



DEEP CLUSTERING

By

ANURAG GOEL

(PhD19015)

Under the supervision of

Prof. Angshul Majumdar

COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI– 110020

NOVEMBER, 2023



DEEP CLUSTERING

By

ANURAG GOEL

PhD19015

A Thesis

submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI– 110020

NOVEMBER, 2023

Certificate

This is to certify that the thesis titled *Deep Clustering* being submitted by *Anurag Goel* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

November, 2023



Prof. Angshul Majumdar

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Dedication

I dedicate this thesis to those who have been the pillars of my journey, providing unwavering support, encouragement, and inspiration.

To my family, whose boundless love and sacrifices have been my foundation. Your belief in me has fueled my determination to reach this academic milestone. Special thanks to my wife Nidhi for her whole-hearted belief in me, constantly motivating me, and for never leaving my side in this journey.

To my friends, who have been my companions through the highs and lows of this academic adventure. Your camaraderie has made the challenges more bearable and the successes more joyful.

To my professors and mentors, who have shared their knowledge generously and guided me with wisdom. Your expertise has been a guiding light, shaping my understanding and passion for this field.

This thesis is a testament to the collective support of these individuals, and I am forever grateful for the impact you have had on my academic and personal growth.

Anurag Goel

Acknowledgements

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Angshul Majumdar, who has been an embodiment of everything I expected from my advisor. I cannot thank him enough for the continuous support, patience, motivation, immense knowledge, valuable guidance, scholarly inputs and consistent encouragement. This thesis and my growth in the past four years are credits to his technical prowess and constant engagement in my work. I am truly blessed to have him as my advisor.

I am much indebted to Dr. Saket Anand, Dr. Debarka Sengupta, Dr. A. V. Subramanyam and Dr. A. Anil Kumar for their insightful comments and encouragement.

I am profoundly grateful to the Indraprastha Institute of Information Technology for providing excellent infrastructure and research environment. I want to thank the University Grants Commission for providing me a research fellowship that helped me financially during my first year of Ph.D.

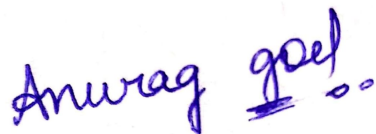
I want to express my sincere thanks to The Sengupta Laboratory headed by Dr. Debarka Sengupta, Dr. Emilie Chouzenoux and Dr. Giovanni Chierchia for their research collaborations.

This thesis would be incomplete without the mention of my support system from the SALSA lab. I thank all my friends and colleagues in this journey, Jyoti Maggu, Aanchal Mongia, Shikha Singh, Shalini Sharma, Pooja Gupta, Priyadarshini Rai, and Kriti Kumar for always keeping me motivated and hopeful in bad times of this journey.

I am profoundly grateful to my employer organization Delhi Technological University (DTU) for helping me financially during my Ph.D. journey. I would also like to thank my colleagues in DTU for being supportive throughout this journey.

This thesis and my Ph.D. itself would not have seen culmination if not for my family. My parents have given me the freedom to choose, my wife has given me the confidence to decide and a faith that no matter what, I have them watching my back. To my beloved daughter Taarini, I would like to express my thanks for always cheering me up. My gratitude to my family is incomplete without special mention for my sisters and brother-in-laws, for always believing in me, for the endless love and supporting me throughout my life. I thank my all time friends, Himanshu and Ashutosh, for the unconditional support.

I also express my regards to all those who supported me in any respect during the completion of my Ph.D.

A handwritten signature in blue ink that reads "Anurag Goel" with a stylized flourish at the end.

Anurag Goel

Abstract

The traditional way of clustering is first extracting the feature vectors according to domain-specific knowledge and then employing a clustering algorithm on the extracted features. Deep learning approaches attempt to combine feature learning and clustering into a unified framework which can directly cluster original images with even higher performance. Therefore, deep clustering approaches rely on deep neural networks for learning high-level representations for clustering.

Auto-encoders are a special instance of deep neural networks which are able to learn representations in a fully unsupervised way. Majority of the prior works on deep clustering are based on auto-encoder framework where the clustering loss is embedded into the deepest layer of an auto-encoder. The problem with auto-encoder is that they require training an encoder and a decoder network. The clustering loss is incorporated after the encoder network; the decoder network is not relevant for clustering. The need of learning an encoder and a decoder network leads to learning twice the number of parameters as that of a standard neural network. This may lead to overfitting especially in the cases where the number of data instances are limited. Moreover, the current state-of-the-art deep clustering approaches are not able to capture the discriminative information in the learned representations due to the lack of supervision [1].

To alleviate the aforementioned problems, we have proposed deep clustering approaches based on Dictionary Learning, Transform Learning, and Convolutional Transform Learning (CTL) frameworks. We have embedded two popular clustering algorithms – K-means clustering and Sparse Subspace clustering. The limitation of unsupervised learning in existing deep clustering approaches is mitigated by incorporating contrastive learning in CTL framework. The proposed deep clustering approaches are evaluated using datasets from multiple domains including computer vision, hyperspectral imaging, text and multiview datasets. The results demonstrate the superiority of the proposed approaches over the current state-of-the-art deep clustering approaches.

Contents

Dedication	i
Acknowledgements	ii
Abstract	iv
List of Tables	ix
List of Figures	xi
1 Introduction	1
1.1 Problem Statement	3
1.2 Background	3
1.2.1 Auto-encoders based deep clustering approaches	4
1.2.2 Convolutional Neural Network (CNN) based deep clustering approaches	5
1.2.3 Generative Adversarial Network (GAN) based deep clustering approaches	6
1.2.4 Contrastive Learning based deep clustering approaches	7
1.3 Datasets Description	8
1.3.1 Computer Vision datasets	9
1.3.2 Hyperspectral Images datasets	11

1.3.3	Text datasets	12
1.3.4	Mutli-view datasets	13
1.4	Evaluation Metrics Description	15
1.4.1	Evaluation Metrics for Clustering	15
1.4.2	Evaluation Metrics for Hyperspectral Imaging	17
1.4.3	Evaluation Metrics for Text datasets	18
1.5	Research Contributions	19
1.5.1	Dictionary Learning based deep clustering approaches .	20
1.5.2	Transform Learning based deep clustering approaches .	21
1.5.3	Convolutional Transform Learning based deep cluster- ing approaches	22
List of Abbreviations		1
2	Dictionary Learning based Deep Clustering Approaches	25
2.1	Dictionary Learning	25
2.2	Deep Dictionary Learning	27
2.3	Proposed Approaches	29
2.3.1	Dictionary Learning based K-means clustering	29
2.3.2	Dictionary Learning based Sparse Subspace clustering .	32
2.3.3	Deep Dictionary Learning based K-means clustering . .	35
2.3.4	Deep Dictionary Learning based Sparse Subspace clus- tering	40
2.4	Experiments and Results	43
2.4.1	Computer Vision	43
2.4.2	Hyperspectral Imaging	45
2.5	Summary	53

3	Transform Learning based Clustering Approaches	54
3.1	Transform Learning	54
3.2	Deep Transform Learning	58
3.3	Literature Review	60
3.4	Proposed Approaches	62
3.4.1	Transformed K-means Clustering	62
3.4.2	Deeply Transformed K-means Clustering	65
3.5	Experiments and Results	72
3.5.1	Text Datasets	72
3.5.2	Hyperspectral Imaging	74
3.5.3	Comparison with Deep Dictionary Learning based deep clustering approaches	78
3.6	Summary	80
4	Convolutional Transform Learning based Clustering Approaches	81
4.1	Convolutional Transform Learning (CTL)	82
4.2	Proposed Approaches	84
4.2.1	Deep Convolutional K-Means Clustering	84
4.2.2	Deep Convolutional Sparse Subspace Clustering	86
4.2.3	Contrastive Deep Convolutional Transform K-means Clustering	89
4.2.4	DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework	96
4.3	Experiments and Results	108
4.3.1	Computer Vision datasets	108
4.3.2	Hyperspectral Imaging	119
4.3.3	Multiview datasets	124

4.4	Summary	130
5	Conclusion	132
5.1	Summary of Contribution	132
5.1.1	Dictionary Learning based clustering approaches	132
5.1.2	Transform Learning based clustering approaches	133
5.1.3	Convolutional Transform Learning based clustering ap- proaches	133
5.2	Future Work	135
	References	137

List of Tables

1.1	Computer Vision datasets: Statistics	11
1.2	Text datasets: Statistics	12
1.3	MVC datasets: Statistics	14
2.1	DL+K-means and DL+SSC: Clustering Results	44
2.2	DL+K-means and DL+SSC: Runtime comparison (in seconds) .	45
2.3	DDLK: Clustering Results	50
2.4	DDLK: Ablation Studies Results	50
2.5	DDLK: Runtime Comparison (in seconds)	53
3.1	DTLK: Table lookup	72
3.2	TLK: Clustering Results on TDT2	73
3.3	TLK: Clustering Results on Reuters	73
3.4	TLK: Clustering Results on Newsgroup	73
3.5	DTLK: Runtime comparison (in seconds)	76
3.6	Comparison of DLK, DLS, DDLK and DTLK	79
4.1	DCKM: Clustering Results	109
4.2	DCKM: Ablation Studies Results	110
4.3	Contrastive DCTLK: Hyperparameters Settings	113
4.4	Contrastive DCTLK: Clustering Results	114

4.5	Contrastive DCTLK: Ablation Results on K-means loss and Contrastive loss	119
4.6	DCTLSSC: Results (Mean \pm Standard Deviation)	121
4.7	DCTLSSC: Runtime comparison (in seconds)	122
4.8	DeConFCluster: Hyperparameters settings	124
4.9	DeConFCluster: Clustering Results (All the metrics are in (%)) .	127
4.10	DeConFCluster: Ablation Studies Results on λ, μ	128
4.11	DeConFCluster: Ablation Studies Results on K-Means Regularizer	129
4.12	DeConFCluster: Ablation Studies Results on Piecemeal and Proposed Formulation	129

List of Figures

1.1	Sample Images of Yale Faces dataset	9
1.2	Sample Images of Extended YaleB dataset	9
1.3	Sample Images of COIL20 dataset	10
1.4	Sample Images of Olivetti Faces dataset	10
2.1	Dictionary Learning (DL)	26
2.2	Neural network interpretation of DL	27
2.3	Schematic diagram: Deep Dictionary Learning	27
2.4	Schematic Diagram of Proposed DL based Clustering Algorithms	35
2.5	DL+K-means and DL+SSC: Empirical Convergence plots. Here, x-axis represents the number of iterations and y-axis represents the normalized cost function value	46
2.6	Spatio-spectral Feature Extraction	47
2.7	DDLS: Empirical Convergence plot	52
3.1	Transform learning (TL)	55
3.2	Neural network interpretation of TL	56
3.3	Schematic diagram: Deep Transform learning	59
3.4	Schematic Diagram of Proposed TL+Kmeans Algorithm	65
3.5	TLK: Empirical Convergence Plot (2 clusters) for TDT2	74
3.6	TLK: Empirical Convergence Plot (2 clusters) for Reuters	74

3.7	TLK: Empirical Convergence Plot (2 clusters) for Newsgroup . . .	75
3.8	DTLK: Empirical Convergence Plot (20 bands)	77
3.9	DTLK: Detailed Results	77
4.1	Overview of the proposed DCKM architecture. L represents number of DCTL layers, M_l^c - filter size and F_l^c - #filters of the respective layer l and channel c . SELU is the activation function.	86
4.2	Contrastive DCTLK Architecture	93
4.3	General view of the DeConFuse Architecture. $C = 5$ represents the number of DeepCTL networks/channels, $L = 2$ is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c	106
4.4	DeConFCluster Architecture. C represents the number of DeepCTL networks/channels, L is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c	108
4.5	DCKM: Empirical Convergence Plot	111
4.6	Contrastive DCTLK: Empirical Convergence Plots	115
4.7	Contrastive DCTLK: Ablation Results on K-means loss regularizer (β)	115
4.8	Contrastive DCTLK: Ablation Results on Contrastive loss regularizer (δ)	116
4.9	Contrastive DCTLK: Ablation Results on Learning rate	117
4.10	Contrastive DCTLK: Ablation Results on μ, ϵ regularizers	118
4.11	DCTLSSC: Ablation Results	123
4.12	DeConFCluster: Empirical Convergence Plots	127
4.13	DeConFCluster: Ablation Results on λ, μ	128
4.14	DeConFCluster: Ablation Results on K-Means Regularizer	129

Chapter 1

Introduction

The conventional clustering algorithms extract the feature vectors according to domain-specific knowledge and employ a clustering algorithm on the extracted features. Deep learning approaches attempt to combine feature learning and clustering into a unified framework which can directly cluster original images with even higher performance.

The regular neural network projects data to form a representation which is then used for supervised tasks. During training, the network weights are learned through various techniques including gradient descent, backpropagation and others. However, when a neural network does not have any output, the backpropagation leads to a trivial solution where the network weights and representations are both zeros, regardless of the cost function used. This makes the conventional feedforward neural networks unsuitable for unsupervised representation learning as even incorporating an unsupervised clustering loss does not change the trivial solution.

Prior research on deep clustering [2–5] has addressed the aforementioned problem by incorporating the clustering losses within the auto-encoder framework. Auto-encoders are typically used for unsupervised learning tasks like feature learning, data reconstruction etc. An auto-encoder is composed of an encoder and a decoder network. The encoder network maps the input data into a lower-dimensional latent space and the decoder network decodes the encoded representation back to the original input space. This allows the auto-encoders to capture meaningful representations of data and makes them well-suited for embedding and clustering tasks. The auto-encoders based deep clustering approaches embed the clustering loss in the latent space after the encoder.

The limitation of auto-encoder is that it requires training an encoder and a decoder network. The clustering loss is incorporated after the encoder network; the decoder network is not relevant for clustering. The need of learning an encoder and a decoder network leads to learning twice the number of parameters as that of a standard feedforward neural network. This may lead to overfitting especially in the cases where the number of data instances are limited [6]. Moreover, the current state-of-the-art deep clustering approaches are not able to capture the discriminative information in the learned representations due to the lack of supervision [1].

1.1 Problem Statement

The current state-of-the-art deep clustering approaches are majorly based on auto-encoders which comprised of encoder-decoder networks. The auto-encoders based deep clustering approaches tend to overfit in data constrained scenarios due to the requirement of learning the weights of encoder as well as decoder. This research dissertation aims to propose the deep clustering approaches based on Deep Dictionary Learning, Deep Transform Learning and Convolutional Transform Learning frameworks. In this dissertation, we have targeted two popular clustering techniques – K-means clustering and Sparse Subspace clustering, to embed in the deep learning frameworks. The proposed deep clustering approaches need to learn fewer parameteres; thus, alleviate the problem of overfitting in auto-encoders based deep clustering approaches and achieve good performance specially in data constrained scenarios.

1.2 Background

Deep clustering approaches use deep learning based networks to learn the deep representations and feed the learnt representations into shallow clustering methods. Several deep learning based architectures are implemented to learn the deep representations for clustering. We will briefly discuss the various current state-of-the art deep clustering approaches based on the deep learning model architecture used for learning the representations for clustering.

1.2.1 Auto-encoders based deep clustering approaches

One of the initial research studies on deep learning based clustering is based on stacked auto-encoder; in there the representation generated from the deepest layer of auto-encoder was fed into a separate clustering algorithm such as k-means or spectral clustering [7]. The auto-encoder regenerative loss and the clustering loss were optimized in a piecemeal fashion. Later, the limitations of piecemeal training were highlighted and a jointly learnt formulation of stacked auto-encoder and sparse subspace clustering was proposed in [2]. The jointly learnt formulation achieved better performance as compared to the piecemeal version. Another deep clustering model was proposed in [3] in which K-means clustering with Student's t-distribution kernel as the distance metric was embedded in the stacked auto-encoder and trained in a joint end-to-end fashion. K-means clustering with standard Euclidean distance was embedded in the auto-encoder in [4, 5]. Later, deep clustering approaches based on convolutional auto-encoder [8] and contractive auto-encoder [9] were proposed. Semi-supervised Deep Embedded Clustering (SDEC) integrated the pairwise constraints to enhance the feature learning process in Deep Embedded Clustering [10]. Deep Clustering under Similarity and Reconstruction constraints (DCSR) jointly optimized the Adaptive Siamese loss and Reconstruction loss to flexibly account for similarities and promote clustering stability [11]. Spectral clustering loss was embedded in the auto-encoder in [12]. The robustness of Deep K-means approach proposed in [5] was improved in [13] by replacing K-means with hierarchical K-means. A dual variational auto-encoder structure is leveraged to impose the reconstruction error

for the latent embedding and its corresponding noise counterpart [14]. Recently, k -Deep Variational Autoencoder (k -DVAE) is proposed, which employs multiple auto-encoders to produce enhanced latent representations for improved clustering outcomes [15].

The limitation of auto-encoder is that it requires training an encoder and a decoder network. The clustering loss is incorporated after the encoder network; the decoder network is not relevant for clustering. The need of learning an encoder and a decoder network leads to learning twice the number of parameters as that of a standard feedforward neural network. This may lead to overfitting especially in the cases where the number of data instances are limited [6].

1.2.2 Convolutional Neural Network (CNN) based deep clustering approaches

The agglomerative clustering is combined with Convolutional Neural Network in a recurrent manner in [16]. This approach divided the time steps into multiple periods and in each period, merging of clusters and representation learning are performed in forward pass and backward pass respectively. DeepClust jointly learns a Convolutional Neural Network and cluster assignments of resulting features [17]. DeepClust iterates between applying k -means clustering on the features generated by the Convolutional Neural Network and updating its weights by predicting the cluster assignments as pseudo-labels in a discriminative loss. A deep clustering framework based on Gated Convolutional Neural Network is proposed in [18]. Gated CNN uses a gated linear unit as non-linear activation function.

CNN-based deep clustering approaches are highly sensitive to noise and variations in input data. Additionally, CNNs are primarily used as supervised learning models. The filters in CNNs are learned in a data-driven manner and are not predefined or handcrafted. However, there is no inherent guarantee that each learned filter will be completely unique. There are several reasons for this like redundancy in training data, learning of local features, random weights initialization, and regularization techniques. Thus, learned representations might be redundant in CNN-based deep clustering approaches [19].

1.2.3 Generative Adversarial Network (GAN) based deep clustering approaches

ClusterGAN samples the latent variables from a mixture of discrete and continuous latent variables and coupled them with an inverse-mapping network, which projects the data to the latent space. The inverse-mapping network is trained with a clustering-specific loss and thus achieves clustering in latent space [20]. Deep Subspace Clustering via Dual Adversarial Generative networks (DSC-DAG) and Self-Supervised Deep Subspace Clustering with Adversarial Generative networks (S2DSC-AG) are proposed in [21]. In DSC-DAG, the distributions of both the inputs and corresponding latent representations are learnt via adversarial training simultaneously. In S2DSC-AG, a self-supervised information learning module substitutes for adversarial learning in the latent space, since both of them play the same role in learning discriminative latent representations. HC-MGAN is a deep clustering method for hierarchical clustering based on GANs with multiple generators [22]. Each generator of MGAN tends to generate

data that correlates with a sub-region of the real data distribution. This clustered generation is used to train a classifier for inferring from which generator a given image came from, thus providing a semantically meaningful clustering for the real distribution.

The limitations in GAN-based deep clustering algorithms are difficulty in convergence and mode collapse problems due to GAN architecture [19].

1.2.4 Contrastive Learning based deep clustering approaches

The deep clustering approaches discussed so far are unsupervised, they may not be capable of capturing discriminative information in the learned representations because of the absence of supervision. To alleviate the negative impact of unsupervised learning in clustering, the concept of Contrastive learning is introduced in Contrastive Clustering [23]. Contrastive Learning is a technique for self-supervised learning that enables a model to learn data features without relying on labeled samples. The approach involves generating pairs of positive and negative data samples, and maximizing similarity between the positive pairs while minimizing similarity between the negative pairs. Contrastive Clustering uses data augmentation to generate positive pairs and negative pairs of data samples that are projected into feature space for instance-level and cluster-level contrastive learning. Contrastive Deep Embedded Clustering integrated the contrastive loss in stacked denoising auto-encoders to obtain more representative features for clustering [1]. But Contrastive Deep Embedded Clustering suffers with the problem of overfitting in auto-encoders specially in data constrained

scenarios [6].

The prior studies on deep clustering are based on auto-encoders, Convolutional Neural Networks and Generative Adversarial Networks. The auto-encoders based approaches have the limitation of high number of learnable parameters due to encoder-decoder network architecture; CNN-based approaches do not guarantee to learn unique and meaningful filters due to lack of supervision and are highly susceptible to noise in input. GAN-based approaches suffer with the limitations of complex GAN architecture. Moreover, all these approaches are not able to learn the representations containing discriminative information for clustering due to lack of supervision. Contrastive learning based clustering overcomes this problem but requires data augmentation to compute the contrastive loss.

1.3 Datasets Description

This dissertation propose deep clustering approaches based on three different deep learning based frameworks namely Dictionary Learning, Transform Learning and Convolutional Transform Learning. While proposing novel deep clustering approaches, multiple datasets covering different domains are explored and used for the evaluation purpose. It helped us realize that the proposed deep clustering approaches are generic enough to be applied to the various application domains of clustering. Thus, the various datasets used for evaluating the approaches proposed in this dissertation are introduced here.



Figure 1.1: Sample Images of Yale Faces dataset

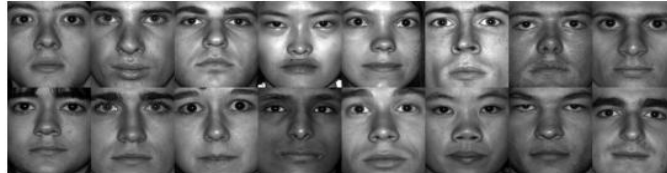


Figure 1.2: Sample Images of Extended YaleB dataset

1.3.1 Computer Vision datasets

The various computer vision datasets used for evaluating the performance of the proposed deep clustering approaches in this dissertation are illustrated below:

- Yale Faces¹: This dataset contains 165 images of 15 people in grayscale mode. There are 11 images per individual corresponding to 11 different facial expressions or configurations. The images are of size 150*150 pixels. Fig. 1.1 shows sample images of this dataset.
- Extended YaleB [24]: This dataset contains 2414 facial images of 38 individuals. There are approximately 64 images per individual. Fig. 1.2 shows sample images of this dataset.
- COIL20 [25]: This dataset contains 1440 grayscale images of 20 objects. There are approximately 72 images per object. Fig. 1.3 shows sample images of this dataset.
- ARFaces [26]: In this dataset, there are 2000 facial images correspond to

¹<http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html>



Figure 1.3: Sample Images of COIL20 dataset



Figure 1.4: Sample Images of Olivetti Faces dataset

different facial expressions, illuminations conditions and occlusions.

- Olivetti Faces²: This dataset comprises of 400 facial images of 40 individuals. There are ten images per individual corresponding to different lighting conditions, time, facial expressions, and facial details. The image size is 64*64 pixels. Fig. 1.4 shows sample images of this dataset.
- Pixraw10P Faces [27]: There are 100 images over 10 classes with 10000 features in this dataset.

The complete statistics of all the above mentioned datasets can be referred from Table 1.1.

²https://scikit-learn.org/0.19/datasets/olivetti_faces.html

Table 1.1: Computer Vision datasets: Statistics

Datasets	#Samples	#Classes	#Features
Yale Faces	165	15	22500
Extended YaleB	2414	38	2016
COIL20	1440	20	1024
ARFaces	2000	100	540
Olivetti Faces	400	40	4096
Pixraw10P Faces	100	10	10000

1.3.2 Hyperspectral Images datasets

The various hyperspectral images datasets used for evaluating the performance of the proposed deep clustering approaches in this dissertation are illustrated below:

- Indian Pines³: The Indian Pines dataset was collected by the airborne visible/infrared imaging spectrometer in Northwestern Indiana, with a size of 145×145 pixels with a spatial resolution of 20 m per pixel and 10-nm spectral resolution over the range of 400–2500 nm. As is the usual protocol, the work uses 200 bands, after removing 20 bands affected by atmospheric absorption. There are 16 classes.
- Pavia University⁴: This dataset is acquired by a reflective optics system imaging spectrometer (ROSIS). The image is of 610×340 pixels covering the Engineering School at the University of Pavia, which was collected under the HySens project managed by the German Aerospace Agency (DLR). The ROSIS-03 sensor comprises 115 spectral channels ranging from 430 to 860 nm. In this dataset, 12 noisy channels have been removed and

³<https://paperswithcode.com/dataset/indian-pines>

⁴<https://paperswithcode.com/dataset/pavia-university>

the remaining 103 spectral channels are investigated. The spatial resolution is 1.3 m per pixel. The available training samples of this dataset cover nine classes of interest.

1.3.3 Text datasets

The various text datasets used for evaluating the performance of the proposed deep clustering approaches in this dissertation are illustrated below:

- TDT2 corpus⁵: This dataset is a collection of 9394 documents collected from six English language news sources on daily basis for a duration of six months.
- Reuters-21578⁵: This dataset is a collection of manually categorized 8293 newswire stories from Reuters Ltd.
- 20 Newsgroups⁵: This dataset is a collection of 18846 documents.

The complete statistics of all the above mentioned datasets can be referred from Table 1.2.

Table 1.2: Text datasets: Statistics

Datasets	#Samples	#Classes	#Features
TDT2	9394	30	36771
Reuters-21578	8293	65	18933
20 Newsgroups	18846	20	26214

⁵<http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html>

1.3.4 Mutli-view datasets

The proposed novel mutiview clustering framework is evaluated on various multiview datasets which are listed below:

- 100leaves [28]: It contains one hundred plant species, each of which has 16 samples per specie. Thus, there are 100 clusters and 1600 total samples. For each sample, shape descriptor, fine scale margin and texture histogram are given.
- Amsterdam Library of Object Images (ALOI) [29]: ALOI dataset consists of 11025 images of 100 small objects. Every image is represented using four features namely - Color similarity, HSV, RGB, and Haralick features.
- Mfeat [28]: Mfeat dataset is from the UCI repository that contains 2000 samples of handwritten digits (0-9). Each image of this dataset is represented using six different features.
- WebKB [28]: It consists of 203 web pages with four classes collected from computer science departments of various universities. Each web page is attributed by the page's content, hyperlink's anchor text of the hyperlink and its title text.

The complete statistics of all the datasets mentioned above can be referred from Table 1.3.

Table 1.3: MVC datasets: Statistics

Datasets	#Samples	#Classes	#Views
100leaves	1600	100	3
ALOI	11025	100	4
Mfeat	2000	10	6
WebKB	203	4	3

The motivation for choosing the above datasets for evaluating the proposed deep clustering approaches can be driven by several factors:

- **Diversity:** These datasets cover a wide range of data types, including facial images, object images, remote sensing data, text data, and more. Their diversity can help researchers evaluate the performance and versatility of deep clustering algorithms across different domains and complexities.
- **Availability and Benchmarking:** These datasets are readily accessible, well-documented and widely used benchmarks. This allows for meaningful benchmarking and facilitates fair comparisons between different approaches.
- **Real-world Applications:** The demonstration of proposed deep clustering approaches on these datasets make the proposed approaches applicable in practical scenarios. For example, clustering facial images is important in various real-world applications, such as face recognition, image retrieval, and surveillance systems. Hyperspectral imaging datasets like Pavia University and Indian Pines are important for land cover classification and

environmental monitoring.

- **Size and Complexity:** These datasets vary in terms of size and complexity. For example, Olivetti Faces is relatively small and simple, making it suitable for initial experimentation and prototyping. In contrast, Extended Yale B and ARFaces contain a larger number of images with variations in lighting, pose, and expression, which can challenge the clustering model's robustness and generalization capabilities.

1.4 Evaluation Metrics Description

This section presents the various evaluation metrics used to evaluate the performance of the proposed deep clustering approaches in this dissertation.

1.4.1 Evaluation Metrics for Clustering

The various metrics used to evaluate the proposed approaches on computer vision datasets are Accuracy, Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI). These evaluation metrics are well-suited for evaluating the proposed deep clustering methods on computer vision datasets since they provide a quantitative, interpretable, and comprehensive assessment of clustering quality. These metrics address the unique challenges of working with visual data and are widely adopted within the computer vision community, promoting consistency and reproducibility in research. These evaluation metrics are described below:

- Accuracy: The accuracy is defined as the ratio of the number of data instances that are assigned the same cluster as in the ground truth to the total number of data instances [30].
- Normalized Mutual Information (NMI): This metric computes the normalized measure of similarity between the labels of same data instances [30]. The range of NMI is [0,1] where 0 signifies no correlation and 1 signifies the perfect correlation. The formula is given by:

$$NMI = \frac{I(l, c)}{\max(H(l), H(c))} \quad (1.1)$$

where $I(l, c)$ denotes the mutual information between the true label l and the assigned cluster c and H denotes the entropy. It is defined as follows:

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \cdot \log \left(\frac{p(x, y)}{p(x) \cdot p(y)} \right) \quad (1.2)$$

Entropy measures the uncertainty and can be computed as follows:

$$H(X) = - \sum_{x \in X} p(x) \cdot \log(p(x)) \quad (1.3)$$

- Adjusted Rand Index (ARI): ARI measures the similarity between two clusters by considering all pairs of data instances that are assigned to the same or different clusters in the actual and predicted labels [31]. The range of ARI is [-1,1]. The higher the ARI value, the better is the clustering.

$$ARI = \frac{(RI - E)}{(\max(RI) - E)} \quad (1.4)$$

where RI is the Rand Index and E is the Expected Rand Index value for random clustering.

$$RI = \frac{(a + b)}{\binom{n}{2}} \quad (1.5)$$

where a and b represents the number of times a pair of elements belongs to the same cluster and different clusters respectively across two clustering methods. $\binom{n}{2}$ is the number of unordered pairs in a set of n elements. Here, $\max(RI) = 1$.

$$E = \frac{\sum(\binom{n_i}{2}) * \sum(\binom{n_j}{2})}{\binom{N}{2}} \quad (1.6)$$

where n_i is the number of samples in cluster i, n_j is the number of samples in cluster j and N is the total number of samples.

1.4.2 Evaluation Metrics for Hyperspectral Imaging

For remote sensing, Overall Accuracy, Average Accuracy, and Kappa Coefficient are well-justified metrics for evaluating deep clustering methods. These metrics provide a quantitative, interpretable, and comprehensive assessment of clustering quality, taking into account the unique characteristics and objectives of remote sensing applications. These metrics are widely accepted and used within the remote sensing and geospatial analysis communities, making them valuable tools for assessing the effectiveness of deep clustering methods in this domain. These evaluation metrics are described below:

- Overall Accuracy (OA): It is a measure of the proportion of correctly classified pixels or samples over the total number of pixels or samples in

the hyperspectral image [32].

- Average Accuracy (AA): It is a metric that calculates the average accuracy of each class separately. It provides a more detailed assessment of classification performance compared to Overall Accuracy [33].
- Kappa Coefficient (κ): It is a statistical measure that assesses the agreement between the observed classification and a random classification. It takes into account the agreement that would be expected by chance [34].

$$\kappa = \frac{OA - EA}{1 - EA} \quad (1.7)$$

where OA is the Overall Accuracy and EA is the Expected Accuracy which can be calculated as follows:

$$EA = \frac{\sum_{i=1}^N ((a) * (b))}{N^2} \quad (1.8)$$

where a is the fraction of pixels or samples in the hyperspectral image that belongs to class i, b is the fraction of pixels or samples in the hyperspectral image that is assigned into class i by clustering algorithm and N is the total number of pixels or samples in the hyperspectral image.

