Unsupervised Learning From Shallow to Deep

Tong Zhang

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

April 2020

Declaration

I hereby declare that this submission is my own work (based on publications in collaboration with the co-authors where due acknowledgement is made) and that, to the best of my knowledge, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma at ANU or any other educational institution, except where due acknowledgment has been made. I also declare that all sources used in this thesis have been fully and properly cited.

Tong Zhang
22 April 2020

To my parents, son, daughter and wife

Acknowledgments

I want to thank many people who helped me along my path to writing this thesis.

Foremost, I would like to express my deep gratitude to my panel members, Prof.r Faith Porikli, Dr. Mehrtash Harandi, and Prof. Richard Hartley. I am indebted to Professor Faith Porikli for his support, understanding, and trust over the years. His encouragement and patience inspire me in many ways. I greatly appreciate the facilities he provided including GPUs and the VARIDESK to relieve pain in my back. I would like to express my sincere gratitude to Dr. Mehrtash Harandi for his unconditional trust and support. He always encourages me to dig deeper into my research topic, and his passion and perseverance always motivate me to devote myself to research. It is my honor to work with Prof. Richard Hartley. Discussion with him is always enlightening and informative, his good research taste will always affect me in my academic career.

I am fortunate enough to work with so many distinguished researchers during the course of Ph.D. study. I would like to thank Prof. Hongdong Li for supporting me all the time in different ways. I can always hear original and novel ideas from him. I would also like to express my gratitude to Dr. Pan Ji for his precious friendship and collaboration throughout these years. We share the same beliefs and opinions on many issues both in life and career, I enjoyed every discussion and conference we attended together. I also owe my thanks to Dr. Wenbing Huang for providing me a chance to visit Tencent AI Lab. It is always interesting and inspiring to meet with him to exchange ideas and opinions.

I would like to thank ANU, NICTA and ACRV vision community for providing a chance to share ideas and funding me to attend several international conferences. During this journey, I am honored to work with brilliant researchers and Ph.D students at ANU, especially Prof. Yuchao Dai, Prof. Xuming He, and Dr Xiaopeng Hong. It is also a great pleasure to thank my dear friends in Canberra and all over the world who supported me in numerous ways during my study: Dr. Jiaolong Yang, Dr. Gao Zhu, Mr. Cristian Rodriguez, Mrs Jing Zhang, Mr. Jun Zhang, Mr. Zeyu Zhuang, Mr. Mina Henein, Mr. Rodrigo Santa Cruz, Mr. Soumava Roy and Mr. Samitha Herath, meanwhile I really enjoyed playing basketball, chatting and hanging out with Mr. Jue Wang, Mr Hongtao Yang, Dr. Haoyang Zhang, Mr. Xiaosong Li and Miss. Xiaolan Wang. I also wish to thank everybody not mentioned here personally, but who contributed in one way or another to the success of this thesis.

At last but not least, I would like to thank my family members. I want to thank my parents, Chunsheng and Xiaochun, for their unconditional love, understanding and never-ending support. I also need to express my gratitude to my parents-in-law, Fushan and Xiulian, for their support and guidance in my life. In addition to them, the rest of my family deserves plenty of thanks. I need to especially thank my grandfather, Shaohui, for supporting me to the best private secondary school and study abroad, both financially and morally. I chose to attend Beihang University mostly because he also graduated from there in 1957. I can not forget that he

traveled from Suzhou to Beijing for introducing me to all his old friends who are professors in Tsinghua and Beihang when I first arrived at Beihang University.

I would like to thank my loving and supportive wife, Xiaofeng, who has always supported me in my endeavors and given me the strength and encouragement to continue my dream. I can not imagine having made it this far without her loyalty, love and occasional kick in the pants. On the other side of every hill, she is waiting and I will never manage to thank her sufficiently for that feat. I also need to thank my precious children, Nina and Alan, for teaching me to be a better man in so many ways.

My many thanks to all of these people, this is a journey I never expected to be on and barely imagined finishing. I hope they have enjoyed the ride as much I have.

Abstract

Machine learning plays an increasingly important role in many research areas, such as computer vision, natural language processing, medical and audio signal processing, especially with the widespread of deep learning. The end-to-end learning paradigm has become the de facto solution of feature learning since massive labeled data provide the desired output for neural networks to build their mapping automatically. Although deep learning has achieved great success in feature learning, it suffers substantially from overfitting. Meanwhile, a large amount of labeled data is difficult and expensive to acquire in many real scenarios. More importantly, the trained deep learning models are hard to adapt to other datasets without fine-tuning under supervision. At the same time, unsupervised learning shows its potential ability to prevent overfitting and influence in machine learning and computer vision research community. Most unsupervised learning algorithms are unable to make good predictions due to the complexity of inferring unknown patterns in data. As a result, how to extract features that can represent high-level concepts has attracted much more attention. Expert practitioners in different fields propose different ways to extract features, which take advantage of important prior knowledge about the task. Therefore, this thesis aims to alleviate the limitations of supervised learning by exploring different frameworks and algorithms to learn good internal representations and invariant feature hierarchies from unlabelled data.

Firstly, we extend the traditional dictionary learning and sparse coding algorithms onto hierarchical image representations in a principled way. To make dictionary atoms capturing additional information from extended receptive fields and attain improved descriptive capacity, we present a two-pass multi-resolution cascade framework for dictionary learning and sparse coding. This cascade method allows collaborative reconstructions at different resolutions using the same dimensional dictionary atoms. The jointly learned dictionary comprises of atoms that adapt to the information available at the coarsest layer, where the support of atoms reaches a maximum range, and the residual images, where the supplementary details refine progressively a reconstruction objective. The residual at a layer is computed by the difference between the aggregated reconstructions of the previous layers and the downsampled original image at that layer. Our method generates flexible and accurate representations using only a small number of coefficients. It is computationally efficient since it encodes the image at the coarsest resolution while yielding very sparse residuals.

In the following work, we started to work on learning features without human supervision through the use of a deeper and nonlinear version of "dictionary learning", namely the deep convolutional autoencoders. Our first idea is to introduce a novel *self-expressive* layer between the encoder and the decoder to mimic the "self-expressiveness" property that has proven effective in traditional subspace clustering. This architecture is built upon deep auto-encoders, which non-linearly map the input data into a latent space. Being differentiable, our new self-expressive layer provides a simple but effective way to learn pairwise affinities between all data points through a standard back-propagation procedure. Being nonlinear, our neural-network

based method is able to cluster data points having complex (often nonlinear) structures.

However, Subspace clustering algorithms are notorious for their scalability issues because building and processing large affinity matrices are quite demanding. We propose two methods to tackle this problem. Our first method is based on k-Subspace Clustering, where we introduce a method that simultaneously learns an embedding space along subspaces within it to minimize a notion of reconstruction error, thus addressing the problem of subspace clustering in an end-to-end learning paradigm. To achieve our goal, we propose a scheme to update subspaces within a deep neural network. This, in turn, frees us from the need of having an affinity matrix to perform clustering. The second attempt is trying to use a feed-forward network to replace the spectral clustering and learn the affinities of each data from the "self-expressive" layer. To that end, We introduce the Neural Collaborative Subspace Clustering (NCSC), where it benefits from a classifier that determines whether a pair of points lies on the same subspace or not. Finally, we also explore to perform dense predictions from a clustering perspective in order to get rid of intensive human labeling. To that end, we present a novel perspective to unsupervised saliency detection through learning from multiple noisy labels generated by "weak" and "noisy" unsupervised handcrafted saliency methods. Our end-to-end deep learning framework for unsupervised saliency detection consists of a latent saliency prediction module and a noise modeling module that work collaboratively and are optimized jointly. Explicit noise modeling enables us to deal with noisy saliency maps in a probabilistic way. Extensive experimental results on various benchmarking datasets show that our model not only outperforms all the unsupervised saliency methods with a large margin, but also achieves comparable performance with the recent state-of-the-art supervised deep saliency methods.

In summary, we make contributions to how to apply unsupervised learning on several tasks in this thesis. Start with traditional sparse coding and dictionary learning in image processing task, we are motivated to propose a framework to learn subspace friendly features and perform subspace clustering by exploiting convolutional neural networks without label information, followed by making deep subspace clustering scalable to large scale datasets. Furthermore, we also extend the clustering on dense prediction task (saliency detection) to reduce human interference and enhance model generalization ability.

Keywords: Unsupervised learning, dictionary learning, subspace clustering, sparse representation, autoencoder, saliency detection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Teaching computers to understand the world as a human is a long and challenging journey. Researchers believe that supervised learning is the first step to start with, and several famous datasets are built for tasks such as image classification and object detection such as MNIST, Caltech 101 and Caltech 256. Through several years of effort, researchers realized that the best algorithm wouldn't work well if the data it learned from didn't reflect the real world. As a result, they began to create large scale datasets for algorithms to learn from, for example, the ImageNet dataset for image classification(Krizhevsky et al. [2012]) and MS COCO dataset(Lin et al. [2014]) for object detection, where the number of images increases from thousands to more than a million, the size and quality of images also improves. These historically unprecedented large datasets brought a revolutionary effect on the development of computer vision and machine learning algorithms. They also stimulate the invention of several crucial deep learning related techniques, which help deep neural networks dominate in many challenges and become the most widely used framework. However, most of the current success heavily relies on massive human labels. Furthermore, the lack of generalization ability degenerates the performance of trained models on other unseen datasets. At the same time, it is impractical to have a large number of human labelings for every sample in different tasks. For example, labeling MRI or other medical images for testing cancer needs top expertise in a certain area, which is at least very expensive. Nevertheless, a large number of unlabeled data is available online without incurring any cost, e.g. the images on the web, different kinds of medical images. As a result, discovering the underlying pattern without massive labels is attracting more and more attention.

As Yann LeCun once said (also is widely known as LeCun's Cake), "Most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake." This metaphor accurately points out the difficulty and importance of unsupervised learning. Unlike supervised learning, unsupervised learning learns the mapping without knowing the desired patterns of output, and it highly depends on how to build the model and cost function. In general, unsupervised learning methods aim to transform the raw input data into a new representation that makes the important characteristics of the input more apparent in a principled way. Besides, unsupervised learning is much more similar to the process of human learning, as we explore the world by observing the environment and analyzing the underlying relation, but not by being taught everything especially after forming

our cognition.

Unsupervised learning is widely used in many computer vision applications from low-level tasks, such as image denoising (Mairal et al. [2009b]; Sulam et al. [2014]), in-painting (Peyré [2009]; Roth and Black [2009]) and dimension reduction (Comon [1994]), to high-level tasks, such as image clustering (Xie et al. [2016]) and feature learning (Kingma and Welling [2013]; Hjelm et al. [2018]). Unsupervised learning is often used to learn the underlying representation of input data, and its techniques are evolving all the time. The most well-known algorithms, such as Principal Component Analysis (PCA) (Wold et al. [1987]), Independent Component Analysis (ICA) (Comon [1994]), have been introduced to reduce the dimension of input to find concise lower dimension representation. Afterward, sparse representation (Olshausen and Field [1997]; Aharon et al. [2006]) has attracted huge attention from researchers and it has formed a complete theoretical system and application methods, where the overcomplete dictionary or basis could be deterministic or data-driven. Nowadays, Convolution Neural Networks (CNN) have become the most popular framework for researchers to conduct their research. Compared to the development of supervised learning, there are fewer studies on training CNN models without supervision. Deep Boltzmann Machines (DBM) (Salakhutdinov and Hinton [2009]; Salakhutdinov and Larochelle [2010]) and Deep Belief Networks (DBN) (Boureau et al. [2008]; Lee et al. [2009]; Hinton [2009]) were proposed in unsupervised learning to learn feature representation. After demystifying several fundamental techniques such as saturating gradient (Glorot et al. [2011]) which help training CNNs much easier, unsupervised learning has been successfully applied in many downstream vision and language tasks Hjelm et al. [2018]; Yang et al. [2017]; Ji et al. [2017b].

Unsupervised learning models can be generally divided into two groups: one is Probabilistic (Generative) Models and the other one is Non-probabilistic Models. This thesis aims to develop unsupervised approaches (Non-probabilistic models), including sparse coding, dictionary learning, auto-encoders, and K-Means, etc. , to narrow the performance gap to the supervised methods, and also apply them on large scale datasets.

1.1 Research Problems

The last twenty years have witnessed the great successes of machine learning. Unsupervised learning is a sub-field of machine learning and has a long and distinguished history. It is a highly interdisciplinary field, which combines knowledge from computer science, engineering, statistics, optimization, information theory and other disciplines of science and mathematics. It has been employed on a lot of tasks and applications such as computer vision, natural language processing, and acoustic processing. The scope of our thesis is only limited to computer vision tasks.

In this thesis, to be clear, we define unsupervised learning as learning without task-specific human annotations and we will discuss several differences and definitions of machine learning algorithms below. Assuming a machine can receive some inputs as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where the $\mathbf{x}_i$ could be an image, a pixel, or a patch drawn from images.

In supervised learning, the machine also receives the other input set $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$, which is the corresponding desired output of the machine. The $\mathbf{y}_i$ could be a label for an image, intensity of a pixel, or a label for a pixel. The goal of this type of learning algorithms is

to "learn" the mapping from $\mathbf{X}$ to $\mathbf{Y}$. In other words, given an input $\mathbf{x}_i$, a CNN will generate an output $\hat{\mathbf{y}}_i$ which is close to $\mathbf{y}_i$ for each sample in the dataset.

In reinforcement learning, an agent will interact with the environment and get the rewards $\mathbf{r}$ after acting a series of actions $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_N$. This is an interactive process, which means states of environment change after every action, and finally, it will result in affecting the final reward. The scalar rewards are usually sparse and time-delayed, it could be given after a series of actions. The goal of reinforcement learning is to optimize these actions to maximize the delayed reward. As a result, deep reinforcement learning is notorious for their training time and stability, and it is usually hard to reproduce the same results.

Different from above, unsupervised learning only has the input data $\mathbf{X}$, it neither receives the rewards nor desired output. At first glance, it seems extremely hard for a machine to learn such a mapping. After a long exploration, it is now practical to develop a framework for unsupervised learning algorithms based on the prior knowledge of different tasks, such as statistical information of datasets, physical laws. Therefore, the goal of unsupervised learning is to exploit the prior information to find the representation of input which could be used for decision making, predicting future input, recovering the raw input from noisy, etc. The following subsections include the problems we are tackling in this thesis, it contains low-level, mid-level, and high-level computer vision tasks.

1.1.1 Dictionary Learning and Sparse coding

The main idea of dictionary learning and sparse coding is to represent one vector as a linear combination of a few atoms in an over-complete dictionary. It is one of the most representative methodologies among linear representation methods. The sparse representation can be employed both in supervised applications, such as image classification and in an unsupervised manner, such as signal processing, image processing, machine learning and computer vision, such as artifacts removal in medical signal processing, image denoising, image inpainting, and image compression.

Sparse representation was first introduced in the 1990s, and it has attracted more and more attention with the blooming of compressive sensing (CS) in the mid of 2000-2010. Technique details and related methodology will be discussed in Chapter 2.2. In the following, we will briefly describe the task we focus on.

In this thesis, we mainly apply sparse representation in unsupervised learning setting on image processing, especially on image restoration (denoising and inpainting), and study the representation ability of dictionary atoms. Image restoration is a field of recovering an original digital image from degraded observations. The source of degrading could be noise, motion blur, and occlusion. The degradation could be described mathematically as :

$$\mathbf{I} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{1.1}$$

where the $\mathbf{x}$ is an original image, $\mathbf{A}$ is a matrix and its function varies as task changes, $\mathbf{n}$ denotes task-dependent noise, and $\mathbf{I}$ is the contaminated image. Note that, in denoising task, the blur kernel $\mathbf{A}$ is an identity matrix, and the noise could be random samples drawn from a Gaussian distribution with a certain variance σ and zero mean (known as additive white noise), or another type of distribution such as salt-and-pepper noise; in image in-painting, the $\mathbf{A}$ is the

mask that excludes the image area where it gets damage or occlusion. As figure 1.1 shown, the pictures are contaminated by Gaussian noise, overlapped by text and missing pixels.

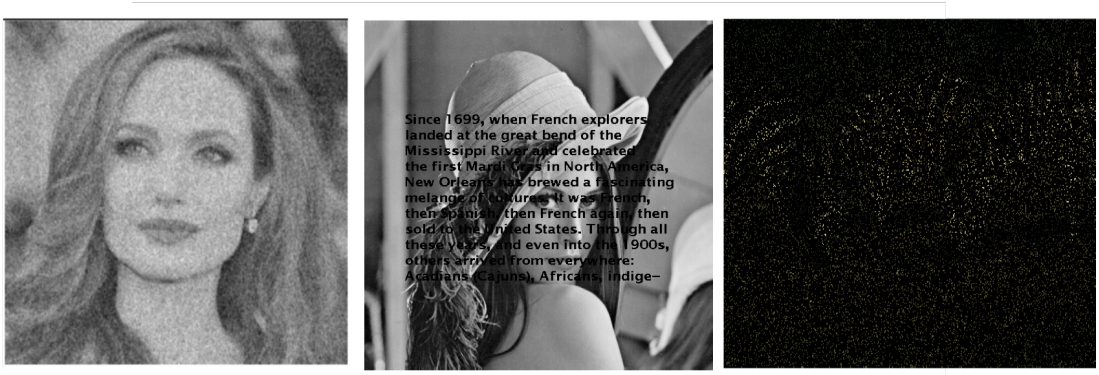


Figure 1.1: Examples of contaminated images, the leftmost one is the summation of original image and gaussian noise, the middle one is overlapped by text, and the rightmost one loses 93% pixels.

There are several techniques which specifically focus on these tasks, such as BM3D (Dabov et al. [2007]) for image denoising, our thesis focus on using sparse coding and dictionary learning approach, which is flexible, simple, efficient and easy to apply to all these restoration tasks. The main idea of applying sparse coding and dictionary learning to image restoration is that the noise in an image patch can be removed by a sparse linear representation of an over-complete dictionary, which can be described as :

$$\mathbf{X} = \mathbf{D}\alpha + \mathbf{n}, \tag{1.2}$$

where the $\mathbf{X}$ is an image patch with size $b^2 \times 1$, α is the coefficients and $\mathbf{D}$ is an over-complete dictionary, which could be deterministic such as DCT, wavelets (more details appear in chapter 2) or data-driven ones.

Therefore, researchers in dictionary learning focus on how to learn the dictionary based on image patches and represent image patches more compactly (in terms of the number of coefficients) and accurately (in terms of reconstruction error).

1.1.2 Subspace Clustering

Clustering is a classic problem in machine learning and pattern recognition, which can be thought of as an unsupervised version of the classification, and it is also a fundamental and core problem in unsupervised learning. Clustering means to divide input samples, such as images and feature vectors, into different groups. It has been addressed in a lot of literature and researchers from various research topics. In general, similarity and distance are two important factors that directly affect the clustering accuracy, as a result, various algorithms and data points have their own preferences. Meanwhile, different rules are also applied in the clustering community, such as partition-based algorithms including K-means(Hartigan and

Wong [1979]) and K-medoids (Park and Jun [2009]), hierarchy based algorithms including agglomerative hierarchical clustering. In our thesis, we narrow the clustering scope to subspace clustering.

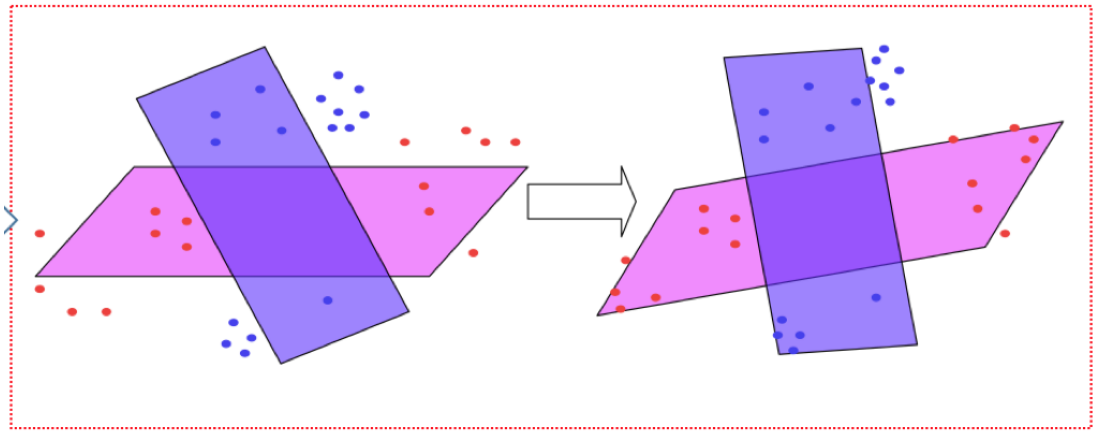


Figure 1.2: The membership of data points are not decided by their distance to the closet centroid but the distance to the closest subspace.

Subspace clustering can be easily understood through its name, which means that data points are grouped into its nearest subspace but not centroids, as Fig. 1.2 shown. Plenty of papers have been proposed in the last three decades to tackle the subspace clustering, especially in face clustering, motion segmentation Elhamifar and Vidal [2013a].

Mathematically, the subspace clustering is defined as modeling a collection of data points with a union of subspaces. Without loss of generality, we can consider affine subspace as being a linear subspace by increasing the dimension of ambient space. Therefore, we use subspace to denote both linear subspace and affine subspace. Let $\mathbf{x}_i \in \mathbb{R}^M, i = 1, 2, \cdots, N$ be samples drawn from an unknown union of $n > 1$ linear subspace $\{\mathbf{S}_j\}$ of unknown dimension $d_j = dims(\mathbf{S}_j), 0 < d_j < M, j = 1, 2, \cdots, K$. This can be described as:

$$\mathbf{S}_j = \{\mathbf{x} \in \mathbb{R}^M : \mathbf{x} = \mathbf{U}_j \alpha\}, j = 1, 2, \cdots, K, \tag{1.3}$$

where the $\mathbf{U}_j \in \mathbb{R}^{M \times d_j}$ is the basis of j-th subspace, and $\alpha \in \mathbb{R}^{d_i}$ is the coefficient vector given the subspace. If there is only 1 subspace, the subspace clustering problem is equivalent to the PCA; however, the number of the subspaces is usually greater than 1 and the problem is becoming more challenging. For example, under Lambertian reflectance, the face images of one subject obtained with a fixed pose and varying lighting conditions lie in a low-dimensional subspace of dimension close to nine. The main challenges can be summarized as follows:

1. Any two subspaces are not guaranteed to be decoupled. In practice, without giving any parameters of subspace (except the number of space) it is hard to explicitly find the subspaces.

2. Subspaces may have an intersection or be very close, which makes the problem extremely hard.

3. Data is usually corrupted by noise, missing entries, outliners, etc. In many applications,

this can cause estimation to become completely wrong.

4. Except for the noise, the data themselves are not usually lie in linear subspaces. For instance, in the example of face image clustering, the reflectance is typically non-Lambertian and the pose of the subject often varies. Under these conditions, the face images of one subject rather lie in a non-linear subspace (or sub-manifold).

5. The scalability is the bottleneck of subspace clustering algorithms since they require the pair-wise affinity, such as the Laplacian matrix. The solution to this problem has become more crucial in recent years as the size of datasets is growing exponentially. Only dealing with small datasets is not satisfactory and can not meet the demand for downstream applications.

The most popular algorithms in subspace clustering, such as sparse subspace clustering and its successors, share a similar spirit with sparse representation, the cost function also can be thought as a variant of (1.2):

$$\mathbf{X} = \mathbf{X}\mathbf{A} + \mathbf{n}, \tag{1.4}$$

where $\mathbf{X}$ is a matrix and its columns are input vectors, which can be considered as a deterministic dictionary, and $\mathbf{A}$ is the affinity matrix for the dataset, and $\mathbf{n}$ denotes the noise or residual. Different sparse constraints, such as ℓ_1, ℓ_2, nuclear norm, applied to the affinity matrix $\mathbf{A}$ yield different properties.

Current state-of-the-art algorithms, to some extent, overcome the above challenges a bit. In Chapter 2, several mainstream state-of-the-art subspace clustering algorithms are discussed in detail. However, the non-linearity and scalability are two major concerns in existing methods, in this thesis we proposed three algorithms to help subspace clustering become more accurate and scalable.

1.1.3 Saliency Detection

Saliency detection aims to locate visually import pixels/regions that attract human attention most. Before the dominance of deep learning, the saliency detection primarily exploits the low-level hand-craft feature, such as center prior (Goferman et al. [2012]), global contrast prior (Cheng et al. [2011]) and background connectivity prior (Zhu et al. [2014]), which are summarized and described with human knowledge. We treated these methods either as unsupervised learning methods or bottom-up methods.

Saliency detection is targeting to build the affinity matrix for every pixel and cut them into two groups, namely foreground, and background. From the perspective of this thesis, those bottom-up methods are thought of a binary subspace clustering problem on the pixel level. Although the number of clusters is reduced to 2, how to establish the affinity between every pixel remains challenging. Given a set of training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where each $\mathbf{x}_i$ is a color image of size $U \times V$ and $\mathbf{y}_i \in \{0,1\}^{U \times V}$ is a binary saliency map, Deep saliency model learns a mapping function $f_\Theta : \mathbb{R}^{U \times V \times 3} \rightarrow [0,1]^{U \times V}$, where Θ is a set of network parameters. Therefore, $f_\Theta(\mathbf{x}_i) = \mathbf{s}_i$ denotes the predicted saliency map, and the empirical risk when learning from clean labelling can be defined as follows:

$$\mathcal{L}(\Theta \mid \mathbf{X}, \mathbf{Y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{(u,v)} \ell(\mathbf{s}_i^{(u,v)}, \mathbf{y}_i^{(u,v)}), \tag{1.5}$$

where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N, \mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$, and (u, v) denote coordinate of pixels in an image, and $\ell : [0, 1] \times \{0, 1\} \to \mathbb{R}$ is the cross-entropy loss defined as:

$$\ell(s, y) = -\big(y \log(s) + (1 - y) \log(1 - s)\big) . \tag{1.6}$$

When clean samples are available, the optimal network model is obtained by minimizing $\mathcal{L}(\Theta|\mathbf{X}, \mathbf{Y})$ using stochastic gradient descent.

