_d = \mathbf{U}_d \mathbf{S}_d \mathbf{V}_{\mathrm{d}}^{\mathrm{T}}.$$

Then the LSA word vectors are defined as the rows of

$$\mathbf{W}_{LSA} = \mathbf{U}_d \mathbf{S}_d.$$

Also, according to Levy et al. [77], the word vectors are row normalized, which has been shown to improve representative accuracy.

Finally, the word2vec word vectors pretrained on 100 billion words of Google News are used to form $\boldsymbol{W}_{w2v} \in \mathbb{R}^{d \times v}$. The word2vec model is a two-layer unsupervised neural network that produces vector representation of words, similar to LSA. In the model, the objective function maximizes the log probability of a context word (w_O), given its input words (w_I). The output is a n-dimensional vector for each word in the

vocabulary that represents the weights of the hidden layers. Words that are similar in context to each other in the training corpus are located close to each other in the vector space representation. In Figure 3.5, two different w2v model architectures are shown, the Continuous Bag of Words (CBOW) and Skip-gram models. The CBOW architecture, predicts a word based on the surrounding context words. The Skip-gram architecture uses the current word to predict the surrounding words in a fixed-size window. For infrequent words, the Skip-gram architecture works better, whereas the CBOW model works faster than the Skip-gram model.

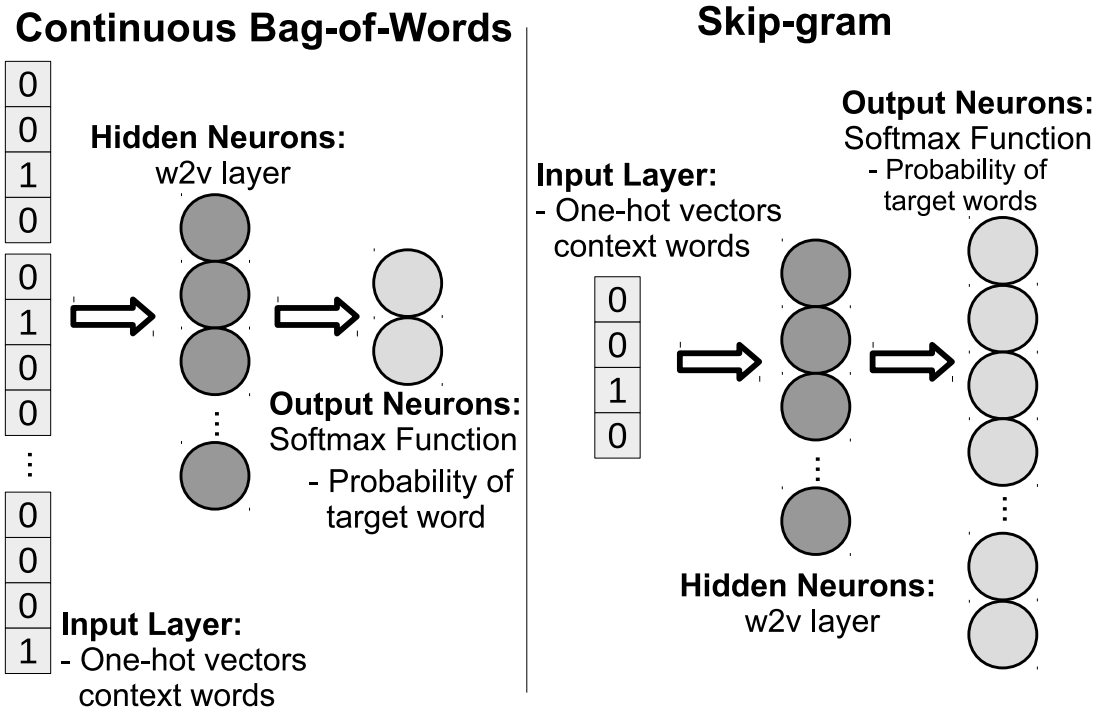

Figure 3.5: Training of word2vec word vectors. The two models of word2vec architectures are shown here.

The $\mathbf{W}_{rand}$ initializing weights are used to test the efficacy of the vectors $\mathbf{W}_{LSA}$ and $\mathbf{W}_{w2v}$ initializing weights. For the subsequent CNN models presented below, the following name convention is used, "{ngram range}-$\mathbf{W}_E$ type-regularization-CNN", where "{ngram range}" indicates the window size used on the word vectors, "$\mathbf{W}_E$"

takes on either $\mathbf{W}_{rand}$, $\mathbf{W}_{LSA}$ or $\mathbf{W}_{w2v}$, and "regularization" is either L_2 weight decay (wt) or dropout (dp) based. The "{ngram range}" and "regularization" are dependent on the model architectures defined below.

3.3.5 LSA-based CNN Model

To successfully apply LSA in a CNN model, two different architectures were developed to take advantage of the LSA word vectors. Specifically, networks with ngram filter sizes = {1,2,3} and {1,2,3,4} are implemented. Each ngram filter size pertains to a separate and parallel convolutional layer in the model. These layers are finally concatenated with each other in the max-over-time pooling layer, prior to the softmax classification layer. The following are the parameters used for both network architectures:

1. **Filters:** The number of filters is set to 128 for each of the ngram filters, which is a common size in convolutional network models [108].

2. **Emedding:** The embedding dimension size was set to 300 to match the word2vector dimensionality [73].

3. **Regularization:** The regularization was applied to each of the weights in convolutional layers using an L_2 parameter weight decay of of $10e^{-5}$ [64, 109].

4. **Activation:** Rectified linear units (ReLu) were used as an activation function, as it has been shown to be sufficient in other studies [102] and also to enable comparison to the w2v-based CNN model specified in Kim [65].

A summary of the architecture is shown for the {1,2,3,4}-LSA-wt-CNN model in Figure 3.6. The {1,2,3}-LSA-wt-CNN model follows the same setup except for the absence of the quadgram filters.

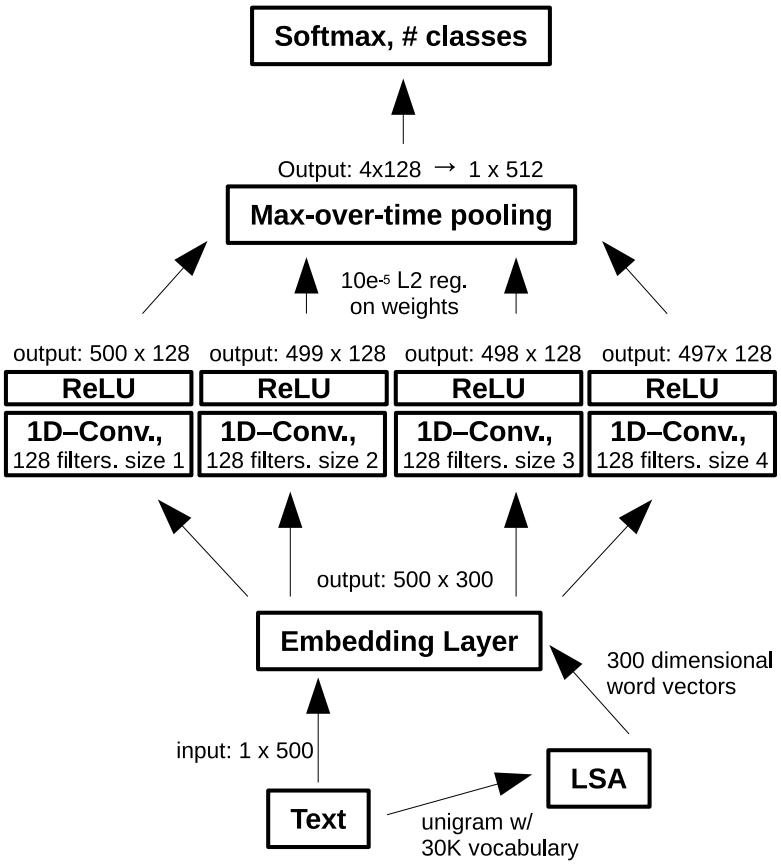

Figure 3.6: Sample CNN model architecture. The above setup shows the architecture and parameters for the top performing LSA-based CNN model.

3.3.6 W2V-based CNN Model

The 1D-CNN architecture from Kim [65] was also used to compare against the LSA-based CNN architecture. The w2v-based CNN architecture uses larger filter sizes since the word2vectors are designed to capture long distance relationships among words [73]. The w2v-based CNN model was shown to be an effective classifier compared to linear and other state-of-the-art classifiers such as RNN's and LSTM's [65]. For the w2v-based CNN architecture, the following parameters as given in [65] were used: filter sizes = {3,4,5}; embedding dimension size = 300; and dropout [110] used for regularization. Only the number of filters for each filter size was increased from 100

to 128, to enable fair comparison to the CNN model architecture developed for LSA word vectors.

3.3.7 Experimental Setup

For all three datasets, the vocabulary size of the training corpus was limited to 30K words for the linear and CNN based models. To test the effectiveness of the LSA and w2v specific CNN architectures, the $\mathbf{W}_{rand}$, $\mathbf{W}_{LSA}$ or $\mathbf{W}_{w2v}$ word vectors were input into both types of architectures. The Adadelta optimizer [111] was used to trained the models for 10 epochs as it was shown to reach convergence quickly [65]. For all datasets, we report the accuracy of document classification on the testing samples.

Chapter 4

Results

The results based on the methods described in the previous section are provided here.

4.1 Clustering Pipeline using ICA and UFL

Performing ICA blind source separation, after the initial matrix factorization step, provided the maximum clustering performance (Table 4.1) in four out of six datasets (COIL100, CMU-PIE, MNIST, and REUTERS-10K). Applying UFL as an initial processing component helped to provide the maximum performance in three out of six datasets (USPS, COIL20, and COIL100). Although on the COIL100 dataset, where both ICA BSS and UFL increased performance, no interaction effect between the two processing components has been shown (see the multivariate analysis of variance provided below). Furthermore, across all datasets the maximum performing clustering algorithms were GNMF (COIL100 and USPS), SPC-SYM, (COIL20 and MNIST) and PCA (CMU-PIE and REUTERS-10K) as shown in Table 4.1. This demonstrates that in four out of six datasets, graph-based clustering provides the maximum performance. None of the datasets exhibited maximum performance without using ICA BSS and/or UFL, which demonstrates that at least one type of the processing components is

necessary to achieve the best clustering.