1.4.3 Evaluation Metrics for Text datasets

The evaluation metrics used for evaluating the proposed deep clustering method on text datasets are Purity and Entropy. These evaluation metrics align well with the objectives and characteristics of text data. These metrics provide interpretable

and relevant assessments of clustering quality, addressing the unique challenges and goals of text clustering tasks. Researchers and practitioners in natural language processing and text analysis frequently use these metrics to assess and compare the performance of clustering algorithms on text data. These evaluation metrics are described below -

- **Purity:** Purity measures the extent to which data points within each cluster are assigned to the same true class [35]. Purity is given by,

$$Purity = \frac{1}{n} \sum_{k=1}^r \max_{1 \leq l \leq q} n_k^l \quad (1.9)$$

where n_k^l is the number of samples in cluster k that belong to original class l . A larger purity value indicates better clustering performance.

- **Entropy:** Entropy measures how classes are distributed on various clusters [35]. Entropy is given by,

$$Entropy = \frac{1}{n \log_2 q} \sum_{k=1}^r \sum_{l=1}^q n_k^l \log_2 \frac{n_k^l}{n_k} \quad (1.10)$$

where $n_k = \sum_l n_k^l$. Generally, a smaller entropy value corresponds to a better clustering quality.

1.5 Research Contributions

This dissertation has three main objectives: 1. To propose novel Dictionary Learning based deep clustering approaches; 2. To propose novel Transform

Learning based deep clustering approaches; and 3. To propose novel Convolutional Transform Learning based deep clustering approaches.

Our contributions towards these objectives are as follows:

1.5.1 Dictionary Learning based deep clustering approaches

We have proposed two dictionary learning based deep clustering algorithms - DL+K-means and DL+SSC in [36]. In DL+K-means and DL+SSC algorithms, K-means clustering loss and Sparse Subspace Clustering (SSC) loss are embedded in dictionary learning formulation respectively and the combined loss functions are jointly optimized in an end-to-end fashion. The proposed algorithms are tested on several computer vision datasets and the results of the proposed algorithms show improvement over the current state-of-the-art deep clustering approaches. Next, we have proposed another novel deep clustering method DDLK i.e. Deep Dictionary Learning based K-means, that embeds the K-means clustering loss in Deep Dictionary Learning (DDL) framework [37]. The joint formulation of DDL loss and K-means clustering loss is solved via Alternative Directed Multipliers Method (ADMM). The proposed DDLK algorithm is applied for scRNA-seq clustering to leverage pathway enrichment scores and yield robust grouping of single cells. We have also proposed DDL+SSC algorithm that incorporates SSC loss in the DDL formulation [38]. DDL+SSC algorithm is applied for hyperspectral image clustering where DDL nonlinearly transforms the data such that the transformed representation (of the data) is separable into subspaces. Comparison with state-of-the-art methods in hyperspectral

image clustering shows the superiority of the proposed DDL+SSC algorithm.

The publications related to the contributions towards the first research objective of this dissertation are as follows:

1. Anurag Goel, and Angshul Majumdar. "Clustering Friendly Dictionary Learning." In Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part I 28, pp. 549-557. Springer International Publishing, 2021.
2. Sarita Poonia, Anurag Goel, Smriti Chawla, Namrata Bhattacharya, Priyadarshini Rai, Yi Fang Lee, Yoon Sim Yap et al."Marker-free characterization of full-length transcriptomes of single live circulating tumor cells." *Genome Research* 33, no. 1 (2023): 80-95.
3. Anurag Goel, and Angshul Majumdar. "Sparse subspace clustering friendly deep dictionary learning for hyperspectral image classification." *IEEE Geoscience and Remote Sensing Letters* 19 (2021): 1-5.

1.5.2 Transform Learning based deep clustering approaches

We have proposed two deep clustering frameworks based on the paradigm of Transform learning. The first approach embeds the K-means clustering loss in Transform learning framework in joint end-to-end fashion and the joint formulation is solved via ADMM method [39]. The proposed approach is applied for document clustering and proves to be more effective than the current state-of-the art deep clustering approaches. In the second approach, we propose

a joint formulation of K-means clustering and deep transform learning based representation learning [40]. The ensuing joint formulation is solved via ADMM method. Application of the proposed solution to the hyperspectral band selection problem shows that the proposed solution improves over the current state-of-the-art by a significant margin.

The publications related to the contributions towards the second research objective of this dissertation are as follows:

1. Anurag Goel, and Angshul Majumdar. "Transformed K-means Clustering." In 2021 29th European Signal Processing Conference (EUSIPCO), pp. 1526-1530. IEEE, 2021.
2. Anurag Goel, and Angshul Majumdar, "K-Means Embedded Deep Transform Learning for Hyperspectral Band Selection", IEEE Geoscience and Remote Sensing Letters 19 (2022): 1-5.

1.5.3 Convolutional Transform Learning based deep clustering approaches

We have proposed four Convolutional Transform Learning (CTL) framework based deep clustering approaches. The first proposed CTL based approach is Deep Convolution K-means clustering that integrates the K-means clustering loss with Deep CTL (DCTL) framework in a joint end-to-end fashion [41]. The proposed approach improves over the current state-of-the-art on several computer vision datasets. We have also incorporated SSC loss in DCTL framework in our next proposed approach DDL+SSC. The proposed DDL+SSC is applied

for hyperspectral band selection. Next, we propose DeConFCluster that applies DCTL based K-means clustering in multiview clustering fusion framework. The proposed framework jointly trains DCTL network, multichannel fusion network and the K-Means clustering module; thus, the representations are diverse and effective as these are guided by K-Means loss. The proposed framework gives superior clustering performance than the current state-of-the-arts multiview clustering approaches on several multiview datasets. Finally, we incorporated contrastive learning in DCTL based K-means clustering framework to capture the discriminative information in the learned representations. The proposed framework outperforms the current state-of-the-art approaches on several facial images datasets.

The publications related to the contributions towards the third research objective of this dissertation are as follows:

1. Anurag Goel, Angshul Majumdar, Emilie Chouzenoux, and Giovanni Chierchia. "Deep Convolutional K-Means Clustering." In 2022 IEEE International Conference on Image Processing (ICIP), pp. 211-215. IEEE, 2022.
2. Anurag Goel, Angshul Majumdar, "Sparse Subspace Clustering Incorporated Deep Convolutional Transform Learning for Hyperspectral Band Selection" (Submitted in ICASSP 2024).
3. Pooja Gupta, Anurag Goel, Angshul Majumdar, Emilie Chouzenoux and Giovanni Chierchia, "DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework" (Submitted in

Information Sciences).

4. Anurag Goel and Angshul Majumdar, "Contrastive Deep Convolutional Transform K-means Clustering" (Submitted in Information Sciences).

The rest of this dissertation is structured as follows:

- The contributions towards the first research objective of the dissertation are explained in Chapter 2. The four proposed deep clustering approaches under these contributions are discussed in Section 2.3. The experiments and results of these proposed approaches are presented in Section 2.4.
- The contributions towards the second research objective of the dissertation are explained in Chapter 3. The two proposed deep clustering approaches under these contributions are discussed in Section 3.4. The experiments and results of these proposed approaches are presented in Section 3.5.
- The contributions towards the third research objective of the dissertation are explained in Chapter 4. The four proposed deep clustering approaches under these contributions are discussed in Section 4.2. The experiments and results of these proposed approaches are presented in Section 4.3.

Chapter 2

Dictionary Learning based Deep Clustering Approaches

This chapter presents four novel deep clustering approaches based on dictionary learning framework. The proposed deep clustering approaches use either K-means clustering or Sparse Subspace clustering to embed in the dictionary learning framework. The proposed approaches are evaluated on the datasets from two different kinds of domains: computer vision and hyperspectral imaging. The results demonstrate that the proposed approaches outperform state-of-the-art deep clustering techniques.

2.1 Dictionary Learning

Dictionary Learning (DL) [42] learns a dictionary/basis to synthesize the data from the latent representation. Consider a data X consists of n training samples where each sample consists of k features, a dictionary D that contains the basis

atoms to learn the representation Z . Then, the DL problem can be expressed as follows:

$$X = DZ \quad (2.1)$$

Dictionary learning aims at learning a dictionary D and the representation Z for the input data X , as shown in figure 2.1. The euclidean cost function of DL is

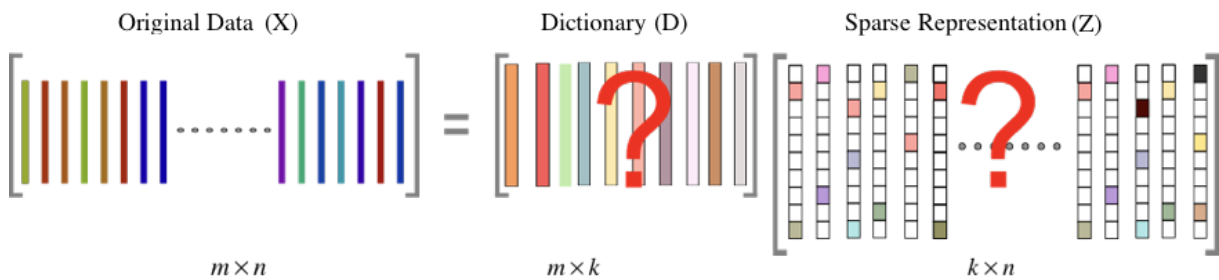


Figure 2.1: Dictionary Learning (DL)

given by:

$$\min_{D,Z} \|X - DZ\|_F^2 \quad (2.2)$$

Here, X is the training data, D is the dictionary to be learned and Z represents the coefficients. $\|\cdot\|_F^2$ represents the Frobenius norm.

The neural network interpretation of DL is shown in Figure 2.2 where dictionary D is considered as an operator being applied on the feature representation to generate the input data. In other words, dictionary D can be considered as weights of the neural network interpretation which are being learned along with the representative features. The direction of weights being learned is opposite to that of a neural network architecture because of the inverse formulation of DL. Please note that it is not a true neural network but its interpretation.

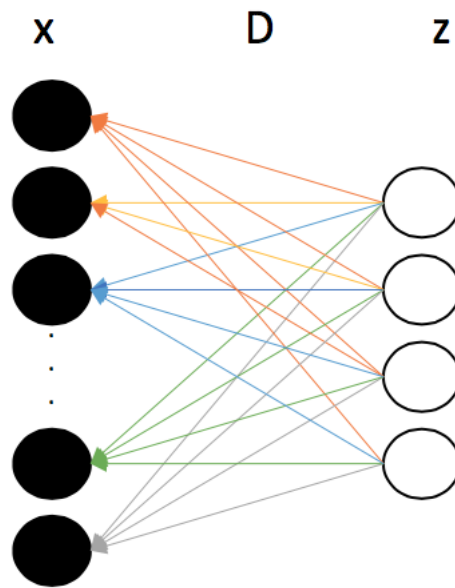


Figure 2.2: Neural network interpretation of DL

2.2 Deep Dictionary Learning

In Deep Dictionary Learning (DDL), the idea is to learn multiple levels of dictionaries. DDL proposes to extend the shallow DL problem into multiple levels. The idea of forming a deep architecture using DL stems arises from the success of deep learning.

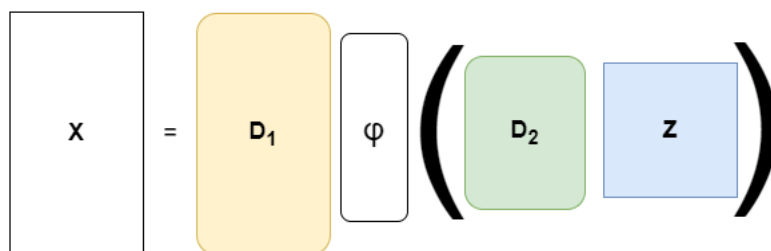


Figure 2.3: Schematic diagram: Deep Dictionary Learning

Mathematically, a DDL problem with two levels (shown in figure 2.3) can be formulated as

$$X = D_1 \phi(D_2 Z) \quad (2.3)$$

where, X is the input data, D_1 and D_2 are dictionaries and Z are the coefficients. D_1 and D_2 are separated by a non-linear activation function ϕ .

Greedy approach to DDL learns one layer of dictionary at a time. For a DDL framework with two layers of dictionary, the first layer of dictionary learns from the training data. The representations learned from the first layer of dictionary are passed as input in the second layer of dictionary to learn the second layer dictionary atoms and the representations. Same concept can be extended to multiple layers of dictionary. The DDL formulation for multiple layers can be expressed as:

$$X = D_1\phi(D_2\phi(\dots\phi(D_N Z))) \quad (2.4)$$

where, D_1, D_2, \dots, D_N are dictionaries corresponding to N levels of dictionaries respectively. Z represents the final level representations and ϕ represents the non-linear activation function. The corresponding optimization problem is expressed as follows:

$$\min_{D_1, D_2, \dots, D_N, Z} \|X - D_1\phi(D_2\phi(\dots\phi(D_N Z)))\|_F^2 + \lambda\|Z\|_1 \quad (2.5)$$

where, X is the input data, D_1, D_2, \dots, D_N are dictionaries corresponding to N levels of dictionaries respectively, Z is the final level representation, ϕ represents the non-linear activation function and $\|\cdot\|_F^2$ represents the Frobenius norm. $\|Z\|_1$ is to promote sparsity on the learned coefficient and λ is a hyperparameter.

2.3 Proposed Approaches

2.3.1 Dictionary Learning based K-means clustering

We present our novel approach *DL+K-means*¹ in which the K-means clustering loss is embedded in the dictionary learning formulation. The popular way to express K-means clustering is via the following formulation:

$$\sum_{i=1}^k \sum_{j=1}^n h_{ij} \|z_j - \mu_i\|_2^2 \quad (2.6)$$

$$h_{ij} = \begin{cases} 1, & \text{if } x_j \in \text{cluster } i \\ 0, & \text{otherwise} \end{cases} \quad (2.7)$$

where z_j denotes the j^{th} sample and μ_i the i^{th} cluster.

In [43], it was shown that the K-means clustering loss shown in equation 2.6 can be alternately represented in the form of matrix factorization as below:

$$\|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (2.8)$$

where Z is the data matrix formed by stacking z_j 's as columns and H is the matrix of binary indicator variables h_{ij} .

In our proposed formulation, the input to the K-means (Z) is not the raw data but the coefficients learned from dictionary learning. In the proposed formulation,

¹Anurag Goel and Angshul Majumdar, "Clustering Friendly Dictionary Learning." In Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part I 28, pp. 549-557. Springer International Publishing, 2021.

we incorporated the K-means cost into the dictionary learning formulation as shown below:

$$\min_{D,Z,H} \underbrace{\|X - DZ\|_F^2}_{\text{Dictionary Learning loss}} + \lambda \|Z\|_1 + \mu \underbrace{\|Z - ZH^\top (HH^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \quad (2.9)$$

In equation 2.9, μ controls the relative importance of the K-means clustering loss. To give equal importance to both the loss functions, we keep $\mu=1$. The sparsity promoting term $\|Z\|_1$ on the coefficients Z , which is usually there in dictionary learning framework, is dropped in the proposed formulation. The sparsity penalty is important for solving inverse problems but plays a less significant role in such representation learning problems apart from being a regularizer. Furthermore, the sparsity promoting term complicates the solution by requiring iterative updates. So, the final proposed formulation is shown as below:

$$\min_{D,Z,H} \underbrace{\|X - DZ\|_F^2}_{\text{Dictionary Learning loss}} + \underbrace{\|Z - ZH^\top (HH^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \quad (2.10)$$

The equation 2.10 can be solved via Alternative Directed Multipliers Method (ADMM) i.e. by updating each variable assuming the others to be constant. This leads to the following sub-problems:

$$\min_D \|X - DZ\|_F^2 \quad (2.11)$$

$$\min_Z (\|X - DZ\|_F^2 + \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2) \quad (2.12)$$

$$\min_H \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (2.13)$$

The closed form update for D is given by –

$$D_k = XZ_k^\dagger \quad (2.14)$$

where k is the iteration number and \dagger represents the pseudo-inverse.

To solve Z, we have to take the gradient of the expression in 2.12 and equate it to 0

$$\nabla_Z(\|X - DZ\|_F^2 + \|Z - ZH^\top(HH^\top)^{-1}H\|_F^2) = 0 \quad (2.15)$$

$$\implies D^\top X - D^\top DZ - Z(I - H^\top(HH^\top)^{-1}H) = 0 \quad (2.16)$$

$$\implies D^\top DZ + Z(I - H^\top(HH^\top)^{-1}H) = D^\top X \quad (2.17)$$

Equation 2.17 is a form of Sylvester's equation $AZ + ZB = C$, where $A = D^\top D$, $B = (I - H^\top(HH^\top)^{-1}H)$ and $C = D^\top X$. We can solve the Sylvester equation in 2.17 to get the update for Z.

The last step is to update H by solving equation 2.13. To get the updated H, we have to simply apply K-means clustering on the updated Z.

The algorithm is shown in a succinct fashion below. Once the convergence is reached, the clusters can be found from H. Since the proposed formulation is a non-convex function, we do not have any guarantees for convergence. We stop the iterations when the H does not change significantly in subsequent iterations.

Algorithm 1: DL+K-means

- 1 Initialize: D_0, Z_0, H_0 ;
 - 2 **repeat**
 - 3 Update D_k using 2.14;
 - 4 Update Z_k by solving Sylvester's equation in 2.17;
 - 5 Update H_k by applying K-means clustering on the updated Z ;
 - 6 **until** *Convergence is reached*;
-

2.3.2 Dictionary Learning based Sparse Subspace clustering

Here, our novel approach $DL+SSC^2$ is presented in which the Sparse Subspace Clustering (SSC) loss is embedded in the dictionary learning framework. In SSC [44], it is assumed that the samples belonging to the same cluster lie in the same subspace. The formulation for SSC is as follows:

$$\sum_i \|z_i - Z_{i^c} c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \quad (2.18)$$

Here z_i is the i^{th} data point, Z_{i^c} represents all the data points barring the i^{th} one and $c_i (\in \mathbb{R}^{m-1})$ corresponds to the sparse linear weights that represent samples in Z_{i^c} belonging to the same cluster as z_i . The l_1 -norm imposes sparsity. λ is a hyperparameter.

Once all the c_i 's are solved, 0's are imputed in appropriate i^{th} locations to make them vectors of length m . For example, $c_1 (\in \mathbb{R}^{m-1})$ will impute 0 in first location to become a vector of length m . Similarly, in c_2 , 0 is imputed in second location and so on. These m -length vectors are then stacked into a matrix $C_{m \times m}$.

²goel2021clustering

The affinity matrix A is computed from C as follows:

$$A = |C| + |C^\top| \quad (2.19)$$

The affinity matrix A is segmented using normalized cuts algorithm as in [45].

The joint optimization formulation of the proposed DL+SSC is expressed as follows:

$$\min_{D, Z, c} \underbrace{\|X - DZ\|_F^2}_{\text{Dictionary Learning loss}} + \underbrace{\mu \left(\sum_i \|z_i - Z_i c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \right)}_{\text{SSC loss}} \quad (2.20)$$

We set $\mu=1$ in the above formulation to give equal importance to the dictionary learning and SSC at the onset. We solve the equation 2.20 using ADMM. The updates for D remains the same as in 2.14. The update for Z is given by -

$$Z \leftarrow \min_Z \|X - DZ\|_F^2 + \|Z - ZC\|_F^2 \quad (2.21)$$

Here Z is formed by stacking the z_i 's as columns. To solve Z , we take the gradient of the expression in 2.21 and equate it to 0.

$$\nabla_Z (\|X - DZ\|_F^2 + \|Z - ZC\|_F^2) = 0 \quad (2.22)$$

$$\implies D^\top DZ + Z(I - C) - D^\top X = 0 \quad (2.23)$$

$$\implies D^\top DZ + Z(I - C) = D^\top X \quad (2.24)$$

Equation 2.24 is a form of Sylvester's equation $PZ + ZQ = R$, where $P = D^\top D$,

$Q = (I - C)$ and $R = D^\top X$. We can solve the Sylvester equation in 2.24 to get the update for Z . The last step is to update the c_i 's,

$$c_{i^k} \leftarrow \min_{c_i} \|z_i - Z_i c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \quad (2.25)$$

The equation 2.25 is solved using SPGL1³ solver.

The DL+SSC algorithm proceeds by iteratively solving for the dictionaries using 2.14, updating the representation by solving Sylvester's equation in equation 2.24 and updating the coefficients c_i 's by SPGL1 solver. As in the case of DL+K-means, 2.20 is non-convex. Hence, we can only expect to reach a local minimum; however we do not have any theoretical guarantees regarding convergence. In practice, we stop the iterations when the values of c_i 's do not change significantly with iterations. We emphasize on c_i 's since it has a direct consequence on the clustering performance.

Once the c_i 's are obtained, the actual clustering proceeds the same as in SSC, i.e. the affinity matrix is computed from the c_i 's which is then segmented using normalized cuts. The complete algorithm is shown in a succinct fashion below.

Algorithm 2: DL+SSC

- 1 Initialize: D_0, Z_0 ;
 - 2 **repeat**
 - 3 Update D_k using 2.14;
 - 4 Update Z_k by solving Sylvester's equation in 2.24;
 - 5 Solve c_i 's using SPGL1;
 - 6 **until** *Convergence is reached*;
 - 7 Compute affinity matrix $A = |C| + |C^\top|$;
 - 8 Use N-cuts [45] to segment A;
-

³<https://www.cs.ubc.ca/mpf/spgl1/index.html>

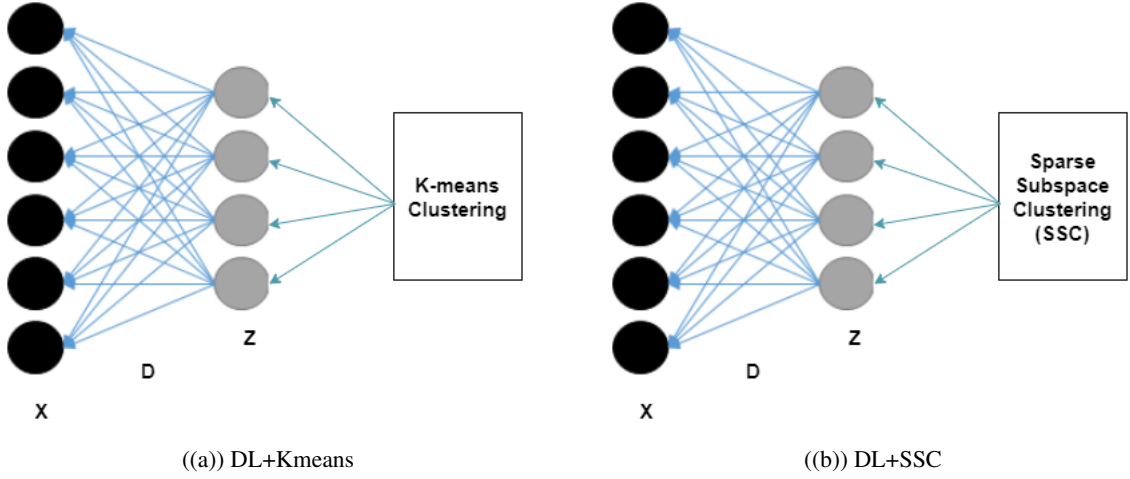


Figure 2.4: Schematic Diagram of Proposed DL based Clustering Algorithms

The schematic diagram of proposed DL based deep clustering algorithms namely DL+Kmeans and DL+SSC are shown in Figure 2.4.

2.3.3 Deep Dictionary Learning based K-means clustering

Our next proposed approach is $DDLK^4$ in which the K-means clustering loss is embedded in the Deep Dictionary Learning (DDL) framework.

In Section 2.3.1, we have seen that the K-means clustering loss can be expressed in the form of matrix factorization [43] as shown below:

$$\|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (2.26)$$

where Z is the data matrix formed by stacking z_j 's as columns and H is the

⁴Sarita Poonia, Anurag Goel, Smriti Chawla, Namrata Bhattacharya, Priyadarshini Rai, Yi Fang Lee, Yoon Sim Yap et al., "Marker-free characterization of full-length transcriptomes of single live circulating tumor cells." *Genome Research* 33, no. 1 (2023): 80-95.

matrix of binary indicator variables h_{ij} .

$$h_{ij} = \begin{cases} 1, & \text{if } x_j \in \text{cluster } i \\ 0, & \text{otherwise} \end{cases} \quad (2.27)$$

where z_j denotes the j^{th} sample and μ_i the i^{th} cluster.

In DDL, instead of learning one layer of dictionary, multiple layers are learned instead. In our proposed formulation, we have used three layers of dictionaries, which is expressed as follows:

$$X = D_1(D_2(D_3Z)) \quad (2.28)$$

Here, D_1, D_2, D_3 are three layers of dictionaries. ReLU type non-linearity is introduced between two consecutive layers of dictionaries. The DDL formulation is expressed as follows:

$$\min_{D_1, D_2, D_3, Z} \|X - D_1(D_2(D_3Z))\|_F^2 \quad (2.29)$$

In this work, ReLU activation function is used between two layers of dictionaries for two reasons – 1. It is easier to incorporate as an optimization constraint, and 2. ReLU has better function approximation capabilities. Therefore, our basic framework for DDL (with ReLU) will be expressed as follows:

$$\min_{D_1, D_2, D_3, Z} \|X - D_1 D_2 D_3 Z\|_F^2 \text{ s.t. } \underbrace{D_2 D_3 Z \geq 0, D_3 Z \geq 0, Z \geq 0}_{\text{ReLU activation}} \quad (2.30)$$

In our proposed formulation DDLK, we incorporate the K-means clustering loss into the DDL formulation. The basic idea is to use the features generated by DDL as inputs for clustering. However, instead of solving it in piecemeal fashion, we jointly optimize the proposed cost function shown below.

$$\begin{aligned}
\min_{D_1, D_2, D_3, Z, H} & \underbrace{\|X - D_1 D_2 D_3 Z\|_F^2}_{\text{DDL loss}} + \mu \underbrace{\|Z - Z H^\top (H H^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \\
s.t. & \underbrace{D_2 D_3 Z \geq 0, D_3 Z \geq 0, Z \geq 0}_{\text{ReLU activation}}
\end{aligned} \tag{2.31}$$

In 2.31, μ controls the relative importance of the dictionary learning and K-means clustering loss. To give equal importance to both the loss functions, we keep $\mu=1$. So, the final proposed formulation is as below:

$$\begin{aligned}
\min_{D_1, D_2, D_3, Z, H} & \underbrace{\|X - D_1 D_2 D_3 Z\|_F^2}_{\text{DDL loss}} + \underbrace{\|Z - Z H^\top (H H^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \\
s.t. & \underbrace{D_2 D_3 Z \geq 0, D_3 Z \geq 0, Z \geq 0}_{\text{ReLU activation}}
\end{aligned} \tag{2.32}$$

The equation 2.32 can be solved via ADMM method i.e. by updating each variable assuming the others to be constant. This leads to the following sub-problems:

$$D_1 \leftarrow \min_{D_1} \|X - D_1 D_2 D_3 Z\|_F^2 \tag{2.33}$$

$$D_2 \leftarrow \min_{D_2} \|X - D_1 D_2 D_3 Z\|_F^2 \tag{2.34}$$

$$D_3 \leftarrow \min_{D_3} \|X - D_1 D_2 D_3 Z\|_F^2 \tag{2.35}$$

$$Z \leftarrow \min_Z \|X - D_1 D_2 D_3 Z\|_F^2 + \|Z - Z H^\top (H H^\top)^{-1} H\|_F^2 \quad (2.36)$$

$$\min_H \|Z - Z H^\top (H H^\top)^{-1} H\|_F^2 \quad (2.37)$$

Initially, we ignore the non-negativity constraints in equation 2.32; later on, we will discuss how they can be handled. The updates for different variables are as follows:

$$D_1^k = X Z_1^\dagger, \text{ where } Z_1 = D_2^{k-1} D_3 Z^{k-1} \quad (2.38)$$

$$D_2^k = (D_1^k)^\dagger X Z_2, \text{ where } Z_2 = D_3^{k-1} Z^{k-1} \quad (2.39)$$

$$D_3^k = (D_1^k D_2^k)^\dagger X (Z_{k-1})^\dagger \quad (2.40)$$

where k is the iteration number and \dagger represents the pseudo-inverse.

To solve Z , we have to take the gradient of the expression in 2.36 and equate it to 0

$$\nabla_Z (\|X - D_1 D_2 D_3 Z\|_F^2 + \|Z - Z H^\top (H H^\top)^{-1} H\|_F^2) = 0 \quad (2.41)$$

$$\implies (D_1 D_2 D_3)^\top X - (D_1 D_2 D_3)^\top (D_1 D_2 D_3) Z - Z (I - H^\top (H H^\top)^{-1} H) = 0 \quad (2.42)$$

$$\implies (D_1 D_2 D_3)^\top (D_1 D_2 D_3) Z + Z (I - H^\top (H H^\top)^{-1} H) = (D_1 D_2 D_3)^\top X \quad (2.43)$$

Equation 2.43 is a form of Sylvester's equation $AZ + ZB = C$, where $A =$

$(D_1 D_2 D_3)^\top (D_1 D_2 D_3)$, $\mathbf{B} = (I - H^\top (H H^\top)^{-1} H)$ and $\mathbf{C} = (D_1 D_2 D_3)^\top X$. We can solve the Sylvester equation in 2.43 to get the update for Z .

The final step is to update H by solving equation 2.37. To get the updated H , we have to simply apply K-means clustering on the updated Z .

In the derivation so far, we have not accounted for the ReLU non-negativity constraints. Ideally imposing the constraints would require solving them via forward-backward type splitting algorithms; such algorithms are iterative and hence would increase the run-time of the algorithm. We account for these constraints by simply putting the negative values in Z , $D_3 Z$ and $D_2 D_3 Z$ to zeroes in every iteration.

The algorithm is shown in a succinct fashion below. Once the convergence is reached, the clusters can be found from H . Since 2.31 is a non-convex function, we do not have any guarantees for convergence. We stop the iterations when the H does not change significantly in subsequent iterations.

Algorithm 3: DDLK

- 1 Initialize: $D_1^0, D_2^0, D_3^0, Z_0, H_0$;
 - 2 **repeat**
 - 3 Update D_1^k, D_2^k, D_3^k using (2.38), (2.39), (2.40);
 - 4 Set the negative coefficients in D_1^k, D_2^k, D_3^k to 0;
 - 5 Update Z_k by solving Sylvester's equation in (2.43);
 - 6 Update H_k by applying K-means clustering on the updated Z ;
 - 7 **until** *Convergence is reached*;
-

2.3.4 Deep Dictionary Learning based Sparse Subspace clustering

As an extension to the proposed approach discussed in Section 2.3.2, in this proposed approach, we have incorporated the Sparse Subspace clustering loss in the Deep Dictionary Learning (DDL) formulation. DDL non-linearly transforms the data such that the transformed representation (of the data) is separable into subspaces.