In recent several years, the saliency algorithms enjoyed the success of deep learning and large scale human labelings. The existing deep learning based methods are exploring different architectures to better utilize the capacity of neural networks to learn the mapping function and also can be generalized to other datasets. However, we are arguing that this intensive per-pixel labeling is not an optimal solution since the generalization ability of a deep learning model is usually limited, which leads to the performance degenerate in some datasets which are dissimilar to the training set. Unsupervised learning not only enables us to train any models without intensive human labelings, but also alleviates the requirement of the model generalization ability.

In this thesis, we propose to consider the saliency detection to be a binary clustering problem in an unsupervised manner, which is a novel perspective for saliency detection community.

1.2 Contributions

This thesis aims to develop unsupervised methods and its applications for computer vision tasks, especially focusing on sparse representation and subspace clustering. Our contributions can be summarized in four folds:

1. We propose a two-pass multi-scale dictionary learning and sparse coding algorithms to represent images more compact and restore the damage images better

2. A novel self-expressive layer is proposed to find a better non-linear mapping to project the samples nonlinearly into linear subspaces.

3. Two methods are introduced to solve the scalability problem of subspace clustering, one is based on k-subspace, the other one connects the clustering with binary classification problem and collaboratively learn from each other.

4. We cast the traditional saliency detection into a clustering problem, which simultaneously estimates the noise distribution and saliency maps.

The details contributions will be listed as follows.

1. Cascade Residuals Guided Nonlinear Dictionary Learning

We present a computationally efficient framework that employs multi-resolution residual maps for dictionary learning and sparse coding in order to address the redundancy and allow dictionary atoms to access larger context for improved descriptive capacity.

To this end, we start with building an image pyramid using bicubic interpolation. In the first pass, we learn a dictionary from the coarsest resolution layer and obtain the sparse representation. We upsample the reconstructed image and compute the residual in the next layer.

The residual at a level is computed by the difference between the aggregated reconstructions from the coarser layers in a cascade fashion and the downsampled original image at that layer. Dictionaries are learned from the residual in every layer. We use the same patch size yet different resolution input images, which is instrumental in reducing computations and capturing larger context through. The computational efficiency stems from encoding at the coarsest resolution and encoding the residuals that are significantly sparse. This enables our cascade to go as deep as needed without any compromise.

In the second pass, we collect all patches from all cascade layers and learn a single dictionary for a final encoding. This naturally solves the problem of determining how many atoms to be assigned at a hierarchical layer. Thus, all atoms in the dictionary have the same dimensionality while their receptive fields vary depending on the layer.

Compared to existing multi-scale approaches operating indiscriminately on image pyramids or wavelets, our dictionary comprises atoms that adapt to the information available at each layer. The details learned from residual images progressively refine our reconstruction objective. This allows our method to generate a flexible image representation using a much smaller number of coefficients. Our extensive experiments demonstrate that our method applies favorably in image coding, denoising, inpainting and artifact removal tasks.

2. Deep Subspace Clustering Networks

We introduce a novel deep neural network architecture to learn (in an unsupervised manner) an explicit non-linear mapping of the data that is well-adapted to subspace clustering. To this end, we build our deep subspace clustering networks (DSC-Nets) upon deep auto-encoders, which non-linearly map the data points to a latent space through a series of encoders. Our key contribution then consists of introducing a novel self-expressive layer – a fully connected layer without bias and non-linear activations – at the junction between the encoder and the decoder. This layer encodes the "self-expressiveness" (Elhamifar and Vidal [2009]) property of data drawn from a union of subspaces, that is, the fact that each data sample can be represented as a linear combination of other samples in the same subspace. To the best of our knowledge, our approach constitutes the first attempt to directly learn the affinities (through combination coefficients) between all data points within one neural network. Furthermore, we propose effective pre-training and fine-tuning strategies to learn the parameters of our DSC-Nets in an unsupervised manner and with a limited amount of data.

3. Scalable Deep k-Subspace Clustering

Instead of constructing the affinity matrix for spectral clustering, we revisit the k-subspace clustering (k-SC) method (Bradley and Mangasarian [2000]; Tseng [2000]; Agarwal and Mustafa [2004]) to design a novel and scalable method. In order to handle non-linear subspaces, we propose to utilize deep neural networks to project data to a latent space where k-SC can be easily applied.

We bypass the step of constructing an affinity matrix and performing spectral clustering, which has been used in mainstream subspace clustering algorithms and accelerates the computation by using a variant of k-subspace clustering. As a result, our method can handle datasets that are orders of magnitudes larger than those considered in traditional methods. In order to address non-linearity, we equip deep neural networks with subspace priors. This in return enables us to learn an explicit non-linear mapping of the data that is well-suited for subspace clustering. We propose a new strategy to update the subspace basis, which can suppress the

effects of outliers and stabilize the nonlinear mapping. When the size of the dataset at hand is manageable, we update subspaces in closed-form using Singular Value Decomposition (SVD) with a simple mechanism to rule out outliers. For large datasets, we update subspaces by making use of the stochastic optimization methods on the Grassmann manifolds.

4. Neural Collaborative Subspace Clustering

We propose a neural structure to improve the performance of subspace clustering while being mindful of the scalability issue. To this end, we first formulate subspace clustering as a classification problem, which in turn removes the spectral clustering step from the computations. Our neural model comprises of two modules, one for classification and one for affinity learning. Both modules collaborate during learning, hence the name "Neural Collaborative Subspace Clustering". During training and in each iteration, we use the affinity matrix generated by the subspace self-expressiveness to supervise the affinity matrix computed from the classification part. Concurrently, we make use of the classification part to improve self-expressiveness to build a better affinity matrix through collaborative optimization.

5. Deep Unspervised Saliency Detection

The idea of considering unsupervised saliency maps as learning from multiple noisy labels is brand new and different from existing unsupervised deep saliency methods (e.g., , Zhang et al. [2017a]). Firstly, we present a novel perspective to unsupervised deep saliency detection and learn saliency maps from multiple noisy unsupervised saliency methods. We formulate the problem as a joint optimization of a latent saliency prediction module and a noise modeling module. Secondly, our deep saliency model is trained in an end-to-end manner without using any human annotations, leading to an extremely cheap solution. At last, extensive performance evaluation on seven benchmarking datasets show that our framework outperforms existing unsupervised methods with a wide margin while achieving comparable results with state-of-the-art deep supervised saliency detection methods (Hou et al. [2017]; Zhang et al. [2017c]).

In summary, the goal of this thesis is to put unsupervised learning in more applications. We explore the potential of applying sparse coding and deep learning methods without labels, the extensive experiments show that we are narrowing the gap between supervised learning methods.

1.3 Thesis Outline

In this section, we briefly introduce the unsupervised learning and its development. At the same time we also discussed the related techniques we are employing, and the contributions of this thesis. Following this introductory chapter, the remaining chapters of this thesis are organized as follows:

In Chapter 2, we will thoroughly discuss the literature in the related areas and some preliminaries of related mathematics. We will begin with some fundamental knowledge in sparse coding, dictionary learning, and subspace clustering since their key ideas and concepts are closely connected and will be used extensively throughout the whole thesis. Their background and development will be included, and the framework, namely the autoencoders employed in our work, will be discussed in detail. Meanwhile, we also discuss the reason why we need to

abandon the massive human labeling to train networks and give a brief background on saliency detection. Finally, we also provide details on the evaluation metrics.

In Chapter 3, we extend the traditional single-layer dictionary learning and sparse coding algorithms onto hierarchical image representations in a principled way. We present a two-pass multi-resolution cascade framework for dictionary learning and sparse coding, which allows collaborative reconstructions at different resolutions using only the same dimensional dictionary atoms. The jointly learned dictionary comprises atoms that adapt to the information available at the coarsest layer, where the support of atoms reaches a maximum range, and the residual images, where the supplementary details refine progressively a reconstruction objective. The residual at a layer is computed by the difference between the aggregated reconstructions of the previous layers and the downsampled original image at that layer. Our method generates flexible and accurate representations using only a small number of coefficients. It is computationally efficient since it encodes the image at the coarsest resolution while yielding very sparse residuals.

In Chapter 4, we naturally bridge the sparse subspace clustering and Convolution Auto-encoder, to the best of our knowledge, to achieve the state-of-the-art subspace clustering results. Our key idea is to introduce a novel self-expressive layer between the encoder and the decoder to mimic the "self-expressiveness" property that has proven effective in traditional subspace clustering. This architecture is built upon deep auto-encoders, which non-linearly map the input data into a latent space. Being differentiable, our new self-expressive layer provides a simple but effective way to learn pairwise affinities between all data points through a standard backpropagation procedure. Being nonlinear, our neural networks based method is able to cluster data points having complex (often nonlinear) structures.

In Chapter 5, we tackle the scalability of subspace clustering by using the idea of k-subspace clustering. we introduce a method that simultaneously learns an embedding space along subspaces within it to minimize a notion of reconstruction error, thus addressing the problem of subspace clustering in an end-to-end learning paradigm. To achieve our goal, we propose a scheme to update subspaces within a deep neural network. This, in turn, frees us from the need of having an affinity matrix to perform clustering.

In Chapter 6, we start to tackle the scalability from a different perspective, which connects clustering with classification. we use a feed-forward network to replace the spectral clustering and learn the affinities of each data from the "self-expressive" layer. To that end, We introduce the Neural Collaborative Subspace Clustering, where it benefits from a classifier that determines whether a pair of points lie on the same subspace or not. Essential to our model is the construction of two affinity matrices, one from the classifier and the other from a notion of subspace self-expressiveness, to supervise training in a collaborative scheme.

In Chapter 7, we explore to perform dense prediction from a clustering perspective in order to get rid of intensive human labelings. To that end, we present a novel perspective to unsupervised saliency detection through learning from multiple noisy labelings generated by "weak" and "noisy" unsupervised handcrafted saliency methods. Our end-to-end deep learning framework for unsupervised saliency detection consists of a latent saliency prediction module and a noise modeling module that work collaboratively and are optimized jointly. Explicit noise modeling enables us to deal with noisy saliency maps in a probabilistic way. Extensive experimental results on various benchmarking datasets show that our model not only outper-

forms all the unsupervised saliency methods with a large margin but also achieves comparable performance with the recent state-of-the-art supervised deep saliency methods.

In Chapter 8, we conclude the thesis by summarizing our contributions of works in this thesis and suggesting our future directions to extend current works.

Chapter 2

Literature Review

Unsupervised learning has made much progress with a collection of algorithms being applied to computer vision problems. In this chapter, we go over the background of our research topics and related mathematics. Our research starts with increasing the capacity and receptive field of dictionary atoms in sparse representation, accordingly more details on sparse representation and its preliminaries mathematics will be first provided. Next, we follow the history steps and introduce how to extend sparse representation (in ℓ_1, ℓ_2 and nuclear norm) to subspace clustering. At the same time, we also review important steps taken in subspace clustering and list different subspace clustering techniques for readers' convenience. At last, we briefly review various traditional unsupervised salience detection algorithms and several supervised saliency detection algorithms based on deep learning. Furthermore, we discuss the drawback of current supervised saliency detection, so that readers can follow our motivation to work on eliminating human labels more clearly.

2.1 Notations

We first need to clarify the notations to avoid some misunderstandings in the following part. In the whole thesis, we use bold upper case letters to denote matrices, and bold lower case letters to denote vectors, and non-bold letters to denote scalar.

Sparse or **sparsity** of a vector denotes the number of non-zero entries. The inner product of two vectors, where the dimension is N, is defined as:

$$\langle x, y \rangle = x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_N y_N. \tag{2.1}$$

The inner product of two matrices is defined as:

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \text{tr}\left(X^T Y\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} Y_{ij}. \tag{2.2}$$

We denote dictionary atoms $\mathbf{d} \in \mathbb{R}^N$ as the column of a dictionary $\mathbf{D} \in \mathbb{R}^{N \times M}$, and we use T to denote the matrix transpose. The identity matrix is denoted as $\mathbf{I}$, where the diagonal part is 1, and $\mathbf{1}$ denotes matrix with all entries being 1. The singular Value Decomposition (SVD) of a matrix $\mathbf{X}$ is represented as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{X} \in \mathbb{R}^{D \times N}$ $\mathbf{U} \in \mathbb{R}^{D \times p}$, $p = min(D, N)$ and $\mathbf{V} \in \mathbb{R}^{N \times p}$, both of these two matrices are orthogonal; meanwhile the $\Sigma \in \mathbb{R}^{p \times p}$ is a

diagonal matrix whose entries are the singular values of the matrix $\mathbf{X}$. The low-rank representation of $\mathbf{X}$ is written as: $\mathbf{X}_r = \mathbf{U}_r\Sigma\mathbf{V}_r^T$, where $\mathbf{U}_r$ is the first r columns of $\mathbf{U}$, Σ_r is the first r largest singular values and $\mathbf{V}_r$ is the first r columns of $\mathbf{V}$

Different norms of matrices and vectors are also need to be clarified. The ℓ_0 norm of a given vector $\mathbf{x}$ can be written as $\|\mathbf{x}\|_0$, which denotes the number of non-zero element in the vector; ℓ_∞ norm is defined as $\|\mathbf{x}\|_\infty = max(|\mathbf{x}|)$. Note that both of these two norm are not differentiable. Meanwhile, $\|\mathbf{x}\|_p$ means the p-norm of $\mathbf{x}$, and is defined as $\|\mathbf{x}\|_p = (\sum_i(|x_i|)^p)^{1/p}$. Among them, ℓ_1 and ℓ_2 are intensively used throughout the thesis. Similarly, ℓ_1 norm on matrices is defined as $\mathbf{X} = \sum_{i,j}|x_{i,j}|$, where the $x_{i,j}$ is the scalar and it is the element located on i-th row and j-th column; $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j}x_{i,j}^2} = \sqrt{trace(\mathbf{X}^T\mathbf{X})}$ denotes the Frobenius norm. The other norm will be mentioned is nuclear norm, we defined it as $\|\mathbf{X}\|_* = \sum_i^p \sigma_i$, where the σ_i is the i-th largest singular values .

Furthermore, the Moore-Penrose pseudo inverse is denoted as $\mathbf{X}^\dagger$, which is widely used in sparse coding and linear inverse problem. It is defined as $\mathbf{X}^\dagger\mathbf{X} = \mathbf{I}$ and $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ when the columns of matrix are linear independent; $\mathbf{X}^\dagger = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ and $\mathbf{X}\mathbf{X}^\dagger = \mathbf{I}$ when the rows of matrix are linear independent. The Moore-Penrose pseudo inverse has several important properties: 1.$\mathbf{X}\mathbf{X}^\dagger\mathbf{X} = \mathbf{I}$; 2. $\mathbf{X}^\dagger\mathbf{X}\mathbf{X}^\dagger = \mathbf{I}$; 3.$(\mathbf{X}\mathbf{X}^\dagger)^T = \mathbf{X}\mathbf{X}^\dagger$; 4. $(\mathbf{X}^\dagger\mathbf{X})^T = \mathbf{X}^\dagger\mathbf{X}$. Note that in this thesis we do not tackle the complex number, the conjugate transpose is equal to transpose.

2.2 Sparse Coding and Dictionary Learning

Sparse representation, also known as sparse approximation, seeks to find sparse solutions for a system of linear equations. In other words, a signal vector can be represented by atoms of a given dictionary or basis. Sparse representation has attracted considerable attention from researchers in a wide range of fields such as signal processing, computer vision, machine learning, harmonic analysis, and statistics. After twenty years' effort from researchers around all the world, we have already built a systematic theory for sparse representation. Many different algorithms have been proposed for sparse representation to tackle problems in different areas.

Sparse representation is closely related to Compressive Sensing (CS). CS proves that a signal can be perfectly recovered by sampling much fewer samples than the number that Shannon-Nyquist sampling theorem requires, as long as the signal can be sparsely represented by a basis which satisfies some theoretical guarantee, for example, the famous restricted isometry property (RIP) theorem (Candes et al. [2006]; Donoho et al. [2006]; Candes and Tao [2005]) which requires the coherence of atoms less than a threshold (in other words the theorem wants the basis to be nearly orthogonal). Driven by the theory of CS, sparse representation has been adapted in many signal processing and image processing tasks (Elad et al. [2010]; Mallat [1999]; Starck et al. [2010]). With the support of CS theory, more and more sparse representation algorithms are applied in different areas, such as image denoising, deblurring, in-painting, super-resolution, restoration, classification, and segmentation. Dictionary learning is one of the typical representatives that successfully and naturally combined with sparse coding.

Sparse representation theory is categorized from different perspectives, as different methods have their motivations and are applied to different applications. Supervised dictionary

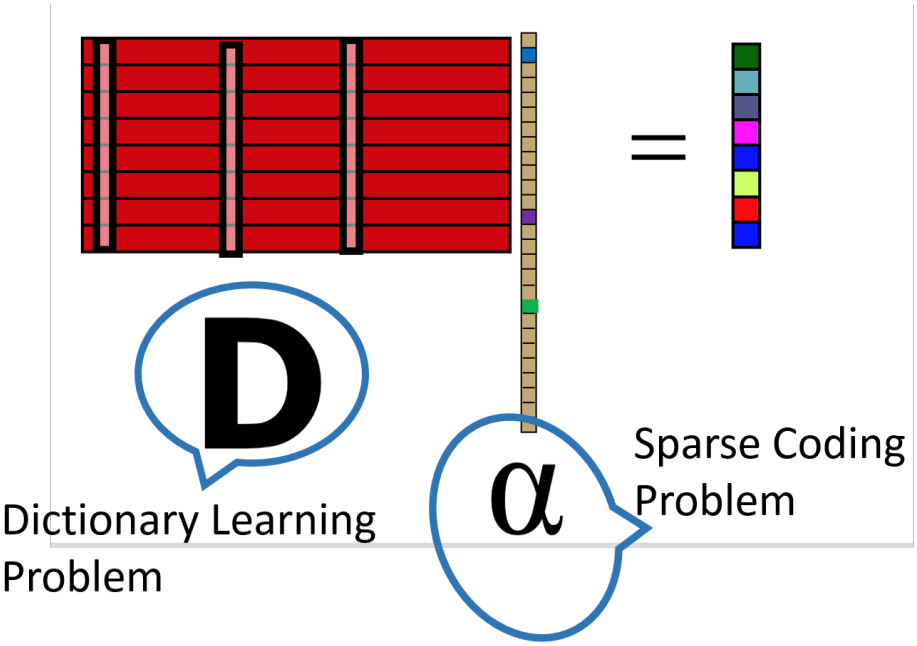


Figure 2.1: Sparse representation is composed by two major steps, one is dictionary learning and the other one is sparse coding.

learning is usually used to work on classification problems, such as texture classification, face classification, and it aims to make the dictionary atoms more discriminative. Nevertheless, in this thesis, we pay our attention to unsupervised learning, which follows different rules from supervised methods. In terms of unsupervised learning, sparse codes and dictionaries are interdependent and mutually conditional as different dictionaries and codes can generate the same signal. Therefore, from the perspective of constructing a dictionary, the sparse representation can be divided into two groups: deterministic dictionary and learning-based dictionary (see Fig. 2.1). In general, the problem has two important steps: 1) update dictionary atoms with fixed codes; 2) pursuing sparse codes with a fixed dictionary.

The dictionary concept can be traced back to Discrete Cosine Transform (DCT) (Ahmed et al. [1974]), Discrete Fourier Transform (DFT) (Brigham and Brigham [1988]), and wavelets based transform such as Haar Transform (Mallat [1999]). The performance of this predetermined dictionary depends on how suitable they are to sparsely describe the samples in question. On the other hand, the learned dictionaries are data-driven and tailored for distinct applications, which result in superior performance compared to the predetermined ones. Noteworthy algorithms of learning dictionary include Method of Optimal Directions (MOD) (Engan et al. [1999], generalized PCA Vidal et al. [2005], KSVD Aharon et al. [2006]), Online Dictionary Learning (ODL) (Mairal et al. [2009a]; Yan et al. [2013b]).

The taxonomy of sparse coding can be addressed from various perspectives. Taking the norm into account, the algorithms can roughly categorized into four groups: ℓ_0 norm minimization, ℓ_p, $(0 < p < 1)$ minimization, ℓ_1 minimization, nuclear norm minimization and $\ell_{2,1}$ norm (also known as group sparsity). Meanwhile, there are also corresponding algorithms

to find the optimal solution for the above norms, such as 1) greedy algorithms; 2) constrained optimization; 3) proximity algorithm-based optimization; and 4) homotopy algorithms.

In the following subsections, we review several methods of sparse coding and dictionary learning applied in this thesis respectively.

2.2.1 Sparse Coding

Sparse coding is a class of unsupervised methods for learning a small portion of coefficients from an over-complete basis to represent data efficiently. Mathematically, it can be described as:

$$\mathbf{x} \approx \sum_{i=1}^{m} \alpha_i \mathbf{d}_i = \mathbf{D}\alpha. \tag{2.3}$$

Given a signal $\mathbf{x} \in \mathbb{R}^{N\times 1}$ and a fixed dictionary $\mathbf{D} \in \mathbb{R}^{N\times M}$ ($M \gg N$), we can always find a code $\alpha \in \mathbb{R}^{M\times 1}$ to represent the signal. The most straightforward solution can be easily found through Moore-Penrose pseudo inverse: $\alpha = \mathbf{D}^{\dagger}\mathbf{x}$, however, it does not yield sparse structure. Therefore, the sparse coding formulate this inverse linear problem as:

$$\arg\min_{\alpha} \|\mathbf{x} - \mathbf{D}\alpha\|_2^2 \qquad \text{s.t.} \|\alpha\|_p \leq T\ , \tag{2.4}$$

or its equivalent form:

$$\arg\min_{\alpha} \|\alpha\|_p \qquad \text{s.t.} \|\mathbf{Y} - \mathbf{D}\mathbf{x}\|_2^2 \leq \epsilon. \tag{2.5}$$

Both of them can be relaxed to:

$$\arg\min_{\alpha} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\alpha\|_2^2 + \lambda\|\alpha\|_p, \tag{2.6}$$

where p-norm yields different solution priors, *i.e.* ℓ_2 norm has the dense solution, $\ell_p(0 \leq p \leq 1)$ has sparse solution. The geometry interpretation of different norms can be illustrated as Fig. 2.2. Since different norms have their feasible regions, the intersection points of the feasible region and data fidelity term locate in different coordinates. In Fig. 2.2, we only use ℓ_1 norm to illustrate the basic idea in two dimension case. It can be seen that solution of ℓ_1 norm minimization is sparse whereas the ℓ_2 norm minimization is hard to rigidly satisfy the condition of sparsity. By introducing $\ell_p(0 \leq p \leq 1)$, the minimization of Eqn. 2.6 requires different algorithms to handle different norms.

In general, the greedy algorithm is often considered as a good way to tackle the ℓ_0 minimization problem since ℓ_0 norm imposes the limited number of the non-zero element which leads the problem to be a non-deterministic polynomial-time hard (NP-hard) problem. Although the greedy algorithm can not guarantee the optimal solution, it finds a balance between the complexity of computation and accuracy of the solution when satisfying certain theoretical guarantees (Tropp et al. [2006]). The two most popular algorithms in sparse representation are Matching Pursuit (MP) and Orthogonal Matching Pursuit (OMP), which updates the representation with the coefficient that decreases the squared representation error the most in each

iteration. At the same time, the solution constrained by ℓ_1 norm demonstrates its advantages. The ℓ_1 norm originates from the least absolute shrinkage and selection operator (LASSO) problem (Efron et al. [2004]; Tibshirani [1996]), it yields an analytical solution and can be solved in polynomial time. Correspondingly, extensive methods are proposed to solve this problem efficiently, such as Basis Pursuit (BP). Moreover, for other norms such as ℓ_p-norm ($0 < p < 1$), group sparsity ($\ell_{2,1}$-norm), different kinds of optimization methods are introduced to solve these problems.

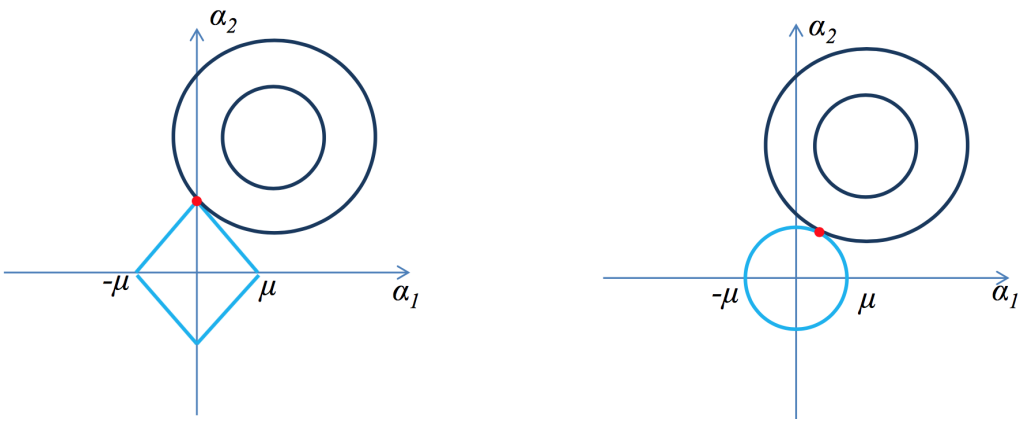


Figure 2.2: The geometric interpretation of ℓ_1 regularized solution and the least square solution.