	Performance of applied processing (NMI, ACC)					
Method	**L_2**	**L_2, ICA**	**L_2, RICA**	**L_2, RICA, ICA**	**L_2, SFT**	**L_2, SFT, ICA**
COIL20	0.918, 0.857^b	0.920, 0.856^b	0.929, 0.894^b	0.914, 0.885^b	**0.965, 0.93**^{b}	0.946, 0.912^a
COIL100	0.914, 0.774^b	0.914, 0.784^a	0.943, 0.813^b	**0.962, 0.897**^{a}	0.932, 0.765^b	0.954, 0.849^a
CMU-PIE	0.941, 0.850^b	**0.986, 0.937**^{c}	0.816, 0.716^b	0.848, 0.721^d	0.844, 0.759^b	0.866, 0.774^b
USPS	0.845, 0.828^a	0.854, 0.810^a	**0.868, 0.926**^{a}	0.850, 0.794^e	0.852, 0.817^b	0.853, 0.813^b
MNIST	0.774, 0.787^a	**0.824, 0.882**^{b}	0.790, 0.828^b	0.787, 0.822^b	0.790, 0.824^b	0.794, 0.853^a
REUTERS-10K	0.446, 0.656^c	**0.460, 0.714**^{c}				

Table 4.1: Comparison of maximum performance across processing components. Bold font within a row indicates the maximum performance obtained for a dataset. The clustering algorithm providing the maximum performance for given processing component and dataset is indicated by the following symbols: ^{a}GNMF; ^{b}SPC-SYM; ^{c}PCA; ^{d}SPC-RW; ^{e}ICA-SYM.

To test the statistical significance and any interaction of the processing components, a three-way multivariate analysis of variance (MANOVA) was performed. The MANOVA revealed that there is no significant interaction among ICA BSS, unsupervised feature learning, and clustering methods. This demonstrates that the performance effects of the three components are independent of each other (as shown by the NMI results in Figure 4.1A and ACC results in Figure 4.1B). The MANOVA revealed that there was a statistically significant difference in applying ICA BSS after the matrix factorizations (Pillais' Trace = 0.058, $F(1{,}166) = 4.05$, $p < 0.05$). The post-hoc pairwise t-tests, show that ICA BSS provides higher mean performance across all datasets (NMI: $\mu_{ICA-ON} = 0.783 \pm 0.174$ and $\mu_{ICA-OFF} = 0.716 \pm 0.20$, $p < 0.05$; ACC: $\mu_{ICA-ON} = 0.745 \pm 0.133$ and $\mu_{ICA-OFF} = 0.653 \pm 0.165$, $p < 0.001$). The MANOVA also showed that there was significant difference in clustering method (Pillais' Trace = 0.54, $F(6, 166) = 8.14$, $p < 0.001$). Specifically, GNMF, SPC-RW, SPC-SYM, and ICA-SYM had the higher mean performance compared to all other clustering methods (all comparisons $p < 0.001$), but there was no statistical difference among these four methods. Although, the UFL processing component was not statistically significant at the $p = 0.05$ level, it was significant at the less stringent $p = 0.1$ level (Pillais' Trace = 0.063, $F(2, 166) = 2.14$, $p < 0.1$). Post-hoc pairwise

t-tests at the $p < 0.05$ level show that the NMI performance is higher with RICA ($\mu_{UFL-RICA} = 0.774 \pm 0.137$) and SFT ($\mu_{UFL-SFT} = 0.786 \pm 0.144$) feature learning in comparison to the absence of feature learning ($\mu_{UFL-OFF} = 0.706 \pm 0.238$). All post-hoc pairwise t-tests were corrected using the False Discovery Rate.

The key results of the processing components on each of the six datasets are summarized below:

1. **COIL20:** Feature learning using SFT with SPC-SYM provided the highest clustering performance (NMI = 0.965, ACC = 0.93), which is better than the baseline SPC-SYM clustering performance (NMI = 0.918, ACC = 0.857). With SFT and ICA BSS, GNMF also showed an improvement in both performance measures (NMI = 0.946, ACC = 0.912) compared to its baseline GNMF clustering performance (NMI = 0.913, ACC = 0.844).

2. **COIL100:** The best clustering performance (NMI = 0.962, ACC = 0.897) was obtained with GNMF clustering using RICA feature learning and ICA BSS. Without feature learning and ICA BSS, GNMF performance is (NMI = 0.9, ACC = 0.713). The addition of the RICA feature learning increases the performance to (NMI = 0.926, ACC = 0.74). The application of ICA BSS to GNMF using only L_2-normalized features increases the clustering performance to (NMI = 0.914, ACC = 0.784).

3. **CMU-PIE:** Clustering performance obtained with SPC-SYM and ICA BSS is (NMI = 0.947, ACC = 0.879), which is an improvement over the baseline SPC-SYM implementation (NMI = 0.941, ACC = 0.85). However, when the simpler method of clustering on the PCA reduced representation of the pixels, followed by ICA BSS, K-means clustering provides the top performance (NMI = 0.985, ACC = 0.937).Without any ICA BSS, clustering in the PCA representation

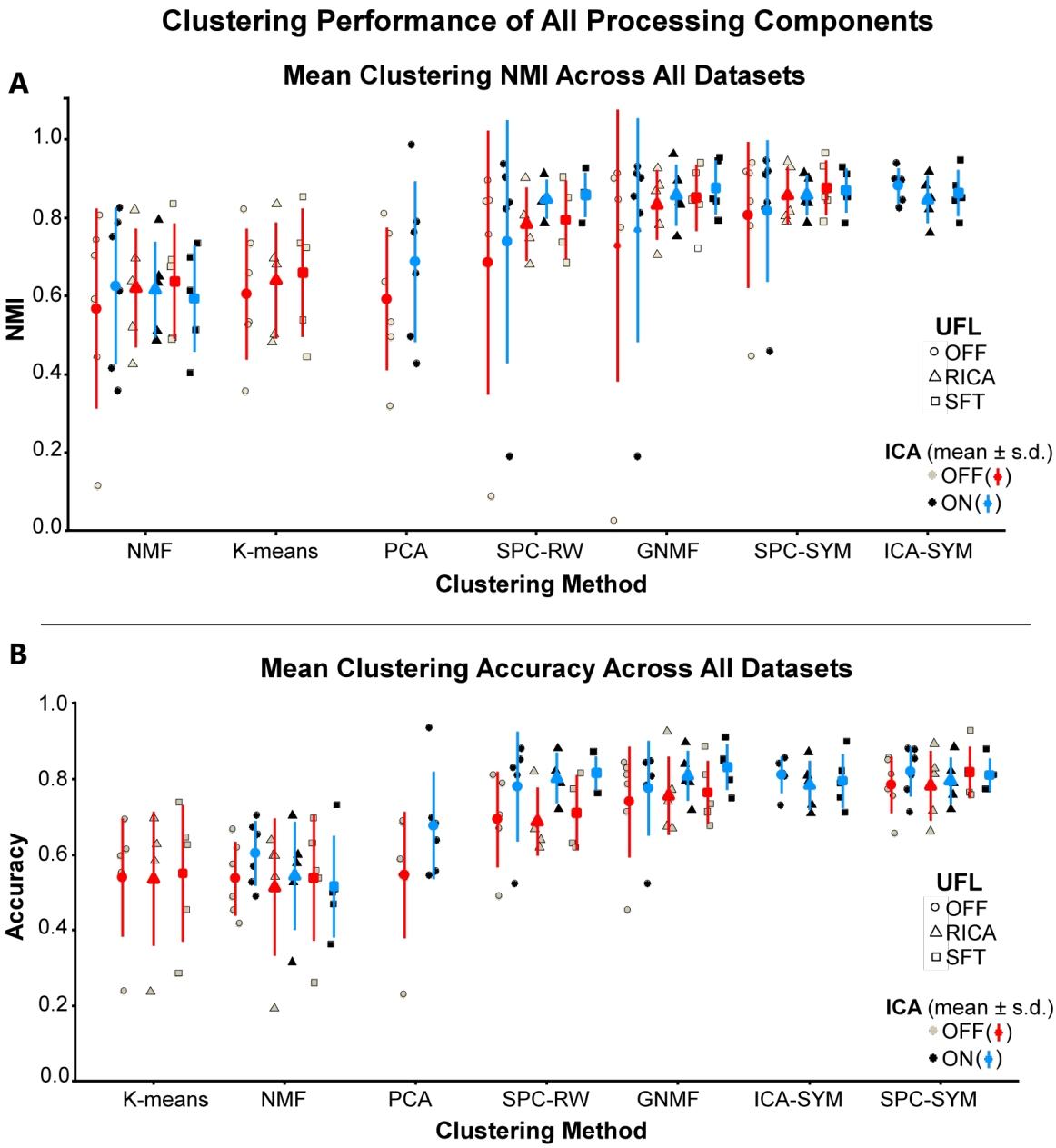

Figure 4.1: Mean clustering performance. The mean performances (with ±s.d. error bars) are place over the individual cluster performances. The clustering methods are organized in ascending order for each performance measure. **(A)** Mean NMI across all datasets. **(B)** Mean ACC across all datasets. ICA BSS tends to have a higher mean NMI and ACC compared to the absence of ICA BSS. Also, the graph-based clustering methods (GNMF, SPC-RW, SPC-SYM, and ICA-SYM) have a higher mean performance. Although ICA-SYM has a high mean NMI, it never provided the maximum one-off performance, such as in SPC-SYM or GNMF. Under the UFL legend, OFF indicates that only L_2-normalized features were used for clustering.

provides only (NMI = 0.534, ACC = 0.231) performance. The best clustering performance using feature learning was obtained with SFT combined with ICA BSS using SPC-SYM (NMI = 0.866, ACC = 0.774), which is still lower than baseline SPC-SYM performance.