The DDL formulation remains same as we discussed in Section 2.3.3, formulated in equation 2.30. DDL is mathematically flexible which allowed us to integrate SSC into it. The SSC formulation is given by -

$$\sum_i \|z_i - Z_{i^c} c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \quad (2.44)$$

Here z_i is the i^{th} data point, Z_{i^c} represents all the data points barring the i^{th} one and $c_i (\in \mathbb{R}^{m-1})$ corresponds to the sparse linear weights that represent samples in Z_{i^c} belonging to the same cluster as z_i . The l_1 -norm imposes sparsity. Once all the c_i 's are solved, 0's are imputed in appropriate i^{th} locations to make them vectors of length m ; the vectors are then stacked into a matrix $C_{m \times m}$. The affinity matrix is computed from C using,

$$A = |C| + |C^T| \quad (2.45)$$

The affinity matrix is segmented using normalized cuts algorithm as in [45]. Our proposed formulation $DDLS^5$ embeds the SSC loss in DDL framework leading

⁵Anurag Goel and Angshul Majumdar, "Sparse subspace clustering friendly deep dictionary learning for hyperspectral image classification", IEEE Geoscience and Remote Sensing Letters 19 (2021): 1-5.

to the following joint optimization formulation,

$$\begin{aligned}
\min_{D_1, D_2, D_3, Z, c} & \underbrace{\|X - D_1 D_2 D_3 Z\|_F^2}_{\text{DDL}} + \underbrace{\mu \left(\sum_i \|z_i - Z_{ic} c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \right)}_{\text{SSC loss}} \\
s.t. & \underbrace{D_2 D_3 Z \geq 0, D_3 Z \geq 0, Z \geq 0}_{\text{ReLU activation}}
\end{aligned} \tag{2.46}$$

We set $\mu=1$ in the above formulation to give equal importance to DDL and SSC loss at the onset. We solve the equation 2.46 using ADMM.

The updates for D_1, D_2 and D_3 remains the same as in equations 2.38, 2.39 and 2.40 respectively. The update for Z is given by -

$$Z \leftarrow \min_Z \|X - D_1 D_2 D_3 Z\|_F^2 + \|Z - ZC\|_F^2 \tag{2.47}$$

Here Z is formed by stacking z_i 's as columns. To solve Z , we take the gradient of the expression in 2.47 and equate it to 0.

$$\nabla_Z (\|X - D_1 D_2 D_3 Z\|_F^2 + \|Z - ZC\|_F^2) = 0 \tag{2.48}$$

$$\implies (D_1 D_2 D_3)^\top (D_1 D_2 D_3) Z + Z(I - C) - (D_1 D_2 D_3)^\top X = 0 \tag{2.49}$$

$$\implies (D_1 D_2 D_3)^\top (D_1 D_2 D_3) Z + Z(I - C) = (D_1 D_2 D_3)^\top X \tag{2.50}$$

Equation 2.50 is a form of Sylvester's equation $PZ + ZQ = R$, where $P = (D_1 D_2 D_3)^\top (D_1 D_2 D_3)$, $Q = (I - C)$ and $R = (D_1 D_2 D_3)^\top X$. We can solve the Sylvester equation in 2.50 to get the update for Z .

The last step is to update the c_i 's,

$$c_{i^k} \leftarrow \min_{c_i} \|z_i - Z_i c_i\|_2^2 + \lambda \|c_i\|_1, \forall i \in \{1, \dots, m\} \quad (2.51)$$

The equation 2.51 is solved using SPGL1⁶ solver.

The algorithm proceeds by iteratively solving for the dictionaries using equations 2.38, 2.39, 2.40, updating the representation by solving Sylvester's equation obtained in equation 2.50 and updating the coefficients c_i 's by SPGL1 solver.

As in the case of K-means, 2.46 is non-convex. Hence, we can only expect to reach a local minimum; however we do not have any theoretical guarantees regarding convergence. In practice, we stop the iterations when the values of c_i 's do not change significantly with iterations. We emphasize on c_i 's since it has a direct consequence on the clustering performance. Once the c_i 's are obtained, the actual clustering proceeds the same as in SSC, i.e. the affinity matrix is computed from the c_i 's which then is segmented using normalized cuts. The complete algorithm is shown in a succinct fashion below.

Algorithm 4: DDLS

- 1 Initialize: $D_1^0, D_2^0, D_3^0, Z_0, C_0$;
 - 2 **repeat**
 - 3 Update D_1^k, D_2^k, D_3^k using (2.38), (2.39), (2.40);
 - 4 Update Z_k by solving Sylvester's equation in 2.50;
 - 5 Solve c_i 's using SPGL1;
 - 6 **until** *Convergence is reached*;
 - 7 Compute affinity matrix $A = |C| + |C^\top|$;
 - 8 Use N-cuts [45] to segment A;
-

⁶<https://www.cs.ubc.ca/mpf/spgl1/index.html>

2.4 Experiments and Results

2.4.1 Computer Vision

We carry out evaluation of our proposed approaches DL+K-means and DL+SSC discussed in Section 2.3.1 and 2.3.2 respectively on three computer vision datasets namely Extended YaleB, ARFaces and COIL20. The datasets details can be referred from Section 1.3.1.

The number of dictionary atoms D in DL+K-means and DL+SSC formulation are fixed as half of the dimensionality of the input data. In DLS formulation (2.20), the value of hyperparameter λ is set as 0.1.

We have compared our proposed approaches DL+K-means and DL+SSC with four state-of-the-art deep clustering approaches namely Deep Convolutional Embedded Clustering (DCEC) [8], Deep K-Means (DKM) [5], Deep Clustering Network (DCN) [4] and Deeply Transformed Subspace Clustering (DTSC) [46].

For the experiments, since the number of clusters is known for the given datasets, two standard clustering metrics namely Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) are used to evaluate the performance of the proposed DL+K-means and DL+SSC algorithms.

The clustering results are presented in Table 2.1. In the case of AR Faces dataset, our proposed approaches demonstrate significantly superior performance compared to the benchmark methods. Specifically, the DL+SSC method achieves the highest performance, followed by DL+K-means. For COIL20 dataset, our

proposed algorithms DL+K-means and DL+SSC show slightly inferior performance compared to the benchmark methods. This outcome is noteworthy as the COIL20 dataset is relatively simpler and smaller in scale. It suggests that our proposed methods excel when faced with challenging datasets but may perform marginally less effectively on datasets with lower complexity.

For Extended YaleB dataset, our proposed algorithm DL+K-means algorithm emerges as the second-best performer, following the benchmark method DTSC [46]. The proposed DL+SSC algorithm also achieves competitive results in this context. Notably, it is evident that the benchmark methods as well as the proposed approaches exhibit relatively lower performance when compared to the results obtained on the other two datasets. This discrepancy can be attributed to the inherent challenges posed by the Extended YaleB dataset, including variations in illumination conditions, limited pose diversity, and the presence of noise. Furthermore, it is worth noting that DL+K-means exhibits superior performance when applied to the Extended YaleB dataset, whereas DL+SSC outperforms in the case of the AR Faces and COIL20 datasets. This divergence in performance suggests that K-means may excel in more challenging scenarios, whereas SSC demonstrates superior performance in datasets characterized by lower complexity.

Table 2.1: DL+K-means and DL+SSC: Clustering Results

Algorithms →	DCEC		DKM		DCN		DTSC		DL+K-means		DL+SSC	
Datasets ↓	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
ARFaces	0.26	0.02	0.45	0.04	0.45	0.05	0.46	0.15	0.56	0.23	0.64	0.26
COIL20	0.79	0.64	0.79	0.60	0.78	0.60	0.78	0.60	0.67	0.56	0.72	0.59
EYaleB	0.20	0.04	0.17	0.02	0.22	0.04	0.42	0.14	0.36	0.11	0.29	0.10

The run-times of different algorithms are shown in Table 2.2. Between DL+K-means and DL+SSC, the former is faster by almost an order of magnitude. This is because it does not require solving a computationally costly l_1 -norm minimization problem like the latter in every iteration. Our proposed algorithms are faster than all the benchmarks since DL+K-means and DL+SSC are relatively shallow techniques.

Table 2.2: DL+K-means and DL+SSC: Runtime comparison (in seconds)

Datasets ↓	DCEC	DKM	DCN	DTSC	DLK	DLS
ARFaces	1013	1548	1495	996	95	389
COIL20	959	517	921	502	29	201
EYaleB	1180	2054	2065	1252	73	564

The empirical convergence plots for the proposed techniques are given in figure 2.5. We observe that although the proposed formulations DL+K-means and DL+SSC are non-convex problems, both the algorithms converge, at least to a local minimum.

2.4.2 Hyperspectral Imaging

We carry out evaluation of our proposed approach DDLS discussed in Section 2.3.4 on two popular hyperspectral images datasets namely Indian Pines and Pavia University. The datasets details can be referred from Section 1.3.2.

For our proposed method, the depth of DDL is kept fixed at 3. The number of dictionary atoms D in DDLS formulation are fixed as half of the dimensionality of the input data. In DDLS formulation (2.46), the value of hyperparameter λ that determines the sparsity level of the linear weights is set as 1.

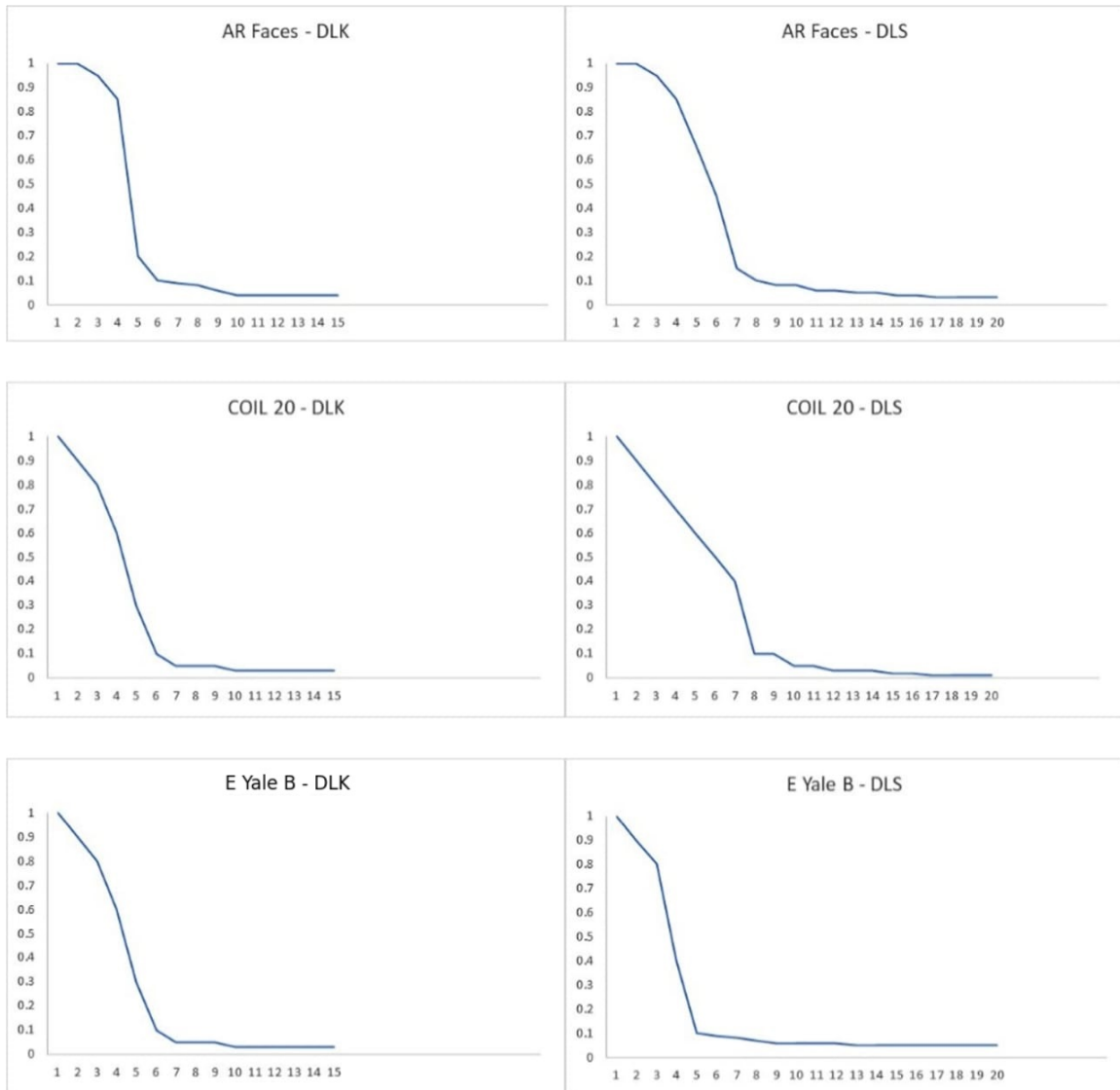


Figure 2.5: DL+K-means and DL+SSC: Empirical Convergence plots. Here, x-axis represents the number of iterations and y-axis represents the normalized cost function value

In prior studies on DDL-based hyperspectral image analysis [47, 48], it was found that the spectral–spatial features generated by taking Principal Component Analysis (PCA) [49] around the pixel of interest turn out to be a good input feature for DDL. Here we do the same. The process is schematically shown in figure 2.6. Around each pixel, a spatial window is considered; within this window, all the bands in the spectral direction are taken. Each such 3-D patch is then vectorized; these are stacked as columns of a matrix. On this matrix, PCA is run to reduce the dimensionality. We have tried three different windows of sizes 3×3 , 5×5 , and 7×7 ; for each window size, 10% of the principal components were kept. For example, in Indian Pines with a 3×3 window, the size of the input to PCA would be 1800 ($3 \times 3 \times 200$); after PCA 180 principal components will be kept.

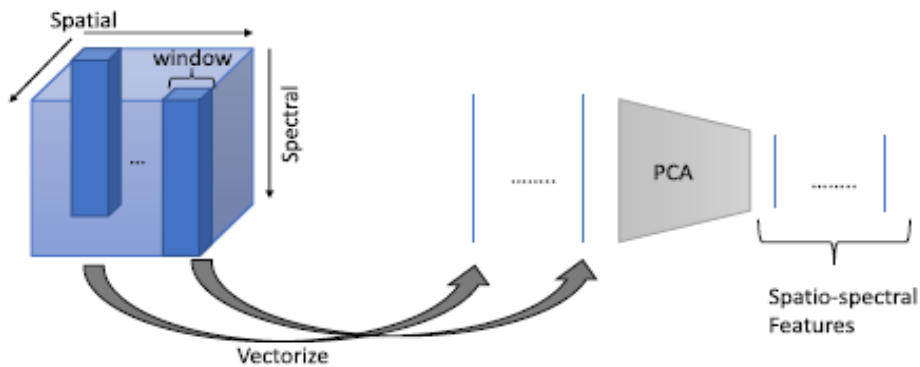


Figure 2.6: Spatio-spectral Feature Extraction

We have compared our proposed approach DDLS with four state-of-the-art deep clustering approaches namely Deep Spatial-Spectral Subspace Clustering (DS^3C) [50], Deep Clustering With Intraclass Distance (DCID) [51], Self-Supervised Deep Subspace Clustering (S^2DSC) [52], and 3-D Convolutional

Auto-encoders (3DCAEs) [53].

For the experiments, two types of evaluation metrics are considered. The first type consists of four generic metrics for clustering – Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), Purity, and Entropy. The second type consists of three metrics that are specific for hyperspectral image classification – Overall Accuracy (OA), Average Accuracy (AA), and Kappa coefficient (κ). Both types of metrics serve different purposes in evaluating clustering and classification tasks. The first type assesses the quality of clustering algorithm by comparing predicted clusters with ground truth labels, while the second type specifically evaluates the accuracy and consistency of hyperspectral image classification.

- **Normalized Mutual Information (NMI):** NMI measures the mutual information between the true labels of data points and the predicted cluster assignments, normalized by the entropy of both sets of labels. NMI takes values between 0 (no mutual information) and 1 (perfect agreement).
- **Adjusted Rand Index (ARI):** ARI quantifies the similarity between the true labels and the predicted clusters, while considering the possibility of randomness. ARI ranges from -1 (no agreement) to 1 (perfect agreement).
- **Purity:** Purity measures the proportion of data points in a cluster that belong to the majority class of that cluster. It evaluates the homogeneity of clusters with respect to class labels.
- **Entropy:** Entropy quantifies the uncertainty or disorder in the distribution

of class labels within clusters. Lower entropy indicates higher purity and better cluster quality.

- **Overall Accuracy (OA):** OA measures the percentage of correctly classified pixels in the hyperspectral image. It provides a general view of the classification performance but may not account for class imbalances.
- **Average Accuracy (AA):** AA calculates the average accuracy across all classes. It can help mitigate the impact of class imbalances and provides a more balanced assessment of the classifier’s performance.
- **Kappa Coefficient (κ):** Kappa coefficient measures the agreement between the predicted classifications and the actual classifications, while accounting for the agreement that could occur by chance. It ranges from -1 (no agreement) to 1 (perfect agreement).

The clustering results are shown in Table 2.3. Our proposed method yields the best results in terms of all metrics for window sizes 3×3 and 5×5 . For the larger window size, we perform worse than DCID. With a larger window size, the dimensionality increases, and with an increase in dimensionality, we need to learn more network weights – this possibly leads to overfitting and thus the result results deteriorate. In the benchmarks, DCID yields the best results. In terms of methodology, this is the most sophisticated technique. 3DCAE and S^2DSC perform somewhat worse than DCID. DS^3C is the simplest approach and consequently yields results that are not at par with the rest.

In the next set of experiments, we carry out ablation studies. We analyze the

Table 2.3: DDLS: Clustering Results

Datasets ↓	Metric	DS^3C	DCID	S^2DSC	3DCAE	Proposed		
						3×3	5×5	7×7
Pavia University	NMI	0.637	0.664	0.648	0.653	0.681	0.673	0.655
	ARI	0.501	0.529	0.507	0.514	0.550	0.537	0.518
	Purity	0.613	0.694	0.645	0.647	0.703	0.700	0.649
	Entropy	0.457	0.442	0.451	0.447	0.404	0.419	0.444
	OA	0.858	0.883	0.866	0.875	0.897	0.888	0.875
	AA	0.820	0.855	0.831	0.839	0.862	0.858	0.841
	Kappa	0.791	0.808	0.793	0.796	0.825	0.814	0.798
Indian Pines	NMI	0.604	0.701	0.685	0.631	0.726	0.711	0.696
	ARI	0.490	0.523	0.508	0.497	0.535	0.531	0.524
	Purity	0.607	0.658	0.633	0.637	0.664	0.660	0.651
	Entropy	0.433	0.426	0.436	0.423	0.415	0.418	0.430
	OA	0.811	0.841	0.835	0.832	0.852	0.846	0.837
	AA	0.775	0.801	0.796	0.790	0.823	0.810	0.792
	Kappa	0.749	0.772	0.770	0.762	0.790	0.783	0.762

effect of depth of dictionaries. For each depth, we see how the metrics change for the proposed joint solution and a piecemeal solution. By a piecemeal solution, we mean that the features are generated separately by DDL and the learned features are input to a separate sparse subspace classifier. The joint solution is the proposed one where DDL and clustering losses are intertwined and optimized jointly. The results are shown in Table 2.4 for the 5×5 window size.

Table 2.4: DDLS: Ablation Studies Results

Datasets ↓	Metric	One layer		Two layers		Three layers		Four layers	
		Joint	Piecemeal	Joint	Piecemeal	Joint	Piecemeal	Joint	Piecemeal
Pavia University	NMI	0.633	0.627	0.654	0.636	0.673	0.649	0.667	0.651
	ARI	0.515	0.509	0.524	0.513	0.537	0.520	0.533	0.523
	Purity	0.611	0.605	0.662	0.614	0.700	0.657	0.688	0.660
	Entropy	0.460	0.468	0.431	0.457	0.419	0.434	0.425	0.432
	OA	0.849	0.832	0.863	0.842	0.888	0.852	0.880	0.853
	AA	0.822	0.809	0.841	0.820	0.858	0.829	0.851	0.832
	Kappa	0.787	0.768	0.802	0.775	0.814	0.784	0.809	0.785
Indian Pines	NMI	0.690	0.685	0.699	0.691	0.711	0.695	0.706	0.697
	ARI	0.511	0.498	0.516	0.509	0.531	0.516	0.528	0.519
	Purity	0.636	0.630	0.651	0.636	0.660	0.650	0.655	0.651
	Entropy	0.429	0.434	0.426	0.428	0.418	0.427	0.421	0.426
	OA	0.819	0.810	0.825	0.818	0.846	0.827	0.836	0.828
	AA	0.781	0.775	0.790	0.779	0.810	0.788	0.805	0.790
	Kappa	0.760	0.751	0.768	0.758	0.783	0.766	0.777	0.767

One observes that the results improve from the dictionary layers one to three and then dip in layer four for the proposed joint formulation. This is expected; in deep learning, one can improve the results by going deeper; however, one cannot go arbitrarily deep since the number of parameters increases with depth. With limited training data, this leads to overfitting and one sees deterioration in results. For every number of dictionary layers, one can see that the joint formulation yields better results than the piecemeal one. This is also expected; the proposed formulation learns projections that are clustering friendly, but this is not the case for the piecemeal formulation. Overall one can notice that the results from Pavia University are always better than that of Indian Pines. The spatial resolution of Indian Pines dataset is relatively coarse. The image pixels are larger, which means that individual objects or features in the scene might not be well-defined, and there can be mixing of information from different land cover types within a single pixel. On the other hand, Pavia University dataset has higher spatial resolution compared to the Indian Pines dataset. This means that individual objects or features can be more clearly delineated in the imagery due to smaller pixel sizes. Therefore, the Pavia University dataset is giving better performance as compared to the Indian Pines dataset. The Indian Pines dataset has a spatial size of 145 x 145 pixels, which translates to a total of 21,025 pixels (samples). On the other hand, the Pavia University dataset has a spatial size of 610 x 340 pixels, resulting in 207,400 pixels (samples). Therefore, Pavia University has more number of pixels due to its larger spatial dimensions which may also be a factor for the better performance of Pavia University dataset as compared to

Indian Pines dataset.

Our proposed DDLS formulation is non-convex and hence, we do not have any convergence guarantee. However, we find that in practice the algorithm converges. The convergence plot for three-layers of dictionaries and 5×5 window size is shown in figure 2.7.

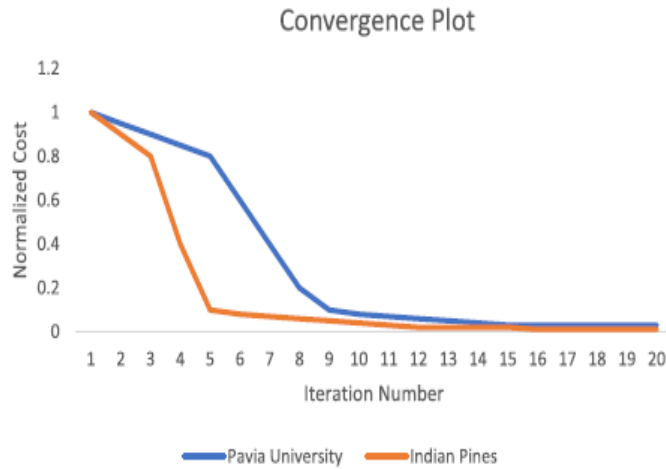


Figure 2.7: DDLS: Empirical Convergence plot

The run times for different algorithms are shown in Table 2.5. The results show that the 3DCAE is the fastest. This is because the said algorithm uses K-means clustering internally; K-means is faster than subspace clustering which all other algorithms use. Our proposed technique (with three layers of dictionaries) is slower than 3DCAE but is faster than the rest. Our proposed algorithm for one layer and two layers of dictionaries is faster than 3DCAE but does not yield the best results at these depths.

Table 2.5: DDLS: Runtime Comparison (in seconds)

Datasets →	Pavia University	Indian Pines
Algorithms ↓		
DS³C	1308	1221
DCID	1276	1195
S²DSC	1093	1115
3DCAE	593	320
Proposed (1 layer)	482	303
Proposed (2 layers)	507	346
Proposed (3 layers)	691	406
Proposed (4 layers)	984	793

2.5 Summary

This chapter proposes four novel deep clustering approaches based on dictionary learning framework namely Dictionary Learning based K-means clustering (DL+K-means), Dictionary Learning based Sparse Subspace Clustering (DL+SSC), Deep Dictionary Learning based K-means clustering (DDLK) and Deep Dictionary Learning based Sparse Subspace Clustering (DDLS). DL+K-means and DL+SSC embeds K-means clustering and Sparse Subspace Clustering in dictionary learning framework respectively and optimize the joint formulation using Alternative Directed Multipliers Method (ADMM). DDLK and DDLS embeds K-means clustering and Sparse Subspace Clustering in deep dictionary learning framework (with three layers of dictionaries) respectively and optimize the joint formulation via ADMM. Experiments have been carried out on two different kinds of problems: computer vision and hyperspectral imaging. DL+K-means and DL+SSC are evaluated on three computer vision datasets while DDLS is evaluated on hyperspectral imaging datasets. In both the set of experiments, the proposed approaches outperform state-of-the-art techniques.

Chapter 3

Transform Learning based Clustering

Approaches

In the previous chapter, we have seen the clustering approaches based on dictionary learning. In this chapter, two novel deep clustering approaches based on Transform Learning (TL) framework are proposed. The conducted experiments, which encompassed two distinct problem domains, namely text datasets and hyperspectral imaging, yielded results demonstrating the superior performance of the proposed methodologies over existing state-of-the-art techniques.

3.1 Transform Learning

Transform Learning (TL) is the analysis equivalent of dictionary learning. It learns an analysis dictionary / transform (T) such that it operates on the data (X) to generate the coefficients (Z) (see figure 3.1). The TL problem can be

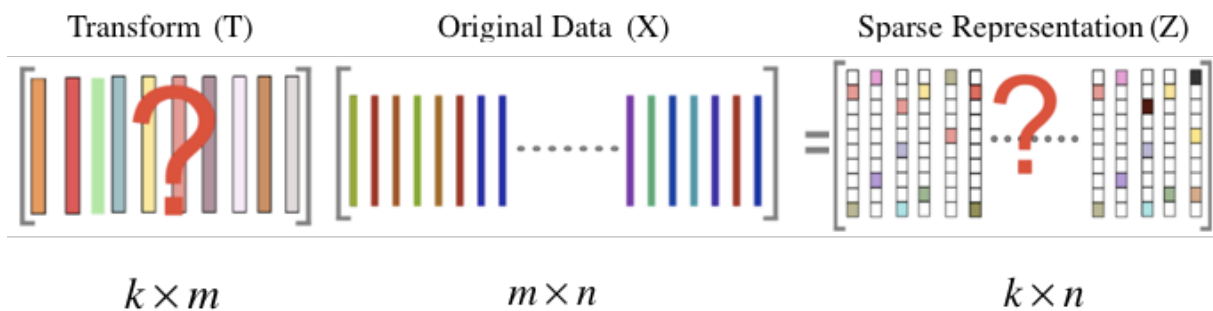


Figure 3.1: Transform learning (TL)

expressed as follows

$$TX = Z \quad (3.1)$$

Suppose we are given the data $X \in \mathbb{R}^{m \times n}$ such that each column x_i represents a training sample of size m , thus we have total n training samples. Mathematically this is represented as,

$$\min_{T \in \mathbb{R}^{k \times m}, Z \in \mathbb{R}^{k \times n}} \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{2} \|Tx_i - z_i\|_2^2 + \mu \|z_i\|_1 \right) \quad (3.2)$$

Here, z_i is learned coefficient corresponding to the training sample x_i , the term $\|z_i\|_1$ is to promote the sparsity on the learned coefficients and μ is the regularization parameter on the sparsity term.

The main motivation that can be carried from TL is to use it beyond signal processing. TL has not been used for solving machine learning problems. We explore if TL features can be general enough to solve machine learning problems, and we have computational cost and run time advantages.

Figure 3.2 shows neural network interpretation of TL where transform T is considered as an operator being applied on the input data to generate feature

representation. In other words, transform T can be considered as weights of the neural network interpretation which are being learned along with the representative features from the given data in the input layer. Please note that it is not a true neural network but its interpretation.

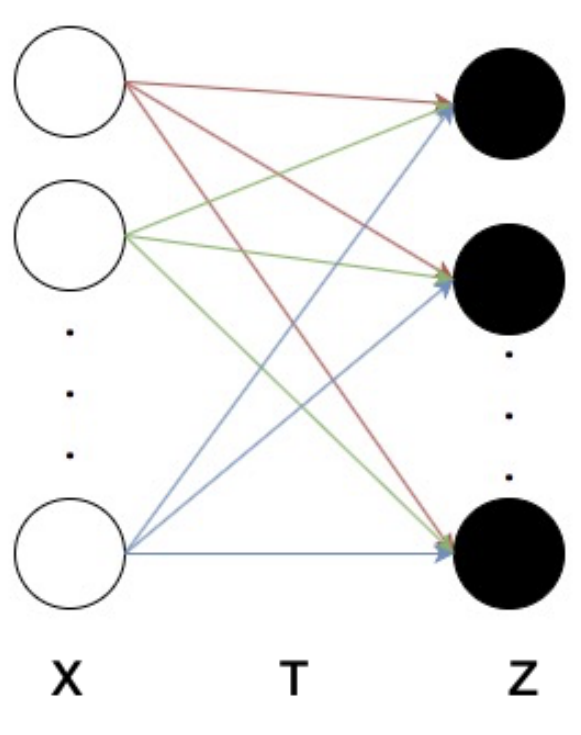


Figure 3.2: Neural network interpretation of TL

One may be enticed to solve the TL problem by formulating,

$$\min_{T,Z} \|TX - Z\|_F^2 + \mu \|Z\|_1 \quad (3.3)$$

Unfortunately such a formulation would lead to degenerate solutions; it is easy to verify the trivial solution $T = 0$ and $Z = 0$. In order to ameliorate this the following formulation was proposed in [54].

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda (\|T\|_F^2 - \log \det(T)) + \mu \|Z\|_1 \quad (3.4)$$

The factor $-\log \det$ ¹ imposes a full rank on the learned transform; this prevents the degenerate solution ($T = 0, Z = 0$). The additional penalty $\|T\|_F^2$ is to balance the scale; without this $-\log \det(T)$ can keep on increasing; producing degenerate results in the other extreme. Note that the sparsity constraint on the coefficients is not mandatory for machine learning problems. It is useful for solving inverse problems in signal processing. In [55], an alternating minimization approach was proposed to solve the TL problem equation 3.4 which leads to the following sub-problems

$$T \leftarrow \min_T \|TX - Z\|_F^2 + \lambda(\|T\|_F^2 - \log \det(T)) \quad (3.5)$$

$$Z \leftarrow \min_Z \|TX - Z\|_F^2 + \mu\|Z\|_1 \quad (3.6)$$

Updating the coefficients (Z) is straightforward. It can be updated via one step of hard thresholding [56], [57]. This is expressed as,

$$^2Z \leftarrow (abs(TX) \geq \mu) \otimes TX \quad (3.7)$$

Here \otimes represents element-wise product.