2.2.1.1 Greedy Algorithm

As mentioned above, greedy algorithms could date back to the 1950s and the core idea of greedy algorithm in sparse coding is to find the entries based on the relationship between the dictionary atom and the probe signal, and then to use least square to obtain the amplitude. Although the greedy algorithm can not guarantee a global optimal solution, in context of dictionary learning and sparse coding the solution can always produce an approximate global optimal solution as long as the dictionary atoms are well constructed (Tropp [2004]). The two main most widely used greedy algorithms for sparse coding are (Mallat and Zhang [1993]) and Orthogonal Matching Pursuit (OMP) (Pati et al. [1993]). Both of them iteratively select the dictionary atom at each step which fits to the residual part of the given signal.

Matching Pursuit

Matching Pursuit algorithm is the earliest and fundamental method of the greedy algorithm applied to the sparse coding problem. Taking an example of sparse decomposition with a vector sample $\mathbf{y}$ over an over-complete dictionary $\mathbf{D}$. The detail of the algorithm is listed as follows.

Set the residual vector $\mathbf{r}_0 = \mathbf{y}$, and every dictionary atom $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N\} \in \mathbb{R}^{d \times N}$ is rescale to 1 through ℓ_2 normalization.

(1). In each iteration, find the best suit atom with respect to residual from dictionary $\mathbf{D}$. The selected atom should satisfy the following condition: $l_n = \arg\max(|\langle \mathbf{r}_{n-1}, \mathbf{d}_i \rangle|)$, where l_n is the selected index from dictionary $\mathbf{D}$ in step n.

(2). Generate the new residual vector by subtracting the new component from the existing residual vector, $\mathbf{r}_n = \mathbf{r}_{n-1} - \alpha_{l_n}\mathbf{d}_{l_n}$, where $\alpha_{l_n} = \langle \mathbf{r}_{n-1}, \mathbf{d}_{l_n} \rangle$.

(3). Keep the iteration until the residual is smaller than the threshold. The final reconstruction signal can be written as: $\hat{\mathbf{y}} = \sum_i^T \alpha_{l_i}\mathbf{d}_{l_i}$, where the iterations times (sparsity) is a very small number $T \ll N$.

Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) is proposed to overcome the convergence issue of Matching Pursuit. It is guaranteed for OMP to converge in a small number of iterations for a finite dimensional space. OMP introduces the orthogonalization in each step to enforce the orthogonal space based on the selected dictionary items and update the existing coefficients. The initial condition is the same as MP, the main step can be summarized as follows:

Set the residual vector $\mathbf{r}_0 = \mathbf{y}$, and every dictionary atom in $\mathbf{D} = \{\mathbf{d}_0, \mathbf{d}_1, ..., \mathbf{d}_N\} \in \mathbb{R}^{d\times N}$ is rescale to 1 through ℓ_2 normalization, the selected dictionary set is marked as $\Lambda = \emptyset$.

Repeat until the residual $\mathbf{r}_i < \epsilon$ or $n < T$:

(1). Find the best matching atom for current residual $\mathbf{r_{n-1}}$ by $\mathbf{d}_{l_n} = \arg\max_{l_n \notin \Lambda}(|\langle \mathbf{r}_{n-1}, \mathbf{d}_i \rangle|)$, and update the selected atom set $\Lambda_n = \Lambda_{n-1} \cup \mathbf{d}_{l_n}$.

(2). Based on the selected atom set, recalculate the residual and coefficient by solving the least square minimization as : $\alpha^* = \arg\min \|\mathbf{y} - \Lambda\alpha\|_2^2$, and $\mathbf{r}_n = \mathbf{y} - \Lambda\alpha^*$

(3). Repeat the above two steps until the algorithm converges.

Most of the greedy algorithm in this area is based on the MP and OMP algorithms, such as regularized version of orthogonal matching pursuit (ROMP) algorithm (Needell and Vershynin [2009]), compressive sampling matching pursuit (CoSaMP) algorithm (Needell and Tropp [2009]), which incorporate insights and ideas from compressive sensing such as restricted isometry property (RIP) and pruning technique to greedy algorithm. A more thorough analysis of greedy algorithms on sparse representation can be found in (Tropp et al. [2006]).

2.2.1.2 Convex Relaxation

Although the ℓ_0-norm minimization algorithm is able to obtain the approximate global optimal sparse representation under some constraint, in some applications or scenarios, ℓ_1 minimization is more desirable since it has a global optimal solution and can be solved in polynomial time. Because of the non-smooth property of ℓ_1 norm at the original point, ℓ_1 minimization is not a trivial task. Furthermore, enjoying the property soft threshold (Donoho [1995]), most algorithms can easily decompose the problem into two or three simple sub-problems. The ℓ_1-minimization along with the sparse coding is widely spread and motivates the following optimization techniques.

With the development of ℓ_1 minimization, related optimization algorithms are proposed and evolved. All of the algorithms can be summarized in several categories.

Constrained Optimization Strategy

The related methods that employ the constrained optimization strategy to solve the original unconstrained non-smooth problem (such as ℓ_1 norm). Gradient projection sparse reconstruction (GPSR) (Figueiredo and Nowak [2005]), Interior-point method based sparse representation strategy.

Homotopy Algorithm Based Sparse Representation

The main idea of homotopy is to solve sequentially the original optimization problems by tracing a continuous parameterized path of solutions along with varying parameters. The concept of homotopy derives from topology and the homotopy technique is mainly applied to address a nonlinear system of equations problem. Homotopy algorithms update the sparse solution by adding or removing elements from the active set. Classic homotopy algorithms include LASSO homotopy algorithm, basis pursuit denoising (BPDN) (Efron et al. [2004]), Iterative re-weighting l1-norm minimization (Asif and Romberg [2013]) and so on.

Proximity Algorithm Based Optimization Strategy

Methods from this category are more favored in recent literature due to efficiency. The fundamental idea of the proximity algorithm is to employ the proximal operator to solve several sub-problems iteratively, similar to the above categories, it is frequently introduced to solve the non-smooth, constrain convex optimization problems.

Suppose the constrained optimization is:

$$\min\{h(\mathbf{x})|\mathbf{x} \in \chi\}, \chi \subset \mathbb{R}^n, \tag{2.7}$$

The general framework of solving this problem can be summarized as :

$$\tilde{\boldsymbol{x}}^t = \arg\min \left\{ h(\boldsymbol{x}) + \frac{\tau}{2} \left\| \boldsymbol{x} - \boldsymbol{x}^t \right\|^2 | \boldsymbol{x} \in \chi \right\}, \tag{2.8}$$

where the τ is the step size and $\mathbf{x}^t$ is the numeric value of $\mathbf{x}$ in t-th iteration. In sparse coding problem with the linear constraint, without losing any generality, we can write the optimization problem as:

$$\arg\min\{F(\mathbf{x}) + G(\mathbf{x})|\mathbf{x} \in \chi\}, \tag{2.9}$$

and the solution can be obtained by:

$$\begin{aligned} \mathbf{x}^{t+1} &= \arg\min \left\{ F(\mathbf{x}) + \left\langle \nabla G\left(\mathbf{x}^t\right), \mathbf{x} - \mathbf{x}^t \right\rangle + \frac{1}{2\tau} \left\| \mathbf{x} - \mathbf{x}^t \right\|^2 \right\} \\ &= \arg\min \left\{ F(\mathbf{x}) + \frac{1}{2\tau} \left\| \mathbf{x} - \boldsymbol{\theta}^t \right\|^2 \right\}, \end{aligned} \tag{2.10}$$

where $\boldsymbol{\theta} = \boldsymbol{x}^t - \tau \nabla G\left(\boldsymbol{x}^t\right)$. Specifically, in sparse coding, the data fidelity term $\|\mathbf{x} - \mathbf{D}\alpha\|_2^2$ is exactly the function G in 2.10 and the F is the ℓ_1 norm constraint. Subsequently, we can decompose the into two-subproblem, one is the Q_t which is used to find approximate solution and the other part is to obtain the sparse solution. Therefore, the sparse solution can be solved by:

$$\boldsymbol{\alpha}^{t+1} = \arg\min Q_t\left(\boldsymbol{\alpha}, \boldsymbol{\alpha}^t\right) + \lambda \|\boldsymbol{\alpha}\|_1. \tag{2.11}$$

The reason why we decompose the original problem as above is that the equation 2.11 is well studied and can be solved very efficiently by soft thresholding (Donoho [1995]) which is a closed-form solution, a simple derivation can be seen in (Selesnick [2009]). Soft thresholding can be described as:

$$\alpha_j^* = \begin{cases} \alpha^t - \lambda, & \text{if } \alpha^t > \lambda \\ \alpha^t + \lambda, & \text{if } \alpha^t < -\lambda \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

We can use a simpler operator to denote the equation 2.12 as $\text{soft}(\alpha, \lambda) = \text{sign}(\alpha) \max\{|\alpha| - \lambda, 0\}$.

Therefore, a number of optimization methods such as iterative soft-threshold algorithm (ISTA) Figueiredo and Nowak [2003]; Daubechies et al. [2004], Fast iterative soft-threshold algorithm (FISTA) (Beck and Teboulle [2009]) are introduced to accelerate the computation. Furthermore, Alternating Direction Method of Multipliers (ADMM) (Boyd et al. [2011]) is also widely used in related fields to solve cost function with compound constraints or regularizations (Selesnick et al. [2014])

2.2.2 Dictionary Learning

The nature of the dictionary learning objective makes it an NP-hard problem since both the dictionary and the coefficients are unknown. To handle this challenge, most dictionary learning algorithms alternate between the sparse coding and dictionary updating steps iteratively by fixing one while optimizing the other. For example, MOD updates the dictionary by solving an analytic solution of the quadratic problem by using Moore-Penrose pseudo-inverse; KSVD incorporates the k-means clustering and singular value decomposition by refining the coefficients and dictionary atoms recursively, and ODL updates the dictionary by using the first-order stochastic gradient descent in small batches. Adding to the complexity, sparse coding itself is an NP-hard problem due to the ℓ_0 norm. This objective is often approximated by greedy schemes such as Matching pursuit (MP) (Mallat and Zhang [1993]) and Orthogonal Matching Pursuit (OMP) (Pati et al. [1993]). Another alternative is to replace the ℓ^0-norm with the ℓ^p-norm with $p \leq 1$. When $p = 1$, the solution can be approximated by Basis Pursuit(BP) (Chen et al. [2001]), FOCal Under-determined System Solver (FOCUSS) (Gorodnitsky and Rao [1997]), and Least Angle Regression (LARS) (Efron et al. [2004]) to count a few.

Multi-scale methods for image encoding have been widely studied in the past. Wavelets are among the premier multi-scale analysis tools in signal processing. Many wavelets variants, e.g., bandlets (Le Pennec and Mallat [2005]), contourlets (Do and Vetterli [2005]), curvelets (Candes and Donoho [2000]) as well as decomposition methods, e.g., wavelet pyramid (Mallat [1989]), steerable pyramid (Simoncelli and Freeman [1995]), and Laplacian pyramid (Burt and Adelson [1983]) have also been proposed. These methods basically improve the frequency-based analysis of Fourier transform by incorporating scale and spatial information. .

There have been few attempts to learn multi-scale dictionaries. In (Mairal et al. [2008]), a quadtree structure is proposed. Dictionaries with different atom dimensions are obtained for different levels of the quadtree and then concatenated together by zero-padding smaller atoms in a dyadic fashion. Unfortunately, the number of scales and the maximum dimension of dictionary atoms are restricted due to the heavy computational and memory requirements. Besides, this approach ignores the coarse-scale information that may be more suitable to represent patches using atoms of the same size.

To overcome the computational issues, (Ophir et al. [2011]) extracts sub-dictionaries in

the wavelet transform domain by exploiting the sparsity between the wavelets coefficients. This work leverages the frequency selectivity of the individual levels of a wavelet pyramid to remove redundancy in the learned representations. Since separate dictionaries are learned for directional subbands, its performance is hampered in comparison to the single-scale KSVD for image denoising tasks. Their following work (Sulam et al. [2014]) exploits multi-scale analysis and single-scale dictionary learning, fusing both outputs by using a weighted joint sparse coding. Since the fused dictionary is several times larger than its single-scale version, the computational complexity is high. Besides, its denoising performance is sensitive to the size and category of images. A similar work (Yan et al. [2013b]) builds multi-resolution dictionaries on the wavelet pyramid by employing the k-means clustering before the ODL step. For each resolution, it clusters the patches of three subbands and then concatenates all dictionary atoms. Although its denoising performance improves due to non-local clustering on the image subbands, each layer requires a large dictionary, which reflects adversely on the computational load.

Multi-resolution sparse representations are also employed for image fusion and super-resolution. Given a pre-trained dictionary, (Liu et al. [2015a]) fuses two images by obtaining sparse coefficients for high-pass and low-pass frequency bands and applying OMP. The fused coefficient columns in each band are chosen by maximal ℓ_1 norm of corresponding coefficients. Towards the same goal, Yin [2015] merges two coefficient vectors; however, the fused coefficient columns are selected by ℓ_2 norm. Instead of training subdictionaries independently, it learns $3S+1$ subdictionaries jointly (S stands for the number of layers), which means the dimension of the matrix is $(3S+1)n \times k$ thus the learning stage is computationally expensive. In (Tarquino et al. [2014]), authors propose a multi-scale approach to super resolve the diffusion-weighted images where the low-resolution dictionary is based on the shearlet transform and the high-resolution one is based on image intensity. In (Dong et al. [2013]), a sparse representation is used to build a model for image interpolation. This model describes each patch as a linear combination of similar non-local patch neighbors, and every patch is represented with a specific dictionary. To decrease the coherence of the representation basis, it clusters patches into multiple groups and learns multiple local PCA dictionaries.

2.3 Autoencoder

Learning useful representations for downstream tasks with little or no supervision is a key challenge in artificial intelligence. Speaking of unsupervised learning in the deep learning era, autoencoder is a necessary topic to be involved in the discussion. In general, autoencoder, which consists of an encoder and a decoder as Fig 2.3, learns a mapping to reconstruct the input. This mapping is an approximation to an identity mapping, however, identity mapping seems a trivial function to learn and not a desirable solution. Therefore, by adding constraints on the network, such as the number of neurons in each layer, sparsity in latent space, autoencoder is able to discover meaningful underlying representation in latent space.

Autoencoders can be traced back to (Baldi and Hornik [1989]), and it was initially used to work on dimension reduction to compare with PCA, where the non-linear is not utilized. In other words, (Baldi and Hornik [1989]) makes autoencoder be a learnable PCA. Mathematically, given a set of unlabelled training samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathbb{R}^n$, an

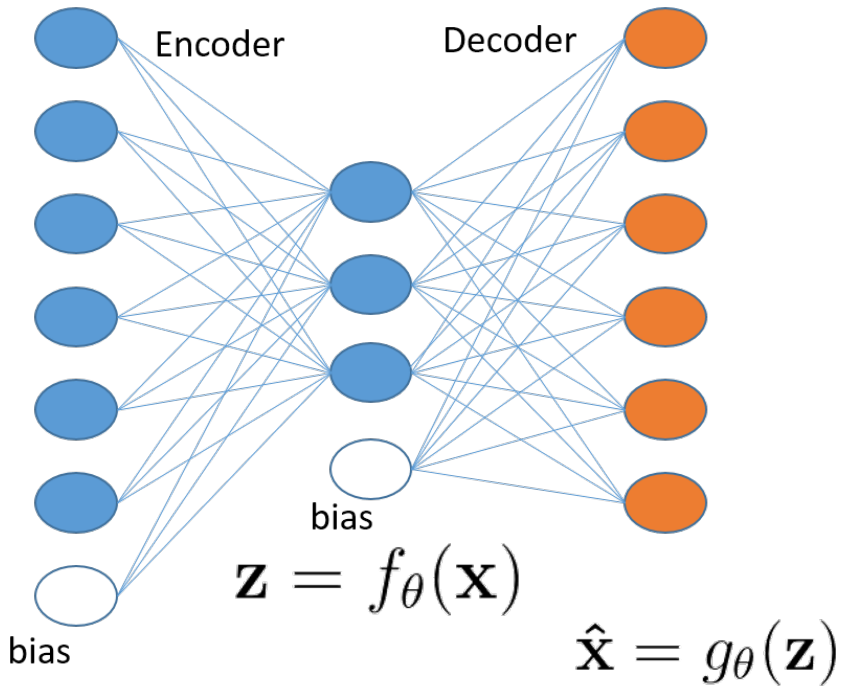


Figure 2.3: A simple example of one-layer autoencoder. It only has one layer to encoder the input sample, and one layer to decoder the latent space. The latent space is usually smaller than the original data space, which captures the underlying structure of training samples.

autoencoder tries to learn a mapping $h(\mathbf{x})_{\mathbf{w},\mathbf{b}} = \mathbf{x}$. Specifically the mapping has two parts, one is called encoder and the other is called decoder:

$$\begin{aligned} \mathbf{z} &= f_\theta(\mathbf{x}) = s_f(\mathbf{b} + \mathbf{W}\mathbf{x}) \\ \hat{\mathbf{x}} &= g_\theta(\mathbf{z}) = s_g(\mathbf{d} + \mathbf{W}'\mathbf{z}), \end{aligned} \tag{2.13}$$

where $\mathbf{z}$ is the latent space vector, $\hat{\mathbf{x}}$ is the reconstruction, f_θ and g_θ represent encoder and decoder, s_f and s_g are the nonlinear activation functions. The autoencoder can be trained by using backpropagation to minimize the cost function:

$$L_\theta(\mathbf{X}) = \sum_t L\left(\mathbf{x}^i, g_\theta\left(f_\theta\left(\mathbf{x}^i\right)\right)\right) \tag{2.14}$$

Shallow Autoencoder

The single-layer autoencoder, Fig. 2.3, and dictionary learning is also closely related, whereas, autoencoder can be considered as a nonlinear version of dictionary learning. Thus, several ideas (Boureau et al. [2008]; Poultney et al. [2007]) are inspired by sparse coding and dictionary learning to learn the sparse representation by using autoencoder. As a result, different sparsity constraints are added to the object function (2.14), which are called sparse

autoencoder. One of the most common forms of the sparse autoencoder is:

$$L_\theta(\mathbf{x}) = L_\theta^{AE} + \lambda \sum_t \sum_{j=1}^{d_h} KL\left(\rho || h_j\left(x_i\right)\right) \tag{2.15}$$

where ρ is the sparsity parameter, and is the mean of a Bernoulli random variable, $h_j\left(x_i\right)$ is the $j-th$ hidden unit's output at $\mathbf{x_i}$, and KL is the KL divergence used to measure the difference between two distributions. If the prior is Bernoulli, the KL divergence can be easily gotten by: $KL\left(\rho||h_j\right) = \rho \log \frac{\rho}{h_j} + (1-\rho) \log \frac{1-\rho}{1-h_j}$. The sparsity level depends on the ρ, lower ρ yields more sparse latent space.

Deep Autoencoder

Limited by computation capability of the hardware and deep learning techniques, the multi-layer autoencoders was very hard to train 20 years ago. For example, saturating gradients problem caused by sigmoid or tanh activation function, the limited storage and poor computation of hardware, and the number of training parameters for fully connected autoencoder is too large. Besides, due to the non-convexity of neural networks, the solution is often located in a different local minimum, especially for unsupervised learning, the results may change a lot with different initializations. Until (Hinton and Salakhutdinov [2006]), when restricted Boltzmann machines (RBMs) were introduced, the potential of layer-wise learning was not fully understood. The other concept that RBMs brought to deep learning area is that creating sequential sets of activations by grouping features and then grouping groups of features, by which the neural networks learn more complex and abstract representations of data. Due to the complexity of RBM, it only extends the single layer to two layers, however, it inspires a lot of works and ideas for deep learning.

Compared to RBM, which is a probability-based model, traditional autoencoder is more straight-forward and easier to train, (Bengio et al. [2007]) also introduces a similar training mechanism to train the traditional autoencoders. It (Bengio et al. [2007]) trains a fully connected network layer by layer in a greedy way. Fig. 2.4 shows an example of training a two-layer encoders and decoders. For the first layer, it is trained to minimize the reconstruction error until reaching a local minimum. Stand upon a local minimum of the first encoder, the second encoder and decoder will be trained until convergence. If adding more layers, the training procedure keeps the same as the example, except adding more layers together to train the encoder. Note that, after training all encoders and decoders separately, the whole autoencoder needs to be fine-tuned as a whole part. This greedy layer-wise training method is called Stacked Autoencoder (SAE), which provides beneficial to the following works.

SAE is widely adopted in many applications and variants of autoencoder, such as denoising autoencoder (Vincent et al. [2008, 2010]) and clustering (Xie et al. [2016]; Yang et al. [2017]). The intuition of the SAE is quite clear and straightforward. Since it is extremely difficult to train all the parameters together, decompose them into several parts and find local saddle points for them one by one. However, this training mechanism and structure also have its drawbacks. Since the following encoders and decoders are trained based on a local minimum of previous layers, it is difficult to jump out the local minimum in the fine-tuning stage. As a consequence, it is often hard to adjust to another task with SAE's initialization.

With the appearance of several crucial techniques, such as using rectified linear unit (ReLu) (Nair

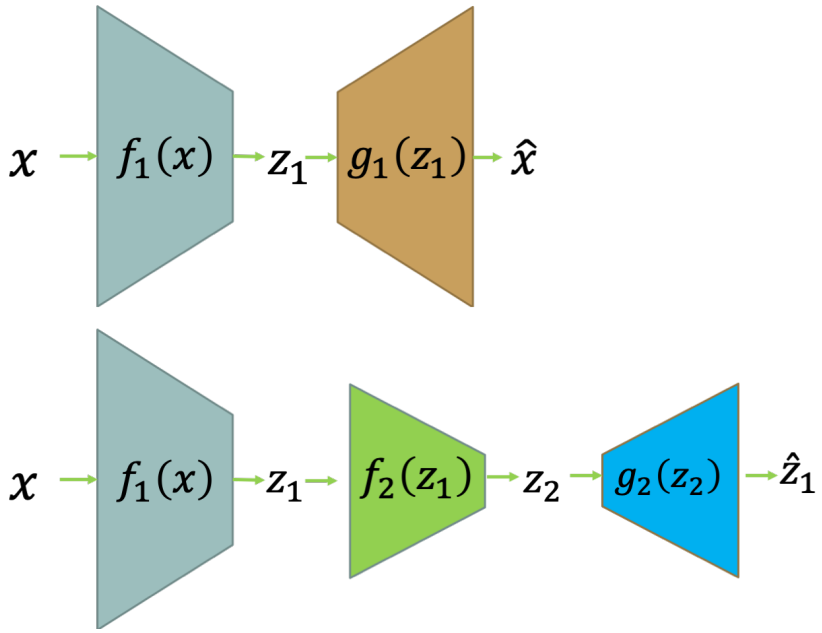


Figure 2.4: A example to show how to greedy train the deeper autoencoder. The upper one is training the first layer and the lower one is training the second layer encoder and decoder based on the first encoder.

and Hinton [2010]) to replace sigmoid or tanh function, employing CNN instead of fully connected network to reduce the number of training parameter, introducing dropout (Srivastava et al. [2014]) to increase the robustness of learning model, autoencoder is also growing deeper and deeper. Many autoencoder based generative models are proposed, one of the most representative frameworks is Variational autoencoder (VAE) (Kingma and Welling [2013]). VAE reinterprets the autoencoder from a probability perspective, and sample the points from latent space to generate new meaningful data. Its formulation is very close to the sparse autoencoder (2.15), and it has the following formulation:

$$L(\theta, \phi; x) = -D_{KL}\left(q_\phi(z|x)||p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right], \tag{2.16}$$

where $z \sim \mathbb{N}(0,1)$, the encoder $q_\phi(z|x)$ is also explicitly represent a gaussian distribution, and $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$ is exactly the autoencoder's reconstruction loss. Unlike sparse autoencoder, VAE assumes the prior of latent space is a Gaussian distribution, and do sampling through reparameterization trick to enable backpropgation. After the training converge, $D_{KL}\left(q_\phi(z|x)||p_\theta(z)\right)$will be minimized, and the samples drawn from the latent space will have large probability equal to sample from a prior $\mathbb{N}(0,1)$ and generate new data.

Following by the big success of GAN (Goodfellow et al. [2014]) and VAE, a bunch of new models are developed, such as adversarial autoencoders (AAEs) (Makhzani et al. [2015]), Wasserstein autoencoder (WAE) (Tolstikhin et al. [2017]), to apply in different research areas such as deep clustering, representation learning, semi-supervised learning, and disentanglement.