4. **USPS:** The best clustering performance was obtained with RICA feature learning and GNMF (NMI = 0.868, ACC = 0.926). The baseline GNMF performance was (NMI = 0.854, ACC = 0.828), which was higher than the baseline SPC-SYM performance (NMI = 0.842, ACC = 0.814). When the ICA BSS was applied to GNMF, the performance of the NMI increased, but the accuracy decreased (NMI = 0.854, ACC = 0.810).

5. **MNIST:** The best clustering performance was obtained with SPC-SYM / SPC-RW using ICA BSS (NMI = 0.824, ACC = 0.882). The performance of the baseline SPC-SYM and SPC-RW were (NMI = 0.779, ACC = 0.756) and (NMI = 0.757, ACC = 0.67), respectively. When RICA feature learning was applied to SPC-SYM, the performance increased to (NMI = 0.79, ACC = 0.828). When ICA BSS was combined with RICA feature learning for SPC-SYM, the performance decreased slightly (NMI = 0.787, ACC = 0.822). With GNMF clustering, the baseline performance (NMI = 0.774, ACC = 0.787) improved when ICA source was applied (NMI = 0.813, ACC = 0.845). Improvement to the GNMF baseline performance was also achieved when SFT feature learning was combined with ICA BSS (NMI = 0.794, ACC = 0.853).

6. **REUTERS-10K:** When PCA dimension reduction followed by ICA BSS is applied directly applied to the TFIDF matrix, K-means clustering provides the top clustering performance (NMI = 0.46, ACC = 0.714). Without ICA BSS, PCA dimension reduction provides an accuracy of (NMI = 0.446, ACC = 0.656).

The next best clustering performance was provided by NMF without (NMI = 0.318, ACC = 0.546) and with ICA BSS (NMI = 0.428, ACC = 0.638).

The results from this study are compared to the deep learning-based clustering methods in the following studies: Deep Embedding Network (DEN) [34], Discriminatively Boosted Clustering (DBC) [112], Infinite Ensemble Clustering (IEC) [113], autoencoder-based Clustering (AEC) [33], Deep Embedded Clustering (DEC) [32], Deep Clustering Network (DCN) [37], Deep Convolutional Embedded Clustering (DCEC) [114], Deep Embedded Regularized Clustering (DEPICT) [13], Variational Deep Embedding (VaDE) [36], autoencoder with K-means clustering (AE+K-means) [32], Information Maximizing Self-Augmented Training (IMSAT) [115], NMF with Deep learning model (NMF-D) [116], Joint Unsupervised Learning (JULE) [38].

The results are also compared to the state-of-the-art clustering methods that are not based on deep learning methods. These methods are: Graph Degree Linkage-based Agglomerative Clustering (AC-GDL) [117] and Agglomerative Clustering via Path Integral (AC-PIC) [118], Spectral Embedded Clustering (SEC) [119] and Local Discriminant Models and Global Integration (LDMGI) [96]. The comparisons are summarized in Table 4.2 and given below.

The implementation of ICA BSS and/or unsupervised feature learning with graph-based clustering algorithms outperformed all other state-of-the-art non-deep learning clustering methods (Table 4.2). With respect to deep learning-based clustering algorithms, our methodology performed second best after the JULE algorithms (JULE-SF and JULE-RC) in three out of six datasets (COIL20, COIL100, and CMU-PIE).

The key results of the processing components on each of the six datasets are summarized below:

1. **COIL20** Our SPC-SYM with SFT feature learning was 3.5 percentage points (p.p.) less than JULE (both JULE-SF and JULE-RC had perfect NMI) in NMI

	Dataset					
Method	**COIL20**	**COIL100**	**CMU-PIE**	**USPS**	**MNIST**	**REUTERS-10K**
Baseline						
K-means	0.735, 0.597	0.822, 0.615	0.532, 0.239	0.659, 0.694	0.527, 0.553	0.356, 0.541
Deep Learning						
AE+K-means					–, 0.818	–, 0.666
NMF-D	0.692, –	0.719, –	0.920, 810	0.287, 0.382	0.152, 0.75	
TSC-D	0.928, 0.899				0.651, 0.692	
DEN	0.870, 0.725					
DBC	0.895, 0.793	0.905, 0.775		0.724, 0.743	0.917, 0.964	
IEC		0.787, 0.546		0.641, 0.767	0.542, 0.609	
AEC				0.651, 0.715	0.669, 0.760	
DCN					0.810, 0.830	
DEC			0.924, 0.801	0.586, 0.619	–, 0.818	–, 0.722
DCEC				0.826, 0.790	0.885, 0.890	
DEPICT			0.974, 0.883	0.927, 0.964	0.917, 0.965	
JULE-SF	1.000,–	0.978, –	0.984, 0.980	0.858, 0.922	0.906, 0.959	
JULE-RC	1.000, –	0.985, –	1.000, 1.000	0.913, 0.950	0.913, 0.964	
VaDE					–, 0.945	–, 0.798
IMSAT					–, 0.984	–, 0.719
SpectralNet					0.924, 0.971	
Non-Deep Learning						
AC-GDL	0.865, –	0.797, –	0.934, 0.842	0.824, 0.867	0.017, 0.113	
AC-PIC	0.855, –	0.840, –	0.902, 0.797	0.840, 0.855	0.017, 0.015	
SEC				0.511, 0.544	0.779, 0.804	
LDMGI				0.563, 0.580	0.802, 0.842	
Ours*	0.965, 0.93	0.962, 0.897	0.986, 0.937	0.868, 0.926	0.824, 0.882	0.460, 0.714

Table 4.2: Comparison of clustering performance across different datasets and clustering techniques. When available, both NMI and ACC are presented in each cell, where the first value is the NMI. If the cell is blank then the clustering method was not used on the dataset. *The maximum clustering performance obtained by the processing components proposed in this study is provided for each dataset.

performance. Overall, SPC-SYM with SFT ranked 2^{nd} (out of 6) compared to 5 other deep clustering methods.

2. **COIL100:** GNMF with RICA feat learning and ICA BSS performed 1.6 p.p. and 2.3 p.p. less than JULE-SF and JULE-RC, respectively, in NMI. GNMF with RICA and ICA BSS ranked 2^{nd} (out of 5) compared to 4 other deep clustering methods.

3. **CMU-PIE:** The simple combination of PCA with ICA BSS performed almost on par with the much more complex JULE algorithm, which uses a combination of recurrent and convolutional neural networks. Specifically, before any model fine-tuning is applied to the JULE algorithm (JULE-SF), our PCA and ICA

combination was 0.2 p.p. higher in NMI performance. However, this advantage is lost once the JULE algorithm is fine-tuned (JULE-RC). PCA with ICA BSS ranked 2^{nd} (out of 5) compared to 4 other deep cluster methods.

4. **USPS** Using RICA with GNMF performed third best after the JULE-RC and DEPICT algorithms, outperforming seven deep learning clustering methods. This new combination even outperformed JULE-SF by 1 p.p. in NMI and 0.4 p.p. in ACC. Overall, RICA with GNMF ranked 3^{rd} (out of 9) compared to 8 other deep clustering methods.

5. **MNIST:** Our clustering algorithm performed better than seven other deep learning clustering algorithms by simply using ICA BSS after eigenvector decomposition of the normalized Laplacian used in SPC-SYM. However, it ranked 8^{th} (out of 15) compared to 14 other deep clustering methods. Nonetheless, it performed better than the popular algorithms of DEC by 0.8 p.p. in NMI and 3.8 p.p. in ACC and DCN by 1.4 p.p. in NMI and 5.2 p.p in ACC.

6. **REUTERS-10K:** Our implementation of clustering in the PCA space with ICA BSS, performed better than AE+K-means by 4.8 p.p. in NMI, however it did not exceed any other deep learning clustering algorithms, ranking 4^{th} (out of 5) in comparison to 4 other deep learning methods. Nonetheless, our PCA and ICA combination performed on par with the DEC and IMSAT algorithms, which were 0.5 and 0.8 p.p. in NMI, respectively, better than our clustering performance.

The computational environment was setup on a computer with 20 GB of memory and AMD - FX-6300 3.5 GHz 6-Core CPU. For the largest dataset, (MNIST), the running time for our best performing algorithm (SPC-SYM with ICA BSS) is 20 minutes. With feature learning, such as SFT, in addition to graph-based clustering,

the running increases to 5 hours for GNMF and 4.9 hours for SPC-SYM. For the USPS dataset (the second largest dataset), our best result using RICA feature learning with GNMF clustering took 18 minutes to run. Adding a separate ICA BSS (∼0.5 seconds), does not noticeably increase the running time because clustering time is reduced by one second on the ICA components.

4.2 Significance Testing of Clusters

The effect of PCA, UFL, and absence of feature extraction on the clustering performance of paintings based on style is displayed in Table 4.3. Following feature extraction with UFL and PCA, the best overall spectral clustering performance provided by feature extraction method was found using the UFL method with 500 features (F-mac = 0.212). The best PCA based clustering performance (F-mac = 0.185) was demonstrated with only 100 principal components. Clustering only with the raw image painting pixels provided an F-mac of 0.197, which was better than using PCA for feature extraction. This indicates that the best UFL features performed 14.37% and 7.57% better than PCA features and raw pixels, respectively, for painting style clustering.