For updating the transform, one can notice that the gradients for different terms in equation 3.5 are easy to compute. Ignoring the constants, this is given

¹ $\log \det(T) = \log(\text{singular values})$. If some singular value ≤ 0 , then the log takes $+\infty$ as output. For the case when T is not square, the algorithm solves $-\log \det(T'T) + \|T\|_F^2$.

²We take absolute of each entry of the matrix and see if any entry of matrix is greater than μ

by:

$$\begin{aligned}
\nabla \|TX - Z\|_F^2 &= X^T(TX - Z) \\
\nabla \|T\|_F^2 &= T \\
\nabla \log \det T &= T^{-T}
\end{aligned} \tag{3.8}$$

In the initial paper on transform learning [55], a non-linear conjugate gradient based technique was proposed to solve the transform update. In the second paper [58], with some linear algebraic tricks they were able to show that a closed form update exists for the transform. This is given by:

$$\begin{aligned}
XX^T + \lambda I &= LL^T \\
L^{-1}YX^T &= QSR^T \\
T &= 0.5R(S + (S^2 + 2\lambda I)^{1/2})Q^T L^{-1}
\end{aligned} \tag{3.9}$$

The first step is to compute the Cholesky decomposition; the decomposition exists since $XX^T + \lambda I$ is symmetric positive definite. The next step is to compute the full Singular Value Decomposition (SVD). The final step is the update step. One must notice that L^{-1} is easy to compute since it is a lower triangular matrix. The cost function is monotonic, decreasing in each step. Moreover, since it is lower bounded, it converges, and its closed-form solution exists.

3.2 Deep Transform Learning

In Deep Transform Learning (DTL), multiple levels of transforms are learned to represent the data in terms of coefficients [59]. The layers of single level transforms have been stacked one after the other leading to the framework of

DTL; DTL is indeed a more powerful tool than the conventional Transform Learning.

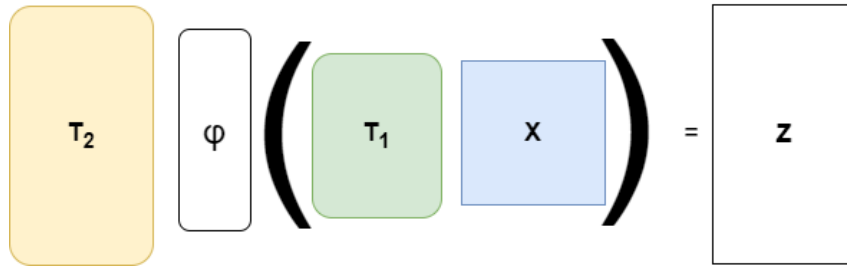


Figure 3.3: Schematic diagram: Deep Transform learning

Deeper representations are learned by stacking one transform after another. The learning is done greedily. The first layer determines the transform and features from the input training samples. The subsequent layers use the features (after activation) from the previous layer as training input as shown in figure 3.3.

For DTL, instead of analyzing the data by single-level transform, multiple levels of transforms are used to produce the final level of coefficients. This is expressed as,

$$T_N \phi (\dots (T_2 \phi (T_1 X))) = Z \quad (3.10)$$

Here T_1 operates on the data X to produce the first level of coefficients. T_2 analyzes the first level of the coefficients to produce the second level. Finally, T_N operates the $(n-1)$ level of coefficients to generate final coefficients Z . ϕ represents the non-linear activation function without which all the transforms will collapse into the single one. The corresponding optimization problem is

expressed as follows:

$$\min_{T_1, T_2, \dots, T_N, Z} \|T_N \phi(\dots(T_2 \phi(T_1 X))) - Z\|_F^2 + \lambda(\|T\|_F^2 - \log \det(T)) + \mu \|Z\|_1 \quad (3.11)$$

Here, the factor $-\log \det(T)^3$ imposes a full rank on the learned transform; this prevents the degenerate solution ($T = 0, Z = 0$). The additional penalty $\|T\|_F^2$ is to balance the scale; without this $-\log \det(T)$ can keep on increasing; producing degenerate results in the other extreme. $\|Z\|_1$ is the sparsity promoting term on the learned coefficients and λ, μ are the regularization parameters.

3.3 Literature Review

Over the years there have been a handful of studies on transform learning based data analysis. In recent years, several variants of transform learning like Kernelized transform learning [60] and Robust transform learning [61] were proposed. In Kernelized transform learning [60], a non-linear version of the data is represented in terms of a transform made up of linear combination of non-linear version of itself. In Kernelized transform learning, a kernel defined by $K = \phi(X)^T \phi(X)$, where ϕ is a non-linear activation function, is used as the data matrix X in the original transform learning formulation; rest all remains the same as in the conventional transform learning. In Robust transform learning [61], a robust version of transform learning is proposed by replacing the Frobenius norm by $l1 - norm$ in transform learning formulation (equation 3.4). Using transform

³ $\log \det(T) = \log(\text{singular values})$. If some singular value ≤ 0 , then the log takes $+\infty$ as output. For the case when T is not square, the algorithm solves $-\log \det(T^T T) + \|T\|_F^2$.

learning as the building block, deeper versions have also been proposed for unsupervised learning [59, 62]. In Greedy deep transform learning [62], the learning proceeds in a greedy fashion. The first layer learns the transform and features from the input training samples while the subsequent layers of transform use the features (after activation) from the previous layers as training input. In Unsupervised deep transform learning [59], the deep architecture of transform learning are learned in a joint end-to-end fashion.

Recently, transform learning based methods are proposed for clustering [46, 63–65]. In [64], the subspace clustering [66] formulations are incorporated into the transform learning framework. The proposed formulation was solved in a joint fashion using the Alternating Direction Method of Multipliers (ADMM). The kernelized version of the aforesaid formulation is also proposed in [64], where the subspace clustering formulations are embedded in kernelized transform learning [60]. A deeper version of Transformed Subspace Clustering [64] is proposed in [46]. In [46], the subspace clustering formulations are incorporated in deep transform learning where three layers of transforms are stacked one over the other and ReLU type non-linearity is introduced in between the transform layers. The results in [46] demonstrate that the deep version of transform learning is more effective as compared to the single layer transform learning framework. However, one cannot go deep to any number of transform layers for better performance since the chances of overfitting increase with increase in the number of transform layers.

3.4 Proposed Approaches

3.4.1 Transformed K-means Clustering

In this work *TL+K-means*⁴, we propose to embed the K-means clustering into the transform learning framework. The basic idea remains the same as in [64]. The learnt representation from transform learning is input for K-means clustering. The K-means embedded transform learning is solved as a single optimization problem.

Transform learning analyses the data by learning a transform/basis to produce coefficients. Mathematically this is expressed as,

$$TX = Z \quad (3.12)$$

Here T is the transform, X is the data and Z the corresponding coefficients. The transform learning formulation is expressed as:

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda(\|T\|_F^2 - \log \det T) + \mu \|Z\|_1 \quad (3.13)$$

The popular way to express K-means clustering is via the following formulation:

$$\sum_{i=1}^k \sum_{j=1}^n h_{ij} \|z_j - \mu_i\|_2^2 \quad (3.14)$$

⁴Anurag Goel and Angshul Majumdar, "Transformed K-means Clustering", In 2021 29th European Signal Processing Conference (EUSIPCO), pp. 1526-1530. IEEE, 2021.

$$h_{ij} = \begin{cases} 1, & \text{if } z_j \in \text{cluster } i \\ 0, & \text{otherwise} \end{cases} \quad (3.15)$$

where z_j denotes the j^{th} sample and μ_i the i^{th} cluster. In [43], it was shown that the K-means loss shown in equation 3.14 can be alternatively represented in the form of matrix factorization as below:

$$\|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (3.16)$$

where Z is the data matrix formed by stacking z_j 's as columns and H is the matrix of binary indicator variables h_{ij} .

In our proposed formulation of TL+K-means, the general idea is to use the coefficients generated by transform learning as inputs to K-means clustering. This is achieved by incorporating the K-means clustering loss into the transform learning formulation as shown below -

$$\min_{T,Z,H} \underbrace{\|TX - Z\|_F^2}_{\text{Transform Learning loss}} + \mu \underbrace{\|Z - ZH^T(HH^T)^{-1}H\|_F^2}_{\text{K-Means loss}} \quad (3.17)$$

In the equation 3.17, transform learning formulation is regularized by the K-means clustering loss in a single joint optimization problem. Note that, compared to the original formulation of transform learning [58], we have dropped the sparsity promoting term Z . This is because transform learning was originally intended to solve inverse problems so sparsity on the coefficients was necessary [67]. However, for our purpose, the l1-norm on the coefficients do not carry any

particular meaning apart from a regularization term. Hence we have dropped it. Note that the same has been done in other transform learning based formulations for machine learning [59–62, 64, 68–71].

The solution to equation 3.17 can be achieved via the alternating direction method of multipliers (ADMM) [72], i.e. in every iteration each variable is updated by assuming the other variables to be constant. This leads to the following three subproblems –

$$T \leftarrow \min_T \|TX - Z\|_F^2 + \lambda(\|T\|_F^2 - \log \det T) \quad (3.18)$$

$$Z \leftarrow \min_Z \|TX - Z\|_F^2 + \mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (3.19)$$

$$H \leftarrow \min_H \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (3.20)$$

The update for equation 3.18 is given by 3.9 in Section 3.1. The subproblem 3.19 can be simplified to the following subproblem -

$$Z \leftarrow \min_Z \|TX - Z\|_F^2 + \mu \|ZK\|_F^2; K = I - H^\top (HH^\top)^{-1} H \quad (3.21)$$

Taking the derivate and equating it to 0,

$$\begin{aligned} \nabla_Z (\|TX - Z\|_F^2 + \mu \|ZK\|_F^2) &= 0 \\ \implies TX &= Z(I + \mu K) \\ \implies Z &= TX(I + \mu K)^{-1} \end{aligned} \quad (3.22)$$

This concludes the closed form update for Z. The solution for H is straightforward; it can be obtained by applying K-means clustering on the updated

Z.

This concludes the derivation of our proposed TL+K-means formulation. The complexity of updating T is dependent on the Cholesky and singular value decompositions; both of which have a complexity of $O(n^3)$. The update for Z has a complexity of $O(n^2)$. The K-means is ideally a NP-hard problem, but the algorithm used here has a complexity of $O(t * k * n^2)$ where t is the number of loops and k is the number of clusters.

The schematic diagram of proposed TL+Kmeans algorithm is shown in Figure 3.4.

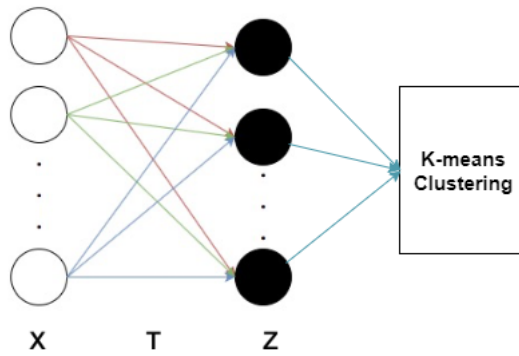


Figure 3.4: Schematic Diagram of Proposed TL+Kmeans Algorithm

3.4.2 Deeply Transformed K-means Clustering

We have already discussed the deep version of Transform Learning i.e. Deep Transform Learning in Section 3.2. In this proposed approach⁵, we have embedded K-means clustering loss in the Deep Transform Learning (DTL) formulation.

As discussed in Section 3.2, for DTL, instead of analyzing the data by single-

⁵Anurag Goel and Angshul Majumdar, "K-Means Embedded Deep Transform Learning for Hyperspectral Band Selection", IEEE Geoscience and Remote Sensing Letters 19 (2022): 1-5.

level transform, multiple levels of transforms are used to produce the final level of coefficients. For example, the DTL formulation with three layers of transforms can be expressed as,

$$T_3(T_2(T_1X)) = Z \quad (3.23)$$

Here T_1 operates on the data X to produce the first level of coefficients. T_2 analyzes the first level of the coefficient to produce the second level. Finally, T_N operates the (n-1) level of coefficients to generate Z . The ReLU type non-linearity is introduced between the transforms layers without which all the transforms will collapse into the single one. The formulation of DTL is given by

$$\begin{aligned} \min_{T_3, T_2, T_1, Z} & \|T_3T_2T_1X - Z\|_F^2 + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i) \\ & s.t. T_2T_1X \geq 0, T_1X \geq 0, Z \geq 0 \end{aligned} \quad (3.24)$$

Here T_1, T_2, T_3 are three layers of transforms. In equation 3.24, the term $\|T_i\|_F^2 - \log \det T_i$ is to prevent the trivial solutions and degenerate solutions to the transforms [73]. The non-negativity constraints in equation 3.24 translate to Rectified Linear Unit (ReLU) activations between the layers. The reason for using ReLU is two-fold. First, it is easy to accommodate in the optimization framework. Second, it is known for its function approximation capability [74].

The K-means clustering loss remains same as in equation 3.16. In our proposed formulation, the input to the K-means clustering is the representation from deep transform learning. Incorporating the K-means clustering loss into the DTL

formulation leads to

$$\begin{aligned}
& \min_{T_1, T_2, T_3, Z, H} \underbrace{\|T_3 T_2 T_1 X - Z\|_F^2 + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i)}_{\text{Transform Learning loss}} + \\
& \underbrace{\mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \quad (3.25) \\
& \underbrace{s.t. T_2 T_1 X \geq 0, T_1 X \geq 0, Z \geq 0}_{\text{ReLU constraints}}
\end{aligned}$$

To solve the formulation in equation 3.25, we introduce two proxies $T_1 X = X_2$ and $T_2 T_1 X = X_3$. The resulting augmented Lagrangian becomes

$$\begin{aligned}
& \min_{T_1, T_2, T_3, X_2, X_3, Z, H} \|T_3 X_3 - Z\|_F^2 + \gamma_1 \|T_2 X_2 - X_3\|_F^2 + \gamma_2 \|T_1 X - X_2\|_F^2 \\
& + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i) + \mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (3.26) \\
& s.t. X_3 \geq 0, X_2 \geq 0, Z \geq 0
\end{aligned}$$

The hyperparameters γ_1 and γ_2 are usually different, but in the given scenario, they represent the relative significance of different layers. We do not see any reason to give more significance to one layer over others. Therefore, we assign equal importance; hence, we keep $\gamma_1 = \gamma_2 = 1$. This leads to

$$\begin{aligned}
& \min_{T_1, T_2, T_3, X_2, X_3, Z, H} \|T_3 X_3 - Z\|_F^2 + \|T_2 X_2 - X_3\|_F^2 + \|T_1 X - X_2\|_F^2 \\
& + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i) + \mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (3.27) \\
& s.t. X_3 \geq 0, X_2 \geq 0, Z \geq 0
\end{aligned}$$

We solve the formulation in equation 3.27 using the alternating direction method

of multipliers (ADMM) [72]. Using ADMM, 3.27 can be segregated into the following subproblems:

$$P_1 : \min_{T_3} \|T_3 X_3 - Z\|_F^2 + \lambda(\|T_3\|_F^2 - \log \det T_3) \quad (3.28)$$

$$P_2 : \min_{T_2} \|T_2 X_2 - X_3\|_F^2 + \lambda(\|T_2\|_F^2 - \log \det T_2) \quad (3.29)$$

$$P_3 : \min_{T_1} \|T_1 X - X_2\|_F^2 + \lambda(\|T_1\|_F^2 - \log \det T_1) \quad (3.30)$$

$$P_4 : \min_{X_3} \|T_3 X_3 - Z\|_F^2 + \|T_2 X_2 - X_3\|_F^2 \text{ s.t. } X_3 \geq 0 \quad (3.31)$$

$$P_5 : \min_{X_2} \|T_2 X_2 - X_3\|_F^2 + \|T_1 X - X_2\|_F^2 \text{ s.t. } X_2 \geq 0 \quad (3.32)$$

$$P_6 : \min_Z \|T_3 X_3 - Z\|_F^2 + \mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \text{ s.t. } Z \geq 0 \quad (3.33)$$

$$P_7 : \min_H \mu \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \quad (3.34)$$

Solutions to $P_1 - P_3$ are the standard transform update given in [55]. It has a closed-form solution

$$\min_T \|TX - Z\|_F^2 + \lambda(\|T\|_F^2 - \log \det T) \quad (3.35)$$

$$XX^\top + \lambda I = LL^\top \quad (\text{Cholesky Decomposition})$$

$$L^{-1}XZ^\top = USV^\top \quad (\text{Singular Value Decomposition}) \quad (3.36)$$

$$T = 0.5U(S + (S^2 + 2\lambda I)^{1/2})V^\top L^{-1}$$

Solutions to sub-problems P_4 and P_5 are akin to the Tikhonov regularization.

The solution can be obtained as follows:

$$\begin{aligned} \min_X \|Z - TX\|_F^2 + \|X - C\|_F^2, T_2X_2 \text{ or } T_1X = C = \text{constant} \\ \implies \hat{X} = (T^\top T + I)^{-1}(T^\top Z + C) \end{aligned} \quad (3.37)$$

Here, we have not specifically accounted for the positivity constraint. This is because the exact solution would require forward–backward-type splitting. We prefer the approximate solution where the aforesaid closed-form solution is thresholded to keep only non-negative values. The same approach was followed in [46].

The solution to P_6 is obtained by taking the derivative of it and equating it to zero

$$\begin{aligned} \nabla_Z(\|T_3X_3 - Z\|_F^2 + \mu\|ZK\|_F^2) = 0, \text{ where } (I - (HH^\top)^{-1}H) = K \\ \implies T_3X_3 = Z(I + \mu K) \\ \implies Z = T_3X_3(I + \mu K)^{-1} \end{aligned} \quad (3.38)$$

In the solution of Z , we follow the same approach as before. We do not explicitly incorporate the non-negativity constraint but put the negative values in it to zero once it is solved in every iteration.

The solution to P_7 is applying K-means clustering on the updated Z .

This concludes the derivation of our proposed algorithm. The complexity of updating the transforms is dependent on the Cholesky decomposition and Singular value decomposition; both of which have a complexity of $O(n^3)$. The update for the coefficients has a complexity of $O(n^2)$. The K-means is ideally a

NP-hard problem, but the algorithm used here has a complexity of $O(t * k * n^2)$ where t is the number of loops and k is the number of clusters.

Application to Hyperspectral Band Selection

We have applied our proposed Deep Transform learning based K-means clustering formulation for Hyperspectral band selection. Most of the clustering-based band selection techniques work with individual band images, i.e., with x_i where i denotes the band. Some kind of a representative feature (z_i) is learned from x_i . This representation is passed into a clustering algorithm where the number of clusters is specified (this is the same as the number of bands to be chosen). The clustering algorithm outputs the cluster centroid. The band images that are closest to the cluster centroid in some sense (normed distance, cosine similarity, and so on) are finally selected.

We do not follow this approach for two reasons:

1. trying to learn deep transforms from a few band images (\sim hundreds) will lead to overfitting [62];
2. clustering from a few samples (\sim hundreds) will not lead to robust cluster centroids.

Instead, we divide the entire hyperspectral datacube into spatial patches $P_{x_{ik}}$ that denotes the k th patch of the i th band image. These patches are input to our proposed deeply transformed learning based K-means clustering (DTKM)

algorithm. This requires solving the following problem:

$$\begin{aligned}
\min_{T_1, T_2, T_3, Z, H} \sum_k & ((\|T_3 T_2 T_1 P_{x_{ik}} - z_{ik}\|_F^2) + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i) + \\
& \mu \|Z - Z H^\top (H H^\top)^{-1} H\|_F^2) \quad (3.39) \\
s.t. & T_2 T_1 X \geq 0, T_1 X \geq 0, Z \geq 0
\end{aligned}$$

The formulation states that K-means clustering is carried out on individual patches (instead of the images) and the transforms are globally learned on all the patches. By learning on a vast number of patches, the problem of overfitting is resolved. Similarly, as the clustering is run patchwise, robustness is ensured following a simple table lookup strategy. For each patch, we will get a band that is closest to the j th cluster in the Euclidean sense. We see which band is appearing to be the closest in the majority of patches and select the same. A toy example is given in Table 3.1. We consider an image with eight patches and want to select five bands. The table entries represent the band number. The cluster center corresponding to each patch is shown in Table 3.1. The final selection, considering all the patches, is obtained from the statistical mode of the column (shown in bold).

One can argue that an identifiability problem may arise in such a scenario. This is avoided via deterministic initialization of all cluster centers for each band.

Table 3.1: DTLK: Table lookup

	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5
Patch 1	15	29	54	11	79
Patch 2	32	29	42	4	77
Patch 3	15	45	56	7	84
Patch 4	15	7	46	4	77
Patch 5	87	16	56	16	77
Patch 6	6	71	15	5	69
Patch 7	15	33	56	4	51
Patch 8	54	29	55	5	26
Selected Band	15	29	56	4	77

3.5 Experiments and Results

3.5.1 Text Datasets

Our proposed work Transformed K-means Clustering focuses on document clustering. Therefore, we used three text datasets namely TDT2 corpus⁶, Reuters-21578 corpus⁶, and 20 Newsgroup⁶. The datasets details can be referred from Section 1.3.3.

In [75], the evaluation metrics are reported by varying the number of clusters from 2 to 10. We follow the same protocol and reports the results using two evaluation metrics - Purity [35] and Entropy [35]. The evaluation metrics details can be referred from Section 1.4.3.

We compare our proposed technique with several state-of-the-art approaches in document clustering. The first one is Improved Spherical K-means (ISKM) [76], Deep Embedding Clustering based on Contractive Autoencoder (DECCA) [9] and Transformed Subspace Clustering (TSC) [64].

⁶<http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html>

Table 3.2: TLK: Clustering Results on TDT2

Clusters ↓	Entropy (lower is better)				Purity (higher is better)			
	ISKM	DECCA	TSC	Proposed	ISKM	DECCA	TSC	Proposed
2	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
4	0.0059	0.0000	0.0000	0.0000	0.9956	1.0000	1.0000	1.0000
6	0.0826	0.0809	0.0013	0.0011	0.9435	0.9954	0.9963	0.9971
8	0.0952	0.0911	0.0465	0.0206	0.9476	0.9801	0.9013	0.9901
10	0.0808	0.0685	0.0178	0.0061	0.9153	0.9224	0.9775	0.9854

Table 3.3: TLK: Clustering Results on Reuters

Clusters ↓	Entropy (lower is better)				Purity (higher is better)			
	ISKM	DECCA	TSC	Proposed	ISKM	DECCA	TSC	Proposed
2	0.0551	0.0454	0.0493	0.0451	0.9012	0.9218	0.9735	0.9912
4	0.2751	0.2400	0.2103	0.2008	0.8834	0.8935	0.8984	0.9061
6	0.2029	0.2021	0.1905	0.1435	0.8719	0.8880	0.8855	0.8995
8	0.2158	0.2029	0.2811	0.2009	0.8686	0.8776	0.9135	0.9524
10	0.2677	0.2464	0.2579	0.2286	0.7690	0.7888	0.8069	0.8331

Table 3.4: TLK: Clustering Results on Newsgroup

Clusters ↓	Entropy (lower is better)				Purity (higher is better)			
	ISKM	DECCA	TSC	Proposed	ISKM	DECCA	TSC	Proposed
2	0.1556	0.1843	0.8172	0.1131	0.8867	0.8200	0.7233	0.9233
4	0.1665	0.1301	0.5911	0.1055	0.8083	0.8183	0.6567	0.8575
6	0.1441	0.1492	0.4697	0.1211	0.8717	0.8322	0.7050	0.9028
8	0.1505	0.1562	0.4673	0.1142	0.8708	0.8700	0.6721	0.8859
10	0.1395	0.1625	0.4449	0.1299	0.8943	0.8677	0.6690	0.9100

The results are shown in Table 3.2, 3.3 and 3.4 for the datasets TDT2, Reuters-21578 and 20 Newsgroup respectively. For all the datasets, across all configurations, one can see that our proposed approach yields the best results. The empirical convergence plots of the proposed approach for the datasets TDT2 corpus, Reuters-21578 and 20 Newsgroup are shown in Fig. 3.5, 3.6 and 3.7 respectively. The results are shown for two clusters. The convergence plots depict that the proposed algorithm converges within 15 to 20 iterations. This has been the case irrespective of the number of clusters. One can observe that for Reuters dataset, the convergence is non-monotonic. This can be the case for

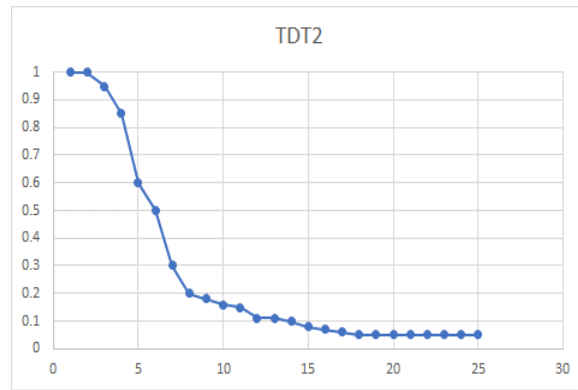


Figure 3.5: TLK: Empirical Convergence Plot (2 clusters) for TDT2

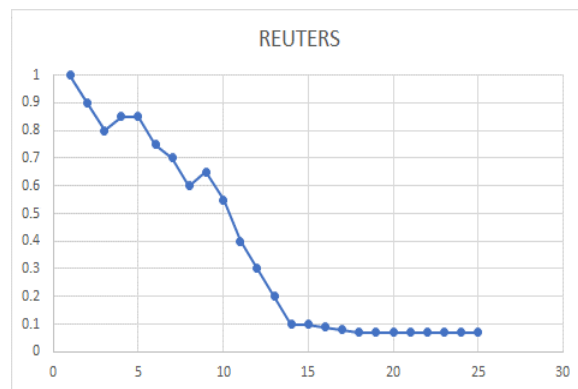


Figure 3.6: TLK: Empirical Convergence Plot (2 clusters) for Reuters

ADMM based solutions.

3.5.2 Hyperspectral Imaging

We carry out evaluation of our proposed approach Deeply Transformed K-means Clustering discussed in Section 3.4.2 on two standard hyperspectral images datasets namely Indian Pines and Pavia University. The datasets details and pre-processing steps can be referred from Section 1.3.2.

There is no straightforward way to evaluate the efficacy of the selected bands. Therefore, it must be evaluated based on some other criterion. Here, we have used classification as the evaluation criterion, i.e., after band selection, the

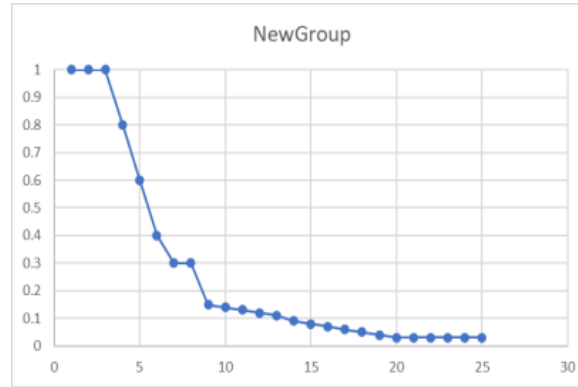


Figure 3.7: TLK: Empirical Convergence Plot (2 clusters) for Newsgroup

selected bands are passed onto a classifier. In particular, we have used kernel sparse representation classifier [77] in this work since it is a highly cited work on hyperspectral classification.

We benchmark against several deep clustering techniques — Unsupervised Deep Subspace Clustering (UDSC) [78], Graph Convolutional Neural Network (GCNN) [79], and BS-Nets [80]. We also compare with Deeply Transformed Subspace Clustering (DTSC) [46]; however, note that DTSC was never meant to be used for the said purpose and, hence, we use the table lookup approach for selecting the bands.

Our proposed technique requires the specification of two parameters — λ and μ . We set the value of λ to 1 for all the experiments; this value has been found to work well on almost all transform learning-based formulations. The parameter μ controls the relative importance of the deep transform learning loss and K-means clustering loss. There is no reason to give more weightage to one than the other; therefore, we set $\mu = 1$. In the experiments, we used overlapping patches of size 8×8 . The patches are obtained by shifting the patches 2 pixels in both

horizontal and vertical directions, as and when needed. The best results were obtained for a three-layer architecture where the number of nodes in subsequent layers was halved, i.e., the architecture was 64-32-16.

The experiments were carried out on a 64 bit Intel Core i5-8265U CPU at 1.60 GHz, 16-GB RAM running Ubuntu. The run times of different algorithms are shown in Table 3.5. One can see that our proposed algorithm is the fastest. It is faster than DTSC because, unlike the latter, we do not need to solve a complex iterative optimization problem every time in the clustering step. Both our proposed algorithm and DTSC are solved using ADMM and hence are considerably faster compared to the traditional backpropagation-based approach used by UDSC, GCNN, and BS-Nets.

Table 3.5: DTLK: Runtime comparison (in seconds)

	Pavia University			Indian Pines		
# Bands →	10	20	30	10	20	30
Algorithms ↓						
UDSC	1002	1338	1609	553	819	1012
GCNN	1190	1581	2086	678	1006	1375
BS-Nets	984	1403	1760	425	664	808
DTSC	817	1119	1531	316	497	633
Proposed	465	650	803	193	302	386

Although we could not give any proof regarding the convergence of our proposed algorithm, we found that it does indeed converge to a local minimum given the choice of parameters. The empirical convergence plot is shown in Fig. 3.8 for 20 bands. Similar convergence results were obtained for the other number of bands. For measuring classification accuracy, we use three standard evaluation metrics in hyperspectral imaging —Overall Accuracy (OA), Average Accuracy (AA), and Kappa coefficient. In Fig. 3.9, the plots for two different datasets and

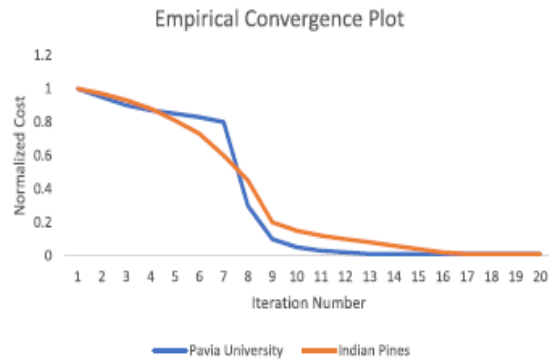


Figure 3.8: DTLK: Empirical Convergence Plot (20 bands)

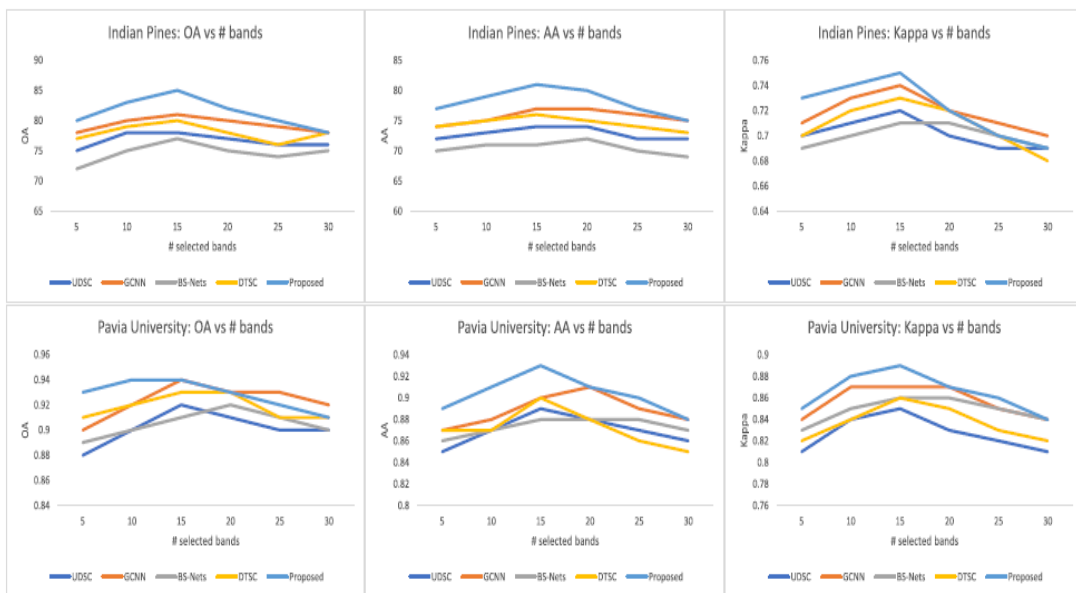


Figure 3.9: DTLK: Detailed Results

three different metrics are shown. From Fig. 3.9, one can note that our proposed method yields the best results, especially for a fewer number of bands. Our proposed method is closely followed by GCNN and DTSC. GCNN performs better than ours in some cases when the number of chosen bands is higher. In general, for all the techniques, one can notice that the results improve as the number of bands increases from 5 to 15 or 20, after that the results deteriorate. The same has been observed by other studies such as [78].