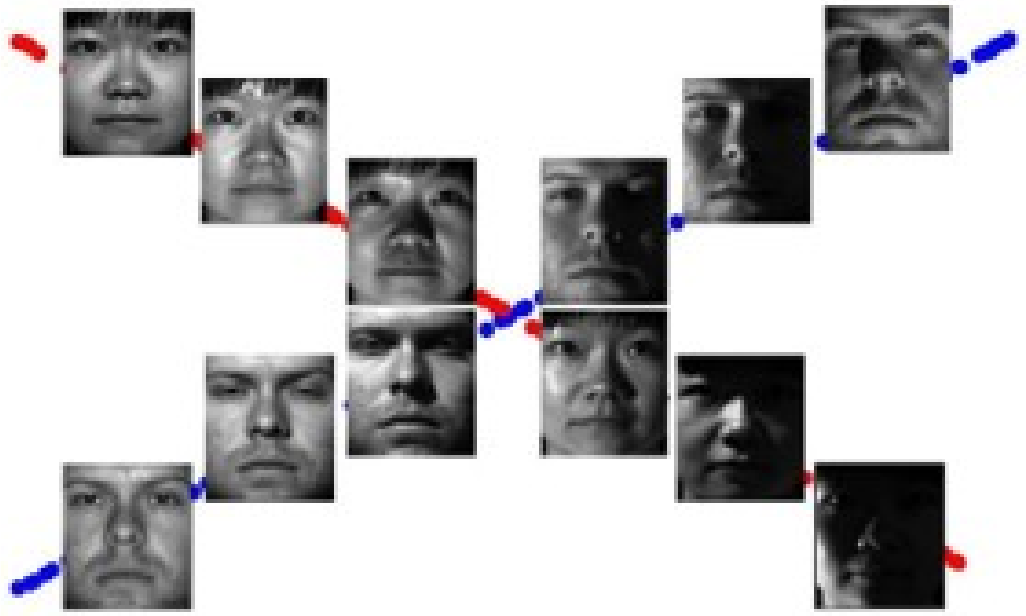


Figure 2.5: A simple example for subspace clustering. It aims to find the membership of data points drawn from multiple subspaces of unknown dimensions, which is different from the traditional clustering methods based on euclidean distance with centroid

2.4 Subspace Clustering

Subspace clustering is to cluster data points drawn from a union of low-dimensional subspaces in an unsupervised manner, which is different from the euclidean distance-based clustering. Mathematically, given a set of K subspaces $\{S_i\}_{i=1}^{K}$ of unknown dimensions, N data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \in \mathbb{R}^D$ are assumed to be drawn from a union of the K subspaces. Subspace clustering is trying to cluster the data into their subspaces. For example, in Figure 2.5 the subspace clustering aims to cluster the face images or high dimensional data points according to the distance to two subspace or some affinity matrix instead of using euclidean distance to their centroids. In practice, the data points are often noisy and have lots of outliners, and some of them can not be even observed. Affine subspace is the main scope of our discussion in this thesis.

Subspace Clustering (SC) has achieved great success in various computer vision tasks, such as motion segmentation (Kanatani [2001]; Elhamifar and Vidal [2009]; Ji et al. [2014b]), face clustering (Ho et al. [2003]; Elhamifar and Vidal [2013a]) and image segmentation (Yang et al. [2006]; Ma et al. [2007]). For example, under Lambertian reflectance, the face images of one subject obtained with a fixed pose and varying lighting conditions lie in a low-dimensional subspace of dimension close to nine (Basri and Jacobs [2003]). Therefore, one can employ subspace clustering to group images of multiple subjects according to their respective subjects.

Over the years, many methods have been developed for linear subspace clustering. In general, these methods consist of two steps: the first and also the most crucial one aims to estimate

an affinity for every pair of data points to form an affinity matrix; the second step then applies normalized cuts (Shi and Malik [2000]) or spectral clustering (Ng et al. [2001]) to the obtained affinity matrix. The resulting methods can then be roughly divided into three categories (Vidal [2011]): factorization methods (Costeira and Kanade [1998]; Kanatani [2001]; Vidal et al. [2008]; Mo and Draper [2012]; Ji et al. [2015],) higher-order model based methods (Yan and Pollefeys [2006]; Chen and Lerman [2009]; Ochs and Brox [2012]; Purkait et al. [2014]), and self-expressiveness based methods (Elhamifar and Vidal [2009]; Liu et al. [2010]; Lu et al. [2012]; Wang et al. [2013]; Ji et al. [2014b]; Feng et al. [2014]; Li and Vidal [2015]; You et al. [2016a]).

In essence, factorization methods build the affinity matrix by factorizing the data matrix, and methods based on higher-order models estimate the affinities by exploiting the residuals of local subspace model fitting. Recently, self-expressiveness based methods, which seek to express the data points as a linear combination of other points in the same subspace, have become the most popular ones. These methods build the affinity matrix using the matrix of combination coefficients. Compared to factorization techniques, self-expressiveness based methods are often more robust to noise and outliers when relying on regularization terms to account for data corruptions. They also have the advantage over higher-order model-based methods of considering connections between all data points rather than exploiting local models, which are often suboptimal.

2.4.1 Self-expressiveness and Sparsity

As mentioned above, a key important step is to construct an affinity matrix from data points. The affinity matrix reflects the pairwise connections for all the given data points. Ideally, the data points drawn from the same subspace have higher similarity, on the other hand, the points come from different subspaces have weaker connections.

Self-expressiveness

Self-expressiveness is first proposed to tackle the subspace clustering in (Elhamifar and Vidal [2009]). The core idea of self-expressiveness is to represent any data point by a linear combination of other points from the same subspace.

Mathematically, the property of self-expressiveness can be summarized by a simple equation:

$$\mathbf{X} = \mathbf{XC}, \tag{2.17}$$

where the $\mathbf{X} \in \mathbb{R}^{d \times N}$ is the data matrix with each column being a data point, and $\mathbf{C} \in \mathbb{R}^{N \times N}$ is the coefficient matrix or affinity matrix we are pursing.

To obtain this affinity matrix, there are several important algorithms which focus on imposing different constraints to the self-expressiveness. With the development of the sparse representation in the last decades, the techniques and intuition are widely spread to different research areas, including subspace clustering. Hence, several sparse formulations are introduced to the subspace clustering to help to find better affinity matrices. The difference is that the dictionary is not learned but pre-defined by the points themselves.

Sparse Subspace Clustering

Sparse Subspace Clustering (SSC) constructs the affinity matrix with few non-zero coefficients, which means to ensure a point can only be linearly expressed by several other points in the dataset. To be more specifically, $\mathbf{C}_{ij} = 0$ if points i and j are drawn from different subspaces, otherwise, $\mathbf{C}_{ij} \neq 0$ if they lie in the same subspace. SSC (Elhamifar and Vidal [2009]) minimizes the ℓ_1 norm of $\mathbf{C}$ to get sparse representation of the data:

$$\begin{aligned} &\min_{\mathbf{C}} \|\mathbf{C}\|_1 \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{C} \\ &\operatorname{diag}(\mathbf{C}) = \mathbf{0} \end{aligned} , \tag{2.18}$$

where the diagonal of coefficient matrix is fixed to be 0 to avoid the trivial solution, namely the identity matrix. An additional constraint on $\mathbf{C}$ is to enfocre the sum of each column to be 1 to deal with affine subspaces, for example:

$$\begin{aligned} &\min_{\mathbf{C}} \|\mathbf{C}\|_1 \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{C} \\ &\mathbf{1}^T\mathbf{C} = \mathbf{1}^T \\ &\operatorname{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \tag{2.19}$$

Furthermore, when data have more noise and sparse outliners, the self-expressiveness term $\mathbf{X} = \mathbf{X}\mathbf{C}$ does not hold any more. The cost function can slightly change by adding extra regularization (Elhamifar and Vidal [2013a]):

$$\begin{aligned} &\min_{\mathbf{C}} \|\mathbf{C}\|_1 + \tfrac{\lambda_1}{2} \|\mathbf{E}_1\|_F^2 + \lambda_2 \|\mathbf{E}_2\|_1 \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E}_1 + \mathbf{E}_2 \\ &\mathbf{1}^T\mathbf{C} = \mathbf{1}^T \\ &\operatorname{diag}(\mathbf{C}) = \mathbf{0}, \end{aligned} \tag{2.20}$$

where $\mathbf{E}_1$ and $\mathbf{E}_2$ are the data noise and sparse outliers and have the same dimension with $\mathbf{X}$, λ_1 and λ_2 are regularizer parameters.

Low Rank Representation Instead of applying sparse coefficient as constraint, sparse nuclear norm is also employed in subspace clustering. Low Rank Representation (LRR) (Liu et al. [2010]) algorithm targets to minimize the rank of $\mathbf{C}$, instead of ℓ_1 norm. However, the rank-minimization is NP-hard and not convex, LRR replaces the rank of $\mathbf{C}$ by its nuclear norm $\|\mathbf{C}\|_* = \sum \sigma_i(\mathbf{C})$, where the $\sigma_i(\mathbf{C})$ is the $i-$th singular value of $\mathbf{C}$. Therefore their noise free formulation can be represented as:

$$\begin{aligned} &\min_{\mathbf{C}} \|\mathbf{C}\|_* \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{C} \\ &\mathbf{1}^T\mathbf{C} = \mathbf{1}^T. \end{aligned} \tag{2.21}$$

When the data is free from noise and drawn from independent linear subspaces, the optimal solution can be obtained through a series of SVD decomposition operations along with the optimization algorithm.

For the case of contaminated data, the cost function becomes:

$$\begin{aligned} &\min_{\mathbf{C}} \|\mathbf{C}\|_* + \mu\|E\|_{2,1} \\ &\text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} \\ &\mathbf{1}^T\mathbf{C} = \mathbf{1}^T, \end{aligned} \tag{2.22}$$

where $\|\mathbf{E}\|_{2,1} = \sum_{k=1}^{N} \sqrt{\sum_{j=1}^{N} \left|\mathbf{E}_{jk}\right|^2}$ is the $\ell_{2,1}$ norm of the error matrix $\mathbf{E}$.

Besides, there are also other algorithms such as Kernel Sparse Subspace Clustering (KSSC) (Patel and Vidal [2014]), Low Rank Subspace Clustering (LRSC) (Vidal and Favaro [2014]) and Efficient Dense Subspace Clustering (EDSC) (Ji et al. [2014b]) are developed to further enhance the subspace clustering methods. Augmented Lagrangian and ADMM are the two most widely used methods to solve these subspace clustering minimization problems.

2.4.2 Spectral Clustering

As mentioned above, the general framework of subspace clustering is to apply spectral clustering on the affinity matrix obtained by the different formulation to get the final clustering results. As a very crucial tool for subspace clustering, we will briefly present the type of spectral clustering used in this thesis in the following paragraphs. We follow the notation from classic tutorial (Von Luxburg [2007]) (more details can be checked in the original paper).

Spectral clustering is developed initially to deal with the Laplacian matrix. Note that, in subspace clustering, we have to do post-processing step to make the affinity matrix be a Laplacian matrix. Suppose a non-negative symmetric affinity matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, $a_{ij} = a_{ji}$ and $a_{ij} \geq 0$, and its Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.23}$$

where $\mathbf{D}$ is called the degree matrix and it is a diagonal matrix:

$$\mathbf{D} = \begin{bmatrix} \sum_j a_{1j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Sigma_j a_{Nj} \end{bmatrix} \tag{2.24}$$

In some applications or algorithms, the laplacian matrix is often required to be normalized, thus we denote the normalized laplacian matrix as:

$$\begin{aligned} \mathbf{L}_{\text{sym}} &:= \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \\ \mathbf{L}_{\text{rw}} &:= \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}. \end{aligned} \tag{2.25}$$

Where the $\mathbf{L}_{\text{sym}}$ is a symmetric matrix and the $\mathbf{L}_{\text{rw}}$ is closely related to a random walk.

One of the most important property of laplacian matrix is that it is always positive semi-

Algorithm 1 Unnormalized spectral clustering

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, number of cluster

1. Compute the unnormalized Laplacian matrix $\mathbf{L}_{\text{un}}$ as 2.23

2. Compute the smallest k eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_k$ of $\mathbf{L}$, and form $\mathbf{U} \in \mathbf{R}^{N\times k}$ with $\mathbf{u}_1, \cdots, \mathbf{u}_k$ as columns.

3.Each row of $\mathbf{U}$ is denoted by $\mathbf{y} \in \mathbb{R}^k, i = 1, \cdots, N$.

4. Cluster the points $\mathbf{y}_i, i = 1, \cdots, N$ with the k-means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \cdots, k\}^N$.

Output: Clusters $\mathbf{A}_1, \cdots, \mathbf{A}_k$ with $\mathbf{A}_i = \{j | \mathbf{y}_j \in \mathbf{C}_i\}$

definite. For any vector $\mathbf{f} \in \mathbb{R}^N$, we can easily derive the following property for unnormalized:

$$\begin{aligned}
\mathbf{f}^T\mathbf{L}\mathbf{f} &= \mathbf{f}^T\mathbf{D}\mathbf{f} - \mathbf{f}^T\mathbf{A}\mathbf{f} \\
&= \sum_{i=1}^{N} d_i f_i^2 - \sum_{i,j=1}^{N} f_i f_j a_{ij} \\
&= \frac{1}{2}\left(\sum_{i=1}^{N} d_i f_i^2 - 2\sum_{i,j=1}^{N} f_i f_j a_{ij} + \sum_{j=1}^{N} d_j f_j^2\right) \\
&= \frac{1}{2}\left(\sum_{i,j=1}^{N} a_{ij} f_i^2 - 2\sum_{i,j=1}^{N} f_i f_j a_{ij} + \sum_{i,j=1}^{N} a_{ij} f_j^2\right) \\
&= \frac{1}{2}\left(\sum_{i,j=1}^{N} a_{ij} f_i^2 - 2\sum_{i,j=1}^{N} f_i f_j a_{ij} + \sum_{i,j=1}^{N} a_{ij} f_j^2\right) \\
&= \frac{1}{2}\sum_{i,j=1}^{N} a_{ij}\left(f_i - f_j\right)^2 \geq 0.
\end{aligned} \tag{2.26}$$

Obviously, the smallest eigenvalue of $\mathbf{L}$ is $\mathbf{0}$ and the corresponding eigenvector is the constant one vector $\mathbf{1}$. When it comes to the normalized Laplacian matrix, the property still holds but the equation changes to:

$$\mathbf{f}'\mathbf{L}_{\text{sym}}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 \tag{2.27}$$

Based on the properties of normalized and unnormalized laplacian matrix, different spectral clustering algorithms are developed.

Unnormalized spectral clustering

The other property from (Von Luxburg [2007]), which related to the number of connected components, can be stated as: "Let G be an undirected graph with non-negative weights A. Then the multiplicity k of the eigenvalue 0 of $\mathbf{L}$ equals the number of connected components $A_1, \cdots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_k}$ of those components. "

The unnormalized spectral clustering algorithm can be summarized as algorithm 1.

Normalized spectral clustering

Algorithm 2 Normalized spectral clustering (Shi and Malik [2000])

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, number of cluster
1. Compute the unnormalized Laplacian matrix $\mathbf{L}_{\text{un}}$ as 2.23
2. Compute the smallest k generalized eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_k$ of generalized eigenproblem $\mathbf{Lu} = \lambda \mathbf{Du}$, and form $\mathbf{U} \in \mathbf{R}^{N\times k}$ with $\mathbf{u}_1, \cdots, \mathbf{u}_k$ as columns.
3. Each row of $\mathbf{U}$ is denoted by $\mathbf{y} \in \mathbb{R}^k, i = 1, \cdots, N$.
4. Cluster the points $\mathbf{y}_i, i = 1, \cdots, N$ with the k-means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \cdots, k\}^N$.
Output: Clusters $\mathbf{A}_1, \cdots, \mathbf{A}_k$ with $\mathbf{A}_i = \{j | \mathbf{y}_j \in \mathbf{C}_i\}$

Algorithm 3 Normalized spectral clustering (Ng et al. [2001])

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, number of cluster
1. Compute the normalized symmetric Laplacian matrix $\mathbf{L}_{\text{un}}$ as 2.25
2. Compute the smallest k eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_k$ of $\mathbf{L}_{\text{sym}}$, and form $\mathbf{U} \in \mathbf{R}^{N\times k}$ with $\mathbf{u}_1, \cdots, \mathbf{u}_k$ as columns. Each row of $\mathbf{U}$ is denoted by $\mathbf{t} \in \mathbb{R}^k, i = 1, \cdots, N$.
3. Normalize $\mathbf{y}_i$ to get $\mathbf{y}_i = \mathbf{t}_i / \|\mathbf{t}_i\|$ for $i = 1, \cdots, N$.
4. Cluster the points $\mathbf{y}_i, i = 1, \cdots, N$ with the k-means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \cdots, k\}^N$.
Output: Clusters $\mathbf{A}_1, \cdots, \mathbf{A}_k$ with $\mathbf{A}_i = \{j | \mathbf{y}_j \in \mathbf{C}_i\}$

Similarly, the normalized version proposition is: "Let G be an undirected graph with non-negative weights $\mathbf{A}$. Then the multiplicity k of the eigenvalue 0 of both $\mathbf{L}_{\text{rw}}$ and $\mathbf{L}_{\text{sym}}$ equals the number of connected components $A_1, \cdots, A_k$ in the graph. For $\mathbf{L}_{\text{rw}}$, the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_k}$ of those components.For $\mathbf{L}_{\text{sym}}$,the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{D}^{-1/2}\mathbf{1}_{A_1}, \cdots, \mathbf{D}^{-1/2}\mathbf{1}_{A_k}$ "

There are two widely used versions of normalized spectral clustering algorithms, one is proposed by (Shi and Malik [2000]) and the other one is introduced by (Ng et al. [2001]). We put the every step of those two type algorithms in algorithm 2 and 3:

Although the three algorithms listed above seem similar, they build three different graph Laplacian matrix. The main idea of them is to change the representation of the data points $\mathbf{x}_i$ to new representation $\mathbf{y}_i$. The representation varies due to the properties of the graph Laplacian change. After the new representation is obtained, the k-means clustering algorithm is applied to cluster the data. In general, the normalized spectral clustering algorithms perform better than the unnormalized one in previous literature, more concrete illustration and analysis about spectral clustering can be referred in (Von Luxburg [2007]). Thus, we refer the spectral clustering to the normalized one in the following context.

2.4.3 Evaluate Clustering

Unlike the classification problem, where each sample has a fixed label, clustering algorithm could change the label for each sample. Therefore, the Hungarian algorithm also known as Munkres algorithm is introduced in clustering task to make the evaluation easier. Hungarian algorithm is a combinatorial optimization method that solves the assignment problem in poly-

nomial time. The assignment problem is to find the optimal assignment that minimizes the total assignment cost. In clustering, the optimal assignment is the ground truth label and the initial assignment is the clustering result, and the total cost is the number of labels which is wrongly labeled.

Given the cost matrix $\mathbf{C} \in \mathbb{R}^{N\times N}$ of moving assigned labels to ground truth label, the main steps of Hungarian algorithm can be summarized as:

1. Subtract the smallest value in each row from all the element of its row,

2. Subtract the smallest value in each column from all the element of its column;

3. Make lines through all zeros for rows and columns in resulting matrix. If N lines are required, an optimal assignment exists among the zeros and the algorithm stops.

4. Find the smallest element which is not covered by a line in Step 3. Subtract the element from all uncovered elements, and add it to all elements which are covered. Return to step 3 until find the N lines.

Therefore, we use this algorithm to find the best matching between predicted clusters and the ground truth labels.

2.5 Saliency detection

Saliency detection aims at identifying the visually interesting objects in images that are consistent with human perception, which is intrinsic to various vision tasks. As Figure 2.6 shown, the algorithm is expected to generate a binary mask for each image to highlight the salient object or region. The applications of saliency detection models are widely spread in many other areas in computer vision, graphics and robotics, such as object detection and recognition (Borji and Itti [2011]), visual tracking (Borji et al. [2012]), human-robot interaction (Sugano et al. [2010]) and so on.

In general, a salient object detection model output a saliency map where the intensity of each pixel represents its probability of belonging to salient objects. Therefore, we can easily formulate the saliency detection as a binary classification problem or binary clustering problem, which can be represented mathematically as:

$$\hat{\mathbf{Y}} = \mathbf{f}(\mathbf{X};\theta), \tag{2.28}$$

where $\mathbf{X} = \{\mathbf{x}_i, i = 1,\cdots,N\}, \mathbf{x}_i \in \mathbb{D}^{h\times w}, \mathbb{D} \subset [0,1,\cdots,255]$, h and w are the height and width of input image, $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_i, i = 1,\ldots,N\}, \hat{\mathbf{y}}_i \subset (0,1)^{h\times w}$, N is number of training images, and θ denotes parameters of the model. In saliency map, if a pixel is a part of salient object the value of the pixel should be close to 1; otherwise, the pixel value is close to 0 when it belongs to background.

Depending on whether human annotations have been used, saliency detection methods can be roughly divided as unsupervised methods and supervised methods. The former ones compute the probability of salient area by exploiting various priors (e.g., , center prior (Goferman et al. [2012]), global contrast prior (Cheng et al. [2011]), background connectivity prior (Zhu et al. [2014])), which are summarized and described with human knowledge. The later ones learn a mapping from RGB images to saliency maps by exploiting the availability of large-scale human-annotated database.

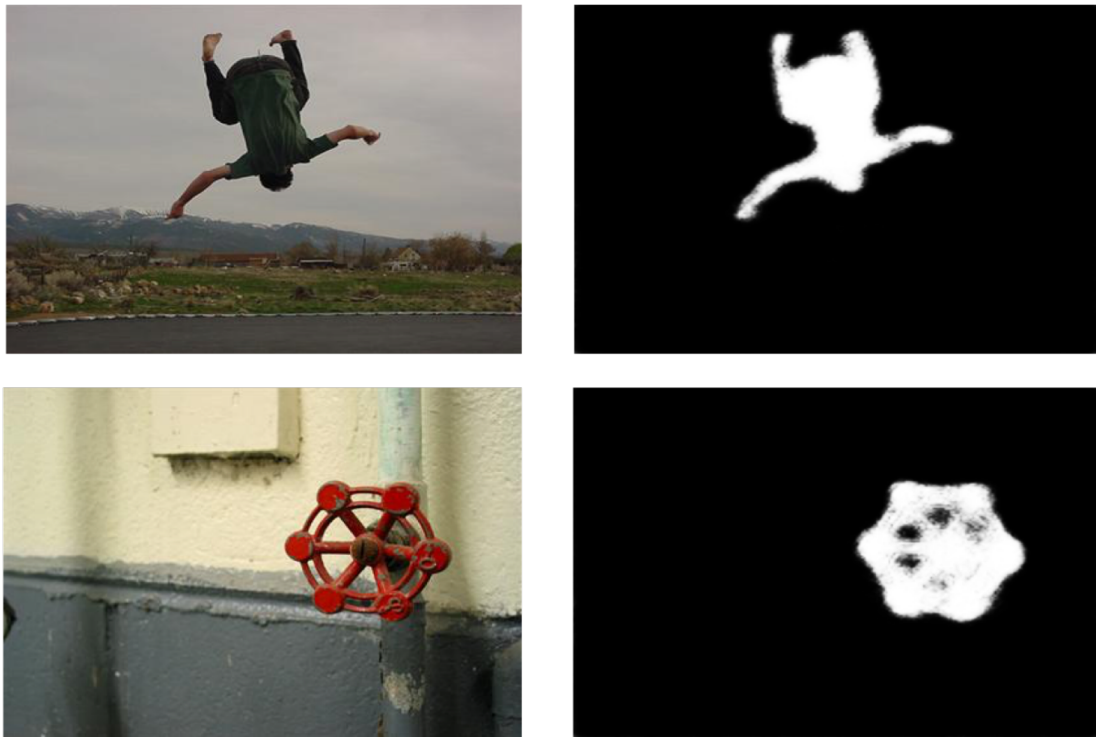


Figure 2.6: Saliency is dependent on what kind of visual stimuli human respond to most. The left column images are the input RGB images, and right column images are the corresponding ground truth saliency map. The saliency region is displayed as white.

Unsupervised method

Prior to the revolution of deep learning bring to computer vision, saliency detection methods mainly relied on employing different priors and handcrafted features (Zhu et al. [2014] Cheng et al. [2013a, 2011]; Goferman et al. [2012]). We refer interested readers to (Borji et al. [2014]) and (Borji et al. [2015]) for surveys and benchmark comparisons. Color contrast prior has been exploited at superpixel level in (Cheng et al. [2011]). (Shen and Wu [2012]) formulated saliency detection as a low-rank matrix decomposition problem by exploiting the sparsity prior for salient objects. Objectness, which highlights the object-like regions, has also been used in (Jiang et al. [2013c]) to mark the regions that have higher possibilities of being an object. (Zhu et al. [2014]) presented a robust background measure, namely "boundary connectivity" along with an optimization framework to measure backgroundness of each superpixel. Building upon the center prior, (Goferman et al. [2012]) detects the image regions that represent the scene, especially those that are near image center.

Supervised method

Conventional supervised techniques, such as (Jiang et al. [2013b]; Kim et al. [2014]), formulate saliency detection as a regression problem, and a classifier is trained to assign saliency at pixel or superpixel level. Recently, deep neural networks have been adopted successfully for saliency detection (Zhang et al. [2017c]; Luo et al. [2017]; Zhang et al. [2017d]; Wang

et al. [2017]; Hou et al. [2017]; Li et al. [2016]; Zhao et al. [2015]; Li and Yu [2015]; Wang et al. [2016]; Li and Yu [2016]; Zhang et al. [2017b]). Deep networks can encode high-level semantic features and hence capture saliency more effectively than both unsupervised saliency methods and non-deep supervised methods. Deep saliency detection methods generally train a deep neural network to assign saliency to each pixel or superpixel. Li and Yu (Li and Yu [2015]) used learned features from an existing CNN model to replace the handcrafted features. Recently, (Hou et al. [2017]) proposed a deep supervised framework with multi-branch short connections embed both high- and low-level features for accurate saliency detection. With the same purpose, a multi-level deep feature aggregation framework is proposed in (Zhang et al. [2017c]). A top-down strategy and a loss function which penalizes errors on the edge is presented in (Luo et al. [2017]).