In Figure 4.2, the best performing clustering of the UFL and PCA feature extraction method, along with the raw pixel features, are visualized as a scatterplot, where it is seen that the UFL has more cohesive and disjointed groupings of paintings compared to the PCA based groups, which have overlapping and diffuse clusters. Also, the groups based on the UFL features are organized in a way which suggests some semantic relationships among different art movements, whereas the raw pixel features have a linear organization, which is unlikely due to some stylistic overlap of art movements. For instance, we see in the bottom right of Figure 4.2A that Romanticism

	Performance of Feature Extraction Methods (F-score/Precision/Recall)								
Style	**Raw Pixels**	**PCA 100**	**PCA 300**	**PCA 500**	**PCA 1000**	**UFL 500**	**UFL 1000**	**UFL 2000**	**UFL 3000**
Baroq.	0.413	0.297	0.047	0.032	0.018	0.362	0.352	0.36	0.353
	0.389	0.237	0.22	0.195	0.211	0.306	0.304	0.304	0.308
	0.44	0.397	0.026	0.018	0.009	0.444	0.418	0.44	0.414
Imprs.	0.165	0.217	0.005	0.038	0.106	0.199	0.198	0.197	0.188
	0.176	0.234	0.133	0.182	0.286	0.257	0.243	0.251	0.233
	0.155	0.203	0.002	0.021	0.065	0.162	0.168	0.162	0.158
Post-	0.21	0.184	0.226	0.247	0.281	0.199	0.106	0.104	0.108
Imprs.	0.165	0.172	0.165	0.199	0.244	0.157	0.164	0.151	0.168
	0.288	0.197	0.36	0.323	0.332	0.273	0.078	0.079	0.079
Real.	*0.096*	0.161	0.018	0.002	0	*0.128*	0.133	0.125	0.131
	0.202	0.218	0.17	0.077	0	0.144	0.141	0.137	0.141
	0.063	0.128	0.009	0.001	0	0.116	0.126	0.115	0.123
Art	0.24	*0.135*	0.206	0.15	0.16	0.229	0.22	0.223	0.219
Nouv.	0.192	0.508	0.18	0.249	0.306	0.194	0.201	0.197	0.2
	0.32	0.078	0.24	0.107	0.109	0.28	0.243	0.259	0.242
Romant.	*0.107*	*0.031*	0.036	0.245	0.009	*0.157*	0.175	0.163	0.168
	0.177	0.226	0.121	0.157	0.174	0.325	0.323	0.333	0.321
	0.077	0.017	0.021	0.557	0.005	0.104	0.12	0.107	0.113
Expres.	0.169	0.234	0.145	0.174	0.179	0.221	0.274	0.282	0.284
	0.256	0.214	0.198	0.19	0.154	0.328	0.217	0.229	0.227
	0.126	0.256	0.115	0.161	0.215	0.166	0.37	0.367	0.378
Renais.	0.175	0.222	0.246	0.2	0.256	0.197	0.203	0.21	0.208
	0.153	0.159	0.158	0.175	0.156	0.186	0.181	0.186	0.178
	0.204	0.367	0.558	0.231	0.714	0.208	0.231	0.242	0.25
F-mac[†]	0.197	0.185	0.116	0.136	0.126	**0.212**	0.208	0.208	0.207
	0.214	0.246	0.168	0.178	0.191	0.237	0.222	0.223	0.222
	0.209	0.205	0.166	0.178	0.181	0.219	0.219	0.221	0.22

Table 4.3: Painting style clustering. [†]F-mac is the average of the F-score; bold indicates the best F-mac score across the different feature extraction methods. Not significant F-score values for the clusters based on raw pixels, PCA 100, and UFL 500 features are indicated by italics.

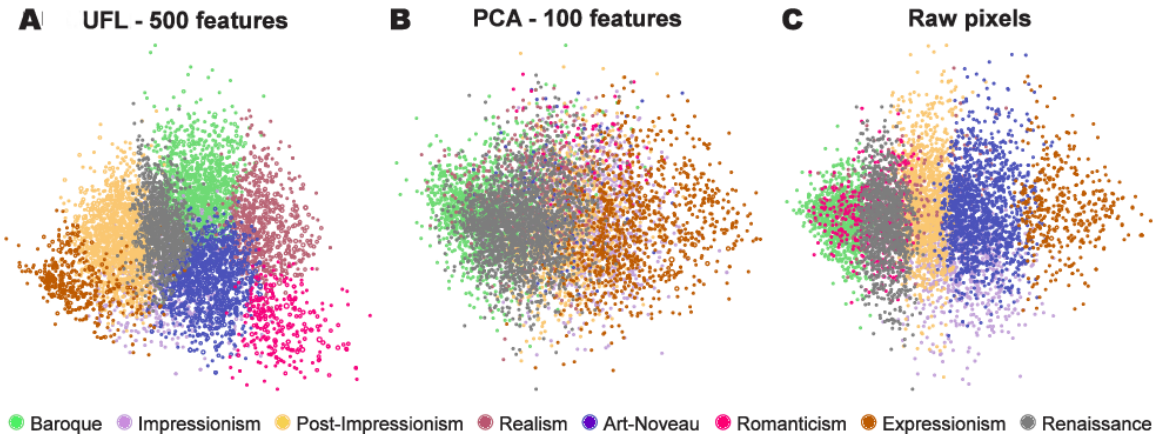

Figure 4.2: Scatterplot of painting styles clusters. The groupings of the painting styles obtained from spectral clustering are plotted as a scatterplot by projecting onto its first two multidimensional scaled components (MDS) of the data. MDS is essentially a visaulization of the similarity of observed data in a dataset. The size of each node, which represents a single painting in the dataset, is determined by its degree (i.e., the number of connections each painting has to other paintings). As a result, the larger a node, the more connections that painting has to other paintings. Also, the distance between nodes (paintings) demonstrates the similarity between paintings. Similar or related paintings and styles are in close proximity, while disparate paintings and styles are distant.

is a neighbor of Realism and Art Nouveau, which is not surprising given that the main Romanticism artistic movement occurred before the Realism and Art Nouveau movements [120]. Also, the Realism stylistic movement developed in response to the Romantic ideals [98]. At the top of Figure 4.2A, we see the Baroque style is surrounded by the Renaissance, Art Nouveau, and Realism styles. Generally, it is seen that the Baroque is highly ornate and eccentric in comparison to the Renaissance style [121], which preceded it. However, the Art Nouveau style is also known as a highly ornamental style, [122] similar to the Baroque style. These semantic organizations of the stylistic movements by using only the information extracted by UFL from the content of the paintings is quite surprising, since it indicates that historical knowledge of the stylistic movement can be directly determined from the paintings without use of any other information.

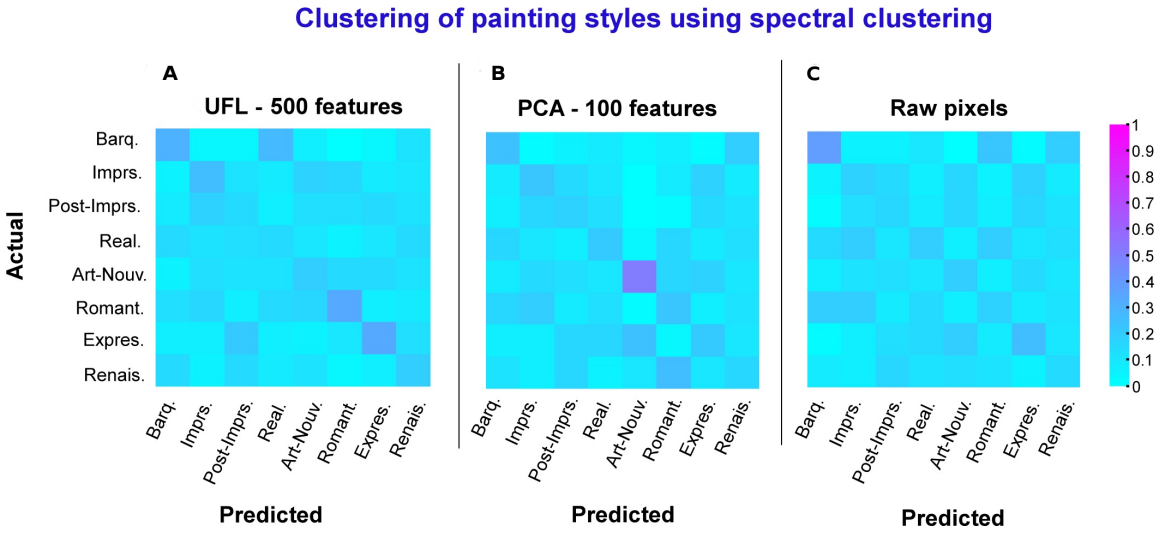

Figure 4.3: Confusion matrices of clustering of painting styles. Each entry of the confusion matrices are scaled by the total paintings in a column, which in essence provides precision, i.e., the ratio of clustered paintings that actually belong to the true style.

The confusion matrix in Figure 4.3A demonstrates that the clustering of styles using 500 UFL features produced the maximum number of paintings correctly in 6 out of 8 styles, as determined by the cost analysis. These styles include the Baroque, Impressionism, Art Nouveau, Romanticism, Expressionism, and Renaissance styles. The confusion matrix in Figure 4.3B shows that the clustering of paintings using 100 PCA features, also assigned the maximum number of paintings in style 6 out of 8 styles a well. These were the Baroque, Impressionism, Post-Impressionism, Realism, Art Nouveau, and Expressionism styles. However, using the 100 PCA features, the clustering algorithm predicted style groupings with poorer recall rates (Table 4.3). This means that only a few of the paintings that actually belong to a style group were selected. For instance, the precision of the Art-Nouveau cluster using 100 PCA features was 0.508, but the recall was 0.078, which is very poor. For the same Art Nouveau cluster using 500 UFL features the precision, although lower, was 0.194. However, the recall was 0.28, which is much better than the recall rate of 0.078 obtained by the

100 PCA features. Clustering with raw pixels (Figure 4.3C) provided the maximum painting counts in the four painting styles of Baroque, Post-Impressionism, Realism, and Expressionism.