3.5.3 Comparison with Deep Dictionary Learning based deep clustering approaches

In this section, we present a comparison between Deep Dictionary Learning (DDL) based deep clustering approach and Deep Transform Learning (DTL) based deep clustering approach.

DDL and DTL are two popular deep learning frameworks for feature extraction and representation learning. Both of these frameworks are used in unsupervised learning tasks, such as image and signal processing, where the goal is to learn a compact and meaningful representation of high-dimensional data.

Deep dictionary learning (DDL) is a technique that learns a dictionary of basis atoms from the input data. The basis atoms are used to represent the data in a sparse and compact manner. In DDL, the input data is transformed by a linear combination of the dictionary elements, and the resulting coefficients are used as features for downstream tasks. DDL is often used in image and signal processing applications, where the input data is a high-dimensional signal or image. The advantage of DDL is that it can learn a compact and interpretable representation of the input data. However, DDL is computationally expensive and requires a large amount of training data to learn an accurate dictionary.

On the other hand, deep transform learning (DTL) is a technique that learns a set of non-linear transformations that map the input data to a low-dimensional feature space. Unlike DDL, DTL does not require a pre-defined dictionary of basis functions. Instead, DTL uses a deep neural network to learn the non-linear transformations from the input data. The advantage of DTL is that it can learn

a highly expressive and flexible representation of the input data. DTL is often used in applications where the input data is complex and high-dimensional, such as natural language processing and computer vision. However, DTL can suffer from overfitting if the network is too large, and it may be difficult to interpret the learned representation [81].

We have compared the performance of DDLK (discussed in Section 2.3.3) and DTLK (discussed in Section 3.4.2) along with several deep clustering benchmarks on computer vision datasets. The results are shown in Table 3.6. The datasets details can be referred from Section 1.3.1. The results show that DDLK gives better results than DTLK on all the four datasets. This might be because all the four datasets are not very complex and high-dimensional.

Table 3.6: Comparison of DLK, DLS, DDLK and DTLK

Algorithms →	DCEC [8]		DKM [5]		DLK		DLS		DDLK		DTLK	
Datasets ↓	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
ARFaces	0.26	0.02	0.45	0.04	0.56	0.23	0.64	0.26	0.58	0.15	0.42	0.03
COIL20	0.79	0.635	0.785	0.60	0.67	0.56	0.72	0.59	0.81	0.68	0.78	0.60
EYaleB	0.20	0.04	0.17	0.02	0.36	0.11	0.29	0.10	0.551	0.224	0.27	0.05
Olivetti	0.709	0.336	0.52	0.101	0.72	0.42	0.68	0.36	0.826	0.533	0.543	0.11

In summary, DDL learns a dictionary of basis functions to represent the input data, while DTL learns a set of nonlinear transformations. DDL is computationally expensive but interpretable, while DTL is highly expressive but may suffer from overfitting [81]. The choice between DDL and DTL depends on the specific application and the trade-off between interpretability and flexibility.

3.6 Summary

This chapter proposes two novel deep clustering approaches based on transform learning framework namely Transformed K-means clustering, Deeply Transformed K-means clustering. Transformed K-means clustering embeds K-means clustering in transform learning framework and optimize the joint formulation using Alternative Directed Multipliers Method (ADMM). Deeply Transformed K-means clustering embeds K-means clustering in deep transform learning framework (with three layers of transforms) and optimize the joint formulation via ADMM. Experiments have been carried out on two different kinds of problems: text datasets and hyperspectral imaging. The first proposed approach Transformed K-means clustering is evaluated on three text datasets while Deeply Transformed K-means clustering is evaluated on hyperspectral imaging datasets. In both the set of experiments, the proposed approaches outperform state-of-the-art techniques.

Chapter 4

Convolutional Transform Learning based Clustering Approaches

This chapter presents four novel deep clustering approaches based on Convolutional Transform Learning (CTL) framework. The first and second proposed approach embeds K-means clustering and Sparse Subspace clustering respectively in Deep CTL (DCTL) framework. The former is evaluated on facial images datasets while the latter is evaluated on hyperspectral imaging data. Both the proposed approaches outperform state-of-the-art deep clustering approaches. The third proposed approach embeds the Contrastive Learning in the DCTL framework embedded with K-means clustering. The incorporation of contrastive learning empowers the proposed model to capture discriminative information of clusters and thus, learns more cluster-friendly representations. The proposed approach is evaluated on five facial images datasets to demonstrate higher clustering performance as compared to the current state-of-the-art deep clustering models. The fourth approach proposes a novel unsupervised multi-channel fu-

sion clustering framework named DeConFCluster. The proposed framework jointly trains DCTL network, multichannel fusion network and the K-Means clustering module; thus, the representations are diverse and effective as these are guided by K-Means loss. The proposed framework gives superior clustering performance than the current state-of-the-arts multiview clustering approaches on several multiview datasets.

4.1 Convolutional Transform Learning (CTL)

Convolutional Transform Learning (CTL) has been introduced in [82]. CTL learns a set of filters $(t_m)_{1 \leq m \leq M}$ operated on observed samples $(s^{(k)})_{1 \leq k \leq K}$ to generate a set of features $(x_m^{(k)})_{1 \leq m \leq M, 1 \leq k \leq K}$. Formally, the inherent learning model is expressed through convolution operation defined as

$$(\forall m \in \{1, \dots, M\}, \forall k \in \{1, \dots, K\}) \quad t_m * s^{(k)} = x_m^{(k)}. \quad (4.1)$$

Following the original study on transform learning, a sparsity penalty was imposed on the features for improving representation ability and limiting overfitting issues [73]. This was expressed as the following optimization problem

$$\begin{aligned} \underset{(t_m)_m, (x_m^{(k)})_{m,k}}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \sum_{m=1}^M \left(\|t_m * s^{(k)} - x_m^{(k)}\|_2^2 + \psi(x_m^{(k)}) \right) \\ & + \mu \sum_{m=1}^M \|t_m\|_F^2 - \lambda \log \det ([t_1 | \dots | t_M]), \end{aligned} \quad (4.2)$$

where ψ is a suitable penalization function. Note that the regularization term

“ $\mu \|\cdot\|_F^2 - \lambda \log \det$ ” promotes diversity and non-degeneracy of the learned filters [83]. Let us introduce the matrix notation

$$T * S - X = \begin{bmatrix} t_1 * s^{(1)} - x_1^{(1)} & \dots & t_M * s^{(1)} - x_M^{(1)} \\ \vdots & \ddots & \vdots \\ t_1 * s^{(K)} - x_1^{(K)} & \dots & t_M * s^{(K)} - x_M^{(K)} \end{bmatrix} \quad (4.3)$$

where $T = \begin{bmatrix} t_1 & \dots & t_M \end{bmatrix}$, $S = \begin{bmatrix} s^{(1)} & \dots & s^{(K)} \end{bmatrix}^\top$, and $X = \begin{bmatrix} x_1^{(k)} & \dots & x_M^{(k)} \end{bmatrix}_{1 \leq k \leq K}$.

The cost function in equation (4.2) could be compactly rewritten as¹

$$F(T, X) = \frac{1}{2} \|T * S - X\|_F^2 + \Psi(X) + \mu \|T\|_F^2 - \lambda \log \det(T), \quad (4.4)$$

where Ψ applied the penalty term ψ column-wise on X .

A local minimizer to (4.4) could be reached efficiently using the alternating proximal algorithm [84–86], which alternates between proximal updates on variables T and X . More precisely, set a Hilbert space $(\mathcal{H}, \|\cdot\|)$, and define the proximity operator [87] at $\tilde{x} \in \mathcal{H}$ of a proper lower-semi-continuous convex function $\varphi : \mathcal{H} \rightarrow [-\infty, +\infty]$ as

$$\text{prox}_\varphi(\tilde{x}) = \arg \min_{x \in \mathcal{H}} \varphi(x) + \frac{1}{2} \|x - \tilde{x}\|^2. \quad (4.5)$$

¹Note that T is not necessarily a square matrix. By abuse of notation, we defined the “log-det” of a rectangular matrix as the sum of logarithms of its singular values.

Then, the alternating proximal algorithm reads

$$\begin{aligned} &\text{For } n = 0, 1, \dots \\ &\left[\begin{array}{l} T^{[n+1]} = \text{prox}_{\gamma_1 F(\cdot, X^{[n]})} (T^{[n]}) \\ X^{[n+1]} = \text{prox}_{\gamma_2 F(T^{[n+1]}, \cdot)} (X^{[n]}) \end{array} \right. \end{aligned} \quad (4.6)$$

with initializations $T^{[0]}$, $X^{[0]}$ and γ_1, γ_2 as positive constants. For more details on the derivations and the convergence guarantees, the readers can refer to [82].

4.2 Proposed Approaches

4.2.1 Deep Convolutional K-Means Clustering

Our proposed approach Deep Convolutional K-Means Clustering² integrated the K-means clustering loss in the Deep CTL (DCTL) framework.

The DCTL approach is the deep version of CTL approach discussed in Section 4.1. In DCTL, we learned a different set of convolutional filters $T_1^{(c)}, \dots, T_L^{(c)}$ and features $X_1^{(c)}, \dots, X_L^{(c)}$. These learned deep features can be computed by stacking many such layers

$$(\forall \ell \in \{1, \dots, L - 1\}) \quad X_\ell = \phi_\ell(T_\ell * X_{\ell-1}), \quad (4.7)$$

²Anurag Goel, Angshul Majumdar, Emilie Chouzenoux, and Giovanni Chierchia, "Deep Convolutional K-Means Clustering." In 2022 IEEE International Conference on Image Processing (ICIP), pp. 211-215. IEEE, 2022.

where $X_0 = S$ and ϕ_ℓ is a given activation function for layer ℓ .

$$\begin{aligned} \underset{T_1, \dots, T_L, X}{\text{minimize}} \quad & \frac{1}{2} \|T_L * \phi_{L-1}(T_{L-1} * \dots * \phi_1(T_1 * S)) - X\|_F^2 + \Psi(X) \\ & + \sum_{\ell=1}^L (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell)) \end{aligned}$$

Our proposed approach Deep Convolutional K-Means Clustering (DCKM) embeds the K-means clustering loss [43] in DCTL formulation. The proposed formulation is expressed as follows:

$$\underset{T_1, \dots, T_L, X, H}{\text{minimize}} \quad \underbrace{F_{\text{conv}}(T_1, \dots, T_L, X | S)}_{\text{DCTL loss}} + \beta \underbrace{\|X - XH^\top (HH^\top)^{-1} H\|_F^2}_{\text{K-Means loss}} \quad (4.8)$$

where

$$\begin{aligned} F_{\text{conv}}(T_1, \dots, T_L, X | S) = & \frac{1}{2} \|T_L * \phi_{L-1}(T_{L-1} * \dots * \phi_1(T_1 * S)) - X\|_F^2 \\ & + \Psi(X) + \sum_{\ell=1}^L (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell)) \end{aligned} \quad (4.9)$$

$$h_{ij} = \begin{cases} 1, & \text{if } x_j \in \text{cluster } i \\ 0, & \text{otherwise} \end{cases} \quad (4.10)$$

Here, S is the input, X is the learned representation, $\beta > 0$ is the regularization weight associated with the K-Means clustering loss, and H is the matrix of binary indicator variables h_{ij} .

The architecture of DCKM framework is shown in figure 4.1. The input data is convolved by a set of convolutional filters T_1 . The resulting feature map is max-

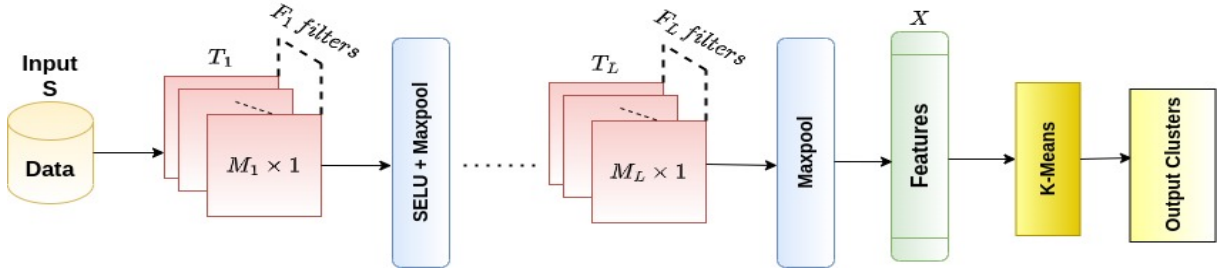


Figure 4.1: Overview of the proposed DCKM architecture. L represents number of DCTL layers, M_l^c - filter size and F_l^c - #filters of the respective layer l and channel c . SELU is the activation function.

pooled followed by Scaled Exponential Linear Unit (SELU) activation function. Then, the resulted features are convolved through second set of convolution filters T_2 . The resulting feature map undergoes max-pooling before being input to K-means clustering.

4.2.2 Deep Convolutional Sparse Subspace Clustering

Our next proposed approach is Deep Convolutional Sparse Subspace Clustering³ that incorporated the Sparse Subspace Clustering (SSC) loss in the DCTL framework.

The DCTL approach is the deep version of CTL approach discussed in Section 4.1. The deeper version of CTL was formed by applying (and learning) multiple convolutional filters one after the other [81]. This was expressed as follows,

$$\begin{aligned}
 \underset{T_1, T_2, T_3, Z}{\text{minimize}} \quad & \|T_3 * (T_2 * (T_1 * X)) - Z\|_F^2 + \Psi(Z) \\
 & + \sum_{\ell=1}^3 (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell))
 \end{aligned} \tag{4.11}$$

³Anurag Goel and Angshul Majumdar, "Sparse Subspace Clustering Incorporated Deep Convolutional Transform Learning for Hyperspectral Band Selection" (Submitted in ICASSP 2024)

In sparse subspace clustering (SSC) [44], it is assumed that the samples belonging to the same cluster lie in the same subspace. The formulation for SSC is as follows:

$$\sum_i \|z_i - Z_{i^c} c_i\|_2^2 + \chi \|c_i\|_1, \forall i \in \{1, \dots, N\} \quad (4.12)$$

Assuming N samples, z_i is the i^{th} data point, Z_{i^c} represents all the data points barring the i^{th} one and $c_i (\in \mathbb{R}^{N-1})$ corresponds to the sparse codes that represent samples in Z_{i^c} belonging to the same cluster as z_i . The l_1 -norm imposes sparsity on the codes. The sparsity is important because it ensures that the i^{th} sample is only represented by the samples of the same cluster. χ is the regularizing parameter on the sparsity promoting term.

It is possible to represent 4.12 in a more compact form as follows:

$$\|Z - ZC\|_2^2 + \chi \|C\|_1, \quad (4.13)$$

such that $\text{diag}(C) = 0$

where

$$Z = \begin{bmatrix} z_1 & | & \dots & | & z_N \end{bmatrix}, C = \begin{bmatrix} \tilde{c}_1 & | & \dots & | & \tilde{c}_N \end{bmatrix}, \tilde{c}_i \in \mathbb{R}^N \quad (4.14)$$

\tilde{c}_i is formed from c_i by imputing it with 0 in the i^{th} position. It is essential to have the constraint $\text{diag}(C) = 0$ so that any sample is not represented by itself.

Therefore, the proposed formulation is as follows,

$$\min_{T_1, T_2, T_3, Z, C} \underbrace{\|T_3 * (T_2 * (T_1 * X)) - Z\|_F^2 + \Psi(Z) + \sum_{\ell=1}^3 (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell))}_{\text{DCTL loss}} + \beta \underbrace{(\|Z - ZC\|_F^2 + \chi \|C\|_1)}_{\text{SSC loss}} \quad (4.15)$$

such that $\text{diag}(C) = 0$.

Here, S is the input, X is the representation learned, $\beta > 0$ is the regularization weight associated with the SSC loss.

Equation 4.15 is solved iteratively in two parts. In the first part, C is assumed to be constant and T_1, T_2, T_3 and Z are updated.

$$P_1 : \min_{T_1, T_2, T_3, Z} \|T_3 * (T_2 * (T_1 * X)) - Z\|_F^2 + \Psi(Z) + \sum_{\ell=1}^3 (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell)) + \beta (\|Z - ZC\|_F^2) \quad (4.16)$$

In the second part, T_1, T_2, T_3 and Z are assumed to be fixed and C is updated.

$$P_2 : \min_C \|T_3 * (T_2 * (T_1 * X)) - Z\|_F^2 + \beta (\|Z - ZC\|_F^2 + \chi \|C\|_1) \quad (4.17)$$

The first part (P_1) is solved via ADAM optimizer and the second part (P_2) is a regularized version of SSC and can be easily solved using any l_1 -norm minimizer. Solutions to P_1 and P_2 are carried out alternately until the convergence is attained, which is defined by the condition that the cluster centers exhibit negligible variance in successive iterations.

Application to Hyperspectral Band Selection

We have applied our proposed approach Deep Convolutional Transform Learning Sparse Subspace Clustering (DCTLSSC) for hyperspectral band selection in a fashion similar to our prior work discussed in Section 3.4.2. This is because our simple scheme yielded very good results in practice.

We divide the entire hyperspectral datacube into spatial patches $P_{x_i^k}$ that denotes the k^{th} patch of i^{th} band image. We modify the proposed formulation slightly to reflect that the transforms are learnt globally on all patches but the clustering is happening on the k^{th} patch. This requires solving the following problem –

$$\begin{aligned} \min_{T_1, T_2, T_3, Z, H} & \sum_i \left(\sum_k (\|T_3 * (T_2 * (T_1 * P_{x_i^k})) - z_{x_i^k}\|_F^2 + \Psi(Z_i)) \right) \\ & + \sum_i \beta (\|Z - ZC\|_F^2 + \chi \|C\|_1) + \lambda \sum_{i=1}^3 (\|T_i\|_F^2 - \log \det T_i) \end{aligned} \quad (4.18)$$

such that $\text{diag}(C_i) = 0$.

For each patch, we will get a band that is closest to the j^{th} cluster in the Euclidean sense. We see which band is appearing to be the closest in the majority of patches and select the same. A toy example is already shown in Table 3.1.

4.2.3 Contrastive Deep Convolutional Transform K-means Clustering

The deep learning based clustering approaches are not able to capture the discriminative information in the learned representations due to the lack of supervision

[1]. To mitigate the negative impact of unsupervised learning in clustering, the concept of Contrastive Learning was proposed in Contrastive Clustering [23]. In Contrastive Clustering, the positive and negative data samples pairs are generated using data augmentation. The constructed data samples pairs are projected into a feature space where the instance-level and cluster-level contrastive learning is performed by maximizing the similarity between positive pair data samples and minimizing the similarity between negative pair data samples. The instance-level and cluster-level contrastive loss are jointly optimized to learn the representations and cluster assignments in joint end-to-end fashion. Later, Contrastive Deep Embedded Clustering embedded the contrastive loss in the stacked denoising auto-encoders to improve the clustering performance [1]. But, Contrastive Deep Embedded Clustering suffers with the limitation of auto-encoders and might overfit in data constrained scenarios.

Here, a novel Contrastive Deep Convolutional Transform K-means Clustering model⁴ is proposed that leverages our previous research work, Deep Convolutional K-means clustering framework [41] discussed in Section 4.2.1, by incorporating the contrastive learning in the DCTL learnt representations. To embed the contrastive loss, the positive and negative pairs of data samples are generated with the input data and the reconstructed data from DCTL learnt representations. Then, the contrastive loss, DCTL loss and K-means clustering loss are jointly optimized together in end-to-end fashion. This ensures the DCTL learnt representations retain the discriminative information of input data features

⁴Anurag Goel and Angshul Majumdar, "Contrastive Deep Convolutional Transform K-means Clustering" (Submitted)

to generate better clusters. Our proposed model incorporates contrastive learning without relying on data augmentation to create positive and negative pairs of data samples, as many other contrastive learning-based methods do. This makes the construction of contrastive sample pairs more convenient. Moreover, our proposed model introduces a skip connection while reconstructing the data samples from DCTL learnt representations to alleviate the degradation problem that may arise due to the deep layers in DCTL framework.

4.2.3.1 Related Work on Contrastive Learning based Clustering

Contrastive Learning is a self-supervised learning technique that allows the model to learn the features of the data using the data samples without their labels. In Contrastive learning, the positive and negative pairs of data samples are generated. The contrastive learning minimizes the similarity between the positive pairs while maximizes the similarity between the negative pairs of data samples. The concept of Contrastive Clustering was introduced in [23] in which the positive pairs and negative pairs of data samples are obtained through data augmentation and then are projected into the feature space where the instance-level contrastive learning is performed in row space and the cluster-level contrastive learning is performed in column space. The joint optimization of clustering loss and instance-level contrastive loss for short-text clustering was proposed in [88]. [89] framed the class contrastive loss in cluster assignment using Nearest Neighbor Matching. Deep Robust Clustering was proposed in [90] that investigates the relationship between mutual information and contrastive

learning. Contrastive Deep Embedded Clustering integrated the contrastive loss along with the regeneration loss of stacked denoising autoencoders to obtain the more representative features for clustering [1].

4.2.3.2 Proposed Formulation

Our proposed network embeds the K-means clustering and Contrastive learning in the Deep Convolutional Transform Learning (DCTL) framework. The complete architecture of the proposed framework is shown in figure 4.2. For the input dataset X , $X \in R^{n*d}$, an input image x , $x \in R^d$, is first passed through the first convolutional layer in which the input image is first convolved with a set of convolutional filters (T_1). Then the max-pooling is applied on the resultant feature map followed by Scaled Exponential Linear Unit (SELU) activation function. The feature map generated from the first convolutional layer is then convolved with another set of convolutional filters (T_2) followed by max-pooling. This constitutes the second convolutional layer. The feature map generated from the second convolutional layer is the representation z on which K-means clustering is applied to generate the clusters.

To embed the contrastive learning in the proposed framework, for each input image x , a corresponding reconstructed image y is generated from the corresponding representation z . To generate the reconstructed image y , the representation z is upsampled and then convolved with a set of convolutional filters. This constitutes the first Deconvolution layer.

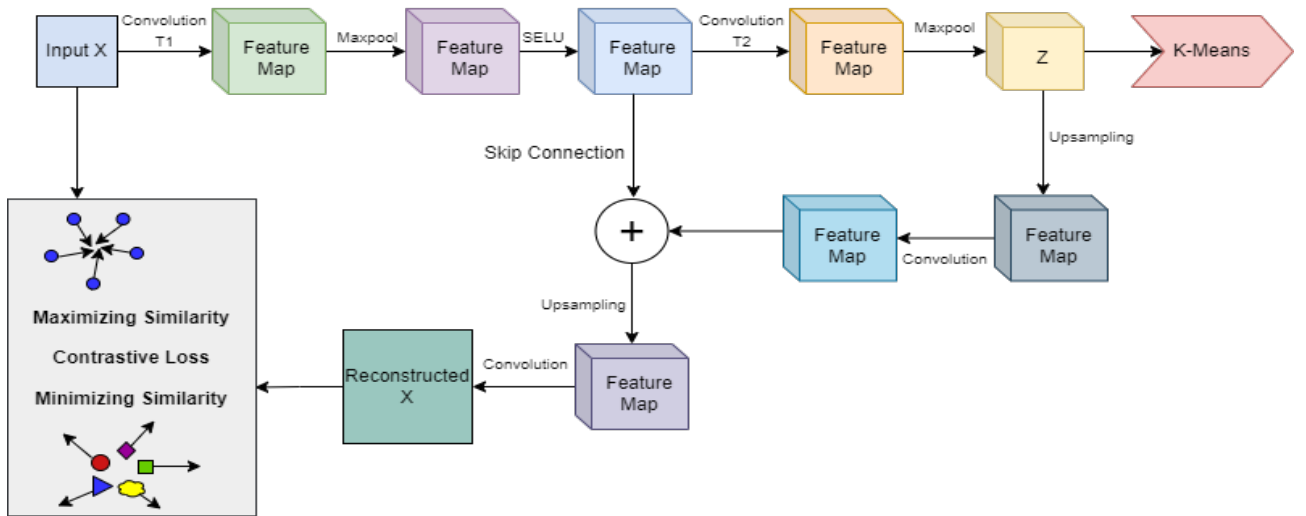


Figure 4.2: Contrastive DCTLK Architecture

The feature map generated from the first deconvolutional layer is then added with the feature map generated from the first convolutional layer through a skip connection. The skip connections are used to solve degradation problems in general. Moreover, the first layer of a deep network learns features with more semantic information since it is close to the input. Therefore, introducing the skip connection from the initial layer, helps in generating the reconstructed image y more similar to the input image x and thus helps in optimizing the contrastive loss.

After the skip connection, the feature map is passed through the second deconvolutional layer in which the feature map is first upsampled and then convolved with a set of convolutional filters. The feature map obtained from the second convolutional layer is the reconstructed image y and has the same dimension as the input image x . Now, the contrastive loss is calculated corresponding to the input image x and the reconstructed image y which is then backpropagated.

In our proposed approach, three loss namely Deep Convolutional Transform Learning (DCTL) loss, K-means clustering loss and Contrastive loss are combined together and the combined loss function is optimize in a joint end-to-end fashion.

The integration of K-means clustering in DCTL framework is already discussed in Section 4.2.1. The DCTL loss can be formulated as follows

$$L_{dctl} = \|T_2 * (T_1 * X) - Z\|_F^2 + \psi(Z) + \sum_{i=1}^2 (\epsilon \|T_i\|_F^2 - \mu \log \det(T_i)) \quad (4.19)$$

Using [43], the K-means clustering loss can be expressed as

$$L_{kmeans} = \|X - XH^T(HH^T)^{-1}H\|_F^2 \quad (4.20)$$

where X is the input data matrix and H is the matrix of binary indicator variables h_{ij} where

$$h_{ij} = \begin{cases} 1, & \text{if } x_j \in \text{cluster } i \\ 0, & \text{otherwise} \end{cases} \quad (4.21)$$

To better learn discriminative information of different clusters, the contrastive learning is applied that maximizes the agreement of positive pairs and minimizes the agreement of negative pairs. In the proposed framework, the original data and the reconstructed data of the same sample is considered as positive pairs.

Suppose that the input data is

$$X^T = \{x_1, x_2, \dots, x_n\} \in R^{d*n}$$

and the corresponding reconstructed output is

$$Y^T = \{y_1, y_2, \dots, y_n\} \in R^{d*n}$$

Now, for each sample x_i where $i \in \{1, \dots, n\}$, one positive pair is generated that comprises of x_i and its corresponding reconstructed sample y_i , n-1 negative pairs are generated by pairing x_i with x_j where $j \in \{1, \dots, n\}, j \neq i$, and n-1 negative pairs are generated by pairing x_i with y_j where $j \in \{1, \dots, n\}, j \neq i$. The similarity between x_i and y_j is given as

$$s(\mathbf{x}_i, \mathbf{y}_j) = \frac{\mathbf{x}_i^t \mathbf{y}_j}{\|\mathbf{x}_i\| \|\mathbf{y}_j\|} \quad (4.22)$$

where $i, j \in [1, n]$. The similarity between y_j and x_i can be computed by swapping the x_i and y_j terms in the equation 4.22. The contrastive loss for $X^T = \{x_1, x_2, \dots, x_n\}$, $Y^T = \{y_1, y_2, \dots, y_n\}$ are obtained at instance-level using the below equations

$$l_i^x = -\log \frac{e^{(s(x_i, y_i)/\tau)}}{\sum_{j=1}^n [e^{(s(x_i, x_j)/\tau)} + e^{(s(x_i, y_j)/\tau)}]} \quad (4.23)$$

$$l_i^y = -\log \frac{e^{(s(y_i, x_i)/\tau)}}{\sum_{j=1}^n [e^{(s(y_i, y_j)/\tau)} + e^{(s(y_i, x_j)/\tau)}]} \quad (4.24)$$

where τ is temperature coefficient that affects the degree of attention given to each negative sample while computing the contrastive loss. A very small value of

τ results in only focusing on a few negative samples that are near to the positive samples while a very high value of τ results in equal weightage given to all the negative samples. Since [1] considered $\tau=0.5$, the same value of τ is kept in our experiments for better comparison. The overall contrastive loss is given by

$$L_{con} = \frac{1}{2n} \sum_{j=1}^n (l_i^x + l_i^y) \quad (4.25)$$

The final loss function of the proposed framework is expressed as follows:

$$L_{final} = L_{dctl} + \beta L_{kmeans} + \delta L_{con} \quad (4.26)$$

$$L_{final} = \underbrace{\|T_2 * (T_1 * X) - Z\|_F^2 + \psi(Z) + \sum_{i=1}^2 (\epsilon \|T_i\|_F^2 - \mu \log \det(T_i))}_{\text{Deep Convolutional Transform Learning loss}} + \underbrace{\beta \|X - XH^T(HH^T)^{-1}H\|_F^2}_{\text{K-Means loss}} + \underbrace{\delta \frac{1}{2n} \sum_{j=1}^n (l_i^x + l_i^y)}_{\text{Contrastive loss}} \quad (4.27)$$

where the β and δ are the K-means loss regularizer and Contrastive loss regularizer respectively.

4.2.4 DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework

With the rapid increase in data collection sources and volume, the exploration of multiview data has become popular. Multiview data is referred to as the data collected from the same data source but with different angles or different

perspectives. For example, the same news is advertised or published in different media with different content; the same statement is labeled with different tags by different individuals, and the same image is captured using different features. Multiview data is richer and more informative but more complex than single-view data. In multiview data, the data belonging to each view has information related to different contexts and also has some complementary information. Clustering is a category of unsupervised learning approach in which the data instances are grouped into several groups or clusters based on the various features of the data instances. Thus, multiview data clustering requires exploring and integrating multiple views of the data to perform the grouping of data instances in possible clusters.