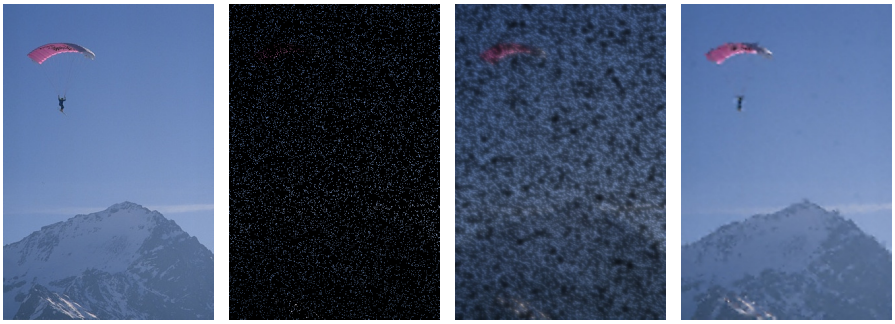
Chapter 3

Cascade Residuals Guided Nonlinear Dictionary Learning

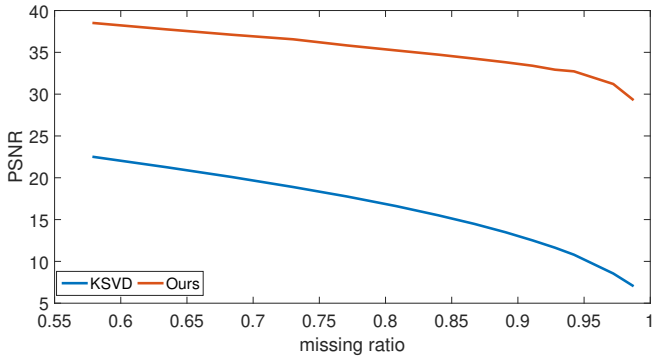
For mathematical convenience, dictionary learning methods often employ in uniform spaces, e.g. in the vector space of 8×8 image patches. In other words, same scale blocks are pulled from overlapping or non-overlapping image patches on a dense grid and a single-scale dictionary is learned. However, dictionary atoms learned in this fashion tend to be myopic and blind to global context since such fixed-scale patches only contain local information within their small support. Simply increasing the patch size results in adverse outcomes, i.e. de- creased the flexibility of the dictionary to fit data and increased computational complexity. Moreover, optimal patch size varies depending on the underlying texture information. For example, finer partitioning by smaller blocks is prefer- able for textured regions, yet larger blocks would suit better for smooth areas. Suppose the image to be encoded is a 256×256 flat (e.g. all pixels have the same value) image. Using the conventional 8×8 overlapping blocks would require more than 60K coefficients, yet the same image can be represented using a small number of coefficients of larger patches, even only a single coefficient in the ideal case of the patch has the size of the image.

Existing multi-scale dictionary learning methods overlook the redundancy between the layers. As a consequence, larger dictionaries are required, and a high number of coefficients are spent unnecessarily on smooth areas. To the best of our knowledge, no method offers a systematic solution where encodings of the coarser scales progressively enhance the reconstructions of the finer layers. Aiming to address the above shortcomings and allow dictionary atoms to access larger context for improved descriptive capacity, we present a computationally efficient cascade framework that employs multi-resolution residual maps for dictionary learning and sparse coding.

To this end, we start with building an image pyramid using bicubic interpolation. In the first pass, we learn a dictionary from the coarsest resolution layer and obtain the sparse representation. We upsample the reconstructed image and compute the residual in the next layer. The residual at a level is computed by the difference between the aggregated reconstructions from the coarser layers in a cascade fashion and the downsampled original image at that layer. Dictionaries are learned from the residual in every layer. We use the same patch size yet different resolution input images, which is instrumental in reducing computations and capturing larger context through. The computational efficiency stems from encoding at the coarsest res-



(a) Original (b) Corrupted (c) KSVD (d) Ours



(e) Reconstruction quality vs. ratio of missing pixels

Figure 3.1: (a) Original image. (b) Corrupt image where 93% of the original pixels are removed. (c) Reconstruction result of KSVD. PSNR is 11.80 dB. (d) Reconstruction result of our method. PSNR is 33.34 dB. (e) Reconstructed quality vs. the rate of missing pixels. As visible, our method is superior to KSVD.

olution and encoding the residuals that are significantly sparse. This enables our cascade to go as deep as needed without any compromise.

In the second pass, we collect all patches from all cascade layers and learn a single dictionary for a final encoding. This naturally solves the problem of determining how many atoms to be assigned at a hierarchical layer. Thus, all atoms in the dictionary have the same dimensionality while their receptive fields vary depending on the layer.

Compared to existing multi-scale approaches operating indiscriminately on image pyramids or wavelets, our dictionary comprises atoms that adapt to the information available at each layer. The details learned from residual images progressively refine our reconstruction objective. This allows our method to generate a flexible image representation using much smaller number of coefficients. Our extensive experiments demonstrate that our method applies favorably in image coding, denoising, inpainting and artifact removal tasks. Figure 3.1 shows an inpainting result generated by our method where the input image was missing 93% of its pixels. As visible, we can recover even the very large areas of missing pixels.

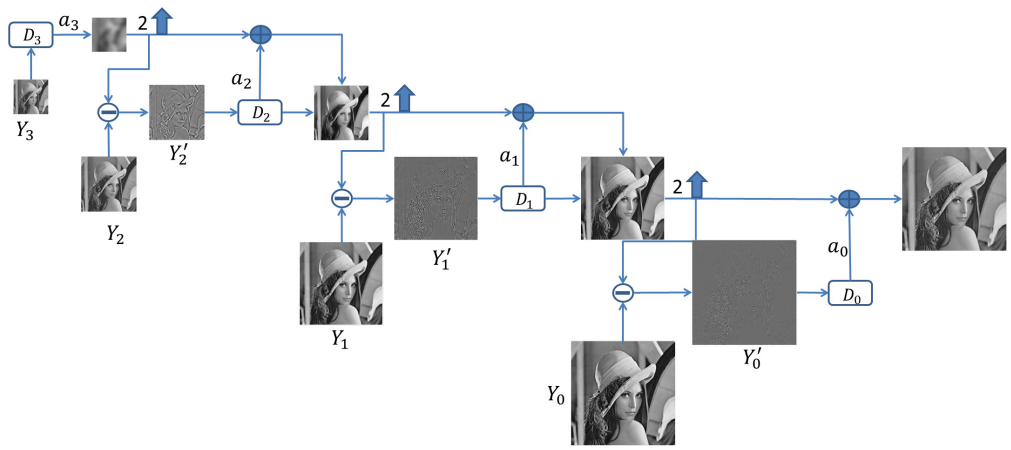


Figure 3.2: The first pass of our method for a 4-layer cascade. $\mathbf{Y}_0$ is the original image, $\{\mathbf{Y}_3, ..., \mathbf{Y}_0\}$ denote each layer of the image $\mathbf{Y}_3$ pyramid, and $\{\mathbf{D}_3, ..., \mathbf{D}_0\}$ are the dictionaries. $\mathbf{D}_3$ is learned from the downsampled image $\mathbf{Y}_3$ and the remaining dictionaries are learned from the residuals $\{\mathbf{Y}_2', \mathbf{Y}_1', \mathbf{Y}_0'\}$. $\mathbf{ff}_n$ are the reconstruction coefficients corresponding to the residual layers $\mathbf{Y}_n'$.

3.1 Sparse Coding on Cascade Layers

As mentioned above, previous dictionary learning algorithms often formulate the problem at hand using a linear model on a fixed dimension thus on a fixed patch scale, which hinders exploiting dictionary atoms in their full potential. In comparison, our approach is nonlinear due to its recursive nature where we encode the resulting residuals of the layers in previous hierarchical levels. In a single layer, we represent the current vector as a linear combination of dictionary atoms, where we keep the same as single layer sparse coding. After each layer, the representations are accumulated into the final reconstruction at the end. Let $\hat{\mathbf{Y}}_n'$ denote the estimated n-th layer and $\hat{\mathbf{Y}}$ denote the reconstructed image, then the overall process can be described as

$$\hat{\mathbf{Y}} = \hat{\mathbf{Y}}_0' + \mathbf{U}(\hat{\mathbf{Y}}_1' + \mathbf{U}(\hat{\mathbf{Y}}_2' + ... + \mathbf{U}(\hat{\mathbf{Y}}_N'))), \tag{3.1}$$

where $\mathbf{U}$ is an upsampling function.

A flow diagram of our framework is shown in Fig. 3.2 for a sample 4-layer cascade, where the input is a 512×512 grayscale image $\mathbf{Y}$. We first construct an image pyramid $\mathbf{Y} = \{\mathbf{Y}_0, \mathbf{Y}_1, ...\mathbf{Y}_N\}$ by bicubic downsampling. Here, $\mathbf{Y}_0$ is the finest (original) resolution and $\mathbf{Y}_N$ is the coarsest resolution.

Other options for the image pyramid would be Gaussian pyramid, Laplacian pyramid, bilinear interpolation, and subsampling. Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts. We employ a two-pass scheme where in the first pass we obtain residuals from layer-wise dictionaries, and in the second pass, we learn a single global dictionary that extracts and refines the atoms of the dictionaries generated in the first pass.

Algorithm 4 Cascade Sparse Coding

Require:

N (the highest pyramid layer), $\mathbf{Y}$(image),

T_n (number of coefficient used in layer n)

Ensure: $\mathbf{Y}^{'}, \hat{\mathbf{Y}}, \hat{\mathbf{D}}_{global}$

$\mathbf{Y}_n \leftarrow \text{subsampling}(\mathbf{Y}, 2^n)$

for $n = \{N, N-1, \cdots, 0\}$ **do**

  if n = N **then**

    $\mathbf{Y}_n^{'} \leftarrow \mathbf{Y}_n$

  else

    $\mathbf{Y}_n^{'} \leftarrow \mathbf{Y}_n - \text{upsample}(\hat{\mathbf{Y}}_{n+1}, 2)$

  Perform KSVD to learn dictionary $\hat{\mathbf{D}}_n$ and encode $\mathbf{Y}_n^{'}$

  $\forall ij \ \{\hat{\mathbf{x}}_n^{ij}, \hat{\mathbf{D}}_n\} \leftarrow \arg\min_{\mathbf{x}_n^{ij}, \mathbf{D}_n} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}_n^{'} - \mathbf{D}_n\mathbf{x}_n^{ij}\|_2^2 \quad \text{s.t } \|\mathbf{x}_n^{ij}\|_0 \leq T_n$

  if n = N **then**

    $\hat{\mathbf{Y}}_n \leftarrow (\sum_{ij} \mathbf{R}_{ij}^T\mathbf{R}_{ij})^{-1}(\sum_{ij} \mathbf{R}_{ij}^T\hat{\mathbf{D}}_n\hat{\mathbf{x}}_n^{ij})$

  else

    $\hat{\mathbf{Y}}_n \leftarrow (\sum_{ij} \mathbf{R}_{ij}^T\mathbf{R}_{ij})^{-1}(\sum_{ij} \mathbf{R}_{ij}^T\hat{\mathbf{D}}_n\hat{\mathbf{x}}_n^{ij}) + \text{upsample}(\hat{\mathbf{Y}}_{n+1}, 2)$

$\mathbf{Y}^{'} \leftarrow \{\mathbf{Y}_N^{'}, \mathbf{Y}_{N-1}^{'} \cdots, \mathbf{Y}_0^{'}\}$

$\forall ij \ \hat{\mathbf{D}}_{\text{global}} \leftarrow \arg\min_{\mathbf{D}} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}^{'} - \mathbf{D}\mathbf{x}^{ij}\|_2^2 \quad \text{s.t } \|\mathbf{x}^{ij}\|_0 \leq T$

Reconstruction:

$\hat{\mathbf{Y}} \leftarrow 0$

for $n = \{N, N-1, \cdots, 0\}$ **do**

  $\mathbf{Y}_n^{'} = \mathbf{Y}_n - \text{upsample}(\hat{\mathbf{Y}}, 2)$

  $\forall ij \ \{\hat{\mathbf{x}}_n^{ij}\} \leftarrow \arg\min_{\mathbf{x}_n^{ij}} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}_n^{'} - \hat{\mathbf{D}}_{global}\mathbf{x}_n^{ij}\|_2^2 \quad \text{s.t } \|\mathbf{x}_n^{ij}\|_0 \leq T_n$

  $\hat{\mathbf{Y}} \leftarrow (\sum_{ij} \mathbf{R}_{ij}^T\mathbf{R}_{ij})^{-1}(\sum_{ij} \mathbf{R}_{ij}^T\hat{\mathbf{D}}_{global}\hat{\mathbf{x}}_n^{ij}) + \text{upsample}(\hat{\mathbf{Y}}, 2)$

return

3.1.1 First Pass

We start at the coarsest (N-th) layer in the cascade. After learning the layer dictionary and finding the sparse coefficients, we propagate consecutively the reconstructed images to the finer layers.In the coarsest layer, we process the downsampled image. In the consecutive layers, we encode and decode the residuals. In each layer, we keep the size of image patches identical, which enable that a $b \times b$ patch in n-th layer corresponds to a $(2^n b) \times (2^n b)$ area in the original image. Algorithm 4 summarizes the first pass.

Dictionary Learning: We learn a dictionary at the coarsest layer and use it to reconstruct the downsampled image. This layer's dictionary $\hat{\mathbf{D}}_N$ is produced by minimizing the objective function using the coarsest resolution image patches

$$\underset{\mathbf{D}_N, \mathbf{x}_N^{ij}}{\arg\min} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}_N - \mathbf{D}_N \mathbf{x}_N^{ij}\|_2^2 + \lambda \|\mathbf{x}_N^{ij}\|_0 \tag{3.2}$$

where the operator $\mathbf{R}_{ij}$ is a binary matrix that extracts a square patch of size $b \times b$ at location (i, j) in the image then arranges the patch pixels into a column vector form. The parameter λ trades off the data fidelity term and the regularization term, and $\mathbf{x}_N^{ij}$ denotes the coefficients for the patch (i, j) .

In Fig. (3.9), we compare the efficiency of different learning algorithms. As shown, KSVD Aharon et al. [2006] underperforms in comparison to a-KSVD Rubinstein et al. [2008] and ODL Mairal et al. [2009a] where both ODL and a-KSVD achieve the same PSNR with fewer coefficients. Our method does not assume a specific dictionary learning technique, and it can use any dictionary learning technique regardless of the way they update dictionary atoms. To demonstrate that our quality and sparsity improvements are not simply due to a specific choice of dictionary learning method, we employ the relatively handicapped and underperforming method, the original KSVD, to obtain our dictionaries. We initialize the dictionary $\mathbf{D}_N$ with a DCT basis by extracting several atoms from the DCT basis and then applying Kronecker product on the atoms to generate an overcomplete matrix, which is similar to KSVD. Notice that using a more efficient initialization scheme may produce better results and improve convergence Sulam et al. [2014].

During the dictionary learning stage, we fix all coefficient vectors $\mathbf{x}_N^{ij}$ and iteratively select dictionary atoms $\mathbf{d}_N^l$ one by one, $l = \{1, 2, \cdots, k\}$. For each atom $\mathbf{d}_N^l$, we extract the patches that are composed by the atom $(i, j) \in \mathbf{d}_N^l$ to compute the corresponding residual without the atom $\mathbf{d}_N^l$. The coefficients are denoted as $\mathbf{x}_N^{ij}(l)$, which are the non-zero entries of the l-th row of the coefficient matrix

$$\mathbf{e}_N^{ij}(l) = \mathbf{R}_{ij}\mathbf{Y}_N - \hat{\mathbf{D}}_N \mathbf{x}_N^{ij} + \mathbf{d}_N^l \mathbf{x}_N^{ij}(l). \tag{3.3}$$

Then, we arrange all $\mathbf{e}_N^{ij}(l)$ as the columns of the overall representation error matrix $\mathbf{E}_N^l$. We update the atom $\hat{\mathbf{d}}_N^l$ and the l-th row of coefficient matrix $\hat{\mathbf{x}}_N(l)$ by solving the equation

$$\{\hat{\mathbf{d}}_N^l, \hat{\mathbf{x}}_N(l)\} = \underset{\mathbf{d}, \mathbf{x}}{\arg\min} \|\mathbf{E}_N^l - \mathbf{d}\mathbf{x}\|_F^2. \tag{3.4}$$

Finally, we perform a SVD decomposition on the error matrix, and update the l-th dictionary atom $\hat{\mathbf{d}}_N^l$ by the first column of $\mathbf{U}$, where $\mathbf{E}_N^l = \mathbf{U}\Sigma\mathbf{V}^T$; the coefficient vector $\hat{\mathbf{x}}_N(l)$ is replaced by the first column of matrix $\Sigma(1,1)\mathbf{V}$. In every iteration, all atoms and coefficients are updated simultaneously.

Sparse Coding: After obtaining the updated dictionary, sparse coding is employed with the Orthogonal Matching Pursuit (OMP), which is a computationally efficient greedy algorithm Tropp [2004]. The sparse coding stops when the number of the non-zero coefficients reaches the upper limit T_N, or the reconstruction error becomes less than the threshold value, which depends on the specific task in hand. We update the coefficient vector $\hat{\mathbf{x}}_N^{ij}$ as

$$\hat{\mathbf{x}}_N^{ij} = \arg\min_{\mathbf{x}_N^{ij}} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}_N^{'} - \hat{\mathbf{D}}_N \mathbf{x}_N^{ij}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}_N^{ij}\|_0 \leq T_N \tag{3.5}$$

and put it back into the dictionary learning stage to update the dictionary atoms and the coefficients.

Residuals: In each layer, we use at most T_n active coefficients for each patch to reconstruct the image and then compute the residual. The number of coefficients governs how strong the residual should emerge. Larger values of T_n favors for more accurate reconstructions; thus the total energy of residuals will decay. Smaller values of T_n cause the residual to increase, not only due to sparse coding but also resampling across layers. Since the dictionary is designed to represent a broad spectrum of patterns to keep the encodings as sparse as possible, T_n should be small. The reconstructed image is a weighted average of the patches that contain the same pixel

$$\hat{\mathbf{Y}}_N = (\sum_{ij} \mathbf{R}_{ij}^T \mathbf{R}_{ij})^{-1} (\sum_{ij} \mathbf{R}_{ij}^T \hat{\mathbf{D}}_N \hat{\mathbf{x}}_N^{ij}). \tag{3.6}$$

After decoding based on the dictionary $\hat{\mathbf{D}}_N$, we obtain the residual image $\mathbf{Y}_{N-1}^{'}$ by subtracting the upsampled reconstruction $\mathbf{U}(\hat{\mathbf{Y}}_N)$ from the next layer image $\mathbf{Y}_{N-1}$, e.g. $\mathbf{Y}_{N-1}^{'} = \mathbf{Y}_{N-1} - \mathbf{U}(\hat{\mathbf{Y}}_N)$. Here, $\mathbf{U}(\cdot)$ denotes the bicubic upsampling operator. Similar to the above dictionary learning and sparse coding procedure for the N-th layer, we reconstruct the residual $\hat{\mathbf{Y}}_{N-1}^{'}$ by training a separate residual dictionary $\mathbf{D}_{N-1}$ from the residual image itself. We keep encoding and decoding on residuals up to the finest layer. The procedure for the cascade residual dictionary learning and reconstruction can be expressed as follows:

$$\{\hat{\mathbf{x}}_n^{ij}, \hat{\mathbf{D}}_n\} = \arg\min_{\mathbf{x}_n^{ij}, \mathbf{D}_n} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{Y}_n^{'} - \mathbf{D}_n \mathbf{x}_n^{ij}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}_n^{ij}\|_0 \leq T_n, \tag{3.7}$$

where residual image is

$$\mathbf{Y}_n^{'} = \begin{cases} \mathbf{Y}_n - \mathbf{U}(\hat{\mathbf{Y}}_{n+1}), & 0 \leq n < N \\ \mathbf{Y}_N, & n = N, \end{cases} \tag{3.8}$$

and the reconstructed residual is:

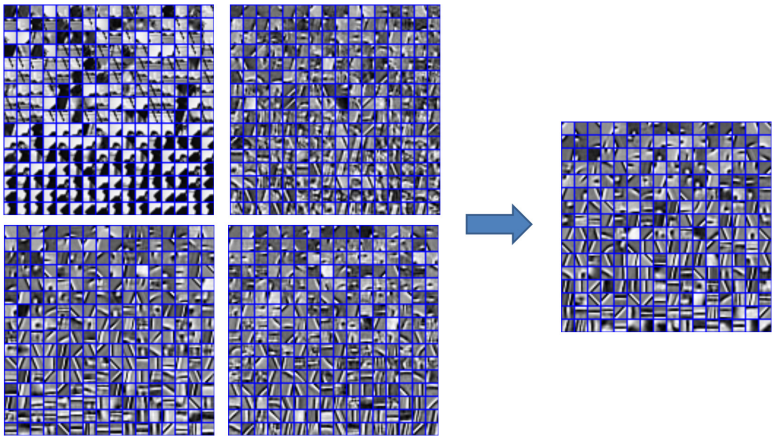


Figure 3.3: Left: The dictionaries learned in the first pass for the different levels (clockwise from the upper left: the coarsest level, the second level, the third level, and the finest level). Right: The unifying dictionary learned in the second pass.

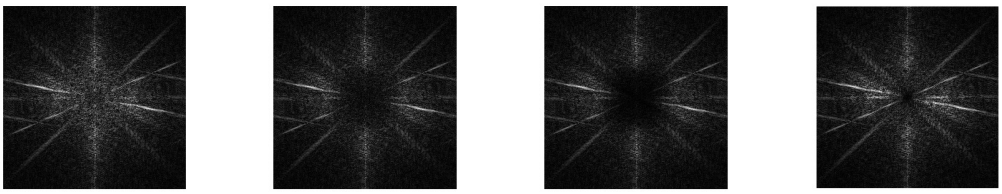
$$\hat{\mathbf{Y}}_n = \begin{cases} (\sum_{ij} \mathbf{R}_{ij}^T \mathbf{R}_{ij})^{-1} (\sum_{ij} \mathbf{R}_{ij}^T \hat{\mathbf{D}}_n \hat{\mathbf{x}}_n^{ij}) + \mathbf{U}(\hat{\mathbf{Y}}_{n+1}), & 0 \leq n < N \\ (\sum_{ij} \mathbf{R}_{ij}^T \mathbf{R}_{ij})^{-1} (\sum_{ij} \mathbf{R}_{ij}^T \hat{\mathbf{D}}_n \hat{\mathbf{x}}_n^{ij}), & n = N. \end{cases} \tag{3.9}$$

Above, Eqn. 5.4 computes the coefficients with respect to the corresponding patches, and Eqn. 3.8 reconstructs the residual image for the next finer layer by subtracting the upsampled version of the coarser layer image from the image pyramid of the given layer. Similarly, Eqn. 3.9 is the general formulation of how we progressively reconstruct the image by adding the estimated residual and the upsampled image from the coarser layers.Increasing the number of non-zero coefficients can reduce the error caused by the sparse representation. There is a trade-off between the number of coefficients and the quality of the reconstructed image. Our goal is to use the minimal number of coefficients while reconstructing an image of highest quality.

3.1.2 Second Pass

In each layer, the more atoms we use, the better quality can be achieved. However, this would not be the best use of the limited number of atoms. For instance, image patches from the coarsest layer are limited both in quantity and variety. The residual images are relatively sparse which imply they do not require many dictionary atoms. However, it is not straightforward to determine the optimal number of atoms for each dictionary since the finer level residuals depend heavily on the coarser ones.

Rather than keeping all dictionaries, we train a global dictionary $\mathbf{D}$ using patches from $\mathbf{Y}' = \{\mathbf{Y}_N, \mathbf{Y}'_{N-1}, \cdots, \mathbf{Y}'_0\}$. As illustrated in Fig. 3.3, the dictionaries learned from $\mathbf{Y}'$ in the first pass are redundant. The overall dictionary is less repetitive thus more effective to reconstruct all four layers. Using a unified dictionary allows us to select most useful atoms automatically without making possibly suboptimal layer-wise decisions. Notice that, in this



$T_n = 1$ $T_n = 4$ $T_n = 10$ Laplacian

Figure 3.4: Residuals of the finest layer in the frequency domain for different values of coefficients used for each patch of Cameraman image is as the input. Right most is the Laplacian pyramid layer of the finest resolution. As visible, our method generates different layers depending on the sparsity level.

procedure, the number of coefficients can be chosen depending on the target quality of each layer.

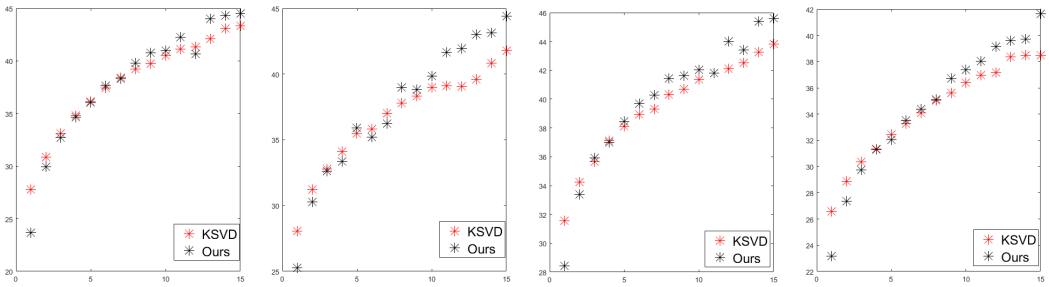
3.2 Analysis

3.2.1 Role of the First Pass

The goal of the second pass is to find a unified and compact dictionary that is suitable for the reconstruction of all layers. From the coarsest to the finest layer, our algorithm reconstructs the input images at each layer. In the coarsest layer, the input image is a thumbnail version of the original image. In the following layers, the images correspond to the residuals between the reconstructed images and the scaled version of the original image. Our layers, except the coarsest one, are different from the corresponding Laplacian pyramid layers. To visualize this, we show the frequency domain versions of the residual in the finest layer for different levels of sparsity (1, 4, 10) applied to all other layers in Fig. 3.4. We also show the frequency transform of the finest level Laplacian pyramid image. As visible, using a higher number of coefficients in our method yields smaller residuals, in particular, the low-frequency components are more accurately reconstructed. When the sparsity level is 1, the finest level image we obtain with our method 3.4-a and the Laplacian pyramid 3.4-d seem similar, yet as the sparsity level increases, their difference dilates significantly. If we learn a dictionary using the Laplacian pyramid and encode all layers using one coefficient per patch, the PSNR is 0.2 dB smaller than our hierarchical method. The PSNR will be less than 1 dB in case our method uses 10 coefficients per patch. These show that our residuals and Laplacian pyramid have different characteristics. Also, the residual pyramid generated by our method in the first pass plays a critical role in the reconstruction performance.