The significance tests for painting style clustering was performed on the top performing UFL and PCA features based on the F-scores. It was also perfored on the clusters obtained by the raw pixels. The random PCA and similarity matrix distributions for simulated data, revealed that the Realism and Romanticism clusters formed by the 500 UFL features and the raw pixels were both due to chance and not due to the features specific to those painting styles. Both random distribution generation schemes provided the same results for the hypothesis tests across all stylistic clusters for the 500 UFL and raw pixel based features, although they had different cutoff points for significant p-values. Also, the significance tests indicated that clusters found using 100 PCA features for the Romanticism and Art Nouveau stylistic groups were not statistically significant. Here again, both types of the random distributions provided the same conclusions based on the hypothesis tests.

4.3 LSA Boosted Convolutional Neural Networks

Table 4.4 presents the experimental results of the proposed CNN architecture for LSA word vectors against the baseline linear model and the CNN architecture designed for w2v word vectors. The best performing CNN architecture and word vector type across all datasets was the {1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN model. It had a higher accuracy than all other models. For the IMDB and AGNews datasets, the $\mathbf{W}_{w2v}$ word vectors reached their maximum accuracy using the {1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN models instead of the {3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN model, which was specifically designed for $\mathbf{W}_{w2v}$. Moreover, on the AGNews dataset, $\mathbf{W}_{w2v}$ word vectors were tied for accuracy on

the {1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN and {1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN models. On the DBPedia dataset, $\mathbf{W}_{\text{w2v}}$ word vectors reached its peak accuracy using the {3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN model. Even then, it only tied the {1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN model.

Model	**IMDB**	**AGNews**	**DBP**
{1,2,3}-TFIDF-Logistic Reg.	0.8942	0.9142	0.9797
{1,2,3}-$\mathbf{W}_{rand}$-wt-CNN	0.8957	0.9201	0.9847
{1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN	0.8996	0.9201	0.9854
{1,2,3}-$\mathbf{W}_{LSA}$-wt-CNN	0.9001	0.9209	0.9858
{1,2,3,4}-$\mathbf{W}_{rand}$-wt-CNN	0.8961	0.9195	0.9849
{1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN	0.8989	0.9201	0.9859
{1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN	**0.9010**	**0.9213**	**0.9862**
{3,4,5}-$\mathbf{W}_{rand}$-dp-CNN	0.8886	0.9159	0.9842
{3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN	0.8965	0.9196	0.9859
{3,4,5}-$\mathbf{W}_{LSA}$-dp-CNN	0.8980	0.9195	0.9855

Table 4.4: Classification accuracy for text documents. Bold face indicates best accuracy for a dataset.

When the {3,4,5}-$\mathbf{W}_E$ type-dp-CNN architecture was used, $\mathbf{W}_{w2v}$ performed better than $\mathbf{W}_{LSA}$ and $\mathbf{W}_{rand}$ on the AGNews and DBP datasets, but not on the IMDB dataset, demonstrating that this architecture is w2v specific. Nonetheless, this architecture never achieved the maximum performance. Also, on all of the CNN architectures, $\mathbf{W}_{rand}$ performed better than the linear ngram models. However, the $\mathbf{W}_{rand}$ word vectors still performed worse than $\mathbf{W}_{LSA}$ word vectors on all datasets, but tied the $\mathbf{W}_{w2v}$ word vectors on the AGNews dataset.

The average accuracy across all datasets was higher for the {1,2,3,4}-$\mathbf{W}_E$ type-wt-CNN (mean = 0.9349) and {1,2,3}-$\mathbf{W}_E$ type-wt-CNN (mean = 0.9347) architectures in comparison to the {3,4,5}-$\mathbf{W}_E$ type-dp-CNN (mean = 0.9326) architecture. Although, the {1,2,3,4}-$\mathbf{W}_E$ type-wt-CNN and {1,2,3}-$\mathbf{W}_E$ type-wt-CNN models have a small difference in mean performance, the extra quadgram in the former model helped to provide the individual maximum classification accuracy with $\mathbf{W}_{LSA}$, which the trigram model never achieved.

Model	AGNews	DBP
{1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN	**0.9213**	0.9862
6-$\mathbf{W}_{char}$-CNN	0.9145	0.9842
9-$\mathbf{W}_{char}$-CNN	0.9083	0.9834
29-$\mathbf{W}_{char}$-CNN	0.9133	**0.9871**

Table 4.5: Comparison of classification accuracies of the LSA-based CNN to character-based CNN models. Accuracy values of the character-based CNNs are taken from their respective studies. Bold face indicates best accuracy for a dataset.

We also compared the {1,2,3}-$\mathbf{W}_E$ type-wt-CNN model to state-of-the-art character-based CNN classifiers used in document classification in Table 4.5. These CNN models use alphanumeric characters to embed the documents rather than words. The idea is to learn the semantic relationships of the words in higher convolutional layers in a similar way to image classification tasks used for CNNs, where the input data are only pixels [102, 123]. The models mainly differ in the number of convolutional layers and the filters sizes in CNN architecture. The model of the character-based CNN by Zhang et al. [102] uses 6 convolutional layers with filter size of 7 (on the first two convolutional layers) and 3 (on the remaining convolutional layers), which is designated as 6-$\mathbf{W}_{char}$-CNN. In Conneau et al. [123], either 9 convolutional layers or 29 convolutional layers with filter size of 3 was used, which are designated as 9-$\mathbf{W}_{char}$-CNN and 29-$\mathbf{W}_{char}$-CNN, respectively. Compared to these models, our best performing {1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN model had the highest accuracy on the AGNews and second highest accuracy on the DBPedia datasets. Comparison results for the IMDB dataset is not available since [102, 123] did not use it for analysis.

Chapter 5

Discussion

In the subsequent sections, the implications of the methods presented in this thesis are discussed in detail.

5.1 Clustering Pipeline using ICA and UFL

It is demonstrated that standard clustering techniques, especially graph-based methods, can be used to achieve equivalent or better clustering performance than deep learning clustering algorithms when ICA blind source separation and unsupervised feature learning algorithms are applied. ICA BSS applied to either the PCA features vectors or to the graph-embedding vectors increased the saliency of the class-specific feature embedding. This is the first study to apply ICA BSS as an improvement to the multiclass problem in graph-based clustering algorithms. UFL using RICA and SFT helped build an improved similarity graph representation of the original input data, which is critical in graph-based clustering algorithms such as spectral clustering and GNMF.

In Figure 5.1A, ICA BSS separates the CMU-PIE face signals into distinct sources, which are indicated by a step function-like feature in each signal. Whereas the PCA

signals (Figure 5.1B) do not exhibit any distinct feature across the different class source signals. With PCA, the majority of the source signal information is carried in the first few components, thus confusing the different class information. Surprisingly, in the CMU-PIE dataset, PCA with ICA BSS and K-means clustering performed better than the more advanced graph-based clustering algorithms that used UFL extracted features. This may be due to the fact that UFL may be extracting noisy features thus decreasing the similarity graph's representation efficacy. This may be the case when UFL feature extraction has the undesired effect of decreasing baseline clustering performance for any of the clustering algorithms.

Furthermore, limiting the number principal components to only the number of classes in K-means combined with PCA, helped to extract only the pertinent class-specific features. First, PCA captures only the necessary variance by eliminating the extra information, and then BSS using ICA separates the signals pertaining to each class. Although unexpected, clustering success in the REUTERS-10K text document dataset is not altogether surprising. Since, ICA has been shown to perform reasonably well on text documents in the task of topic classification and information retrieval [124, 125].

Figure 5.2 plots the estimated clusters obtained from GNMF for the USPS dataset using both the L_2-normalized and RICA features. The L_2-normalized features used for clustering fail to highlight distinct groupings (Figure 5.2B) in the USPS digit data, while the RICA features provide much more compact clusters (Figure 5.2A), which is also evidenced by the higher clustering NMI and ACC. This shows that employing UFL for feature extraction is better able to capture the underlying structure in data, which is important for building an accurate similarity matrix [32]. Furthermore, without any hyper-parameter search in the feature learning procedure, a high clustering performance that is better than many advanced deep clustering techniques