Multiview data knowledge extraction is vital in big data mining and analytics nowadays. In this regard, many recent works suggest CNN based clustering objectives [91, 92]. These generally lie on the encoder-decoder framework. In such a work, the clustering loss is included after the encoder network, which ensues the problem of additional training of a decoder network and hence, incurs extra learning of weights. In data-constrained scenarios, this can make the model prone to overfitting [6]. Also, some works learn representations independently and apply clustering algorithms like K-Means in a piecemeal fashion which may lead to representations being less effective for clustering task.

Deep Convolutional Transform Learning (DCTL) based unsupervised and supervised frameworks have been proposed in [93] and [94] respectively for performing the classification and regression tasks. In this chapter, an unsupervised

multi-channel multiview clustering framework based on DCTL - DeConFCluster is introduced that bridges all the gaps mentioned earlier, namely

1. it avoids additional decoder training,
2. it learns unique transforms and
3. it learns representations from joint training of deep CTL multiview layers and K-Means algorithm.

The proposed framework is evaluated on four standard multiview clustering datasets. The results demonstrate that the proposed framework outperforms the state-of-the-art multiview deep clustering approaches.

4.2.4.1 Related Work on Multiview Clustering

MultiView Clustering (MVC) clusters subjects into subgroups using multiview data and has gained significant attention rapidly as it caters to solving real-world problems that fall under big data analytics. Recently many solutions have been proposed to perform the same. These are broadly classified into two categories generative and discriminative approaches. The generative approaches try to learn the underlying data distribution. These use generative models with each model representing the individual view and then find the clustering solution. In contrast, discriminative approaches seek to optimize an objective function with pairwise similarities. The average similarity in intra-clusters and inter-clusters is minimized and maximized respectively [95]. The former usually

includes Expectation Maximization (EM) and mixture models. The latter, being larger in number, can be further categorized into sub-categories like multiview spectral clustering, multiview subspace clustering, multiview non-negative matrix factorization clustering, multi-kernel clustering, Canonical Correlation Analysis (CCA), etc. [95].

In generative approaches, the work in [96] assumes independent views and adopts multinomial distribution for the document clustering problem. Similarly, based on different assumptions and criteria, two versions of the multiview EM algorithm for finite mixture models are proposed in [97]. Using Convex Mixture Models (CMMs) for single-view clustering, the multiview version proposed in [98] could find the global optimum. It also avoided the initialization and local optima problems of standard mixture models, as the latter requires multiple EM algorithms executions.

Next, the multiview spectral clustering method is discussed in discriminative approaches. This method obtains a common clustering result and assumes that the same or similar eigenvector matrix is shared among all views. There are two characteristic methods. First is co-training spectral clustering [99–102] when both labeled and unlabelled data are available. Second is co-regularized spectral clustering [103, 104], which is a semi-supervised learning technique. The objective function generally requires the difference between the predictor functions of the two views to be minimized.

There are methodologies based on subspace clustering in the multiview data

[105–107]. It requires finding the underlying low dimensional common subspace from each view which is, in general, obtained by making each of the view's coefficient matrix as similar as possible. The other works suggest Non-Negative Matrix Factorization (NMF) that seeks two non-negative matrix factors called basis and indicator. In the case of MVC, some studies point to learning a common indicator matrix across each view [108, 109] for NMF. Some works propose using multiview K-Means clustering to deal with the extensive data. These works use K-Means as it is computationally less expensive than eigen-decomposition. In [110], authors proposed a multiview K-Means clustering method that adopted a common indicator matrix across different views. Besides Non-negative Matrix Factorization (NMF), the authors in [111] introduced a categorical utility function to measure the similarity between the indicator matrix from each view and the common indicator matrix and proposed a consensus based MVC method.

Also, there are methods in which direct view combination via a kernel is used as a common approach to perform MVC. Usually, it is done by designating a kernel for each view and then combining these kernels in a convex combination [112–114]. Another technique - CCA combines multiple views after projection [115, 116]. All methods mentioned earlier have achieved satisfactory performance for the clustering task. But, it may be challenging to handle the data with high-dimensional features and nonlinear property using the above stated methods since they majorly adopt shallow and linear embedding functions to reveal the intrinsic structure of the multiview data.

Recently, graph based MVC has also gained momentum. The authors in [28] proposed a solution wherein the graph matrices of multiple views are combined into a unified graph matrix by generating the Similarity Induced Graph (SIG) matrices for all the available views. Then the rank constraint on the graph Laplacian matrix is applied, and the number of connected components are produced from the unified graph, which gives the final number of clusters.

Deep learning has emerged as a highly utilized technique to solve almost all real-world problems and is used in the case of MVC. In [92], multiple autoencoders are utilized for multiview data to generate multiple latent representations and apply heterogeneous graph learning to fuse the generated latent representations followed by the K-Means network for the final clusters. Further, in the study[91], based on autoencoders, Deep Embedded Multiview Clustering with collaborative training (DEMVC) is proposed. It utilizes complementary and consensus information from multiple views and collaboratively learns the deep latent feature representations and clustering assignments.

A Graph Neural Network (GNN) [117] is applied to deep representation-based MVC to completely benefit from the features embedded in the attributed multiview graph data. Further, the work in [118] used Graph Convolutional Network (GCN) as an encoder with the most reliable view as input. In another study, multiple GCN decoders capture the view-consistent low-dimensional feature representation among different views [119]. Here, the issue is with the additional weights training incurred from the decoder network, which could lead to overfitting in data-constrained scenarios [6, 41]. Also, another shortcoming

of existing solutions is due to CNN. Additionally, CNN ends up in a trivial solution without an output. Employing Deconvolutional layers is the lone way to prevent the trivial solution. However, even using the mentioned solution, there are chances of over-fitting.

The work in [39] embedded K-Means clustering in the Transform Learning framework and trained in a joint end-to-end fashion. Also, DCTL was utilized to perform clustering by jointly training it with K-Means to perform single-view clustering [41]. In this chapter, a MVC framework DeConFCluster is introduced that jointly trains and optimizes DeConFuse and K-Means clustering modules and it overcomes the aforementioned shortcomings.

4.2.4.2 DeConFuse: a deep convolutional transform-based unsupervised fusion framework

ConFuse framework was proposed for the unsupervised construction of representation features of multi-channel data [120]. A natural strategy was to learn, for each channel $c \in \{1, \dots, C\}$, a distinct set of convolutional filters $(T^{(c)})_{1 \leq c \leq C}$ and associated features $(X^{(c)})_{1 \leq c \leq C}$, by solving a CTL-based formulation:

$$\begin{aligned} \underset{T^{(c)}, X^{(c)}}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \left(\|S_k^{(c)} T^{(c)} - X_k^{(c)}\|_F^2 + \Psi(X_k^{(c)}) \right) \\ & + \mu \|T^{(c)}\|_F^2 - \lambda \log \det(T^{(c)}). \end{aligned} \quad (4.28)$$

Then, the learned channel-wise features were stacked as $X_k = [X_k^{(1)\top} | \dots | X_k^{(C)\top}]^\top$ for each k , and fused by a transform learning procedure acting as a fully-

connected layer:

$$\begin{aligned} \underset{\tilde{T}, Z}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \|\tilde{T} X_k - Z_k\|_F^2 + \iota_+(Z) \\ & + \mu \|\tilde{T}\|_F^2 - \lambda \log \det(\tilde{T}), \end{aligned} \quad (4.29)$$

where \tilde{T} denoted the fusion stage transform (not assumed to be convolutional), Z is the row-wise concatenation of the fusion stage features $(Z_k)_{1 \leq k \leq K}$, and ι_+ is the indicator function for positive orthant, equals to zero if all the entries of Z are non-negative, and $+\infty$ otherwise. Such non-negativity constraint allowed us to avoid trivial solutions.

However, the disjoint resolution of Problems (4.28) and (4.29) might lead to unstable solutions that were too sensitive to initialization. Thus, an alternative strategy is proposed in this work where all the variables are learned in an end-to-end fashion by solving a joint optimization problem. To this aim, we relied on the key property that the solution $(\hat{X}^{(c)})_{1 \leq c \leq C}$ of the CTL problem assuming fixed filters $(T^{(c)})_{1 \leq c \leq C}$ could be reformulated as the simple application of an element-wise activation function, that is, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} \hat{X}_k(T) &= \left[\hat{X}_k^{(c)}(T) \right]_{1 \leq c \leq C} \\ &= \left[\Phi(S_k^{(c)} T^{(c)}) \right]_{1 \leq c \leq C}, \end{aligned} \quad (4.30)$$

with Φ the proximity operator of Ψ [121]. For example, if Ψ was the indicator function of the positive orthant, then Φ identified with the famous rectified linear unit (ReLU) activation function. Many other examples are provided in [121].

Consequently, in the proposed formulation, the Equation (4.30) is plugged into Problem (4.29), leading to our final *ConFuse* formulation:

$$\begin{aligned} \underset{T, \tilde{T}, Z}{\text{minimize}} \quad & \frac{1}{2} \sum_{k=1}^K \|\tilde{T} \hat{X}_k(T) - Z_k\|_F^2 + \iota_+(Z) + \mu \|\tilde{T}\|_F^2 \\ & + \mu \|T\|_F^2 - \lambda \left(\log \det(\tilde{T}) + \sum_{c=1}^C \log \det(T^{(c)}) \right). \end{aligned} \quad (4.31)$$

Although Problem (4.31) was still non-convex, this new formulation had two notable advantages. First, as soon as the involved activation function was smooth, all terms of the cost function in (4.31) were differentiable, except the indicator function. Thus, the accelerated stochastic projected gradient descent, Adam, from [122] can be employed. The latter used automatic differentiation and stochastic approximations to deal with large datasets efficiently. Second, any (sub-)differentiable activation function Φ could be plugged into our model (4.30), for instance, SELU [123], or Leaky ReLU [124]. This flexibility will play a key role in the performance, as shown in the experimental section.

In the formulation of ConFuse, the non-negativity constraint on Z is imposed to avoid trivial solutions. Regarding the representation filters stacked in matrices (T, \tilde{T}) , the log-det regularization imposed a full rank on those. Thus, it helped to enforce the diversity and to prevent the degenerate solution $(T = 0, X = 0, \tilde{T} = 0, Z = 0)$. The Frobenius regularization ensured that the matrices entries remain bounded.

In DeConFuse framework, the ConFuse architecture is extended with more Convolutional layers based on CTL [93]. Here, the number of Transforms is

same as the number of CTL Layers. Thus, a different set of convolutional filters $T_1^{(c)}, \dots, T_L^{(c)}$ and features $X_1^{(c)}, \dots, X_L^{(c)}$ are learned. These learned deep features can be computed by stacking many such layers

$$(\forall \ell \in \{1, \dots, L-1\}) \quad X_\ell = \phi_\ell(T_\ell * X_{\ell-1}), \quad (4.32)$$

where $X_0 = S$ and ϕ_ℓ a given activation function for layer ℓ . Further, these features were processed in the same manner as in the ConFuse architecture, i.e., with fusion transform \tilde{T} and common representation Z learned subsequently.

This led to the joint optimization problem

$$\underset{T, X, \tilde{T}, Z}{\text{minimize}} \quad \underbrace{F_{\text{fusion}}(\tilde{T}, Z, X) + \sum_{c=1}^C F_{\text{conv}}(T_1^{(c)}, \dots, T_L^{(c)}, X^{(c)} | S^{(c)})}_{J(T, X, \tilde{T}, Z)} \quad (4.33)$$

where

$$\begin{aligned} F_{\text{conv}}(T_1, \dots, T_L, X | S) &= \frac{1}{2} \|T_L * \phi_{L-1}(T_{L-1} * \dots * \phi_1(T_1 * S)) - X\|_F^2 \\ &+ \Psi(X) + \sum_{\ell=1}^L (\mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell)). \end{aligned} \quad (4.34)$$

and

$$\begin{aligned} F_{\text{fusion}}(\tilde{T}, Z, X) &= \frac{1}{2} \left\| Z - \sum_{c=1}^C \text{flat}(X^{(c)}) \tilde{T}_c \right\|_F^2 + \iota_+(Z) \\ &+ \sum_{c=1}^C (\mu \|\tilde{T}_c\|_F^2 - \lambda \log \det(\tilde{T}_c)), \end{aligned} \quad (4.35)$$

where the operator “flat” transformed $X^{(c)}$ into a matrix where each row contained the “flattened” features of a sample. The complete architecture is shown

in Figure 4.3.

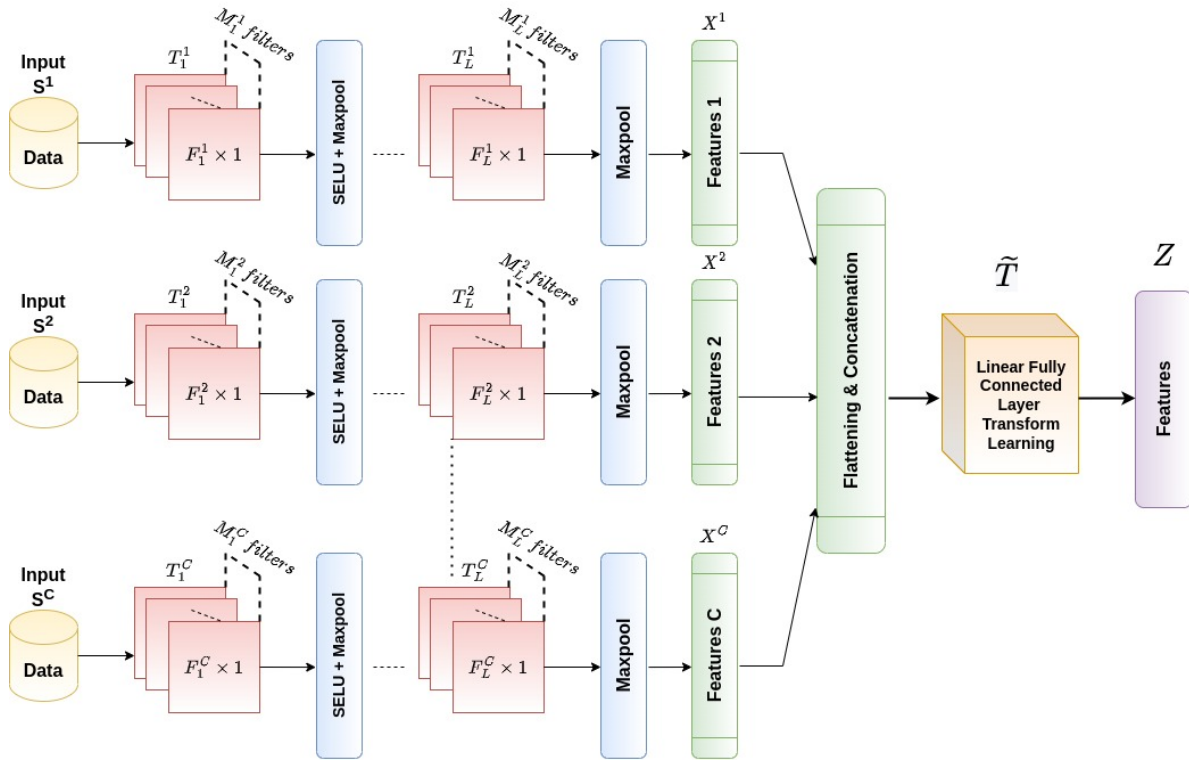


Figure 4.3: General view of the DeConFuse Architecture. $C = 5$ represents the number of DeepCTL networks/channels, $L = 2$ is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c .

4.2.4.3 Proposed Formulation

In this chapter, an unsupervised multi-channel fusion framework called DeConF-Cluster⁵ is proposed to perform multiview clustering. Previously, the framework DCKM [41] combined DCTL [81] with K-Means for Single View Clustering (SVC). In contrast, multiview clustering task is targeted here. Hence, DeConFCluster was a multi-channel clustering framework that extended DeConFuse Network [125] based on DCTL by embedding the K-Means clustering loss as was done in DCKM [41]. Here, fusion was happening that was not present

⁵P. Gupta, A. Goel, A. Majumdar, E. Chouzenoux and G. Chierchia, "DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework" (Submitted in Pattern Recognition)

in DCKM. It jointly trained and globally optimized DeConFuse Network and K-Means module. There were as many channels as the number of views in any of the considered datasets, i.e., $C = V$. Each channel was processed based on the DCTL network. This amounted to learning unique transforms $(T_c)_{1 \leq c \leq C}$ and thus, diverse and interpretable representation $(X_c)_{1 \leq c \leq C}$, for each channel input $(S_c)_{1 \leq c \leq C}$. These channel wise representations were further fused using TL [73] to learn a common representation Z and transform \tilde{T} . This completed the first module of the architecture. The representations are then fed as input to the second part of the framework K-Means clustering module that gives the clustering results. Thus, the representations learned are also guided by the K-Means loss. The learning problem reads:

$$\begin{aligned} \underset{T, X, \tilde{T}, Z, H}{\text{minimize}} \quad & F_{\text{fusion}}(\tilde{T}, Z, X) + \sum_{c=1}^C F_{\text{conv}}(T_1^{(c)}, \dots, T_L^{(c)}, X^{(c)} | S^{(c)}) + \\ & \beta \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \end{aligned} \quad (4.36)$$

The complete architecture of the DeConFCluster is summarized in the figure 4.4. The network's pipeline consisted of multiple channels wherein each channel was designated for one of the views of the multiview dataset. Next, the representations were learned from these channels' networks that gave the individual view's contribution. Then these representations were flattened and concatenated to pass through a fully connected layer learned via TL. Here, a common representation across all channels' representations is learned that provides the cross-channel information or shared information from each view. Finally, clusters were obtained by inputting the representation into the K-Means module.

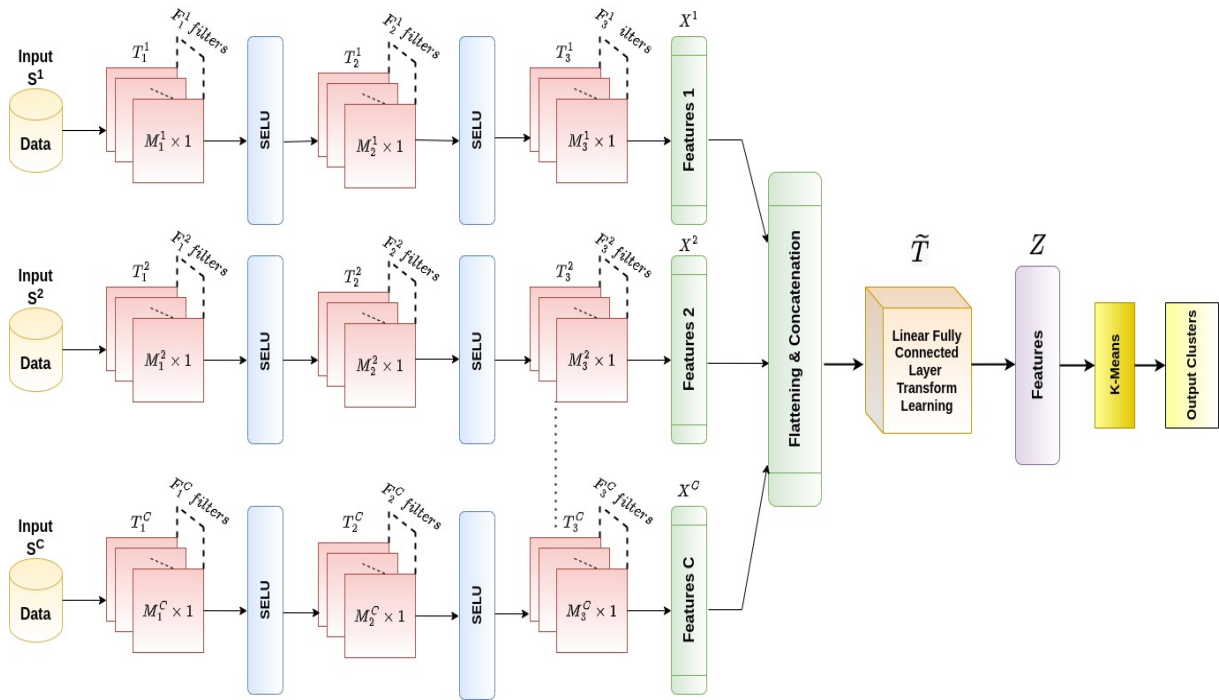


Figure 4.4: DeConFCluster Architecture. C represents the number of DeepCTL networks/channels, L is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c .

All the variables were learned in an end-to-end fashion. Typically, SGD could be used as an optimizer for all the variables except H . This latter variable was updated directly via K-Means clustering [43] at each iteration using the current Z estimate as an input.

4.3 Experiments and Results

4.3.1 Computer Vision datasets

4.3.1.1 Results on Deep Convolutional K-means Clustering (DCKM)

The evaluation of our proposed approach Deep Convolutional K-means Clustering (DCKM) discussed in Section 4.2.1 is carried out on three benchmark facial images datasets: Yale, Extended YaleB and ARFaces. The datasets details can

be referred from Section 1.3.1.

For all the datasets, the dense shift invariant feature transform (DSIFT) features were first extracted; then principal component analysis (PCA) was used to further reduce the dimensions to 300. This protocol was followed by several prior clustering studies [2, 46, 64, 126].

Our proposed formulation is compared with several benchmarks including Deep Learning friendly Clustering (DLC) [4], Deep K-Means (DKM) [5], Deep Clustering with Convolutional Autoencoder (DCEC) [8] and AutoEncoded K-Means (AEKM) [7]. The K-means algorithm with standard Euclidean distance as distance metric is also used for comparison.

Normalized Mutual Information (NMI), Adjusted Rand Index (ARI) and Accuracy are used as metrics [2, 127]. The values of μ and λ are set as 1 and 0.001 in all the experiments. For all the experiments, 3 filters of sizes 9x9 have been used in both the first and second layer of convolutions. The max-pooling kernel size is 2x2.

Table 4.1: DCKM: Clustering Results

Models	Yale				Extended YaleB				ARFaces			
	Acc	NMI	ARI	Time	Acc	NMI	ARI	Time	Acc	NMI	ARI	Time
K-Means	0.618	0.669	0.448	24	0.098	0.131	0.014	1769	0.146	0.457	0.047	377
DKM	0.338	0.430	0.158	397	0.087	0.125	0.010	7789	0.133	0.449	0.042	1414
DLC	0.386	0.477	0.186	428	0.098	0.151	0.016	5436	0.138	0.455	0.045	1418
AEKM	0.376	0.462	0.169	124	0.103	0.103	0.018	2536	0.137	0.456	0.045	522
DCEC	0.539	0.624	0.377	2112	0.295	0.453	0.173	24367	0.074	0.26	0.02	7008
Proposed	0.649	0.708	0.510	51	0.349	0.448	0.132	2417	0.159	0.463	0.051	757

The results are shown in Table 4.1. For Yale and AR Faces, our proposed method yields the best results; for Extended YaleB, our results are a close second.

The Yale and the AR Face datasets are more challenging compared to Extended YaleB. This is because the first two have a larger number of clusters (an order of magnitude higher than Extended YaleB) and fewer images (an order of magnitude lower than Extended YaleB). On these two challenging datasets, our method performed better than the existing benchmarks. In the relatively simpler case (Extended YaleB), our method is slightly worse than DCEC in terms of ARI and NMI but is better in terms of accuracy. It is interesting to note that existing deep learning algorithms are doing worse than the simple K-means specially in data-constrained scenarios.

The runtime (in seconds) for various techniques are shown in Table 4.1 under Time column. Unsurprisingly K-means is the fastest, our proposed method is in the same order as that of K-means and is much faster than the rest of the deep learning based clustering techniques.

Ablation Studies

Table 4.2: DCKM: Ablation Studies Results

	Yale				Extended YaleB				ARFaces			
Metric	Prop1L	Piece1L	Prop2L	Piece2L	Prop1L	Piece1L	Prop2L	Piece2L	Prop1L	Piece1L	Prop2L	Piece2L
Acc	0.620	0.588	0.649	0.636	0.146	0.135	0.349	0.320	0.142	0.122	0.159	0.151
NMI	0.686	0.648	0.708	0.691	0.204	0.193	0.448	0.432	0.453	0.434	0.463	0.456
ARI	0.447	0.412	0.510	0.462	0.033	0.031	0.132	0.123	0.044	0.040	0.051	0.048

This section shows how the results vary with the number of layers. The study also illustrates the variation in results when solving the problem jointly (as proposed) compared to the piecemeal solution. In the piecemeal approach, features are generated from (deep) convolutional transform learning, and these

features are subsequently fed into K-means clustering. The results are shown in Table 4.2. The joint solution, be it one layer or two-layer, yields better results than the piecemeal solution. This is expected; even in the past, jointly formulated solutions yielded better results than piecemeal ones. For both the piecemeal and joint solutions, going deeper helps, that is, the results obtained from two layers are always better than the ones from one layer.

Finally the empirical convergence plot is shown in figure 4.5. It is observed that the proposed algorithm converges within 50 iterations. The convergence plot for other depths show a similar trend.

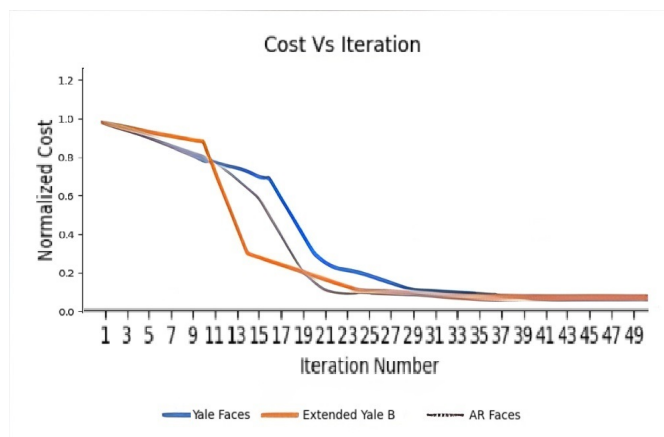


Figure 4.5: DCKM: Empirical Convergence Plot

4.3.1.2 Results on Contrastive Deep Convolutional Transform K-means Clustering

The evaluation of our proposed approach Contrastive Deep Convolutional Transform K-means Clustering discussed in Section 4.2.3 is carried out on five benchmark facial images datasets namely Yale Faces, Extended YaleB, ARFaces, Olivetti Faces and Pixraw10P datasets. The datasets details can be referred from Section 1.3.1.

Since the number of clusters (classes) for the given datasets are known, When conducting experiments, the standard clustering evaluation metrics namely Accuracy [30], Normalized Mutual Information (NMI) [30], and Adjusted Rand Index (ARI) [31], are commonly employed. The evaluation metrics details can be referred from Section 1.4.1.

The proposed model is compared with five state-of-the-art models to evaluate the performance. The benchmark models are briefly described as follows:

- K-Means: This refers to the standard K-means clustering algorithm [128] with Euclidean distance as the distance metric.
- Deep Clustering Network (DCN): This refers to the Deep Clustering Network proposed in [4]. DCN framework learns the deep representations via Deep Neural Network and apply K-means clustering on the learned deep representations. DCN optimizes the deep representations and the cluster assignments in an alternate fashion.
- Deep K-Means (DKM): DKM refers to Deep K-Means approach proposed in [5] that jointly learns the deep representations via auto-encoders and K-means clustering.
- Efficient Deep Embedded Subspace Clustering (EDESC): EDESC [129] is a deep learning based subspace clustering approach that learns the subspace bases from deep representation in an iterative refining manner.
- Contrastive Deep Embedded Clustering (CDEC): CDEC [1] embeds the con-

trastive loss in the stacked denoising auto-encoders to learn discriminative latent representations by optimizing the reconstruction loss and contrastive loss. In CDEC, the KL-divergence is optimized as the clustering loss.

In the experiments, Adam Optimizer is used, the values of ϵ and μ both are set as 10^{-4} for all the datasets. All other hyperparameters' values are grid-searched and the ones that gave best results are set as the final values. The final values for these hyperparameters are summarized in Table 4.8.

Table 4.3: Contrastive DCTLK: Hyperparameters Settings

Parameter	Yale	EYaleB	ARFaces	Olivetti	Pixraw10P
# Epochs	50	50	50	50	110
Learning Rate	1e-4	1e-3	1e-4	1e-3	1e-5
Kernel Sizes ¹	(9,9)	(3,3)	(3,3)	(3,3)	(3,3)
#Kernel Filters ²	(9,9)	(64,64)	(32,32)	(64,64)	(64,64)
β^3	1.0	1.0	1.0	1.0	1.0
δ^4	0.8	1.0	1.0	1.0	1.0

¹ Kernel Filters size for DCTL layers 1, 2

² Number of Kernel Filters for DCTL layers 1, 2

³ K-Means loss regularizer

⁴ Contrastive loss regularizer

The performance of the proposed framework is compared with five state-of-the-art approaches including the K-means algorithm, Deep Clustering Network (DCN), Deep K-Means (DKM), Efficient Deep Embedded Subspace Clustering (EDESC), and Contrastive Deep Embedded Clustering (CDEC) on five facial images datasets namely Yale Faces, Extended YaleB, ARFaces, Olivetti Faces, and Pixraw10P Faces dataset. The results are shown in Table 4.4. The results demonstrate that the proposed framework outperforms all the benchmarks with significant margin for all the datasets. The EDESC approach shows the second best results for ARFaces and Pixraw10P datasets while CDEC shows the second

best result for Extended YaleB dataset. For Yale Faces and Olivetti Faces datasets, K-means algorithm gives the second best results.

The loss plots of the proposed framework for the three datasets: Yale Faces, Olivetti Faces and Pixraw10P Faces are shown in figure 4.6. The proposed framework converges within 50 epochs. For Pixraw10P faces, the convergence is reached in 110 epochs due to the lower learning rate.

Table 4.4: Contrastive DCTLK: Clustering Results

Models	Metrics	Yale	EYaleB	ARFaces	Olivetti	Pixraw10P
K-means	Acc	0.618	0.098	0.146	0.612	0.91
	NMI	0.669	0.131	0.457	0.779	0.92
	ARI	0.448	0.014	0.047	0.464	0.83
DCN ¹	Acc	0.386	0.098	0.138	0.275	0.674
	NMI	0.477	0.151	0.455	0.513	0.776
	ARI	0.186	0.016	0.045	0.092	0.555
DKM ²	Acc	0.338	0.087	0.133	0.286	0.608
	NMI	0.430	0.125	0.449	0.523	0.752
	ARI	0.158	0.010	0.042	0.101	0.484
EDESC ³	Acc	0.588	0.113	0.147	0.53	0.94
	NMI	0.648	0.168	0.453	0.727	0.93
	ARI	0.408	0.024	0.043	0.386	0.87
CDEC ⁴	Acc	0.521	0.146	0.132	0.407	0.82
	NMI	0.577	0.233	0.423	0.631	0.87
	ARI	0.302	0.042	0.024	0.221	0.74
Proposed	Acc	0.685	0.214	0.159	0.642	0.94
	NMI	0.71	0.271	0.462	0.811	0.95
	ARI	0.511	0.052	0.053	0.493	0.89

¹ Deep Clustering Network

² Deep K-means

³ Efficient Deep Embedded Subspace Clustering

⁴ Contrastive Deep Embedded Clustering

Ablation Studies

In this section, five ablation studies are performed on the various important hyperparameters using three datasets namely Yale Faces, Olivetti Faces and Pixraw10P Faces.