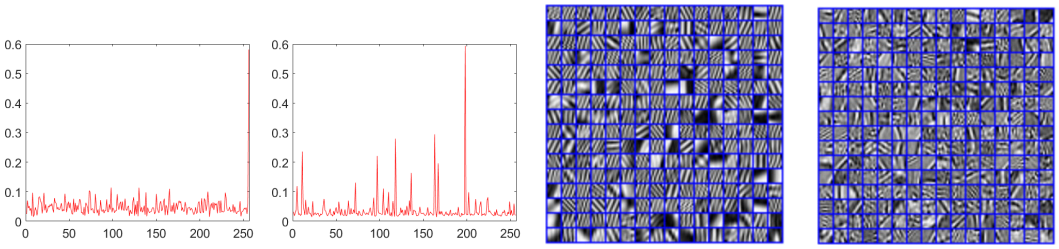
3.2.2 Second Pass: Generating a Unified Dictionary

The non-convex nature of the optimization algorithm for dictionary learning, i.e., updating the steps of learning the dictionary and then the corresponding sparse coefficients in a loop, may cause the solution to converge into one of the local minima. In our method, we utilize



Barbara Boat Lena Cameraman

Figure 3.5: Reconstruction quality between the single layer learned dictionary and our dictionary. Horizontal axis is the sparsity (T_n per patch), and the vertical axis is PSNR in dB. Red: conventional dictionary, Black: dictionary generated by our algorithm.



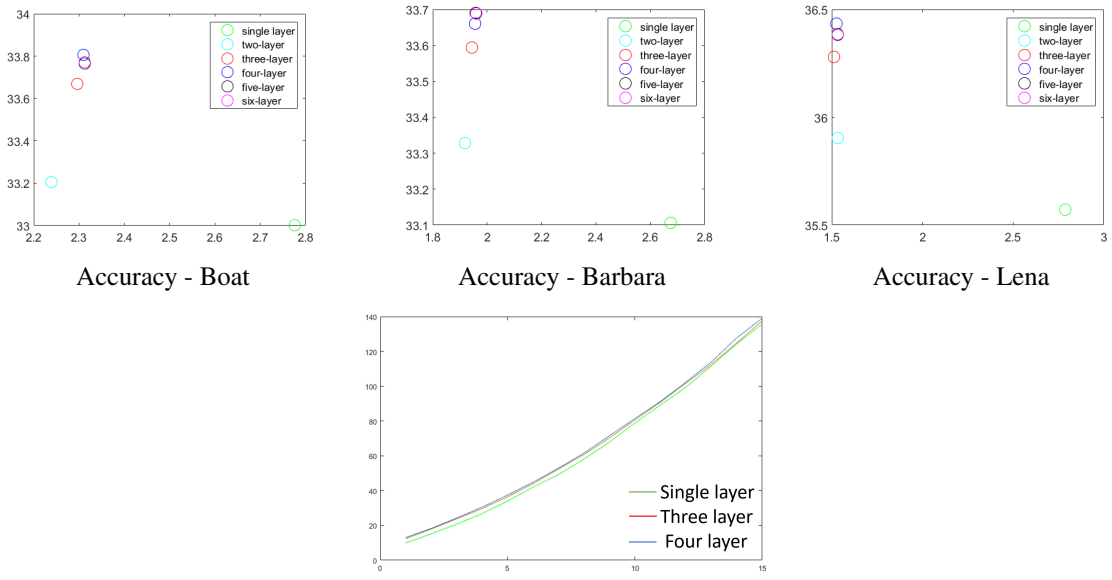
KSVD Ours KSVD Ours

Figure 3.6: Left: The frequency graphs of atoms when 15 coefficients are used in reconstruction. Right: the dictionaries generated by the KSVD and our method.

the OMP for sparse encoding, which is a greedy algorithm that does not guarantee the global minimum. Although we are seeking for a linear model for every layer, the final dictionary is based on the dictionaries of the previous layers. Thus, the solution we obtain can be regarded as a combination of the previous local minima.

To assess which dictionary learning method provides a higher reconstruction performance, we compare the reconstruction power of the dictionaries learned by the original KSVD method and our algorithm. We reconstruct the same single layer image by using OMP with a different number of coefficients. Figure 3.5 shows that our approach achieves higher PSNR values than using the original KSVD.

We also notice that the probability of each dictionary atom utilized in our reconstruction is different from the KSVD dictionary. In Sandin and Martin-del Campo [2016] a method called Equiprobable Matching Pursuit (EMP) where a probability constraint is incorporated to prevent a few atoms dominating the reconstruction is proposed. Our nonlinear dictionary learning also generates a dictionary that can avert having one or two atoms to become dominant to others, achieving the same goal as EMP without imposing any additional constraints. Figure 3.6 shows that the atoms in our dictionary are utilized more uniformly. In comparison, KSVD exploits one atom more often than others. At the same time, the dictionary atoms learned by our algorithm are more diverse than the ones in the KSVD dictionary.



Accuracy - Boat | Accuracy - Barbara | Accuracy - Lena

Computational load

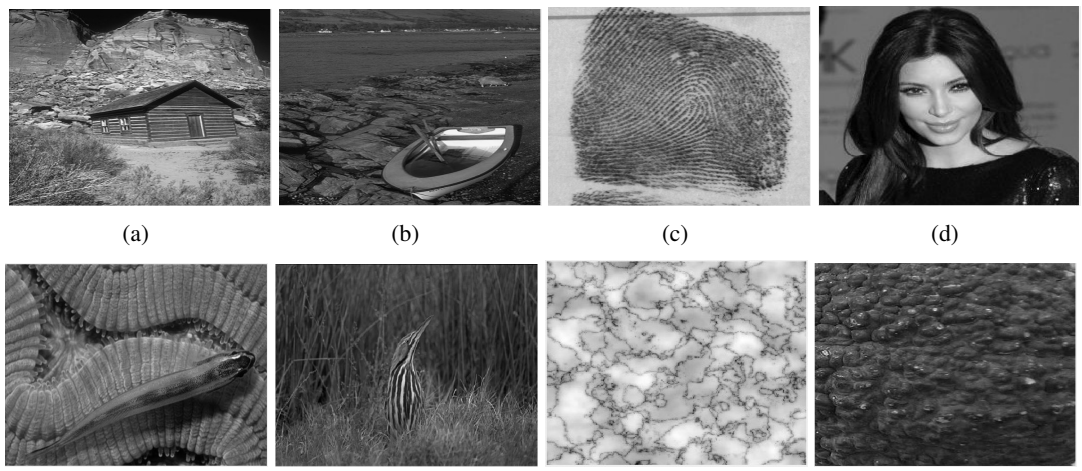
Figure 3.7: Top: The PSNR vs the average number of coefficients per pixel for different layer versions of our method and single-layer version. Bottom: Computational times with respect to the number of coefficients used (single-layer is KSVD, others are our cascade method).

3.2.3 Layers Matter

There is a positive correlation between the quality of the reconstruction and the number of layers in our cascaded framework. We also notice in the bottom graph in Fig. 3.7 that the computational complexity does not change much with the increase of the layers. Does this mean the deeper hierarchical models are better?

To seek an answer to the question of the optimal number of the layers, we analyze the reconstruction results for different number of layers from 1 to 6 on three test images (Boat, Barbara, Lena) as reported in Fig. 3.7. We observe that our multi-layer reconstruction is more accurate than single layer reconstruction while using a smaller number of coefficients. However, the results do not improve remarkably after the fourth-layer reconstruction. Since the number of patches extracted from the fifth and sixth layers are only 625 and 72, respectively, which is only approximately $1/400$ and $1/4000$ of the number of patches extracted from the finest layer, they hardly influence the dictionary building, leading a larger error for these two layers (as a result, using more coefficients in the following layers to fix this). On the other hand, reconstructing a 8×8 patch in the fifth layer is equal to a 128×128 patch in the finest layer, which is too large to estimate accurately using small dictionary atoms. We find that in most images, a four-layer pyramid provides an optimal hierarchical representation.

As in Fig. 3.7, our method does not increase the computational load in comparison to a single layer and it would benefit from faster optimization techniques for a single layer. A discussion on the converge analysis of different optimization techniques for a single layer such as K-SVD, Accelerated Plain Dictionary Learning, etc. can be found in Bao et al. [2016].



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.8: Sample images from 5 datasets.

3.3 Experimental Analysis

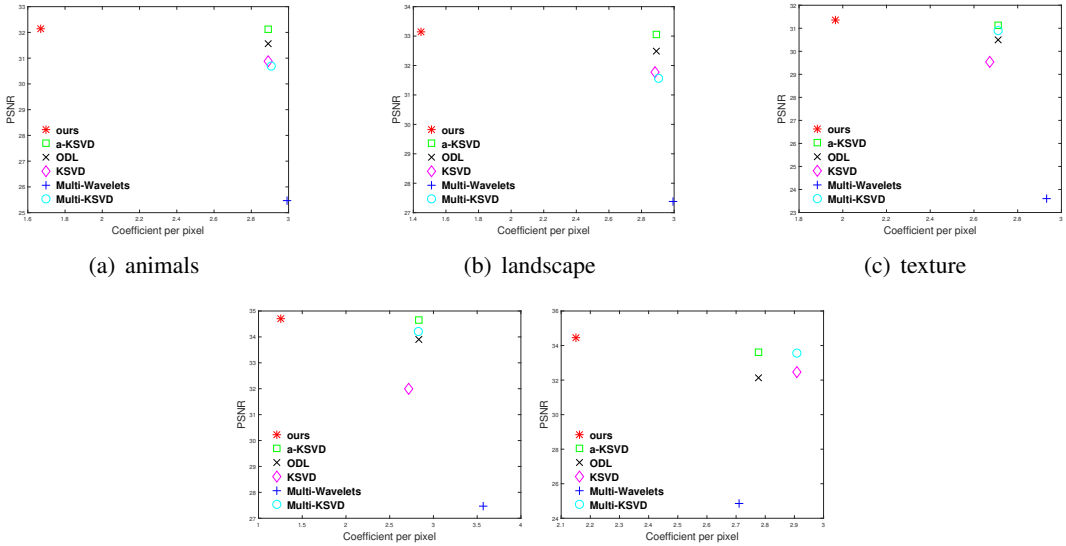
To demonstrate the flexibility of our method, we evaluate its performance on three different and popular image processing tasks: image coding, image denoising, and image inpainting. For a comprehensive evaluation, we build five different image datasets, where each dataset contains 50 images of specific object classes: animals, landscapes, textures, faces, and fingerprints (all color except the fingerprint images, which are grayscale). Some of these images are selected from the BSD300 Martin et al. [2001] and CelebA Liu et al. [2015b] datasets, and the rest are downloaded from the websites. The size of the images in these datasets varies from 256×256 to 480×440. The grayscale versions of sample images are shown in Fig. (3.8).

3.3.1 Image Coding

We compare our method with 5 state-of-the-art dictionary learning algorithms including both single and multi-scale methods: approximate KSVD (a-KSVD) Rubinstein et al. [2008], ODL Mairal et al. [2009a], KSVD Aharon et al. [2006], multi-scale KSVD Mairal et al. [2008], multi-scale KSVD using wavelets (multi-wavelets) Ophir et al. [2011].

For objectiveness, we use the same number of dictionary atoms for our and all other methods. Notice that, a larger dictionary would generate a sparser representations. We employ $4\times$ overcomplete dictionaries, i.e. $\mathbf{D} \in \mathbb{R}^{64\times256}$ except for the multi-wavelets where the dictionary in each sub-band has as many atoms as our dictionary (in favor of the multi-wavelets). For multi-scale KSVD, the maximum dimension of dictionary atom can be 16 due to the storage issue and only 2 scales can be performed. Thus, we extracted 128 atoms at each scale.

Figure 3.9 depicts the number of coefficients per pixel as the function of the number of coefficient per each pixel. Each point is the average score per pixel for the corresponding method. As seen, our method is the best performing algorithm among the state-of-the-art. In all five image datasets, it achieves the highest PSNR scores with significantly much less number of coefficients. In these experiments, the patches are extracted by 1-pixel overlapping



(a) animals (b) landscape (c) texture

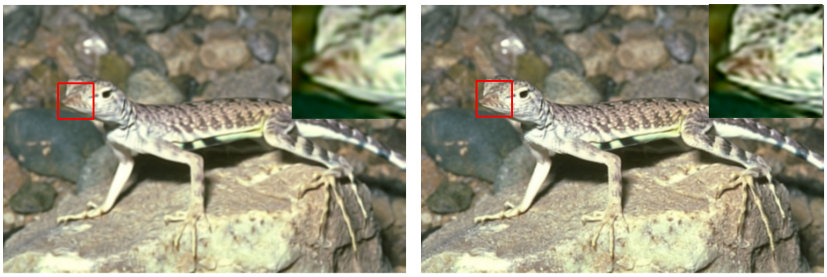
(d) face (e) fingerprint

Figure 3.9: Reconstruction results on different 5 different image datasets. The horizontal axis represents the number of coefficient per pixel and the vertical axis is the quality in terms of PSNR (dB).

in all images. We use 8×8 blocks on each layer, and the cascade comprises 4 layers. Since the blocks in every layer have the same size, the lower resolution blocks efficiently represent larger receptive fields when they are upsampling onto a higher resolution.

When decoding on the coarsest resolution, our method employs 8×8 blocks, which corresponds to $8 \cdot 2^{n-1} \times 8 \cdot 2^{n-1}$ patches on the finest (original) resolution using the same dictionary atoms. Since there is a single global dictionary after the second pass, all layers share the same atoms. Even though this may resemble the quadtree structure, our method is not limited by the size of the dictionary (patch size, i.e., the dimensionality of the atoms, and the number of the atoms). Furthermore, it is as fast as the baseline single-scale dictionary learning and sparse coding methods.

Compared with other algorithms, our method can save an outstanding 55.6%, 42.23% and 49.95% coefficients for the face, animals, and landscape datasets, respectively. For the image classes where spatial texture is dominant, our method is also superior by decreasing the number of coefficient by 27.74% and 22.38% for the texture and fingerprint datasets. The ratio is defined as $(c_1 - c_0)/c_1$, where c_0 is the number of the coefficients employed by our algorithm and c_1 is the number of the coefficients used by the second best algorithm. Note that, for all the five datasets, our algorithm achieves the highest PSNR while using much fewer coefficients. The second best algorithm is a-KSVD (Fig. (3.9)).Sample image coding results for qualitative assessment are given in Fig. 3.10. As shown, a-KSVD image coding generates inferior results even though it uses more coefficients.



(a) a-KSVD: 28.68 db PSNR (b) Our method: 32.62 db PSNR

Figure 3.10: Image coding results the comparison between a-KSVD and our method. Our method uses 1309035 coefficients and achieves 32.62 db PSNR score, while a-KSVD uses 1332286 coefficients to get 28.65 dB PSNR. our method is almost **4 dB** better. Enlarged red regions are shown on the top-right corner of each image. As visible, our method produces more detailed reconstructions.

Table 3.1: Denoising results (PSNR) on different test images for $\sigma = 10$

	KSVD	ODL	a-KSVD	M-W	m-KSVD	Ours
a	31.10	30.98	31.05	30.95	**31.16**	30.61
b	32.93	**33.05**	32.93	32.74	33.02	32.91
c	34.05	**34.09**	34.01	33.99	33.42	**34.09**
d	35.61	35.67	35.62	32.36	35.52	**35.70**
e	34.18	**34.38**	34.20	34.13	34.07	34.33
f	34.35	**34.57**	34.38	34.51	34.47	34.52
g	33.18	33.52	33.22	33.49	33.50	**33.74**
h	33.90	33.91	**34.00**	33.85	33.85	**34.00**

3.3.2 Image Denoising

We also analyze the image denoising performance of our method and make comparisons with five dictionary learning algorithms. We note that the state of the art in denoising use collaborative and non-local techniques such as BM3D Dabov et al. [2007] and LSSC Mairal et al. [2009c]. However, our goal here is not to design a yet another collaborative scheme. Instead, we aim to understand how our method compares to other dictionary learning methods.

We minimize the cost function in Eqn. (3.10) for denoising. We use the difference between the downsampled input image and aggregated reconstructions at each layer to terminate the OMP.

$$\begin{aligned} &\hat{\mathbf{x}}_n^{ij} = \arg\min_{\mathbf{x}_i} \sum_{ij} \|\mathbf{x}_n^{ij}\|_0 \\ &\text{s.t.} \|\mathbf{R}_{ij}\mathbf{Y}_n - \mathbf{D}_n\mathbf{x}_n^{ij} + \mathbf{R}_{ij}\mathbf{U}(\hat{\mathbf{Y}}_{n+1})\|_2^2 \leq C\sigma. \end{aligned} \tag{3.10}$$

Above, the reconstructed residual $\hat{\mathbf{Y}}_{n+1}$ is defined as in Eqn. (3.9), and σ is chosen according to the variance of the noise. As before, we choose the 4-layer cascade and 8×8 patch size. The

Figure 3.11: Denoised images. Additive zero-mean Gaussian noise with $\sigma = 30$.

Table 3.2: Denoising results (PSNR) on different test images for $\sigma = 30$.

	KSVD	ODL	a-KSVD	M-W	m-KSVD	Ours
a	25.03	25.04	25.06	25.08	**25.10**	25.03
b	27.79	27.84	27.78	27.83	27.78	**27.85**
c	27.48	26.96	27.46	**28.38**	27.77	28.01
d	30.33	**30.39**	30.35	30.11	30.13	30.29
e	28.36	28.30	28.32	**29.10**	28.53	29.08
f	28.50	28.46	28.44	**29.21**	28.59	29.06
g	27.71	27.46	27.69	28.12	27.86	**28.20**
h	28.30	28.29	28.27	28.69	28.37	**28.83**

Table 3.3: Denoising results (PSNR) on test images for $\sigma = 50$.

	KSVD	ODL	a-KSVD	M-W	m-KSVD	Ours
a	22.75	20.80	22.74	**23.10**	22.85	22.88
b	25.75	24.27	25.73	**26.06**	25.63	25.95
c	24.19	22.65	24.16	**26.15**	24.66	25.92
d	27.80	25.09	27.84	27.79	27.52	**27.85**
e	26.65	26.05	26.63	27.09	26.42	**27.19**
f	26.72	25.27	26.70	**27.14**	26.43	26.85
g	26.04	25.73	26.05	**26.27**	25.80	26.19
h	26.45	25.82	26.43	**26.63**	26.20	26.56

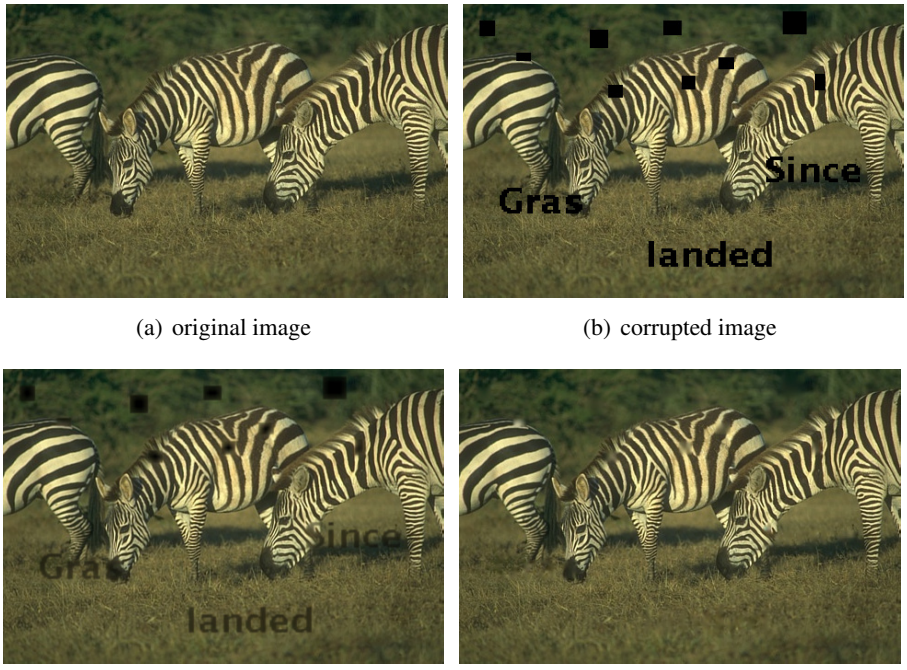
parameters of KSVD and multi-scale wavelets are set as recommenced by original authors. We fixed all hyperparameters for all test images. Since the denoising task is totally different from image coding, we do not need to force the size of dictionary to be identical for all algorithms. In multi-scale methods, the residuals in the finer layers are mostly noise, which cannot be used to learn an efficient dictionary. Therefore, we learn a dictionary for each layer per class from the clean images, which is similar to the multi-wavelets. As shown in Fig. 3.11 for the 320$\times$ 480 animal image and 256$\times$256 face image, our method achieves comparable or higher PSNR scores than the state-of-the-art methods. In addition, our method can render finer details more accurately.

We also conducted extensive experiments with varying noise levels on a set of different types of images in Fig. 3.8. Table 3.1, 3.2, and 3.3 present the denoising results (PSNR) when the Gaussian variance is 10, 30, and 50, respectively. The leftmost columns of these tables are the corresponding ID in Fig. 3.8. As visible, the multi-scale wavelets perform well on images with complex textures and when the noise level is high, and ODL is suitable for lower noise levels. In comparison, our algorithm is more consistent and stable.

3.3.3 Image Inpainting

Image inpainting is often used for the restoration of the damaged photographs and the removal of specific artifacts such as missing pixels. Previous dictionary learning based algorithms work only when the missing area is smaller than the corresponding patch size of the dictionary atom dimensionality.

We observed that our method generates the best image inpainting results. As demonstrated in Fig. 3.1 our method can restore the missing image regions that are remarkably much larger than the dimension of dictionary atoms, outperforming the state-of-the-art methods. By reconstructing the image starting at the coarsest layer, we can fix completely missing regions. The larger the missing area, the smoother the restored image becomes. In comparison, single-scale based methods fail completely.



(a) original image (b) corrupted image

PSNR 28.74 PSNR 34.88

(c) KSVD (d) Ours

Figure 3.12: A sample 480×320 image from the animal dataset is corrupted with large artifacts and missing blocks. The sizes of the artifacts range from 8 to 32 pixels. Our method efficiently removes the artifacts.

Ours: 40 dB KSVD: 26 dB Ours: 36 dB KSVD: 22 dB

Figure 3.13: (a-b) Inpainting results for 8×8 and 14×14 missing blocks. (c-d) Results for 16×16 to 32×32 missing blocks.

Table 3.4: Image In-painting Results

	8	14	20	26	32	38	44	50
KSVD	34.76	26.96	22.03	20.18	18.86	16.43	16.48	14.24
Ours	41.59	40.78	37.54	33.80	30.25	25.87	26.12	23.44

Given the mask $\mathbf{M}$ of missing pixels, our formulation in each layer is

$$\begin{aligned} \hat{\mathbf{x}}_n^{ij} &= \arg\min_{\mathbf{x}_n} \sum_{ij} \|\mathbf{R}_{ij}\mathbf{M} \otimes (\mathbf{R}_{ij}\mathbf{Y}'_n - \mathbf{D}_n\mathbf{x}_n)\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{x}_n^{ij}\|_0 \leq T_n \end{aligned} \tag{3.11}$$

where we denote $\otimes$ as the element-wise multiplication between two vectors.

Figure 3.12 shows that our algorithm can fill in the big holes where the KSVD fails. To analyze our algorithm further, we randomly remove 8 different sized squares (8, 14, 20, 26, 32, 38, 44, and 50) at 1 to 6 image locations each (8 to 48 holes at each try) in the given image in Fig. 3.13. When the missing area is small, e.g. 8×8 and 14×14, our algorithm can recover with a high PSNR of 40 dB, which is approximately 14 dB higher than the KSVD. When the missing area size is between 16×16 to 32×32, our method can still recover with 36 dB PSNR but KSVD degrades to around 22 dB. With the missing areas growing, our algorithm still outperforms the KSVD almost 10 dB. Here, we compare with the KSVD algorithm since the multi-scale KSVD simply increases the dimension of atoms, which leads proportionally more atoms to form an overcomplete dictionary. At the same time, multi-scale KSVD still fails to handle holes larger than the dimensionality of the atoms.

3.4 Summary

We presented a non-linear dictionary learning and sparse coding method on cascaded residuals. Our cascade allows capturing both local and global information. Its coarse-to-fine structure prevent from reconstructing the regions that can be well represented by the coarser layers. Our sparse coding can be used to progressively improve the quality of the decoded image.

Our method provides significant improvement over the state-of-the-art solutions in terms of the quality of reconstructed image, reduction in the number of coefficients, and computational complexity. It generates much higher quality images using less number of coefficients. It produces superior results on image inpainting, in particular, in handling of very large ratios of missing pixels and large gaps.

Chapter 4

Deep Subspace Clustering Networks

In Chapter 3, we have seen that extending the single layer dictionary learning to nonlinear multi-layer dictionary learning is able to enhance the performance on low-level image processing task. In this chapter, we also extend this concept to subspace clustering, which share similar concept and formulation with dictionary learning.