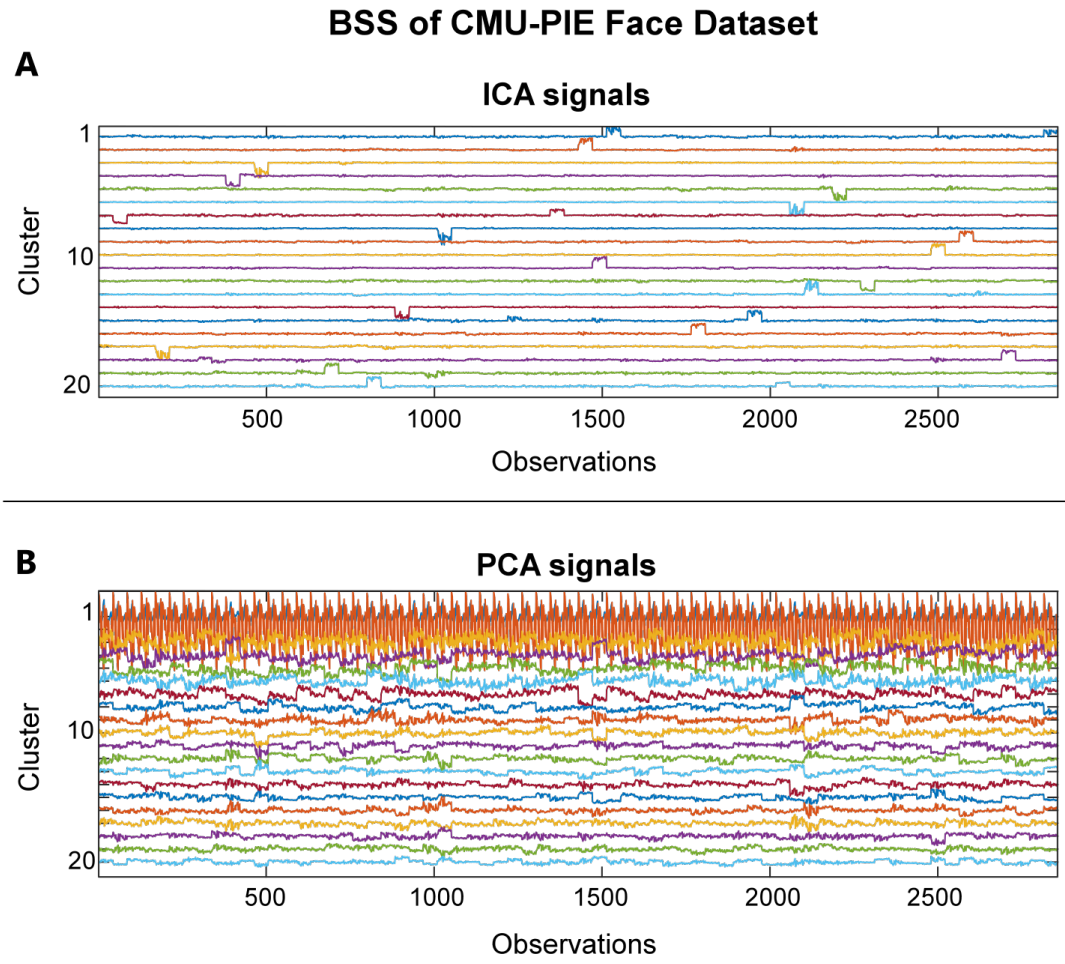

Figure 5.1: CMU-PIE ICA blind source separation. The first 20 signals are shown from a total of 68, which is the total number of possible clusters. **(A)** Each of the ICA sources on the top panel has a distinct signal. **(B)** The PCA sources do not show much separation, which is the reason for its poor clustering performance.

is achieved.

The limitations of the clustering methodology presented in our study are mainly due to the processing speed of the similarity graph and the matrix factorizations [43]. However, the speed limitation of the matrix factorization is due to the initial eigendecomposition and NMF computations. Otherwise, once the k basis vectors or eigenvectors have been obtained, the ICA computation is fast since the dimensionality of these vectors are small. Nonetheless, as a simple and effective boost to spectral clustering and GNMF, ICA blind source separation can be used alongside UFL with RICA or SFT.

Given the multiple stages of processing in the proposed clustering pipeline, guidelines for its recommended usage are provided. Initially, PCA with the number components set to the number of classes and ICA BSS using K-means should be applied as a simple baseline. This should be an effective baseline for text documents. In image datasets, another effective baseline would be to use GNMF or SPC-SYM using ICA BSS. GNMF or SPC-SYM using UFL with RICA and SFT for feature extraction can also be used as alternate clustering algorithm. Finally, GNMF or SPC-SYM using ICA BSS and UFL feature extraction can be used as a last attempt to obtain high clustering performance.

The proposed pipeline is effective for small (1,000 – 2,000 observations, such as COIL20, COIL100 and CMU-PIE) to medium-large datasets (10,000 – 50,000 observations, such as USPS and REUTERS-10K). Beyond 50,000 observations, deep clustering methods may be more suitable due to computation time of the similarity graphs, basis vectors, and eigenvectors [43]. The pipeline can be very useful for medical data, specifically in the analysis of electronic health records, where the number of observed patients is typically $\sim$1000 [126], due to the difficulty of collecting pertinent patient data. On a dataset with less than 50,000 observations, deep clustering algo-

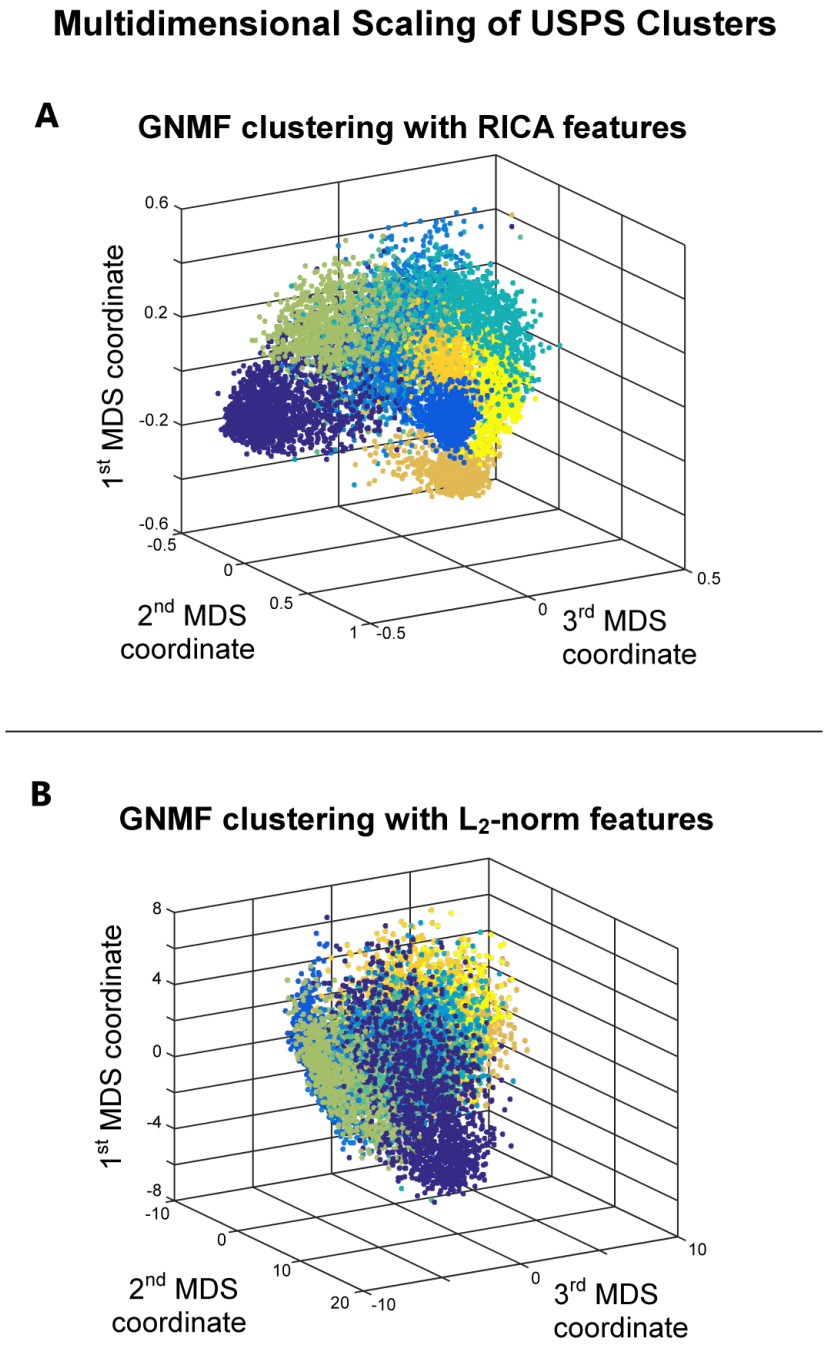

Figure 5.2: Visualization of the USPS clusters in multidimensional space. MDS is performed on the similarity matrix of the observations. **(A)** Clusters based on RICA extracted features are more distinct and homogeneous. **(B)** Clustering on the pixels only after applying L_2-normalization did not provide sufficient separation of the clusters.

rithms may require too much overhead in terms of hyper-parameter setup, without providing a large improvement in terms of clustering performance.

5.1.1 Overall Evaluation

Graph-based clustering performance can easily be improved by applying ICA blind source separation during the graph Laplacian embedding step. Applying unsupervised feature learning to input data using RICA or SFT, also improves clustering performance. Surprisingly in some cases, high clustering performance can be achieved by simply performing K-means clustering on the ICA components after PCA dimension reduction on the input data.

Compared to state-of-the-art non-deep learning clustering methods, ICA BSS and/or UFL with graph-based clustering algorithms outperformed all other methods. With respect to deep learning-based clustering algorithms, our methodology obtained the following rankings on the six different datasets: COIL20, 2^{nd} out of 5; COIL100, 2^{nd} out of 5; CMU-PIE, 2^{nd} out of 5; USPS, 3^{rd} out of 9; MNIST 8^{th} out of 15; and REUTERS-10K, 4^{th} out of 5.

These findings demonstrate the robustness of standard clustering methods when implemented with careful processing. Instead of developing complex deep learning clustering implementations, equivalent results can be achieved from existing standard techniques with the modifications presented herein. Furthermore, the clustering implementation presented in this study may also be used as an empirical baseline to justify use of sophisticated deep clustering learning networks and to reduce computation times. For future studies, increasing the speed of building the k nearest neighbor graph used in spectral clustering and GNMF will be investigated.

5.2 Significance Testing of Clusters

We demonstrated two new types of data distributions used in performing a reliability analysis of clusters for hypothesis testing: (1) a uniform distribution projected onto the principal components of the original data and (2) a randomized, weighted adjacency matrix. These distributions are used to form the null-hypothesis, which were used to assess the statistical significance of painting style clusters.

Also in this study it was demonstrated the feasibility of capturing stylistic information from a large dataset of digitized paintings without any prior knowledge of the painting styles or the types of features. We also demonstrated it may be possible to capture historical knowledge of the stylistic movements by utilizing only the UFL features extracted from the content of painting images. Thus, we have introduced a method to capture the stylistic characteristics of art in an objective manner. This greatly simplifies the feature extraction process in art, as it has been indicated by Shamir [127, 128, 129] that a great variety of handcrafted features are necessary to successfully capture the characteristics of art. Our success is due to that fact that we utilized the recently proposed UFL algorithm [12, 15, 28], which facilitates the use of a deep learning-like algorithm.

Shamir et al. [130] have shown that computers are able to exceed in accuracy the detection of abstract art created by children versus actual artists. In this study, Shamir et al. [130] used their specialized feature extraction algorithm [131, 132], however, it may be possible to obtain similar results using the simple UFL method demonstrated here. Another similar use of the UFL features could be to detect when an artist decides to change their painting style. For instance, it is known that Edouard Manet painted in both Realism and Impressionist styles.