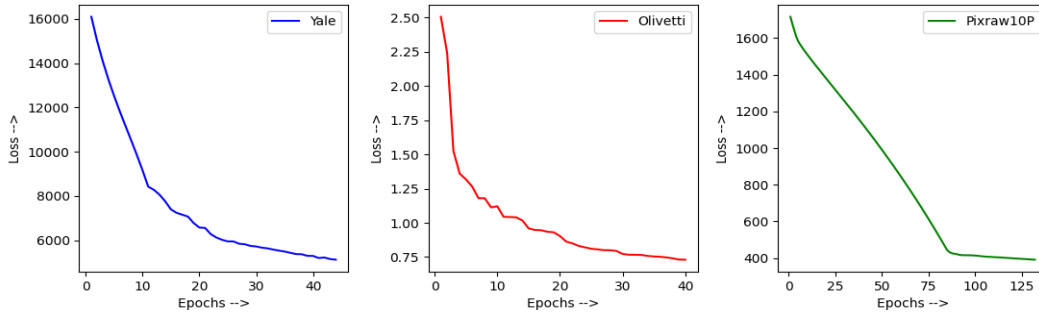


Figure 4.6: Contrastive DCTLK: Empirical Convergence Plots

4.3.1.3 Ablation on K-means clustering loss Regularizer (β)

The first ablation study experimented on the value of K-means clustering loss regularizer in the final loss function of the proposed framework which is represented by β in equation 4.27. The values of β are taken in the range of [0.2, 1.2]. The results are shown in figure 4.7. From the results, it can be observed that all the three datasets are giving best results for $\beta = 1.0$. This signifies the importance of K-means clustering loss term in the final loss function.

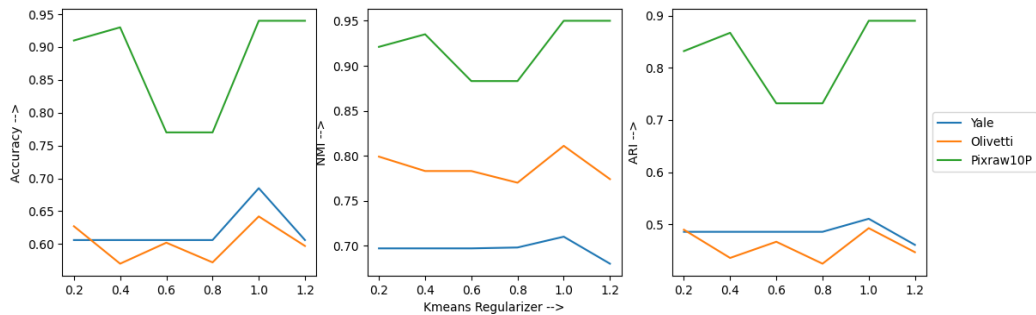


Figure 4.7: Contrastive DCTLK: Ablation Results on K-means loss regularizer (β)

4.3.1.4 Ablation on Contrastive loss Regularizer (δ)

The second ablation study experimented on the value of Contrastive loss regularizer in the final loss function of the proposed framework which is represented by δ in equation 4.27. The values of δ are taken in the range of [0.2, 1.2]. The results are shown in figure 4.8. From the results, it can be observed that Olivetti Faces and Pixraw10P Faces datasets have shown the best performance for $\delta = 1.0$ while Yale Faces is giving the best results for $\delta = 0.8$. This means that a higher weightage to the contrastive loss term in the final loss function helps in achieving better clustering performance.

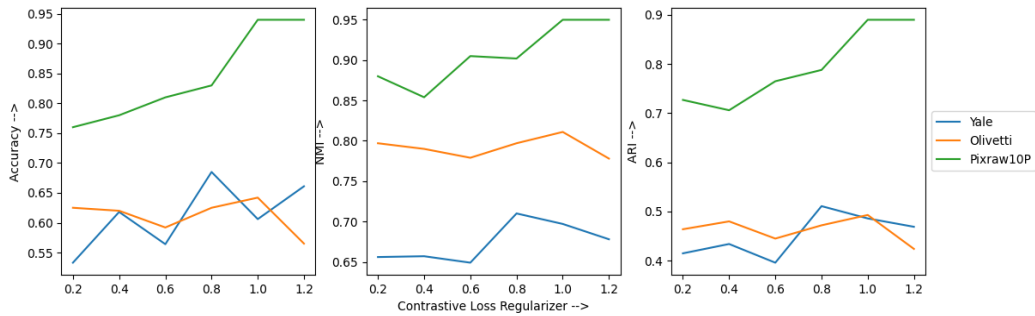


Figure 4.8: Contrastive DCTLK: Ablation Results on Contrastive loss regularizer (δ)

4.3.1.5 Ablation on Learning Rate

The third ablation study is performed on the learning rate. Learning rate is considered as an important hyperparameter that affects the final results. The values of learning rate are considered in the range of $[10^{-5}, 10^{-1}]$ for conducting the experiments in this ablation study. The results are shown in figure 4.9. From the results, it can be interpreted that the Yale Faces dataset gives the best results

with the learning rate value 10^{-4} while the Olivetti Faces and Pixraw10P Faces dataset give the best performance with the learning rate values of 10^{-3} and 10^{-5} respectively.

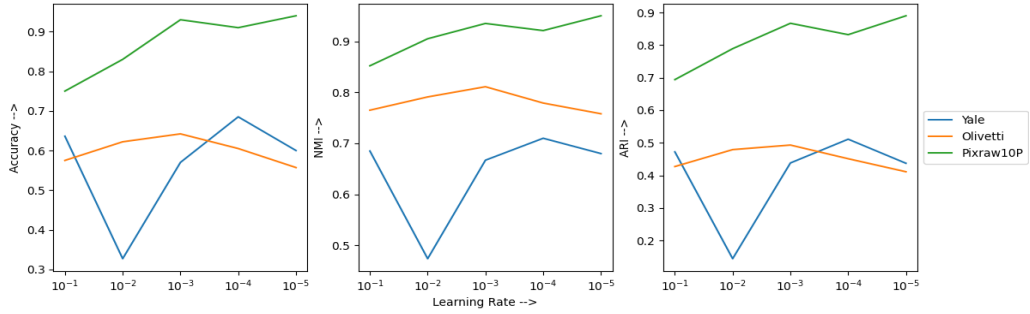


Figure 4.9: Contrastive DCTLK: Ablation Results on Learning rate

4.3.1.6 Ablation Studies Results on μ, ϵ regularizers

This ablation study experimented with the regularizers μ and ϵ associated with the penalty terms log-det and frobenius norms in the final loss function of the proposed framework formulated in equation 4.27. The set of values taken as a combination for both the penalty regularizers μ and ϵ respectively are - $((10^{-2}, 10^{-3}), (10^{-3}, 10^{-4}), (10^{-4}, 10^{-4}), (10^{-4}, 10^{-3}), (10^{-3}, 10^{-2}))$. The metric-wise results for all three datasets are shown in in figure 4.10. This indicates that the proposed framework is robust for the regularizers of the penalization terms since the results of Yale Faces and Pixraw10P Faces have not shown any variation for different values of μ and ϵ regularizers. The performance of Olivetti Faces dataset slightly varies for different values of μ and ϵ regularizers and gives the best performance for $\mu = 10^{-4}$ and $\epsilon = 10^{-4}$.

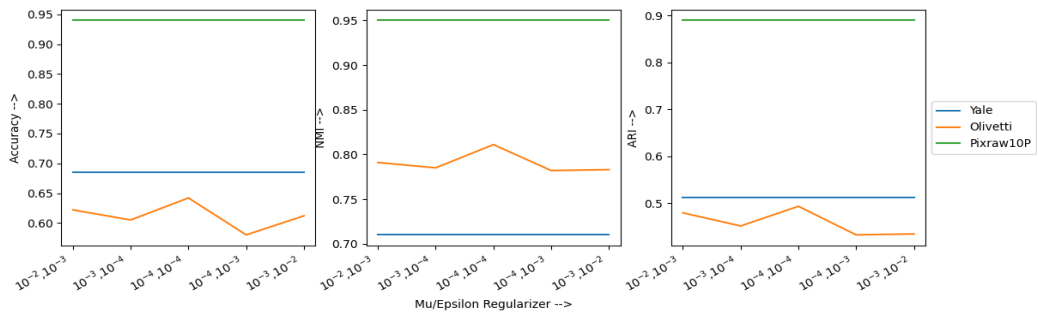


Figure 4.10: Contrastive DCTLK: Ablation Results on μ, ϵ regularizers

4.3.1.7 Ablation on K-means clustering loss and Contrastive Loss

The last ablation study is performed on the loss functions: K-means clustering loss and Contrastive loss. There are four variants of the proposed framework which are evaluated in this experiment. The first two variants are based on the piecemeal approach where the representations are first learned using the deep convolutional transform filters and once the representations are learned, they are fed to the K-means clustering to obtain the final clusters. Therefore, in the piecemeal version, β is set to 0 and thus, the K-means clustering loss is not backpropagated. In the piecemeal versions, the first variant is without contrastive loss while the second variant is with contrastive loss i.e. in the first variant, both β and δ are set to 0 while in the second variant only β is set to 0 while δ is set to the best value of contrastive loss regularizer obtained in Section 5.2.2. The third and fourth variants show the importance of contrastive loss in the proposed framework where the third variant is without contrastive loss i.e. δ is set to 0 while the fourth variant is with the contrastive loss. The results are shown in Table 4.5. The results show that the best results are obtained in the fourth variant where both the K-means clustering loss and contrastive loss are optimized. This

signifies that both the K-means clustering loss and contrastive loss are crucial in the proposed framework for achieving the best clustering performance.

Table 4.5: Contrastive DCTLK: Ablation Results on K-means loss and Contrastive loss

Models	Metrics	Yale	Olivetti	Pixraw10P
Piecemeal w/o Contrastive loss	Acc	0.594	0.577	0.59
	NMI	0.693	0.769	0.764
	ARI	0.449	0.42	0.449
Piecemeal with Contrastive loss	Acc	0.679	0.59	0.77
	NMI	0.702	0.782	0.883
	ARI	0.499	0.447	0.732
Proposed w/o Contrastive loss	Acc	0.649	0.58	0.67
	NMI	0.708	0.774	0.709
	ARI	0.510	0.409	0.379
Proposed with Contrastive loss	Acc	0.685	0.642	0.94
	NMI	0.71	0.811	0.95
	ARI	0.511	0.493	0.89

4.3.2 Hyperspectral Imaging

The proposed approach Deep Convolutional Sparse Subspace Clustering discussed in Section 4.2.2 is evaluated on two benchmark hyperspectral images datasets: Indian Pines and Pavia University. The datasets details and pre-processing steps can be referred from Section 1.3.2.

There is no straightforward way to evaluate the efficacy of the selected bands. Therefore, it must be evaluated based on some other criterion. Here, classification is used as the evaluation criterion, i.e., after band selection, the selected bands are passed onto a classifier. In particular, kernel sparse representation classifier [77] is used since it is a highly cited work on hyperspectral classification. The same was followed in our previous work discussed in Section 3.4.2.

In choosing the benchmarks, diversity in the selection strategy is maintained.

The first one is based on ranking – Similarity Based Ranking (SBR) [130]. The second one is based on Searching – FastVGBS [131]. The third one is a hybrid approach based on Dilation Distances and Ranking (DDR) [132]. The final one is deep learning formulation based on attention networks – DARecNet [133]. As our method is based on SSC, SSC is also used as a benchmark. Our previous work Deeply Transformed K-Means clustering (DTKM) (discussed in Section 3.4.2) is also used as a benchmark.

Our proposed technique requires the specification of four hyperparameters — λ , μ , β and χ . The value of both λ and μ is set as 1 for all the experiments; this value has been found to work well on almost all transform learning-based formulations. The parameter μ controls the relative importance of the deep transform learning and the SSC costs. There is no reason to give more weightage to one than the other; therefore, μ is set to 1. The fourth parameter χ controls the sparsity; as the number of clusters is relatively few (compared to the size of the image) the number of samples falling in each cluster will be moderate. Keeping that in mind, χ value is set as 0.2 in all the experiments; perhaps trying to optimize it could have given somewhat better results.

In the experiments, overlapping patches of size 16x16 are used. The patches are obtained by shifting the patches 1 pixel in both horizontal and vertical directions, as and when needed. The proposed architecture with three layers is used. For all the experiments, 3 filters of sizes 5x5 have been used in both the first, second, and third layers of convolutions. The maxpooling kernel size is 2x2.

In this work, an extremely challenging evaluation protocol shown in [134] is followed. For each class, only 15 labeled samples are used for training and the rest for testing. These 15 samples are randomly picked, and 100 such trials are carried out. The means and the standard deviations are being reported in the Table 4.6. For measuring classification accuracy, three standard measures in hyperspectral imaging – Overall Accuracy, Average Accuracy (AA), and Kappa coefficient are used.

Table 4.6: DCTLSSC: Results (Mean \pm Standard Deviation)

Metric \downarrow	Datasets \rightarrow	Pavia University			Indian Pines		
	# Bands \rightarrow	10	20	30	10	20	30
	Algorithms \downarrow						
Overall Accuracy	DTKM	0.94 \pm 0.16	0.93 \pm 0.13	0.91 \pm 0.12	0.83 \pm 0.13	0.82 \pm 0.11	0.79 \pm 0.11
	SBR	0.81 \pm 0.11	0.82 \pm 0.11	0.80 \pm 0.09	0.73 \pm 0.09	0.73 \pm 0.08	0.71 \pm 0.07
	FastVGBS	0.72 \pm 0.09	0.73 \pm 0.09	0.70 \pm 0.08	0.63 \pm 0.08	0.64 \pm 0.07	0.62 \pm 0.06
	DDR	0.78 \pm 0.12	0.80 \pm 0.11	0.79 \pm 0.11	0.68 \pm 0.12	0.69 \pm 0.12	0.68 \pm 0.11
	DARecNet	0.93 \pm 0.15	0.92 \pm 0.14	0.92 \pm 0.13	0.82 \pm 0.13	0.81 \pm 0.13	0.80 \pm 0.11
	SSC	0.69 \pm 0.07	0.71 \pm 0.07	0.71 \pm 0.06	0.62 \pm 0.07	0.64 \pm 0.06	0.63 \pm 0.05
	Proposed	0.95 \pm 0.12	0.94 \pm 0.12	0.93 \pm 0.12	0.85 \pm 0.11	0.84 \pm 0.12	0.82 \pm 0.10
Average Accuracy	DTKM	0.91 \pm 0.14	0.91 \pm 0.13	0.88 \pm 0.13	0.80 \pm 0.12	0.80 \pm 0.12	0.76 \pm 0.10
	SBR	0.78 \pm 0.10	0.79 \pm 0.10	0.77 \pm 0.09	0.71 \pm 0.09	0.71 \pm 0.09	0.77 \pm 0.09
	FastVGBS	0.71 \pm 0.08	0.71 \pm 0.07	0.70 \pm 0.06	0.63 \pm 0.08	0.64 \pm 0.08	0.62 \pm 0.07
	DDR	0.76 \pm 0.11	0.77 \pm 0.11	0.76 \pm 0.09	0.69 \pm 0.10	0.70 \pm 0.09	0.70 \pm 0.09
	DARecNet	0.90 \pm 0.14	0.90 \pm 0.13	0.89 \pm 0.13	0.81 \pm 0.13	0.80 \pm 0.13	0.78 \pm 0.11
	SSC	0.67 \pm 0.06	0.68 \pm 0.06	0.68 \pm 0.05	0.59 \pm 0.06	0.62 \pm 0.05	0.60 \pm 0.05
	Proposed	0.93 \pm 0.13	0.93 \pm 0.12	0.91 \pm 0.12	0.84 \pm 0.12	0.82 \pm 0.11	0.80 \pm 0.11
Kappa Coefficient	DTKM	0.88 \pm 0.14	0.87 \pm 0.14	0.84 \pm 0.13	0.74 \pm 0.13	0.72 \pm 0.12	0.69 \pm 0.11
	SBR	0.75 \pm 0.10	0.76 \pm 0.08	0.73 \pm 0.07	0.64 \pm 0.08	0.65 \pm 0.07	0.64 \pm 0.07
	FastVGBS	0.64 \pm 0.09	0.65 \pm 0.09	0.63 \pm 0.08	0.59 \pm 0.08	0.60 \pm 0.08	0.59 \pm 0.08
	DDR	0.71 \pm 0.11	0.73 \pm 0.12	0.71 \pm 0.11	0.62 \pm 0.12	0.63 \pm 0.11	0.63 \pm 0.10
	DARecNet	0.86 \pm 0.14	0.85 \pm 0.12	0.85 \pm 0.11	0.73 \pm 0.14	0.71 \pm 0.13	0.70 \pm 0.13
	SSC	0.62 \pm 0.06	0.64 \pm 0.06	0.63 \pm 0.05	0.55 \pm 0.06	0.57 \pm 0.05	0.56 \pm 0.04
	Proposed	0.89 \pm 0.12	0.89 \pm 0.12	0.87 \pm 0.12	0.77 \pm 0.13	0.76 \pm 0.12	0.74 \pm 0.12

The running time of different algorithms are shown in Table 3.5. One can see that DTKM, DARecNet and our proposed are slower than the rest. SSC is the fastest. This is understandable since the aforesaid are based on deep learning while others are not. However, speed comes at the expense of accuracy. Note that, while comparing with our previous work DTKM, the results are slightly

different; this is because the patch size chosen in DTKM and the one used here are different.

Table 4.7: DCTLSSC: Runtime comparison (in seconds)

	Pavia University			Indian Pines		
# Bands →	10	20	30	10	20	30
Algorithms ↓						
DTKM	505	781	1004	229	396	501
SBR	257	306	433	130	199	226
FastVGBS	109	139	166	63	80	91
DDR	163	188	204	105	131	155
DARecNet	794	1107	1569	402	564	732
SSC	106	112	127	61	63	68
Proposed	329	404	530	154	187	215

The general observation is that the deep learning based techniques that are slower (observed in Table 3.5) are indeed more accurate than others. Our method outperforms the benchmarks including DTKM. One can notice that addition of convolutional transforms vastly improves the results over vanilla SSC. For all the algorithms, the best results are obtained between 10 to 20 bands and results deteriorate when one goes up to 30 bands. The same was observed in prior studies like in our previous work DTKM and DARecNet [134]. It is observed that the algorithms that give higher accuracy also show higher deviations. Furthermore, it is noticed that the deviations in general decrease for all the techniques with the increase in the number of bands.

It is interesting to compare our proposed work with our previous work DTKM. The prior study used a fully connected network; empirical observations in deep learning have established that convolutional filters typically yield better results than fully connected networks, therefore our improvement was expected. The other deep learning based method DARecNet [133] performs almost at par

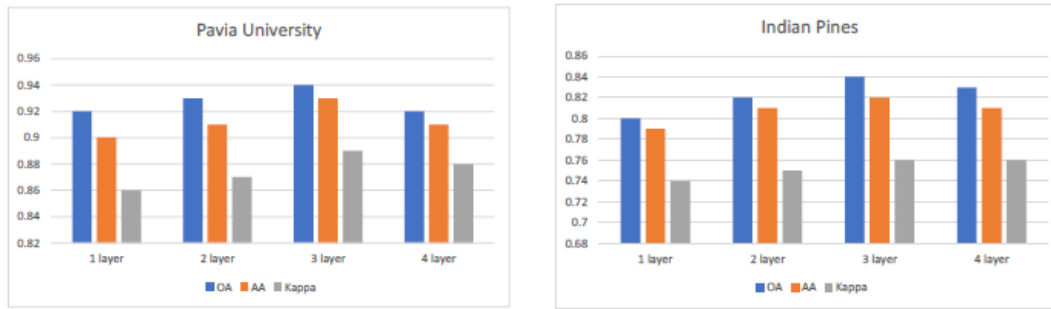


Figure 4.11: DCTLSSC: Ablation Results

with DTKM. This may owe to two reasons. In terms of architecture, it uses state-of-the-art attention networks hence one can expect good performance. Unfortunately, it does not have an embedded band selection strategy, like DTKM or ours; perhaps this is the reason it was below par.

4.3.2.1 Ablation Study

This set of experiments analyze the effect of depth on the performance of our proposed method. The depth of convolutional transform layers is increased from one to four and see how the performance metrics vary. The number of chosen bands is kept constant at 20. The number of filters and the size of filters remain as before. The results are shown in figure 4.11.

One can see that the results improve from Layers one to three, but then dips. The deterioration from layers three to four is likely due to overfitting; there is not enough data to learn further layers [81].

4.3.3 Multiview datasets

The performance of our proposed approach DeConFCluster (discussed in Section 4.2.4) is illustrated on various multiview clustering datasets - 100leaves, ALOI, Mfeat and WebKB. The datasets details can be referred from Section 1.3.4.

4.3.3.1 Hyperparameters Settings

Stochastic Gradient Descent (SGD) is used as the optimizer, $\lambda = 0.01$, $\mu = 0.0001$ and weight decay as 0.001 for all the datasets. Another hyperparameter - feature_ratio is also defined that indicated the percentage of features kept in the final representation Z . All other hyperparameters can be referred from Table 4.8.

Table 4.8: DeConFCluster: Hyperparameters settings

Parameter	100leaves	WebKB	Mfeat	ALOI
Batch size	1600	203	128	11025
Epochs	25	25	40	25
Learning Rate	5e-6	1e-4	1e-4	5e-6
Kernel Sizes ¹	(3,3,3)	(3,3,3)	(5,3,3)	(3,3,3)
#Filters ²	(4,8,16)	(4,8,16)	(2,4,8)	(4,8,16)
feature_ratio	0.15	0.15	0.25	0.25
β^3	1.0	0.5	0.8	0.5

¹ Kernel sizes for DCTL layers 1,2,3

² #Filters for DCTL layers 1,2,3

³ K-Means loss regularizer

4.3.3.2 Benchmarks Used

The results are compared with four state-of-the-art works. The compared models are briefly described here as follows:

- MCGL: It is a graph based learning method. Starting graphs were learned

using different views' data points that were further optimized with a rank constraint on the Laplacian matrix. Next, optimized graphs were integrated into a global graph. The graph was learned with the same rank constraint on its Laplacian matrix. Cluster indicators were obtained from the global graph only without conducting any graph cut technique and the K-Means clustering [135].

- GMC: In this approach, each view was weighted and the SIG matrices and the unified graph matrix were jointly learned [28]. The latter was obtained by the fusion of the graph matrices of each view.
- DEMVC: This method proposed a framework based on autoencoders. It utilized complementary and consensus information from multiple views and learned the deep latent feature representations and clustering assignments in a collaborative manner [91].
- RRA-MVC: This technique proposed a simple baseline model (SiMVC) that aligned the distributions of the views. Further, it added the contrastive module and selective views alignment by prioritizing the views and, thus, improved the baseline model's performance calling it as CoMVC framework [136].

4.3.3.3 Metrics Used

The performance of the proposed model is evaluated using three metrics - Accuracy, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI).

4.3.3.4 Results Analysis

The results of our model and benchmarks on all the four datasets are reported in Table 4.9. It can be observed from Table 4.9 that for all the datasets, the proposed model has shown better performance than the state-of-the-arts specially in the case of 100leaves and WebKB datasets. The proposed model achieves a significant improvement of 10.62% and 5.81% in the accuracy over the second best performing benchmark for 100leaves and WebKB datasets respectively. For Mfeat dataset, the proposed model outperforms all the benchmarks in terms of accuracy and ARI. The NMI value of the proposed model is slightly worse than the NMI value of the benchmark models MCGL [135] and GMC [28]. For ALOI dataset, the proposed model performs better than all the state-of-the-arts in terms of accuracy. For the other two metrics NMI and ARI, the proposed model is the second best performer after the benchmark RRA-MVC [136]. This is probably because the feature variability across classes is low in case of Mfeat and ALOI datasets. It is worth noting that the proposed model performed well in the case of 100leaves and WebKB, both of which have fewer samples per class. For example, in 100leaves dataset, there are only 16 images available for each class. It is quite challenging for a model to learn the feature representations using few samples per class. Thus, the proposed method performed well for challenging datasets and slightly worse for the ones where we have relatively higher number of samples per class.

Also, the convergence plot for all the datasets were plotted that can be referred

Table 4.9: DeConFCluster: Clustering Results (All the metrics are in (%))

Models	100leaves			WebKB			Mfeat			ALOI		
	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
MCGL	81.06	91.30	51.50	54.19	8.60	4.01	85.30	90.55	83.13	46.25	66.57	4.41
GMC	82.38	92.92	49.74	76.35	41.64	42.80	88.20	90.50	85.02	57.05	73.50	43.05
DEMVC	6.69	24.53	0.60	49.75	10.05	8.43	46.45	37.53	24.59	13.52	41.30	8.45
RRA-MVC	73.25	92.56	71.58	40.89	13.43	9.22	81.20	83.19	74.36	55.22	80.79	49.34
Proposed	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84

from figure 4.12. Using SGD as an optimizer, it could be clearly inferred that the given solution converged to the point of stability. The SGD parameters, such as mini-batch size and learning rate, are given in Table 4.8 for all the considered datasets.

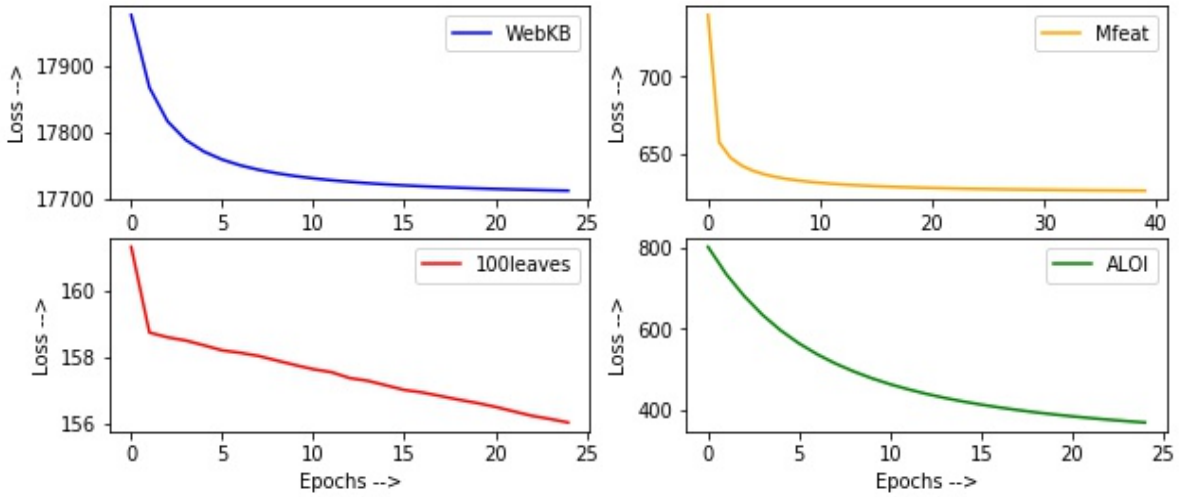


Figure 4.12: DeConFCluster: Empirical Convergence Plots

4.3.3.5 Ablation studies

This section shows the results corresponding to the three ablation studies performed for all the datasets. We first experimented with changing the values of the regularizers λ and μ associated with the penalty terms log-det and Frobenius norms in both CTL and TL equations 4.32 and 4.35 respectively. The set of values taken as a combination for both the penalty regularizers are -

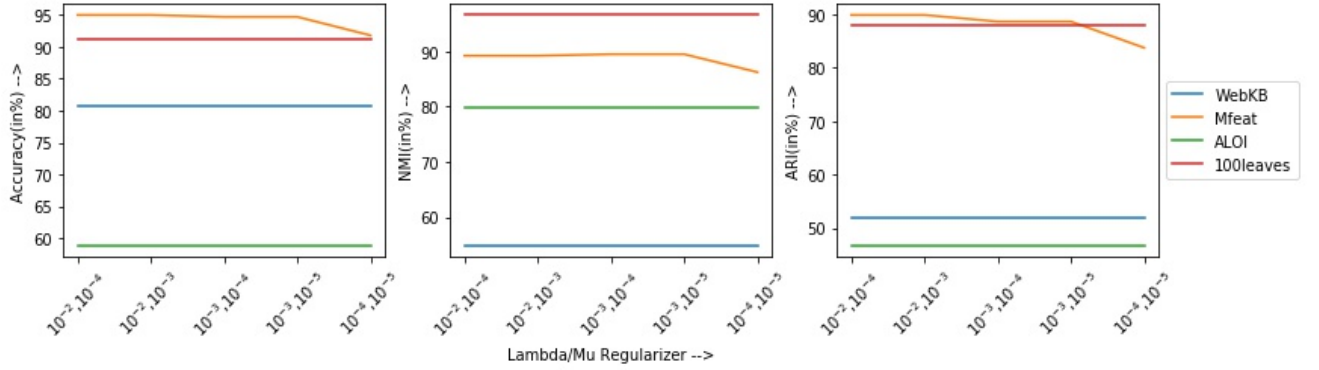


Figure 4.13: DeConFCluster: Ablation Results on λ, μ

$((10^{-2}, 10^{-4}), (10^{-2}, 10^{-3}), (10^{-3}, 10^{-4}), (10^{-3}, 10^{-5}), (10^{-4}, 10^{-5}))$. The results can be referred from Table 4.10. We also displayed them graphically for all three metrics Accuracy, NMI and ARI in figure 4.13. We concluded from the results that our method was robust for the regularizers of the penalization terms except for the Mfeat dataset. In the case of Mfeat, the results were depleted for lower values of regularizers. Thus, these penalizations played an essential role in our formulation and helped to learn better representations.