Most recent works on subspace clustering Yan and Pollefeys [2006]; Chen and Lerman [2009]; Elhamifar and Vidal [2013b]; Liu et al. [2013]; Wang et al. [2013]; Lu et al. [2012]; Ji et al. [2015]; You et al. [2016a] focus on clustering linear subspaces. However, in practice, the data do not necessarily conform to linear subspace models. For instance, in the example of face image clustering, reflectance is typically non-Lambertian and the pose of the subject often varies. Under these conditions, the face images of one subject rather lie in a non-linear subspace (or sub-manifold). A few works Chen et al. [2009]; Patel et al. [2013]; Patel and Vidal [2014]; Yin et al. [2016]; Xiao et al. [2016] have proposed to exploit the kernel trick Shawe-Taylor and Cristianini [2004] to address the case of non-linear subspaces. However, the selection of different kernel types is largely empirical, and there is no clear reason to believe that the implicit feature space corresponding to a predefined kernel is truly well-suited to subspace clustering.

In this chapter, we introduce a novel deep neural network architecture to learn (in an unsupervised manner) an explicit non-linear mapping of the data that is well-adapted to subspace clustering. To this end, we build our deep subspace clustering networks (*DSC-Nets*) upon deep auto-encoders, which non-linearly map the data points to a latent space through a series of encoder layers. Our key contribution then consists of introducing a novel *self-expressive* layer – a fully connected layer without bias and non-linear activations – at the junction between the encoder and the decoder. This layer encodes the "self-expressiveness" property Rao et al. [2008]; Elhamifar and Vidal [2009] of data drawn from a union of subspaces, that is, the fact that each data sample can be represented as a linear combination of other samples in the same subspace. To the best of our knowledge, our approach constitutes the first attempt to directly learn the affinities (through combination coefficients) between all data points within one neural network. Furthermore, we propose effective pre-training and fine-tuning strategies to learn the parameters of our DSC-Nets in an unsupervised manner and with a limited amount of data.

We extensively evaluate our method on face clustering, using the Extended Yale B Lee et al. [2005] and ORL Samaria and Harter [1994] datasets, and on general object clustering, using COIL20 Nene et al. [1996b] and COIL100 Nene et al. [1996a]. Our experiments show

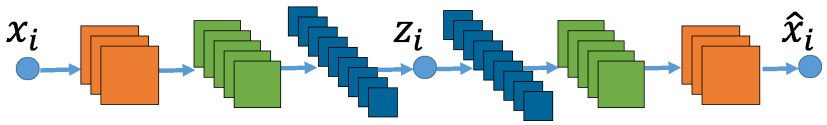


Figure 4.1: Deep Convolutional Auto-Encoder: The input $\mathbf{x}_i$ is mapped to $\mathbf{z}_i$ through an encoder, and then reconstructed as $\hat{\mathbf{x}}_i$ through a decoder. We use shaded circles to denote data vectors and shaded squares to denote the channels after convolution or deconvolution. We do not enforce the weights of the corresponding encoder and decoder layers to be coupled (or the same).

that our DSC-Nets significantly outperform the state-of-the-art subspace clustering methods.

4.1 Deep Subspace Clustering Networks (DSC-Nets)

Our deep subspace clustering networks leverage deep auto-encoders and the self-expressiveness property. Before introducing our networks, we first discuss this property in more detail.

4.1.1 Self-Expressiveness

Given data points $\{\mathbf{x}_i\}_{i=1,\cdots,N}$ drawn from multiple linear subspaces $\{\mathcal{S}_i\}_{i=1,\cdots,K}$, one can express a point in a subspace as a linear combination of other points in the same subspace. In the literature Rao et al. [2008]; Elhamifar and Vidal [2009], this property is called self-expressiveness. If we stack all the points $\mathbf{x}_i$ into columns of a data matrix $\mathbf{X}$, the self-expressiveness property can be simply represented as one single equation, i.e., $\mathbf{X} = \mathbf{X}\mathbf{C}$, where $\mathbf{C}$ is the self-representation coefficient matrix. It has been shown in Ji et al. [2014b] that, under the assumption that the subspaces are independent, by minimizing certain norms of $\mathbf{C}$, $\mathbf{C}$ is guaranteed to have a block-diagonal structure (up to certain permutations), i.e., $c_{ij} \neq 0$ iff point $\mathbf{x}_i$ and point $\mathbf{x}_j$ lie in the same subspace. So we can leverage the matrix $\mathbf{C}$ to construct the affinity matrix for spectral clustering. Mathematically, this idea is formalized as the optimization problem

$$\min_{\mathbf{C}} \|\mathbf{C}\|_p \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C}, \ (\text{diag}(\mathbf{C}) = \mathbf{0}) \ , \tag{4.1}$$

where $\|\cdot\|_p$ represents an arbitrary matrix norm, and the optional diagonal constraint on $\mathbf{C}$ prevents trivial solutions for sparsity inducing norms, such as the ℓ_1 norm. Various norms for $\mathbf{C}$ have been proposed in the literature, e.g., the ℓ_1 norm in Sparse Subspace Clustering (SSC) Elhamifar and Vidal [2009, 2013b], the nuclear norm in Low Rank Representation (LRR) Liu et al. [2010, 2013] and Low Rank Subspace Clustering (LRSC) Favaro et al. [2011]; Vidal and Favaro [2014], and the Frobenius norm in Least-Squares Regression (LSR) Lu et al. [2012] and Efficient Dense Subspace Clustering (EDSC) Ji et al. [2014b]. To account for data corruptions, the equality constraint in (6.2) is often relaxed as a regularization term, leading to

$$\min_{\mathbf{C}} \|\mathbf{C}\|_p + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2 \quad \text{s.t.} \quad (\text{diag}(\mathbf{C}) = \mathbf{0}) \ . \tag{4.2}$$

Unfortunately, the self-expressiveness property only holds for linear subspaces. While kernel based methods Patel et al. [2013]; Patel and Vidal [2014]; Yin et al. [2016]; Xiao et al.

[2016] aim to tackle the non-linear case, it is not clear that pre-defined kernels yield implicit feature spaces that are well-suited for subspace clustering. In this work, we aim to learn an explicit mapping that makes the subspaces more separable. To this end, and as discussed below, we propose to build our networks upon deep auto-encoders.

4.1.2 Self-Expressive Layer in Deep Auto-Encoders

Our goal is to train a deep auto-encoder, such as the one depicted by Figure 4.1, such that its latent representation is well-suited to subspace clustering. To this end, we introduce a new layer that encodes the notion of self-expressiveness.

Specifically, let Θ denote the auto-encoder parameters, which can be decomposed into encoder parameters Θ_e and decoder parameters Θ_d. Furthermore, let $\mathbf{Z}_{\Theta_e}$ denote the output of the encoder, i.e., the latent representation of the data matrix $\mathbf{X}$. To encode self-expressiveness, we introduce a new loss function defined as

$$L(\Theta, \mathbf{C}) = \frac{1}{2}\|\mathbf{X} - \hat{\mathbf{X}}_\Theta\|_F^2 + \lambda_1\|\mathbf{C}\|_p + \frac{\lambda_2}{2}\|\mathbf{Z}_{\Theta_e} - \mathbf{Z}_{\Theta_e}\mathbf{C}\|_F^2 \quad \text{s.t.} \quad (\text{diag}(\mathbf{C}) = \mathbf{0}) \ , \tag{4.3}$$

where $\hat{\mathbf{X}}_\Theta$ represents the data reconstructed by the auto-encoder. To minimize (4.3), we propose to leverage the fact that, as discussed below, $\mathbf{C}$ can be thought of as the parameters of an additional network layer, which lets us solve for Θ and $\mathbf{C}$ jointly using backpropagation.Note that one could also alternate minimization between Θ and $\mathbf{C}$. However, since the loss is non-convex, this would not provide better convergence guarantees and would require investigating the influence of the number of steps in the optimization w.r.t. Θ on the clustering results.

Specifically, consider the self-expressiveness term in (4.3), $\|\mathbf{Z}_{\Theta_e} - \mathbf{Z}_{\Theta_e}\mathbf{C}\|_F^2$. Since each data point $\mathbf{z}_i$ (in the latent space) is approximated by a weighted linear combination of other points $\{\mathbf{z}_j\}_{j=1,\cdots,N}$ (optionally, $j \neq i$) with weights c_{ij}, this linear operation corresponds exactly to a set of linear neurons without non-linear activations. Therefore, if we take each $\mathbf{z}_i$ as a node in the network, we can then represent the self-expressiveness term with a fully-connected linear layer, which we call the *self-expressive layer*. The weights of the self-expressive layer correspond to the matrix $\mathbf{C}$ in (4.3), which are further used to construct affinities between all data points. Therefore, our self-expressive layer essentially lets us directly learn the affinity matrix via the network. Moreover, minimizing $\|\mathbf{C}\|_p$ simply translates to adding a regularizer to the weights of the self-expressive layer. In this work, we consider two kinds of regularizations on $\mathbf{C}$: (i) the ℓ_1 norm, resulting in a network denoted by DSC-Net-L1; (ii) the ℓ_2 norm, resulting in a network denoted by DSC-Net-L2.

For notational consistency, let us denote the parameters of the self-expressive layer (which are just the elements of $\mathbf{C}$) as Θ_s. As can be seen from Figure 4.2, we then take the input to the decoder part of our network to be the transformed latent representation $\mathbf{Z}_{\Theta_e}\Theta_s$. This lets us re-write our loss function as

$$\tilde{L}(\tilde{\Theta}) = \frac{1}{2}\|\mathbf{X} - \hat{\mathbf{X}}_{\tilde{\Theta}}\|_F^2 + \lambda_1\|\Theta_s\|_p + \frac{\lambda_2}{2}\|\mathbf{Z}_{\Theta_e} - \mathbf{Z}_{\Theta_e}\Theta_s\|_F^2 \quad \text{s.t.} \quad (\text{diag}(\Theta_s) = \mathbf{0}) \ , \tag{4.4}$$

where the network parameters $\tilde{\Theta}$ now consist of encoder parameters Θ_e, self-expressive layer parameters Θ_s, and decoder parameters Θ_d, and where the reconstructed data $\hat{\mathbf{X}}$ is now a

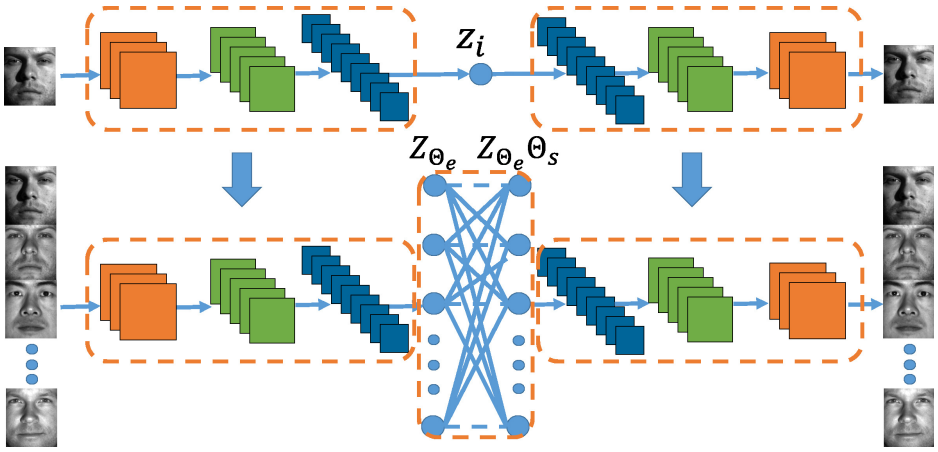


Figure 4.2: Deep Subspace Clustering Networks: As an example, we show a deep subspace clustering network with three convolutional encoder layers, one self-expressive layer, and three deconvolutional decoder layer. During training, we first pre-train the deep auto-encoder without the self-expressive layer; we then fine-tune our entire network using this pre-trained model for initialization.

function of $\{\Theta_e, \Theta_s, \Theta_d\}$ rather than just $\{\Theta_e, \Theta_d\}$ in (4.3).

4.1.3 Network Architecture

Our network consists of three parts, i.e., stacked encoders, a self-expressive layer, and stacked decoders. The overall network architecture is shown in Figure 4.2. In this thesis, since we focus on image clustering problems, we advocate the use of convolutional auto-encoders that have fewer parameters than the fully connected ones and are thus easier to train. Note, however, that fully-connected auto-encoders are also compatible with our self-expressive layer. In the convolutional layers, we use kernels with stride 2 in both horizontal and vertical directions, and rectified linear unit (ReLU) Krizhevsky et al. [2012] for the non-linear activations. Given N images to be clustered, we use all the images in a single batch. Each input image is mapped by the convolutional encoder layers to a latent vector (or node) $\mathbf{z}_i$, represented as a shaded circle in Figure 4.2. In the self-expressive layer, the nodes are fully connected using linear weights without bias and non-linear activations. The latent vectors are then mapped back to the original image space via the deconvolutional decoder layers.

For the i^{th} encoder layer with n_i channels of kernel size $k_i \times k_i$, the number of weight parameters is $k_i^2 n_{i-1} n_i$, with $n_0 = 1$. Since the encoders and decoders have symmetric structures, their total number of parameters is $\sum_i 2k_i^2 n_{i-1} n_i$ plus the number of bias parameters $\sum_i 2n_i - n_1 + 1$. For N input images, the number of parameters for the self-expressive layer is N^2. For example, if we have three encoder layers with 10, 20, and 30 channels, respectively, and all convolutional kernels are of size 3×3, then the number of parameters for encoders and decoders is $\sum_{i=1}^{3} 2(k_i^2 n_{i-1} + 1)n_i - n_1 + 1 = 14671$. If we have 1000 input images, then the number of parameters in the self-expressive layer is 10^6. Therefore, the network parameters are typically dominated by those of the self-expressive layer.

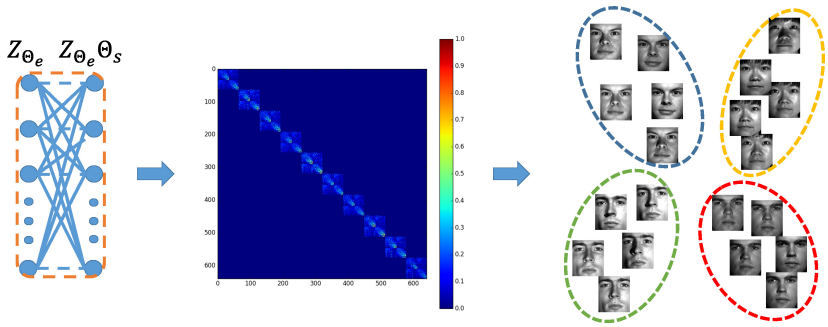


Figure 4.3: From the parameters of the self-expressive layer, we construct an affinity matrix, which we use to perform spectral clustering to get the final clusters. Best viewed in color.

4.1.4 Training Strategy

Since the size of datasets for unsupervised subspace clustering is usually limited (e.g., in the order of thousands of images), our networks remain of a tractable size. However, for the same reason, it also remains difficult to directly train a network with millions of parameters from scratch. To address this, we design the pre-training and fine-tuning strategies described below. Note that this also allows us to avoid the trivial all-zero solution while minimizing the loss (4.4).

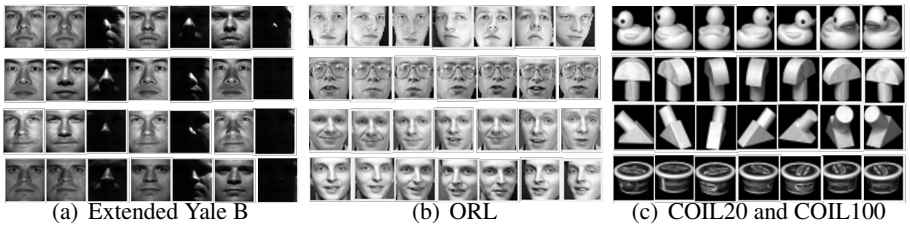
As illustrated in Figure 4.2, we first pre-train the deep auto-encoder without the self-expressive layer on all the data we have. We then use the trained parameters to initialize the encoder and decoder layers of our network. After this, in the fine-tuning stage, we build a big batch using all the data to minimize the loss $\tilde{L}(\Theta)$ defined in (4.4) with a gradient descent method. Specifically, we use Adam Kingma and Ba [2014], an adaptive momentum based gradient descent method, to minimize the loss, where we set the learning rate to 1.0×10^{-3} in all our experiments. Since we always use the same batch in each training epoch, our optimization strategy is rather a deterministic momentum based gradient method than a stochastic gradient method.

Note also that, since we only have access to images for training and not to cluster labels, our training strategy is unsupervised (or self-supervised).

Once the network is trained, we can use the parameters of the self-expressive layer to construct an affinity matrix for spectral clustering Ng et al. [2001], as illustrated in Figure 4.3. Although such an affinity matrix could in principle be computed as $|\mathbf{C}| + |\mathbf{C}^T|$, over the years, researchers in the field have developed many heuristics to improve the resulting matrix.

Since there is no globally-accepted solution for this step in the literature, we make use of the heuristics employed by SSC Elhamifar and Vidal [2013b] and EDSC Ji et al. [2014b]. Due to the lack of space, we refer the reader to the publicly available implementation of SSC and Section 5 of Ji et al. [2014b], as well as to the TensorFlow implementation of our algorithm [1] for more detail.

[1] https://github.com/panji1990/Deep-subspace-clustering-networks



(a) Extended Yale B (b) ORL (c) COIL20 and COIL100

Figure 4.4: Sample images from Extended Yale B, ORL , COIL20 and COIL100.

4.2 Experiments

We implemented our method in Python with Tensorflow-1.0 Abadi et al. [2016], and evaluated it on four standard datasets, i.e., the Extended Yale B and ORL face image datasets, and the COIL20/100 object image datasets. We compare our methods against the following baselines: Low Rank Representation (LRR) Liu et al. [2013], Low Rank Subspace Clustering (LRSC) Vidal and Favaro [2014], Sparse Subspace Clustering (SSC) Elhamifar and Vidal [2013b], Kernel Sparse Subspace Clustering (KSSC) Patel and Vidal [2014], SSC by Orthogonal Matching Pursuit (SSC-OMP) You et al. [2016b], Efficient Dense Subspace Clustering (EDSC) Ji et al. [2014b], SSC with the pre-trained convolutional auto-encoder features (AE+SSC), and EDSC with the pre-trained convolutional auto-encoder features (AE+EDSC). For all the baselines, we used the source codes released by the authors and tuned the parameters by grid search to the achieve best results on each dataset. Since the code for the deep subspace clustering method of Peng et al. [2016] is not publicly available, we are only able to provide a comparison against this approach on Extended Yale B and COIL20, for which the results are provided in Peng et al. [2016]. Note that this comparison already clearly shows the benefits of our approach.

For all quantitative evaluations, we make use of the clustering error rate, defined as

$$\text{err } \% = \frac{\text{\# of wrongly clustered points}}{\text{total \# of points}} \times 100\% \,. \tag{4.5}$$

4.2.1 Extended Yale B Dataset

The Extended Yale B dataset Lee et al. [2005] is a popular benchmark for subspace clustering. It consists of 38 subjects, each of which is represented with 64 face images acquired under different illumination conditions (see Figure 4.4(a) for sample images from this dataset). Following the experimental setup of Elhamifar and Vidal [2013b], we down-sampled the original face images from 192×168 to 42×42 pixels, which makes it computationally feasible for the baselines Elhamifar and Vidal [2013b]; Liu et al. [2013]. In each experiment, we pick $K \in \{10, 15, 20, 25, 30, 35, 38\}$ subjects (each subject with 64 face images) to test the robustness w.r.t. an increasing number of clusters. Taking all possible combinations of K subjects out of 38 would result in too many experimental trials. To get a manageable size of experiments, we first number the subjects from 1 to 38 and then take all possible K consecutive subjects. For example, in the case of 10 subjects, we take all the images from subject $1-10$, $2-11$, $\cdots$, $29-38$, giving rise to 29 experimental trials. We experimented with different architectures

layers	encoder-1	encoder-2	encoder-3	self-expressive	decoder-1	decoder-2	decoder-3
kernel size	5×5	3×3	3×3	–	3×3	3×3	5×5
channels	10	20	30	–	30	20	10
parameters	260	1820	5430	5914624	5420	1810	251

Table 4.1: Network settings for Extended Yale B.

for the convolutional layers of our network, e.g., different network depths and number of channels. While increasing these values increases the representation power of the network, it also increases the number of network parameters, thus requiring larger training datasets. Since the size of Extended Yale B is quite limited, with only 2432 images, we found having three-layer encoders and decoders with $[10, 20, 30]$ channels to be a good trade-off for this dataset. The detailed network settings are described in Table 4.1. In the fine-tuning phase, since the number of epochs required for gradient descent increases as the number of subjects K increases, we defined the number of epochs for DSC-Net-L1 as $160 + 20K$ and for DSC-Net-L2 as $50 + 25K$. We set the regularization parameters to $\lambda_1 = 1.0, \lambda_2 = 1.0 \times 10^{\frac{K}{10}-3}$.

The clustering performance of different methods for different numbers of subjects is provided in Table 4.2. For the experiments with K subjects, we report the mean and median errors of $39 - K$ experimental trials. From these results, we can see that the performance of most of the baselines decreases dramatically as the number of subjects K increases. By contrast, the performance of our deep subspace clustering methods, DSC-Net-L1 and DSC-Net-L2, remains relatively stable w.r.t. the number of clusters. Specifically, our DSC-Net-L2 achieves 2.67% error rate for 38 subjects, which is only around $1/5$ of the best performing baseline EDSC. We also observe that using the pre-trained auto-encoder features does not necessarily improve the performance of SSC and EDSC, which confirms the benefits of our joint optimization of all parameters in one network. The results of Peng et al. [2016] on this dataset for 38 subjects was reported to be $92.08 \pm 2.42\%$ in terms of clustering accuracy, or equivalently $7.92 \pm 2.42\%$ in terms of clustering error, which is worse than both our methods – DSC-Net-L1 and DSC-Net-L2. We further notice that DSC-Net-L1 performs slightly worse than DSC-Net-L2 in the current experimental settings. We conjecture that this is due to the difficulty in optimization introduced by the ℓ_1 norm as it is non-differentiable at zero.

4.2.2 ORL Dataset

The ORL dataset Samaria and Harter [1994] is composed of 400 human face images, with 40 subjects each having 10 samples. Following Cai et al. [2007], we down-sampled the original face images from 112×92 to 32×32. For each subject, the images were taken under varying lighting conditions with different facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses)(see Figure 4.4(b) for sample images). Compared to Extended Yale B, this dataset is more challenging for subspace clustering because (i) the face subspaces have more non-linearity due to varying facial expressions and details; (ii) the dataset size is much smaller (400 vs. 2432). To design a trainable deep auto-encoder on 400 images, we reduced the number of network parameters by decreasing the number of channels in each encoder and decoder layer. The resulting network is specified in Table 4.3. Since we already verified the robustness of our method to the number of clusters in the previous experiment, here, we only provide results for clustering all 40 subjects. In this setting, we set $\lambda_1 = 1$ and

Method	LRR	LRSC	SSC	AE+ SSC	KSSC	SSC-OMP	EDSC	AE+ EDSC	DSC-Net-L1	DSC-Net-L2
10 subjects										
Mean	22.22	30.95	10.22	17.06	14.49	12.08	5.64	5.46	2.23	**1.59**
Median	23.49	29.38	11.09	17.75	15.78	8.28	5.47	6.09	2.03	**1.25**
15 subjects										
Mean	23.22	31.47	13.13	18.65	16.22	14.05	7.63	6.70	2.17	**1.69**
Median	23.49	31.64	13.40	17.76	17.34	14.69	6.41	5.52	2.03	**1.72**
20 subjects										
Mean	30.23	28.76	19.75	18.23	16.55	15.16	9.30	7.67	2.17	**1.73**
Median	29.30	28.91	21.17	16.80	17.34	15.23	10.31	6.56	2.11	**1.80**
25 subjects										
Mean	27.92	27.81	26.22	18.72	18.56	18.89	10.67	10.27	2.53	**1.75**
Median	28.13	26.81	26.66	17.88	18.03	18.53	10.84	10.22	2.19	**1.81**
30 subjects										
Mean	37.98	30.64	28.76	19.99	20.49	20.75	11.24	11.56	2.63	**2.07**
Median	36.82	30.31	28.59	20.00	20.94	20.52	11.09	10.36	2.81	**2.19**
35 subjects										
Mean	41.85	31.35	28.55	22.13	26.07	20.29	13.10	13.28	3.09	**2.65**
Median	41.81	31.74	29.04	21.74	25.92	20.18	13.10	13.21	3.10	**2.64**
38 subjects										
Mean	34.87	29.89	27.51	25.33	27.75	24.71	11.64	12.66	3.33	**2.67**
Median	34.87	29.89	27.51	25.33	27.75	24.71	11.64	12.66	3.33	**2.67**

Table 4.2: Clustering error (in %) on Extended Yale B. The lower the better.

layers	encoder-1	encoder-2	encoder-3	self-expressive	decoder-1	decoder-2	decoder-3
kernel size	5×5	3×3	3×3	–	3×3	3×3	5×5
channels	5	3	3	–	3	3	5
parameters	130	138	84	160000	84	140	126

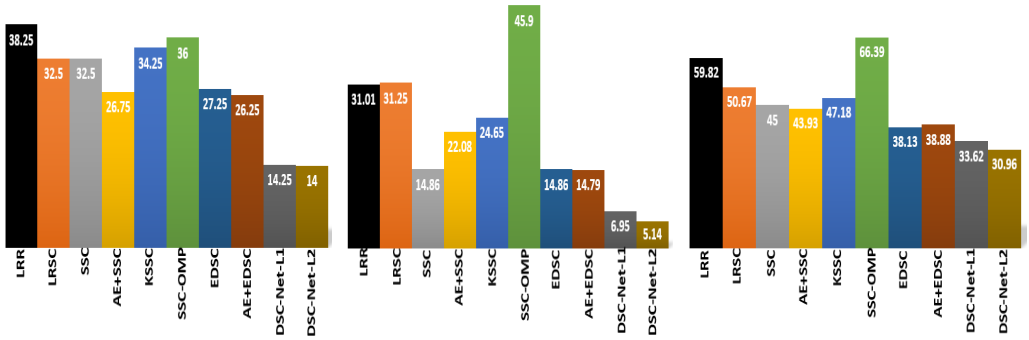
Table 4.3: Network settings for ORL.