Previous studies of clustering paintings by style [133, 56] had not analyzed the accuracy and reliability or significance of grouping paintings together by style, i.e.,

whether the style clusters were due to chance occurrence of the painting image data. Grouping paintings based on style is not a trivial problem [56], where naïve individuals can easily be influenced by the content of a painting, leading to incorrect style clusters [133]. Spehr et al. [133] clustered 772 paintings, using kernel PCA feature transformation and K-means clustering, however, they employed handcrafted features, such as template matching, semantic content, shape segmentation, and color distribution features, which required an intermediate feature selection step to choose the relevant features. Furthermore, Spehr et al. [133] failed to provide a performance metric such as the accuracy or F-score, but rather used a derived measure, which only provided the relative discriminability among clusters, despite having the ground truth style labels, most likely due to the issue of cluster label assignment [101].

Other clustering studies [134, 128], which also utilized handcrafted features to group paintings by their creating artists, failed to provide quantitative metrics regarding the performance of the clusters. In Lombardi [134], clusters were only used as a visualization tool, demonstrating the divergence among artists by their paintings. In Shamir and Tarakhovsky [128], using only 994 paintings from a total of 34 painters, phylogenetic trees were created to demonstrate the relationships between artists, but the confidence level [50] of the tree branches was not provided. Without the confidence levels, it is not possible to objectively assess whether the relationships between artists existed due to a chance occurrence of the collected data, which could cause a false acceptance of the style clusters. For instance, in our clustering of the styles with our best performing algorithm using 500 ULFK features, the performance of the Realism and Romanticism clusters were better than the chance F-score of 12.5%, but were not significant due to the null distribution tests. As the confusion matrix in Figure 4.3A demonstrates, Realism was incorrectly clustered as Baroque, and Romanticism was incorrectly clustered as Impressionism, Post-Impressionism, and Art

Nouveau.

5.2.1 Overall Evaluation

As a result, our null distribution methods provide a way to objectively assess whether groupings of paintings by a clustering algorithm are significant. These methods allow art researchers to determine the reliability of the groups of paintings independent of technical knowledge, in essence providing a second opinion regarding the clusters. The success of the two novel random distributions formed by a uniform distribution projected onto the PCA space of the original data and by a randomized weighted adjacency matrix indicate tath for clustering complex and nonstandard dataset such as paintings indicates we will be able assess reliablity of atypical datasets. This is important for datasets obtained in medical and biological studies because the sample size of the observations may be small compared to the population size of the actual data available.

5.3 LSA Boosted Convolutional Neural Networks

This study has shown that natively trained LSA word vectors can be used as an alternative to pretrained w2v word vectors in a CNN model for text document classification. Also we have introduced a novel architecture for CNN classification of text documents, using small word window sizes. The LSA-based CNN model has also been shown to perform better than all other character-based CNN models on the AGNews dataset and second best on the DBPedia dataset.

Most CNN architecture for document classifications have used combinations of window sizes ranging from 3, 5 and 7 words [123]. Although, the goal was to model short and long distance relationships among words [123], smaller window sizes of 1 or

2 words have not been analyzed, as in this current study. As shown in Johnson and Zhang [64], effective linear classification with {1,2,3}-ngram is heavily dependent on unigrams rather than bigram and trigrams. The larger window sizes may better at modeling semantic similarity tasks, but the short distances may be better suited for classification.

Thus, the success of the LSA-based CNN model architecture could be attributed to the use of the smaller filter sizes together with the LSA word vectors. Since the LSA word vectors are extracted using unigrams, the LSA only encodes short distance relationships among words. Thus, using filter sizes larger than quadgrams would not be able to capture meaningful semantic relationships among words. Whereas the w2v word vectors perform better in CNNs with larger filter sizes, since w2v word vectors are trained with large windows sizes ranging from 5 and 10 words in length [77]. Also the results show that carefully tuning traditional methods, such as LSA word vectors, gives equivalent results to deep methods, such as w2v, as shown in Levy et al. [77].

A limitation of linear models is that they cannot use ngram representations that are not present in the training dataset [64]. CNNs are able to find and use ngrams that are not wholly in the training set, which is why CNN based ngram models perform better than linear ngram models. For instance, CNN based ngrams can learn a general trigram "best *X-positive* ever", where *X-positive* represents a positive word. This general trigram can then be used to classify the testing data as long as it follows the general form [64].

One limitation of the LSA-based CNN model and the related small word window architecture, can be that in datasets long documents (i.e., many and long sentences per document), the model may begin to perform worse against models that take advantage of larger window sizes. In this case, CNN architecture based on w2v word vectors or character-based models may perform better, since they are more likely to

capture long distance semantic relationships among words.

5.3.1 Overall Evaluation

In this study we have shown that LSA word vectors using CNNs with small window sizes can be used as a baseline classifier for document classification. Moreover, the LSA word vectors out performed pretrained w2v word vectors in all CNN models presented in this study. One reason for this is that the LSA word vectors are more domain specific than pretrained w2v word vectors. Furthermore, with the LSA-based CNN models, LSA word vector dimensionality can be easily adjusted to any size. Whereas the pretrained w2v word vectors are set to a dimensionality of 300, unless a costly pretraining process is undertaken. For future studies, we would like to analyze the effect of using different LSA word vector dimensions. Also, we would like to implement other types of traditional word vectors, which could have an effect on larger text datasets.

Chapter 6

Conclusions

This thesis has demonstrated that with careful parameter selection and a combination of standard matrix factorization and clustering techniques, state-of-the-art clustering and classification performance can be achieved, independent of deep learning methods. The main findings of this thesis are:

1. Applying independent component analysis during the graph Laplacian embedding step in graph-based clustering methods will increase the signal difference among clusters, thus boosting accuracy.

2. Features extracted with the unsupervised feature learning procedure, using reconstruction ICA or Sparse Filtering as the feature learners also improves clustering performance.

3. Applying ICA in graph-based clustering using unsupervised feature learning can provide cumulative improvement to clustering performance.

4. High clustering performance can also be achieved by simply performing K-means clustering on the ICA components after PCA dimension reduction on the input

data. However, the key step here is to limit the PCA dimensions to the number of classes.

5. Two novel reference distributions for the statistical testing of clusters were demonstrated. One is based on the randomizing data in PCA space and the other randomizes the similarity measures among data observations.

6. A new single hidden layer CNN architecture utilizing small word window sizes and LSA-based word vectors was developed for classifying text documents. The accuracy of the classification was better than many deep layered CNNs.

Altogether these findings demonstrate the robustness of standard clustering and matrix factorization techniques in comparison to deep learning based methods. Equivalent results can be achieved from existing standard techniques with the modifications presented herein. Furthermore, the clustering and classification schemes presented here can be used as an empirical baseline to justify use of sophisticated deep learning networks and to reduce algorithm implementation overhead.

For future research directions, we would like to investigate methods for building large scale graphs based on content derived from data, where no graph existed before. In graph-based clustering techniques, a major limitation is the speed at which a similarity graph among observations can be constructed.

Bibliography

[1] M. Kevin, *Machine learning: a probabilistic perspective*. The MIT Press, 2012, vol. 1.

[2] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[3] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of intelligent information systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[4] S. Haykin, "A comprehensive foundation," *Neural networks*, vol. 2, no. 2004, p. 41, 2004.

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

[6] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[7] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[8] G. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-w

[11] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[12] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[13] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5747–5756.

[14] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A Simple Deep Learning Baseline for Image Classification?" *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[15] A. P. Coates, "Demystifying unsupervised feature learning," Ph.D. dissertation, Stanford University, 2012.

[16] "Unsupervised Feature Learning and Deep Learning Tutorial," May 2017. [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[17] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[19] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2003.

[20] C. M. Bishop, *Pattern recognition and machine learning*. Springer-Verlag New York, 2006.

[21] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[22] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.

[23] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 267–273.

[24] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[25] A. Hyvrinen, J. Karhunen, and E. Oja, *Independent component analysis*. John Wiley & Sons, 2004, vol. 46.

[26] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 561–580.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[31] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[32] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[33] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Iberoamerican Congress on Pattern Recognition*, 2013, pp. 117–124.

[34] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, 2014, pp. 1532–1537.

[35] 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 criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[36] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[37] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," *arXiv preprint arXiv:1610.04794*, 2016.

[38] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[39] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[40] Z. Zhang and M. I. Jordan, "Multiway Spectral Clustering: A Margin-Based Perspective," *Statistical Science*, vol. 23, no. 3, pp. 383–403, 2008.

[41] X. Y. Stella and J. Shi, "Multiclass spectral clustering," in *null*. IEEE, 2003, p. 313.

[42] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering." in *AAAI*, 2014, pp. 1293–1299.

[43] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger, "SpectralNet: Spectral Clustering using Deep Neural Networks," *arXiv preprint arXiv:1801.01587*, 2018.

[44] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "Ica with reconstruction cost for efficient overcomplete feature learning," in *Advances in neural information processing systems*, 2011, pp. 1017–1025.

[45] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng, in *Advances in neural information processing systems*, 2011, pp. 1125–1133.

[46] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko, "Permutation tests for classification," in *International Conference on Computational Learning Theory*. Springer, 2005, pp. 501–515.

[47] T. Hsing, S. Attoor, and E. Dougherty, "Relation between permutation-test p values and classifier error estimates," *Machine Learning*, vol. 52, no. 1-2, pp. 11–30, 2003.

[48] P. Good, *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer Science & Business Media, 2013.

[49] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[50] M. Nei and S. Kumar, *Molecular evolution and phylogenetics*. Oxford university press, 2000.

[51] J. Felsenstein, "Confidence limits on phylogenies: an approach using the bootstrap," *Evolution*, vol. 39, no. 4, pp. 783–791, 1985.