Table 4.10: DeConFCluster: Ablation Studies Results on λ, μ

Value (λ, μ)	100leaves ($\beta = 1.0$)			WebKB ($\beta = 0.5$)			Mfeat ($\beta = 0.8$)			ALOI ($\beta = 0.5$)		
	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
$(10^{-2}, 10^{-4})$	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84
$(10^{-2}, 10^{-3})$	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84
$(10^{-3}, 10^{-4})$	91.13	96.59	88.01	80.79	54.98	52.02	94.70	89.49	88.66	58.95	79.75	46.84
$(10^{-3}, 10^{-5})$	91.13	96.59	88.01	80.79	54.98	52.02	94.70	89.49	88.66	58.95	79.75	46.84
$(10^{-4}, 10^{-5})$	91.13	96.59	88.01	80.79	54.98	52.02	91.80	86.24	83.74	58.95	79.75	46.84

Secondly, we experimented with the regularizer β associated with K-Means clustering loss in the equation 4.36. The set of values for β lie in range $[0, 1]$, specifically, these are $(0.0, 0.1, 0.3, 0.5, 0.8, 1.0)$. We represented results both in text and graphically that can be referred from Table 4.11 and figure 4.14 respectively. It could be observed that for all the datasets, in general, the K-

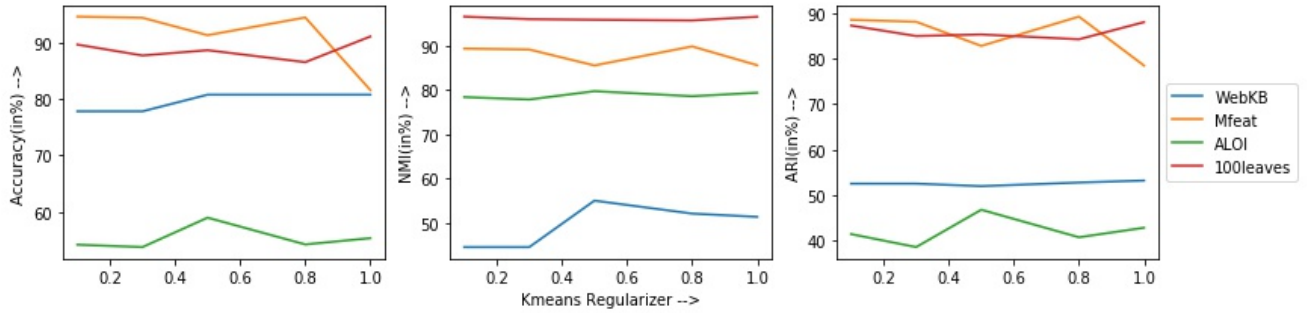


Figure 4.14: DeConFCluster: Ablation Results on K-Means Regularizer

Means regularizer $\beta \geq 0.5$ gave better performance. This signified that K-Means loss was an important term associated with the final objective function. It helped in learning better representations as guided by it and was thus responsible for better clustering performance.

Table 4.11: DeConFCluster: Ablation Studies Results on K-Means Regularizer

Value	100leaves			WebKB			Mfeat			ALOI		
β	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
0.0	89.56	96.17	86.50	77.83	44.47	52.57	91.10	85.37	82.15	55.27	78.34	41.16
0.1	89.69	96.62	87.26	77.83	44.47	52.57	94.65	89.39	88.51	54.15	78.41	41.49
0.3	87.75	96.05	84.96	77.83	44.47	52.57	94.45	89.18	88.08	53.72	77.87	38.65
0.5	88.69	95.91	85.30	80.79	54.98	52.02	91.35	85.57	82.79	58.95	79.75	46.84
0.8	86.56	95.76	84.24	80.79	52.05	52.81	95.00	89.22	89.89	54.20	78.61	40.80
1.0	91.13	96.59	88.01	80.79	51.32	53.24	81.60	85.62	78.46	55.31	79.40	42.89

Table 4.12: DeConFCluster: Ablation Studies Results on Piecemeal and Proposed Formulation

Methods	100leaves			WebKB			Mfeat			ALOI		
	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
Piecemeal	89.56	96.17	86.50	77.83	44.47	52.57	91.10	85.37	82.15	55.27	78.34	41.16
Proposed	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84

The second experiment inference was also validated by the third experiment conducted, where we performed the piecemeal version of our model. It means we first learned representations from the DeConFuse network separately and then passed the learned representations via the K-Means clustering module to get the final clusters, i.e., here $\beta = 0$. The results could be referred from Table 4.12. Here, it was clearly inferred that the joint optimization of the DeConFuse

and K-Means clustering module is better than the piecemeal approach.

4.4 Summary

This chapter proposes four novel deep clustering approaches based on CTL framework. The first proposed approach is Deep Convolutional K-means clustering that embeds K-means clustering in CTL framework with two layers of convolutional transform layers. The proposed approach is evaluated on three standard computer vision datasets, demonstrated higher clustering scores as compared to the current state-of-the-art deep clustering frameworks. The second proposed approach embeds SSC in K-means clustering in CTL framework. The proposed approach is evaluated on two standard hyperspectral imaging datasets, demonstrated higher performance as compared to the current state-of-the-art hyperspectral imaging approaches. The third work proposes a novel unsupervised multi-channel fusion clustering framework named DeConFCluster. The proposed framework jointly trains the DCTL based DeConFuse and K-Means clustering modules in an end-to-end fashion. The advantage of this framework is that it does not have the additional overhead of learning the weights of decoder or deconvolutional layers, which is the case in existing multiview clustering approaches. Another advantage of this framework is that it promotes diversity among filters and thus, in turn, helps to learn more interpretable filters that are further guided by K-Means loss. Therefore, due to these advantages, the proposed framework DeConFCluster, evaluated on four standard multiview datasets,

demonstrated higher clustering scores as compared to the current state-of-the-art multiview clustering frameworks. The last proposed approach is Contrastive Deep Convolutional Transform K-means clustering. This approach highlights the problem of existing deep clustering approaches being not able to capture discriminative information due to lack of supervision. In the fourth proposed approach, the said problem is alleviated by incorporating Contrastive learning in Deep Convolutional Transform K-means clustering. The proposed framework jointly trains the contrastive loss, DCTL loss and K-means clustering loss in a joint end-to-end fashion. The proposed approach is evaluated on five standard computer vision datasets, demonstrated higher clustering scores as compared to the current state-of-the-art deep clustering frameworks.

Chapter 5

Conclusion

This dissertation proposes novel deep clustering approaches based on dictionary learning, transform learning and convolutional transform learning frameworks.

5.1 Summary of Contribution

In this section, we will briefly summarize the chapter-wise contribution in the area of Deep Clustering, giving a bird's eye view of the dissertation.

5.1.1 Dictionary Learning based clustering approaches

In this part of the dissertation, we have modeled clustering approaches based on dictionary learning framework, namely - Dictionary Learning + K-means (DLK) and Dictionary Learning + Sparse subspace clustering (DLS). The proposed algorithms DLK and DLS are evaluated on computer vision datasets and both the algorithms obtained good performance. We have also proposed Deep Dictionary

Learning + K-means (DDLK) and Deep Dictionary Learning + Sparse subspace clustering (DDLK) algorithms. DDLK algorithm is applied for hyperspectral image classification problem and DDLK outperforms several state-of-the-art approaches of hyperspectral imaging.

5.1.2 Transform Learning based clustering approaches

This chapter presented the clustering approaches based on transform learning framework. Transform learning is the analysis equivalent of dictionary learning. In this chapter, we proposed two approaches namely, Transform Learning + K-means (TLK) and Deep Transform Learning + K-means (DTLK). The first framework TLK embeds the K-means clustering loss in Transform learning framework. TLK algorithm is applied for document clustering and achieved good results. DTLK algorithm is applied for hyperspectral band selection and achieved promising results as compared to other state-of-the-arts.

5.1.3 Convolutional Transform Learning based clustering approaches

In this chapter, we presented four Convolutional Transform Learning (CTL) based clustering approaches. The first proposed approach is Deep Convolutional K-means Clustering (DCKM) that embeds K-means clustering loss in Deep CTL (DCTL) framework and optimize the joint formulation in an end-to-end fashion. The performance of DCKM is evaluated on facial images datasets and achieves good performance. The second proposed approach incorporates Sparse Subspace

Clustering loss in DCTL framework. This approach is applied for hyperspectral band selection and achieved promising results.

The third proposed approach is a Multiview Clustering Framework based on CTL - DeConFCluster. The proposed DeConFCluster is an unsupervised multiview multi-channel fusion framework that performs multiview clustering task utilizing representations from the fusion of the individual view's representations; thus, it learns individual view information and then learns cross-channel information via fusion. The framework comprised a DeConFuse network (discussed in Section 4.2.4.2) and a K-means module that are jointly trained and optimized. Therefore, representations learned are beneficial since those have the advantages of CTL and are well-guided through the K-Means loss also. The same is observed in terms of performance from the experimental results too.

The fourth proposed clustering approach based on CTL is Contrastive Deep Convolutional Transform k-means clustering that leverages DCKM by incorporating the contrastive learning in the DCTL learnt representations. To embed the contrastive loss, the positive and negative pairs of data samples are generated with the input data and the reconstructed data from DCTL learnt representations. Then, the contrastive loss, DCTL loss and K-means clustering loss are jointly optimized together in end-to-end fashion. This ensures the DCTL learnt representations retain the discriminative information of input data features to generate better clusters. Our proposed model incorporates contrastive learning without relying on data augmentation to create positive and negative pairs of data samples, as many other contrastive learning-based methods do. This makes

the construction of contrastive sample pairs more convenient. Moreover, our proposed model introduces a skip connection while reconstructing the data samples from DCTL learnt representations to alleviate the degradation problem that may arise due to the deep layers in DCTL framework. Extensive experimental results on several facial images datasets demonstrate the superiority of our proposed model.

The most significant advantage of the proposed deep clustering approaches is that the proposed frameworks prevented the additional training from the decoder/deconvolutional network that is generally applied in the auto-encoders based deep clustering approaches. This significantly reduces the total number of learnable parameters. Therefore, the proposed frameworks avoid overfitting even in data-constrained scenarios where the number of data instances is low and the number of classes is high.

5.2 Future Work

Convolutional Dictionary Learning [137] extends the traditional dictionary learning framework to incorporate local spatial relationships within the data. Instead of using a single atom as in traditional dictionary learning, convolutional dictionary learning uses a set of filters that are convolved with the input signal. These filters capture local patterns and can be interpreted as the dictionary elements. It provides a powerful framework for capturing meaningful features and has been successfully applied in various areas of signal processing and computer vision.

In future, we will leverage Convolutional Dictionary Learning framework to integrate the clustering modules.

The visual and audio modalities exhibit a significant correlation while carrying distinct information. This strong correlation enables accurate prediction of the semantics of one modality from the other. Leveraging their inherent differences, cross-modal prediction emerges as a potentially more beneficial approach for self-supervised learning of video and audio representations, surpassing the benefits of learning within a single modality [138]. Therefore, in the future, we plan to extend the deep clustering approaches proposed in this dissertation to cross-modal clustering.

The robustness of the clustering network is susceptible to being attenuated especially when it encounters an adversarial attack. A small perturbation in the embedding space will lead to diverse clustering results since the labels are absent. In future, we plan to incorporate adversarial learning into the proposed deep clustering networks to enhance the clustering robustness. Adversarial learning generates an adversarial sample to learn a small perturbation in the embedding space which can fool the clustering layers but not impact the deep embedding. Then, contrastive learning [139] is utilized to force the results generated by the original embedding space features and their perturbed versions to be similar to improve the robustness and the overall performance of the clustering network [140].

References

- [1] G. Sheng, Q. Wang, C. Pei, and Q. Gao, “Contrastive deep embedded clustering,” *Neurocomputing*, vol. 514, pp. 13–20, 2022.
- [2] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, and Z. Yi, “Deep subspace clustering with sparsity prior.” in *IJCAI*, 2016, pp. 1925–1931.
- [3] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*. PMLR, 2016, pp. 478–487.
- [4] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” in *international conference on machine learning*. PMLR, 2017, pp. 3861–3870.
- [5] M. M. Fard, T. Thonet, and E. Gaussier, “Deep k-means: Jointly clustering with k-means and learning representations,” *Pattern Recognition Letters*, vol. 138, pp. 185–192, 2020.
- [6] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.

- [7] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, “Learning deep representations for graph clustering,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.
- [8] X. Guo, X. Liu, E. Zhu, and J. Yin, “Deep clustering with convolutional autoencoders,” in *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*. Springer, 2017, pp. 373–382.
- [9] B. Diallo, J. Hu, T. Li, G. A. Khan, X. Liang, and Y. Zhao, “Deep embedding clustering based on contractive autoencoder,” *Neurocomputing*, vol. 433, pp. 96–107, 2021.
- [10] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. Hoi, and Z. Xu, “Semi-supervised deep embedded clustering,” *Neurocomputing*, vol. 325, pp. 121–130, 2019.
- [11] L. Yu and W. Wang, “Dcsr: Deep clustering under similarity and reconstruction constraints,” *Neurocomputing*, vol. 411, pp. 216–228, 2020.
- [12] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, “Deep spectral clustering using dual autoencoder network,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4066–4075.
- [13] S. Huang, Z. Kang, Z. Xu, and Q. Liu, “Robust deep k-means: An effective and simple method for data clustering,” *Pattern Recognition*, vol. 117, p. 107996, 2021.
- [14] L. Yang, W. Fan, and N. Bouguila, “Deep clustering analysis via dual vari-

- ational autoencoder with spherical latent embeddings,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [15] A. Caciularu and J. Goldberger, “An entangled mixture of variational autoencoders approach to deep clustering,” *Neurocomputing*, vol. 529, pp. 182–189, 2023.
- [16] 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.
- [17] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.
- [18] L. Li and H. Kameoka, “Deep clustering with gated convolutional networks,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 16–20.
- [19] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, “A survey of clustering with deep learning: From the perspective of network architecture,” *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.
- [20] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, “Clustergan: Latent space clustering in generative adversarial networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4610–4617.

- [21] Z. Yu, Z. Zhang, W. Cao, C. Liu, C. P. Chen, and H.-S. Wong, “Gan-based enhanced deep subspace clustering networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 7, pp. 3267–3281, 2020.
- [22] D. P. de Mello, R. M. Assunção, and F. Murai, “Top-down deep clustering with multi-generator gans,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7770–7778.
- [23] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, “Contrastive clustering,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8547–8555.
- [24] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, “From few to many: Illumination cone models for face recognition under variable lighting and pose,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 643–660, 2001.
- [25] S. A. Nene, S. K. Nayar, H. Murase *et al.*, “Columbia object image library (coil-20),” 1996.
- [26] A. Martinez and R. Benavente, “The ar face database: Cvc technical report, 24,” 1998.
- [27] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, “Feature selection: A data perspective,” *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.
- [28] H. Wang, Y. Yang, and B. Liu, “Gmc: Graph-based multi-view clustering,”

- IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1116–1129, 2019.
- [29] X. Zhang, L. Zhao, L. Zong, X. Liu, and H. Yu, “Multi-view clustering via multi-manifold regularized nonnegative matrix factorization,” in *Proceedings of the IEEE International Conference on Data Mining (ICDM 2014)*, 2014, pp. 1103–1108.
- [30] 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*, 2003, pp. 267–273.
- [31] K. Y. Yeung and W. L. Ruzzo, “Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data,” *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.
- [32] F. A. Kruse, A. Lefkoff, J. Boardman, K. Heidebrecht, A. Shapiro, P. Barloon, and A. Goetz, “The spectral image processing system (sips)—interactive visualization and analysis of imaging spectrometer data,” *Remote sensing of environment*, vol. 44, no. 2-3, pp. 145–163, 1993.
- [33] G. M. Foody and A. Mathur, “A relative evaluation of multiclass image classification by support vector machines,” *IEEE Transactions on geoscience and remote sensing*, vol. 42, no. 6, pp. 1335–1343, 2004.

- [34] J. Cohen, “A coefficient of agreement for nominal scales,” *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.
- [35] Z. Yang and E. Oja, “Linear and nonlinear projective nonnegative matrix factorization,” *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 734–749, 2010.
- [36] A. Goel and A. Majumdar, “Clustering friendly dictionary learning,” in *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part I* 28. Springer, 2021, pp. 549–557.
- [37] S. Poonia, A. Goel, S. Chawla, N. Bhattacharya, P. Rai, Y. F. Lee, Y. S. Yap, J. West, A. A. Bhagat, J. Tayal *et al.*, “Marker-free characterization of full-length transcriptomes of single live circulating tumor cells,” *Genome Research*, vol. 33, no. 1, pp. 80–95, 2023.
- [38] A. Goel and A. Majumdar, “Sparse subspace clustering friendly deep dictionary learning for hyperspectral image classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.
- [39] —, “Transformed k-means clustering,” in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1526–1530.
- [40] —, “K-means embedded deep transform learning for hyperspectral band selection,” *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

- [41] A. Goel, A. Majumdar, E. Chouzenoux, and G. Chierchia, “Deep convolutional k-means clustering,” in *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 211–215.
- [42] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, “Deep dictionary learning,” *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.
- [43] C. Bauckhage, “K-means clustering is matrix factorization,” *arXiv preprint arXiv:1512.07548*, 2015.
- [44] E. Elhamifar and R. Vidal, “Sparse subspace clustering: Algorithm, theory, and applications,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.
- [45] 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.
- [46] J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia, “Deeply transformed subspace clustering,” *Signal Processing*, vol. 174, p. 107628, 2020.
- [47] V. Singhal, H. K. Aggarwal, S. Tariyal, and A. Majumdar, “Discriminative robust deep dictionary learning for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 5274–5283, 2017.
- [48] V. Singhal and A. Majumdar, “Row-sparse discriminative deep dictionary learning for hyperspectral image classification,” *IEEE Journal of Selected*

- Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 12, pp. 5019–5028, 2018.
- [49] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
- [50] J. Lei, X. Li, B. Peng, L. Fang, N. Ling, and Q. Huang, “Deep spatial-spectral subspace clustering for hyperspectral image,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2686–2697, 2020.
- [51] J. Sun, W. Wang, X. Wei, L. Fang, X. Tang, Y. Xu, H. Yu, and W. Yao, “Deep clustering with intraclass distance constraint for hyperspectral images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 5, pp. 4135–4149, 2020.
- [52] K. Li, Y. Qin, Q. Ling, Y. Wang, Z. Lin, and W. An, “Self-supervised deep subspace clustering for hyperspectral images with adaptive self-expressive coefficient matrix initialization,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 3215–3227, 2021.
- [53] J. Nalepa, M. Myller, Y. Imai, K.-i. Honda, T. Takeda, and M. Antoniak, “Unsupervised segmentation of hyperspectral images using 3-d convolutional autoencoders,” *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 11, pp. 1948–1952, 2020.

- [54] S. Ravishankar and Y. Bresler, “Sparsifying transform learning for compressed sensing mri,” in *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE, 2013, pp. 17–20.
- [55] S. Ravishankar, B. Wen, and Y. Bresler, “Online sparsifying transform learning - Part I,” *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 625–636, 2015.
- [56] T. Blumensath and M. E. Davies, “Iterative thresholding for sparse approximations,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, Dec 2008. [Online]. Available: <https://doi.org/10.1007/s00041-008-9035-z>
- [57] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [58] S. Ravishankar and Y. Bresler, “Online sparsifying transform learning - Part II,” *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 637–646, 2015.
- [59] J. Maggu and A. Majumdar, “Unsupervised deep transform learning,” in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 6782–6786.
- [60] J. Maggu and A. Majumdar, “Kernel transform learning,” *Pattern Recognition Letters*, vol. 98, pp. 117–122, 2017.

- [61] J. Maggu and A. Majumdar, “Robust transform learning,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 1467–1471.
- [62] J. Maggu and A. Majumdar, “Greedy deep transform learning,” in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1822–1826.
- [63] J. Maggu, A. Majumdar, and E. Chouzenoux, “Transformed locally linear manifold clustering,” in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1057–1061.
- [64] ———, “Transformed subspace clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1796–1801, 2020.
- [65] J. Maggu and A. Majumdar, “Kernelized transformed subspace clustering with geometric weights for non-linear manifolds,” *Neurocomputing*, vol. 520, pp. 141–151, 2023.
- [66] R. Vidal, “Subspace clustering,” *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.
- [67] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [68] J. Maggu, H. K. Aggarwal, and A. Majumdar, “Label-consistent transform learning for hyperspectral image classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 9, pp. 1502–1506, 2019.

- [69] V. Singhal, J. Maggu, and A. Majumdar, “Simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep dictionary learning and deep transform learning,” *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2969–2978, 2018.
- [70] J. Maggu and A. Majumdar, “Semi-coupled transform learning,” in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part III 25*. Springer, 2018, pp. 141–150.
- [71] S. Nagpal, M. Singh, R. Singh, M. Vatsa, A. Noore, and A. Majumdar, “Face sketch matching via coupled deep transform learning,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5419–5428.
- [72] Y. Wang, W. Yin, and J. Zeng, “Global convergence of admm in nonconvex nonsmooth optimization,” *Journal of Scientific Computing*, vol. 78, pp. 29–63, 2019.
- [73] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms,” *IEEE Transactions on Signal Processing*, vol. 61(5), pp. 1072–1086, 2012.
- [74] B. Hanin, “Universal function approximation by deep neural nets with bounded width and relu activations,” *Mathematics*, vol. 7, no. 10, p. 992, 2019.
- [75] X. Pei, C. Chen, and W. Gong, “Concept factorization with adaptive neigh-

- bors for document clustering,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 343–352, 2016.
- [76] H. Kim, H. K. Kim, and S. Cho, “Improving spherical k-means for document clustering: Fast initialization, sparse centroid projection, and efficient cluster labeling,” *Expert Systems with Applications*, vol. 150, p. 113288, 2020.
- [77] Y. Chen, N. M. Nasrabadi, and T. D. Tran, “Hyperspectral image classification via kernel sparse representation,” *IEEE Transactions on Geoscience and Remote sensing*, vol. 51, no. 1, pp. 217–231, 2012.
- [78] M. Zeng, Y. Cai, Z. Cai, X. Liu, P. Hu, and J. Ku, “Unsupervised hyperspectral image band selection based on deep subspace clustering,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 12, pp. 1889–1893, 2019.
- [79] Y. Cai, Z. Zhang, Z. Cai, X. Liu, X. Jiang, and Q. Yan, “Graph convolutional subspace clustering: A robust subspace clustering framework for hyperspectral image,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 5, pp. 4191–4202, 2020.
- [80] Y. Cai, X. Liu, and Z. Cai, “Bs-nets: An end-to-end framework for band selection of hyperspectral image,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 3, pp. 1969–1984, 2019.
- [81] J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia, “Deep convolutional transform learning,” in *Neural Information Processing: 27th*

- International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V* 27. Springer, 2020, pp. 300–307.
- [82] J. Maggu, E. Chouzenoux, G. Chierchia, and A. Majumdar, “Convolutional transform learning,” *In International Conference on Neural Information Processing*, pp. 162–174, Dec 2018.
- [83] —, “Convolutional transform learning,” in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part III* 25. Springer, 2018, pp. 162–174.
- [84] H. Attouch, J. Bolte, and B. Svaiter, “Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods,” *Mathematical Programming*, vol. 137, 01 2011.
- [85] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, “A block coordinate variable metric forward-backward algorithm,” *Journal of Global Optimization*, vol. 66, 11 2016.
- [86] J. Bolte, S. Sabach, and M. Teboulle, “Proximal alternating linearized minimization for nonconvex and nonsmooth problems,” *Mathematical Programming*, vol. 146, 08 2013.
- [87] H. Bauschke, R. Burachik, P. Combettes, and D. Luke, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer Optimization and Its Applications, 11 2009.

- [88] D. Zhang, F. Nan, X. Wei, S. Li, H. Zhu, K. McKeown, R. Nallapati, A. Arnold, and B. Xiang, “Supporting clustering with contrastive learning,” *arXiv preprint arXiv:2103.12953*, 2021.
- [89] Z. Dang, C. Deng, X. Yang, K. Wei, and H. Huang, “Nearest neighbor matching for deep clustering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 693–13 702.
- [90] H. Zhong, C. Chen, Z. Jin, and X.-S. Hua, “Deep robust clustering by contrastive learning,” *arXiv preprint arXiv:2008.03030*, 2020.
- [91] J. Xu, Y. Ren, G. Li, L. Pan, C. Zhu, and Z. Xu, “Deep embedded multi-view clustering with collaborative training,” *Information Sciences*, vol. 573, pp. 279–290, 2021.
- [92] X. Yang, C. Deng, Z. Dang, and D. Tao, “Deep multiview collaborative clustering,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [93] P. Gupta, J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia, “Deconfuse: a deep convolutional transform-based unsupervised fusion framework,” *EURASIP Journal on Advances in Signal Processing*, vol. 2020, no. 1, pp. 1–32, 2020.
- [94] P. Gupta, A. Majumdar, E. Chouzenoux, and G. Chierchia, “Superdeconfuse: a supervised deep convolutional transform based fusion framework

- for financial trading systems,” *Expert Systems with Applications*, vol. 169, p. 114206, 2021.
- [95] G. Chao, S. Sun, and J. Bi, “A survey on multiview clustering,” *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 146–168, 2021.
- [96] S. Bickel and T. Scheffer, “Multi-view clustering,” in *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM 2004)*, 2004, pp. 19–26.
- [97] X. Yi, Y. Xu, and C. Zhang, “Multi-view em algorithm for finite mixture models,” in *Pattern Recognition and Data Mining*, S. Singh, M. Singh, C. Apte, and P. Perner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 420–425.
- [98] D. Lashkari and P. Golland, “Convex clustering with exemplar-based models,” in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2007.
- [99] A. Kumar and H. D. III, “A co-training approach for multi-view spectral clustering,” in *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML 2011)*. Madison, WI, USA: Omnipress, 2011, p. 393–400.
- [100] J. Sun, J. Lu, T. Xu, and J. Bi, “Multi-view sparse co-clustering via proximal alternating linearized minimization,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, ser. Pro-

- ceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37, Lille, France, 07–09 Jul 2015, pp. 757–766.
- [101] T. Liu, “Guided co-training for large-scale multi-view spectral clustering,” *CoRR*, vol. abs/1707.09866, 2017. [Online]. Available: <http://arxiv.org/abs/1707.09866>
- [102] W. Cai, H. Zhou, and L. Xu, “A multi-view co-training clustering algorithm based on global and local structure preserving,” *IEEE Access*, vol. 9, pp. 29 293–29 302, 2021.
- [103] A. Kumar, P. Rai, and H. Daume, “Co-regularized multi-view spectral clustering,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.
- [104] Y. Ye, X. Liu, J. Yin, and E. Zhu, “Co-regularized kernel k-means for multi-view clustering,” in *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016)*, 2016, pp. 1583–1588.
- [105] M. Brbić and I. Kopriva, “Multi-view low-rank sparse subspace clustering,” *Pattern Recognition*, vol. 73, pp. 247–258, 2018.
- [106] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, and D. Xu, “Generalized latent multi-view subspace clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 86–99, 2020.
- [107] J. Tan, Y. Shi, Z. Yang, C. Wen, and L. Lin, “Unsupervised multi-view

- clustering by squeezing hybrid knowledge from cross view and each view,” *IEEE Transactions on Multimedia*, vol. 23, pp. 2943–2956, 2021.
- [108] S. Yu, L. Tranchevent, X. Liu, W. Glanzel, J. A. Suykens, B. De Moor, and Y. Moreau, “Optimized data fusion for kernel k-means clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 1031–1039, 2012.
- [109] X. Chen, X. Xu, J. Z. Huang, and Y. Ye, “Tw-k-means: Automated two-level variable weighting clustering algorithm for multiview data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 932–944, 2013.
- [110] X. Cai, F. Nie, and H. Huang, “Multi-view k-means clustering on big data,” in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*. AAAI Press, 2013, p. 2598–2604.
- [111] H. Liu and Y. Fu, “Consensus guided multi-view clustering,” *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 4, apr 2018. [Online]. Available: <https://doi.org/10.1145/3182384>
- [112] T. Joachims, N. Cristianini, and J. Shawe-Taylor, “Composite kernels for hypertext categorisation,” in *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, 01 2001, pp. 250–257.
- [113] T. Zhang, A. Popescul, and B. Dom, “Linear prediction models with graph regularization for web-page categorization,” in *Proceedings of the*

12th ACM International Conference Knowledge Discovery Data Mining (SIGKDD 2006), 2006, p. 821–826.

- [114] G. Chao and S. Sun, “Multi-kernel maximum entropy discrimination for multi-view learning,” *Intelligence Data Analysis*, vol. 20, no. 3, p. 481–493, jan 2016.
- [115] M. B. Blaschko and C. H. Lampert, “Correlational spectral clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008, pp. 1–8.
- [116] W. Shao, L. He, C.-t. Lu, and P. S. Yu, “Online multi-view clustering with incomplete views,” in *Proceedings of the IEEE International Conference on Big Data (Big Data 2016)*, 2016, pp. 1012–1017.
- [117] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
- [118] H. Zhang, G. Lu, M. Zhan, and B. Zhang, “Semi-supervised classification of graph convolutional networks with laplacian rank constraints,” *Neural Processing Letters*, vol. 54, pp. 1–12, 08 2022.
- [119] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, “One2multi graph autoencoder for multi-view graph clustering,” in *In Proceedings of The Web Conference 2020 (WWW ’20)*, 04 2020, pp. 3070–3076.
- [120] P. Gupta, J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia,

- “Confuse: Convolutional transform learning fusion framework for multi-channel data analysis,” in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1986–1990.
- [121] P. Combettes and J.-C. Pesquet, “Deep neural network structures solving variational inequalities,” *Set-Valued and Variational Analysis*, 2020, <https://arxiv.org/abs/1808.07526>.
- [122] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. of ICLR*, 2015.
- [123] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” in *Proc. of NeurIPS*, Long Beach, California, USA, 4-9 Dec. 2017.
- [124] A. Mass, A. Hannun, and A. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. of ICML*, Atlanta, USA, 16-21 June 2013.
- [125] P. Gupta, J. Maggu, E. Majumdar, A. Chouzenoux, and G. Chierchia, “Deconfuse: a deep convolutional transform-based unsupervised fusion framework,” *EURASIP J. Adv. Signal Process*, vol. 26, 2020.
- [126] Y. Chen, G. Li, and Y. Gu, “Active orthogonal matching pursuit for sparse subspace clustering,” *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 164–168, 2017.
- [127] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, “Deep subspace clustering,”

- IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5509–5521, 2020.
- [128] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
- [129] J. Cai, J. Fan, W. Guo, S. Wang, Y. Zhang, and Z. Zhang, “Efficient deep embedded subspace clustering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1–10.
- [130] B. Xu, X. Li, W. Hou, Y. Wang, and Y. Wei, “A similarity-based ranking method for hyperspectral band selection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 11, pp. 9585–9599, 2021.
- [131] L. Ji, L. Zhu, L. Wang, Y. Xi, K. Yu, and X. Geng, “Fastvgbs: A fast version of the volume-gradient-based band selection method for hyperspectral imagery,” *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 3, pp. 514–517, 2020.
- [132] A. Challa, G. Barman, S. Danda, and B. D. Sagar, “Band selection using dilation distances,” *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.
- [133] S. K. Roy, S. Das, T. Song, and B. Chanda, “Darecnet-bs: Unsupervised dual-attention reconstruction network for hyperspectral band selection,” *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 12, pp. 2152–2156, 2020.

- [134] S. Jia, X. Deng, J. Zhu, M. Xu, J. Zhou, and X. Jia, “Collaborative representation-based multiscale superpixel fusion for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7770–7784, 2019.
- [135] K. Zhan, C. Zhang, J. Guan, and J. Wang, “Graph learning for multiview clustering,” *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2887–2895, 2018.
- [136] D. J. Trosten, S. Lokse, R. Jenssen, and M. Kampffmeyer, “Reconsidering representation alignment for multi-view clustering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, Los Alamitos, CA, USA, jun 2021, pp. 1255–1265.
- [137] C. Garcia-Cardona and B. Wohlberg, “Convolutional dictionary learning,” *arXiv preprint arXiv:1709.02893*, 2017.
- [138] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and D. Tran, “Self-supervised learning by cross-modal audio-video clustering,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9758–9770, 2020.
- [139] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [140] X. Yang, C. Deng, K. Wei, J. Yan, and W. Liu, “Adversarial learning

for robust deep clustering,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9098–9108, 2020.