[52] M. K. Kerr and G. A. Churchill, "Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments," *Proceedings of the national academy of sciences*, vol. 98, no. 16, pp. 8961–8965, 2001.

[53] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer, "Testing for nonlinearity in time series: the method of surrogate data," *Physica D: Nonlinear Phenomena*, vol. 58, no. 1-4, pp. 77–94, 1992.

[54] D. Mandic, M. Chen, T. Gautama, M. Van Hulle, and A. Constantinides, "On the characterization of the deterministic/stochastic and linear/nonlinear nature of time series," in *Proceedings of The Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 464, no. 2093. The Royal Society, 2008, pp. 1141–1160.

[55] T. Schreiber and A. Schmitz, "Surrogate time series," *Physica D: Nonlinear Phenomena*, vol. 142, no. 3-4, pp. 346–382, 2000.

[56] C. Wallraven, R. Fleming, D. Cunningham, J. Rigau, M. Feixas, and M. Sbert, "Categorizing art: Comparing humans and computers," *Computers & Graphics*, vol. 33, no. 4, pp. 484–495, 2009.

[57] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8595–8598. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6639343/

[58] K. Javed, S. Maruf, and H. A. Babri, "A two-stage Markov blanket based feature selection algorithm for text classification," *Neurocomputing*, vol. 157, pp. 91–104, 2015.

[59] A. J. J. Yepes, L. Plaza, J. Carrillo-de Albornoz, J. G. Mork, and A. R. Aronson, "Feature engineering for MEDLINE citation categorization with MeSH," *BMC bioinformatics*, vol. 16, no. 1, p. 1, 2015.

[60] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[61] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive datasets*. Cambridge university press, 2014.

[62] H. Kim, P. Howland, and H. Park, "Dimension reduction in text classification with support vector machines," in *Journal of Machine Learning Research*, 2005, pp. 37–53.

[63] C. Silva, U. Lotri?, B. Ribeiro, and A. Dobnikar, "Distributed text classification with an ensemble kernel-based learning approach," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 3, pp. 287–297, 2010.

[64] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[65] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[66] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," *arXiv preprint arXiv:1602.02373*, 2016.

[67] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[68] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[69] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[70] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[71] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012.

[73] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[74] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, 2014, pp. 512–519.

[75] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[76] E. Altszyler, M. Sigman, S. Ribeiro, and D. F. Slezak, "Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database," *arXiv preprint arXiv:1610.01520*, 2016.

[77] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[78] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-20)," 1996.

[79] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on.* IEEE, 2002, pp. 53–58.

[80] "(Matlab) Codes and Datasets for Subspace Learning (Dimensionality Reduction)." [Online]. Available: http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

[81] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[82] E. Oja, "Principal components, minor components, and linear neural networks," *Neural networks*, vol. 5, no. 6, pp. 927–935, 1992.

[83] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[84] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 1991, pp. 586–591.

[85] E. Gultepe, T. E. Conturo, and M. Makrehchi, "Predicting and grouping digitized paintings by style using unsupervised feature learning," *Journal of Cultural Heritage*, 2017.

[86] S. Choi, A. Cichocki, H.-M. Park, and S.-Y. Lee, "Blind source separation and independent component analysis: A review," *Neural Information Processing-Letters and Reviews*, vol. 6, no. 1, pp. 1–57, 2005.

[87] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on signal processing*, vol. 45, no. 2, pp. 434–444, 1997.

[88] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of neuroscience methods*, vol. 134, no. 1, pp. 9–21, 2004.

[89] R. N. Vigrio, "Extraction of ocular artefacts from EEG using independent component analysis," *Electroencephalography and clinical neurophysiology*, vol. 103, no. 3, pp. 395–404, 1997.

[90] V. D. Calhoun, T. Adali, G. D. Pearlson, and J. Pekar, "A method for making group inferences from functional MRI data using independent component analysis," *Human brain mapping*, vol. 14, no. 3, pp. 140–151, 2001.

[91] M. J. McKeown and T. J. Sejnowski, "Independent component analysis of fMRI data: examining the assumptions," *Human brain mapping*, vol. 6, no. 5-6, pp. 368–372, 1998.

[92] M. Nascimento, e. F. F. Silva, T. Sfadi, A. C. C. Nascimento, T. E. M. Ferreira, L. M. A. Barroso, C. Ferreira Azevedo, S. E. F. Guimares, and N. V. L. Sero, "Independent Component Analysis (ICA) based-clustering of temporal RNA-seq data," *PLOS ONE*, vol. 12, no. 7, p. e0181195, Jul. 2017.

[93] A. Hyvrinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[94] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural computation*, vol. 12, no. 2, pp. 337–365, 2000.

[95] S. K. Thompson, *Sampling*, 3rd ed., ser. Wiley series in probability and statistics, Hoboken, N.J, 2012.

[96] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2761–2773, 2010.

[97] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1?2, pp. 83–97, 1955.

[98] A. R. Finocchio, "Nineteenth-Century French Realism | Essay | Heilbrunn Timeline of Art History | The Metropolitan Museum of Art." [Online]. Available: http://www.metmuseum.org/toah/hd/rlsm/hd_rlsm.htm

[99] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition, 2010," *Cited on*, vol. 80.

[100] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[101] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, pp. 379–390.

[102] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[103] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, 2012, pp. 90–94.

[104] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[105] *AG's corpus of news articles*. [Online]. Available: https://www.di.unipi.it/ gulli/AG_corpus_of_news_articles.html

[106] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and others, "DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[107] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[108] *CS231n Convolutional Neural Networks for Visual Recognition.* [Online]. Available: http://cs231n.github.io/convolutional-networks/

[109] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[110] 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*, vol. 15, no. 1, pp. 1929–1958, 2014.

[111] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[112] F. Li, H. Qiao, B. Zhang, and X. Xi, "Discriminatively Boosted Image Clustering with Fully Convolutional Auto-Encoders," *arXiv preprint arXiv:1703.07980*, 2017.

[113] H. Liu, M. Shao, S. Li, and Y. Fu, "Infinite ensemble for image clustering," 2016, pp. 1745–1754.

[114] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep Clustering with Convolutional Autoencoders," in *International Conference on Neural Information Processing*, 2017, pp. 373–382.

[115] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning Discrete Representations via Information Maximizing Self Augmented Training," *arXiv preprint arXiv:1702.08720*, 2017.

[116] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-nmf model for learning hidden representations," in *International Conference on Machine Learning*, 2014, pp. 1692–1700.

[117] W. Zhang, X. Wang, D. Zhao, and X. Tang, "Graph degree linkage: Agglomerative clustering on a directed graph," in *European Conference on Computer Vision*, 2012, pp. 428–441.

[118] W. Zhang, D. Zhao, and X. Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, 2013.

[119] F. Nie, Z. Zeng, I. W. Tsang, D. Xu, and C. Zhang, "Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering," *IEEE Transactions on Neural Networks*, vol. 22, no. 11, pp. 1796–1808, 2011.

[120] "Romanticism," *Encyclopedia Britannica*. [Online]. Available: https://www.britannica.com/art/Romanticism

[121] "Baroque art and architecture," *Encyclopedia Britannica*. [Online]. Available: https://www.britannica.com/art/Baroque-art-and-architecture

[122] "Art Nouveau | artistic style," *Encyclopedia Britannica*. [Online]. Available: https://www.britannica.com/art/Art-Nouveau

[123] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, 2016.

[124] T. Kolenda, L. K. Hansen, and S. Sigurdsson, "Independent components in text," in *Advances in Independent Component Analysis*, 2000, pp. 235–256.

[125] P. Mirowski, M. Ranzato, and Y. LeCun, "Dynamic auto-encoders for semantic indexing," in *Proceedings of the NIPS 2010 Workshop on Deep Learning*, 2010, pp. 1–9.

[126] E. Gultepe, J. P. Green, H. Nguyen, J. Adams, T. Albertson, and I. Tagkopoulos, "From vital signs to clinical outcomes for patients with sepsis: a machine learning basis for a clinical decision support system," *Journal of the American Medical Informatics Association*, 2013. [Online]. Available: http://jamia.bmj.com/content/early/2013/08/19/amiajnl-2013-001815.abstract

[127] L. Shamir, T. Macura, N. Orlov, D. M. Eckley, and I. G. Goldberg, "Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art," *ACM Transactions on Applied Perception*, vol. 7, no. 2, pp. 1–17, Feb. 2010.

[128] L. Shamir and J. A. Tarakhovsky, "Computer analysis of art," *Journal on Computing and Cultural Heritage*, vol. 5, no. 2, pp. 1–11, Jul. 2012.

[129] L. Shamir, "What makes a Pollock Pollock: A machine vision approach," *International Journal of Arts and Technology*, vol. 8, no. 1, pp. 1–10, 2015.

[130] L. Shamir, J. Nissel, and E. Winner, "Distinguishing between Abstract Art by Artists vs. Children and Animals: Comparison between Human and Machine Perception," *ACM Transactions on Applied Perception*, vol. 13, no. 3, pp. 1–17, May 2016.

[131] L. Shamir, N. Orlov, D. M. Eckley, T. J. Macura, and I. G. Goldberg, "IICBU 2008: a proposed benchmark suite for biological image analysis," *Medical & biological engineering & computing*, vol. 46, no. 9, pp. 943–947, 2008.

[132] N. Orlov, L. Shamir, T. Macura, J. Johnston, D. M. Eckley, and I. G. Goldberg, "WND-CHARM: Multi-purpose image classification using compound image transforms," *Pattern recognition letters*, vol. 29, no. 11, pp. 1684–1693, 2008.

[133] M. Spehr, C. Wallraven, and R. W. Fleming, "Image Statistics for Clustering Paintings According to their Visual Appearance." *Computational Aesthetics*, vol. 2009, pp. 1–8, 2009.

[134] T. E. Lombardi, *Classification of Style in Fine-art Painting*. Citeseer, 2005.