$\lambda_2 = 0.2$ and run 700 epochs for DSC-Net-L2 and 1500 epochs for DSC-Net-L1 during fine-tuning. Note that, since the size of this dataset is small, we can even use the whole data as a single batch in pre-training, because we found this to be numerically more stable and converge faster than stochastic gradient descent using randomly sampled mini-batches.

Figure 4.5(a) shows the error rates of the different methods, where different colors denote different subspace clustering algorithms and the length of the bars reflects the error rate. Since there are much fewer samples per subject, all competing methods perform worse than on Extended Yale B. Note that both EDSC and SSC achieve moderate clustering improvement by using the features of pre-trained convolutional auto-encoders, but their error rates are still around twice as high as those of our methods.

4.2.3 COIL20 and COIL100 Datasets

The previous experiments both target face clustering. To show the generality of our algorithm, we also evaluate it on the COIL object image datasets – COIL20 Nene et al. [1996b] and COIL100 Nene et al. [1996a]. COIL20 consists of 1440 gray-scale image samples, distributed over 20 objects such as duck and car model (see sample images in Figure 4.4(c)). Similarly, COIL100 consists of 7200 images distributed over 100 objects. Each object was placed on a turntable against a black background, and 72 images were taken at pose intervals of 5 degrees. Following Cai et al. [2011], we down-sampled the images to 32×32. In contrast with the previous human face datasets, in which faces are well aligned and have similar structures, the



(a) ORL (b) COIL20 (c) COIL100

Figure 4.5: Subspace clustering error (in %) on the ORL, COIL20 and COIL100 datasets. Different colors indicate different methods. The height of the bars encodes the error, so the lower the better.

object images from COIL20 and COIL100 are more diverse, and even samples from the same object differ from each other due to the change of viewing angle. This makes these datasets challenging for subspace clustering techniques. For these datasets, we used shallower networks with one encoder layer, one self-expressive layer, and one decoder layer. For COIL20, we set the number of channels to 15 and the kernel size to 3×3. For COIL100, we increased the number of channels to 50 and the kernel size to 5×5. The settings for both networks are provided in Table 4.4. Note that with these network architectures, the dimension of the latent space representation $\mathbf{z}_i$ increases by a factor of 15/4 for COIL20 (as the spatial resolution of each channel shrinks to 1/4 of the input image after convolutions with stride 2, and we have 15 channels) and 50/4 for COIL100. Thus our networks perform dimensionality lifting rather than dimensionality reduction. This, in some sense, is similar to the idea of Hilbert space mapping in kernel methods Shawe-Taylor and Cristianini [2004], but with the difference that, in our case, the mapping is explicit, via the neural network. In our experiments, we found that these shallow, dimension-lifting networks performed better than deep, bottle-neck ones on these datasets. While it is also possible to design deep, dimension-lifting networks, the number of channels has to increase by a factor of 4 after each layer to compensate for the resolution loss. For example, if we want the latent space dimension to increase by a factor of 15/4, we need $15 \cdot 4$ channels in the second layer for a 2-layer encoder, $15 \cdot 4^2$ channels in the third layer for a 3-layer encoder, and so forth. In the presence of limited data, this increasing number of parameters makes training less reliable. Figure 4.5(b) and (c) depict the error rates of the different methods on clustering 20 classes for COIL20 and 100 classes for COIL100, respectively. Note that, in both cases, our DSC-Net-L2 achieves the lowest error rate. In particular, for COIL20, we obtain an error of 5.14%, which is roughly 1/3 of the error rate of the best-performing baseline AE+EDSC. The results of Peng et al. [2016] on COIL20 were reported to be $14.24 \pm 4.70\%$ in terms of clustering error, which is also much higher than ours.

	COIL20			COIL100		
layers	encoder-1	self-expressive	decoder-1	encoder-1	self-expressive	decoder-1
kernel size	3×3	–	3×3	5×5	–	5×5
channels	15	–	15	50	–	50
parameters	150	2073600	136	1300	51840000	1251

Table 4.4: Network settings for COIL20 and COIL100.

4.3 Summary

We have introduced a deep auto-encoder framework for subspace clustering by developing a novel self-expressive layer to harness the "self-expressiveness" property of a union of subspaces. Our deep subspace clustering network allows us to directly learn the affinities between all data points with a single neural network. Furthermore, we have proposed pre-training and fine-tuning strategies to train our network, demonstrating the ability to handle challenging scenarios with small-size datasets, such as the ORL dataset. Our experiments have demonstrated that our deep subspace clustering methods provide significant improvement over the state-of-the-art subspace clustering solutions in terms of clustering accuracy on several standard datasets.

Chapter 5

Scalable Deep k-Subspace Clustering

In Chapter 4, it has proven that along with self-expressiveness CNN is able to provide more subspace friendly feature representation. However, due to the workflow of DSC-nets remains the same as traditional subspace clustering methods (applying spectral clustering on affinity matrix) the scalability is a big issue for subspace clustering.

With the current trend in analyzing big data, SC algorithms should be able to deal with a large volume of data. However, most of the state-of-the-art methods for SC make use of an affinity matrix along norm regularization (e.g., , ℓ_1 Elhamifar and Vidal [2009, 2013b], ℓ_2 Ji et al. [2014b] or nuclear Liu et al. [2013]; Vidal and Favaro [2014]). Not only building an affinity matrix demands for solving large scale optimization problems but also performing spectral clustering on an affinity matrix, whose size is dictated by the number of samples, is overwhelming.

In this chapter, instead of constructing the affinity matrix for spectral clustering, we revisit the *k*-subspace clustering (*k*-SC) method Bradley and Mangasarian [2000]; Tseng [2000]; Agarwal and Mustafa [2004] to design a novel and scalable method. In order to handle non-linear subspaces, we propose to utilize deep neural networks to project data to a latent space where *k*-SC can be easily applied. We mainly have three highlights in this chapter:

- We bypass the steps of constructing an affinity matrix and performing spectral clustering, which have been used in mainstream subspace clustering algorithms, and accelerate the computation by using a variant of *k*-subspace clustering. As a result, our method can handle datasets that are orders of magnitudes larger than those considered in traditional methods.

- In order to address non-linearity, we equip deep neural networks with subspace priors. This in return enables us to learn an explicit non-linear mapping of the data that is well-suited for subspace clustering.

- We propose novel strategies to update subspace bases. When the size of the dataset at hand is manageable, we update subspaces in closed-form using Singular Value Decomposition (SVD) with a simple mechanism to rule out outliers. For large datasets, we update subspaces by making use of the stochastic optimization methods on the Grassmann manifolds.

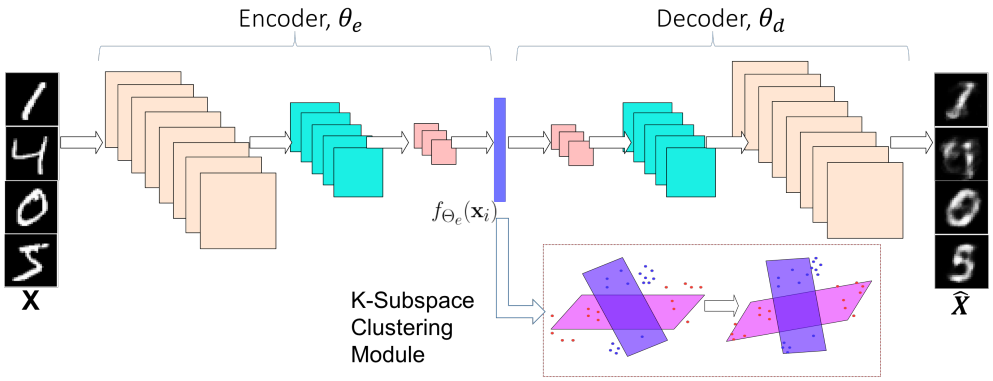


Figure 5.1: Scalable deep k-subspace structure. As an example, we show our scalable batch-based subspace clustering with three convolutional encoder layers and three deconvolutional decoder layers. During the training, we first pre-train the deep convolutional auto-encoder by simply reconstructing the corresponding images, and then fine-tune the network using this pre-trained model as initialization. During the fine-tuning, the network minimizes the sum of distances of each sample in the latent space to its closet subspace.

5.1 k-Subspace Clustering(k-SC) Networks

Our k-subspace clustering networks leverage on the properties of deep convolutional auto-encoder and the k-subspaces clustering. In this section we will discuss the k-subspace property and the whole framework in detail.

5.1.1 k-Subspace Clustering

Consider a collection of points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \in \mathbb{R}^d$ belonging to a union of k subspaces $\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_k$ of dimensions $p_1, p_2, \cdots, p_k$, respectively [1]. With slight abuse of notation, we will use $\mathbf{S}_i$ to represent the basis of the subspace index by i, that is $\mathbf{S}_i \in \mathbb{R}^{d\times p_i}$ and $\mathbf{S}_i^\top \mathbf{S}_i = \mathbf{I}_{p_i}$ with $\mathbf{I}_p$ denoting $p \times p$ identity matrix. The goal of subspace clustering is to learn the subspaces and assign points to their nearest subspaces.

Once every data point is assigned to a subspace, the corresponding subspace basis can be re-calculated by SVD (will be shown shortly). Different from self-expressiveness-based methods which obtain the affinity matrix by solving large-scale optimization problems, k-SC seeks to minimize the sum of residuals of points to their nearest subspaces. The cost function of k-SC can be written as

$$\begin{aligned} &\min_{\{\mathbf{S}_i\},\{w_{ij}\}} \sum_{j}^{n}\sum_{i}^{k} w_{ij}\|\mathbf{x}_j - \mathbf{S}_i\mathbf{S}_i^\top \mathbf{x}_j\|_2^2, \\ &\text{s.t.} \quad w_{ij} \in \{0,1\} \quad \text{and} \quad \sum_{i=1}^{k} w_{ij} = 1\ . \end{aligned} \tag{5.1}$$

[1] We assume $p = p_i, \forall i$ in the remainder.

Given the subspace basis $\{\mathbf{S}_1, \cdots, \mathbf{S}_n\}$, the optimal value for w_{ij} can be written as

$$w_{ij} = \begin{cases} 1 & \text{if} \quad i = \arg\min_m \|\mathbf{x}_j - \mathbf{S}_m \mathbf{S}_m^\top \mathbf{x}_j\|_2^2 \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{5.2}$$

For the sake of discussion, let us arrange w_{ij} into a membership matrix $\mathbf{W} \in \mathbb{R}^{k \times n}$. Beginning with an initialization of k candidate subspaces bases, k-SC updates the membership assignments w_{ij} and subspaces in an alternating fashion: 1) cluster points by assigning the nearest subspace as in Eqn. (5.2); 2) re-estimate the new subspace bases by performing SVD on the points of each cluster (the columns of $\mathbf{W}$ where the i-th row is 1). Similar to k-means, the whole algorithm works in an Expectation Maximization (EM) style, and is guaranteed to converge to a local minimum in a finite number of iterations. We will shortly show how stochastic optimization techniques can be applied to minimize the problem depicted in (5.1), equipping our solution with the ability to handle large-scale data.

subsection$k$-SC with Convolutional Auto-Encoder Network

Denoising fully-connected Auto-Encoders (AEs) are widely used with generic clustering algorithms Yang et al. [2017]; Xie et al. [2016]. We have found such structures difficult to train (due to the large number of parameters in the fully-connected layers) and propose to use convolutional AEs to learn the embeddings for k-SC.

Specifically, let θ denote the AE parameters, which can be decomposed into encoder parameters θ_e and decoder parameters θ_d. Let $f_{\theta_e}(\cdot)$ be the encoder mapping function and $g_{\theta_d}(\cdot)$ as the decoder mapping function, both of which are composed of a sequence of convolution kernels and nonlinear activation functions. Our overall loss can be written as

$$\ell(\theta, \{\mathbf{S}_i\}, \mathbf{W}) = \ell_{\text{ae}}(\theta) + \lambda \ell_{\text{ksc}}(\{\mathbf{S}_i\}, \mathbf{W}) \,, \tag{5.3}$$

where λ is a regularization parameter to balance the reconstruction loss and the k-subspace clustering loss. The auto-encoder reconstruction loss ℓ_{ae} is defined as

$$\ell_{\text{ae}}(\theta) = \sum_j \|\mathbf{x}_j - g_{\theta_d}(f_{\theta_e}(\mathbf{x}_j))\|_2^2. \tag{5.4}$$

The $\ell_{\text{ksc}}(\theta)$ is the loss for subspace clustering and is written as

$$\begin{aligned} &\ell_{\text{ksc}}(\{\mathbf{S}_i\}, \theta) = \sum_{i,j} w_{ij} \|f_{\theta_e}(\mathbf{x}_j) - \mathbf{S_i}\mathbf{S_i}^\top f_{\theta_e}(\mathbf{x}_j)\|_2^2 \\ &\text{s.t.} \quad \mathbf{S}_i \in \mathcal{G}(d, p),\ w_{ij} \in \{0, 1\} \,,\ \sum_{i=1}^{k} w_{ij} = 1,\ \forall ij \,, \end{aligned} \tag{5.5}$$

where $\mathcal{G}(d, p)$ denotes the Grassmann manifold consisting of p-dimensional subspaces with ambient dimension d.

As a pre-processing step, some of traditional algorithms such as You et al. [2016b]; Zhang et al. [2012] use PCA to project images onto a low-dimensional space. However, the mapping by PCA projection is linear and fixed. By contrast, our encoder function f_{θ_e} can update its

parameters to adapt to a space which is subspace-clustering-friendly.

5.2 Optimization

Algorithm 5 Scalable k-Subspace Clustering (SVD update)

Input: dimensionality of subspaces p, number of class K, epochs number T, batch size b and dataset $\mathbf{x}_j$, $j = 1, \cdots, N$
Pre-train CAE using $\mathbf{x}_j$, $j = 1, \cdots, N$
Generate $\{\mathbf{S}_i\}$ based on the pre-trained model and initial cluster labels
for $m = 1 : T$ **do**
  for $n = 1 : b : N$
    Update the CAE parameters θ by Eqn. (5.6)
  end
  Recalculate the latent space for the whole data set,
  $\mathbf{z}_j = f_{\theta_e}(\mathbf{x}_j), j = 1...N$
  Assign the membership for every $\mathbf{z}_i$ as Eqn.(5.2) and
  rule out the farthest 10% points as outliers, for each
  class we have set $\mathbf{Z}_i$
  Update each subspace $\mathbf{S}_i$ through SVD decomposition on the $\mathbf{Z}_i$
end
Output: Subspaces $\{\mathbf{S}_i\}$, and membership assignment w_{ij}

The cost function (7.2) is highly non-convex and three sets of variables (i.e., , $\mathbf{W}, \theta$, and $\{\mathbf{S}_i\}$) should be updated alternatively. It is known that alternating optimization problems are not without difficulties. A strategy such as wake-and-sleep is a common practice to update one set of variables while fixing the others. As mentioned before, we first pre-train a CAE without having any information about $\mathbf{W}$ and $\mathbf{S}_i$. Therefore, it is natural to obtain an initial state for $\mathbf{W}$ and $\{\mathbf{S}_i\}$ directly from the output of the pre-trained CAE. This is exactly how we initialize $\mathbf{W}$ and $\{\mathbf{S}_i\}$.

As shown in Fig (5.1), the gradient of the encoder comes from the loss of reconstruction and the loss of k-subspace clustering loss, i.e., ,

$$\nabla_{\theta_e} \ell = \frac{\partial \ell_{\text{ae}}}{\partial \theta_e} + \lambda \frac{\partial \ell_{\text{ksc}}}{\partial \theta_e}. \tag{5.6}$$

By fixing $\{\mathbf{S}_i\}$, the assignments $\mathbf{W}$ for a mini-batch can be obtained easily and the required gradient for updating the CAE follows by back-propagating the error. The most difficult part in our problem is to find a way to update the subspaces efficiently and accurately. Here we will explain two approaches to update the subspaces. The first method is based on the SVD decomposition and the second one makes use of the Riemannian geometry of Grassmannian to update the subspaces.

5.2.1 SVD Update

Although SVD decomposition is computationally more expensive, we empirically observe that the SVD can provide satisfactory results. In our optimization, we update the encoder through back-propagation, batch by batch, and update the subspaces by employing the SVD once per epoch. This is mainly because updating subspaces more frequently hinders the convergence.

Intuitively, if the gradient takes the network to a bad direction, updating subspaces accordingly could intensify the negativity and worsen the CAE. Empirically, we observe updating subspaces after every epoch can neutralize the good and the bad directions of the gradient, yielding a stable framework.

The outliers may affect the subspace clustering badly, especially for k-subspace clustering. Therefore, when updating each subspace, we rule out the farthest 10% points as outliers. That is, after back propagation on CAE, we pass all the data through the encoder and assign their membership. We then sort the distance between each sample and the subspace it belongs to, and remove the outliers. Finally, we apply SVD on the remainder of points assigned to a subspace to obtain its new basis. Note that we only need to compute the largest p singular values and corresponding vectors to update a subspace. Specifically, after fixing w_{ij} and θ in Eqn. (5.5), updating the subspace basis $\mathbf{S}_i$ translates to solving the following problem

$$\underset{\mathbf{S}_i \in \mathcal{G}(d,p)}{\arg\min} \|\mathbf{Y}_i - \mathbf{S}_i \mathbf{S}_i^\top \mathbf{Y}_i\|_F^2 \,, \tag{5.7}$$

where $\mathbf{Y}_i$ consists of $\{f_{\theta_e}(\mathbf{x}_j)\}$ (as columns) that belong to cluster i. The solution to (5.7) corresponds to the column space $\mathbf{Y}_i$, which can be obtained by applying SVD on $\mathbf{Y}_i$ and taking the top p left singular vectors.

5.2.2 Gradient based update

If more frequent updates are required, the SVD solution can be replaced by a Riemannian gradient descent method based on the geometry of Grassmannian. In particular, let $\nabla \ell_{ksc}(\mathbf{S}_i)$ be the gradient of the loss with respect to $\mathbf{S}_i$ after an iteration (or accumulated gradient after a few iterations). In Riemannian optimization, $\mathbf{S}_i$ is updated according to the following rule;

$$\mathbf{S}_i^{(t+1)} = \mathrm{Y}_{\mathbf{S}_i^{(t)}}\Big(-\eta \Pi_{\mathbf{S}_i^{(t)}}\big(\nabla \ell_{ksc}(\mathbf{S}_i)\big)\Big) \,. \tag{5.8}$$

We explain Eqn. (5.8) with the aid of Fig. 5.2. First we note that a global coordinate system on a Riemannian manifold cannot be defined. As such Riemannian techniques make extensive use of the tangent bundle of the manifold to achieve their goal. Note that moving in the direction of $\nabla \ell_{ksc}(\mathbf{S}_i)$ will take us off the manifold. For a Riemannian manifold embedded in a Euclidean space (our case here), an ambient vector such as $\nabla \ell_{ksc}(\mathbf{S}_i)$ can be projected orthogonally on the tangent space at the current solution $\mathbf{S}_i^{(t)}$. We denote this operator by Π in Eqn. (5.8). The resulting tangent vector shown by the green arrow in Fig. 5.2 identifies a geodesic on the manifold. Moving along this geodesic (sufficiently) will guarantee to decrease the loss while preserving the orthogonality of the solution. In Riemannian optimization, this is achieved by a retraction which is local approximation to the exponential map on the manifold.

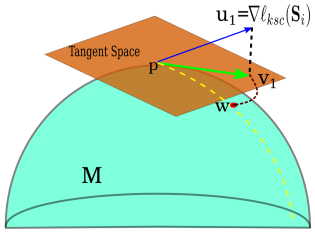


Figure 5.2: Illustration of how we update the gradient and keep the subspaces lie on the manifold

We denote the retraction in Eqn. (5.8) by Y. The only remaining bit is η which is the learning rate. For the Grassmannian, we have

$$\Pi_{\mathbf{S}}(\boldsymbol{u}) = \left(\mathbf{I}_d - \mathbf{S}\mathbf{S}^\top\right)\boldsymbol{u}\,, \tag{5.9}$$

$$\mathrm{Y}_{\mathbf{S}}(\boldsymbol{u}) = \mathrm{qf}\left(\mathbf{S} + \boldsymbol{u}\right)\,, \tag{5.10}$$

In Eqn. (5.10), qf is the Q part of the QR decomposition which is much faster than SVD. Although the SVD can perform good enough in experiments, we provide the other method which is faster in order to deal with very large datasets.

Algorithm 6 Scalable k-Subspace Clustering(Gradients update)

Input: dimensionality of subspaces p, number of class K, initial $\{\mathbf{S}_i\}$, epochs number T, pre-trained CAE, batch size b and dataset $\mathbf{x}_j$, $j = 1, ..., N$
for $m = 1 : T$ **do**
  for $n = 1 : b : N$
    Assign the membership for every point based on the distance
    to each subspace as Eqn. (5.2)
    Compute the gradients with respect to each subspace as
    Eqn. (5.5) $\ell_{ksc}(\mathbf{S}_i)$ and store them
    Update the CAE parameters Θ by Eqn. (5.6)
  end
  Project the gradients on Grassmannian manifolds
  based on Eqn. (5.9)
  Apply the gradients on the corresponding subspace
  accord to Eqn. (5.10)
end
Output: subspaces $\{\mathbf{S}_i\}$, and membership assignment w_{ij}

5.3 Experiment

We use Tensorflow Abadi et al. [2016] to build our networks. We used MNIST dataset LeCun et al. [1998] in our first experiment. MNIST is not considered as a standard dataset for previous subspace clustering algorithms, since the size of this dataset is far beyond the size that

traditional algorithms can handle. In addition, the original images do not follow the structure of linear subspaces. Taking advantage of CAE with our k-subspace clustering module, we aim to project all the MNIST data into a space which is more friendly for subspace clustering. In order to enforce our conclusion, we also evaluate our method on Fashion-MNIST dataset Xiao et al. [2017], a similar dataset to MNIST but with fashion images. Fashion-MNIST has 10 classes, with image being gray scale and of size 28×28. The images of Fashion-MNIST come from the fashion products which are classified based on a certain assortment and manually labeled by in-house fashion experts and reviewed by a separate team. It contains more variations within each class and it is thus more challenging compared to MNIST.

5.3.0.1 Baseline Methods

For most of the baselines and our method, we evaluate them on the whole datasets of MNIST and Fashion-MNIST with all 70000 images (including both training and testing sets). We compare our solution with the following generic clustering algorithms:

1) k-Means Lloyd [1982]: k-means finds clusters based on spatial closeness. As an EM method, it heavily relies on good initialization. Hence, for k-means (and other k-means based methods), we run the algorithm 20 times with different centroid seeds and report the best result.

2) Deep Embedded Clustering (DEC) Xie et al. [2016]: A rich structure for the MNIST dataset is proposed in Xie et al. [2016] which we follow here. In particular, stacked autoencoder(SAE) Bengio et al. [2007] along layer-wise pre-training was considered. The structure of the network reads as 784-500-500-2000-10. Image brightness is scaled from 0-1 to 0-5.2 to boost the performance. We observe that this method is highly sensitive to network parameters in the sense that even a small change in the structure will result in a significant performance drop. However, the feature extracted by the pre-trained model is very discriminative, i.e., , even simply using k-means on top of it can achieve competitive results. We call the feature extracted by this network the SEA features in the sequel.

3) Deep Clustering Network (DCN) Yang et al. [2017]: Based on the vanilla SAE, Yang *et al.*propose to add k-means clustering loss in addition to the data reconstruction loss of SAE.

4) Stacked Auto-Encoder followed by k-Means (SAE-KM): Extract features with SAE followed by applying k-means.

5) PCA followed by k-subspace (PCA-KS): It projects the original data onto a low-dimensional space at first, then use k-subspace to obtain the final results. Since PCA is a linear projection, it helps the readers to understand where the improvements come from compared to our nonlinear projection. The results are reported based on the 10 trails due to the randomness of initialization when employing k-subspace.

6) Convolutional Auto-Encoder followed by k-Means (CAE-KM): Extract features with SAE and then apply k-means. This is also the initialization for our method. It also can be considered as an evaluation of the quality of our initialization.

For those subspace clustering algorithms that rely on affinity matrix construction and spectral clustering, since they are not scalable to the whole dataset, we can report their results on the test sets (with 10000 images) only. We list several state-of-the-art subspace clustering algorithms for baselines: Sparse Subspace Clustering (SSC) Elhamifar and Vidal [2013b], Low Rank Representation (LRR) Liu et al. [2013], Kernel Sparse Subspace Clustering (KSSC) Pa-

tel and Vidal [2014], SSC by Orthogonal Matching Pursuit(SSC-OMP) You et al. [2016b] and the latest one Deep Subspace Clustering Networks (DSC-Net) Ji et al. [2017b].

5.3.0.2 Evaluation Metric

For all quantitative evaluations, we make use of the unsupervised clustering accuracy rate, defined as

$$\text{ACC } \% = \max_{M} \frac{\sum_{i=1}^{n} \mathbf{1}(l_i = M(s_i))}{n} \times 100\% \,. \tag{5.11}$$

where l_i is the ground-truth label, s_i is the subspace assignment produced by the algorithm, and M ranges over all possible one-to-one mappings between subspaces and labels. The mappings can be efficiently computed by the Hungarian algorithm. We also use normalized mutual information (NMI) as the additional quantitative standard. NMI scales from 0 to 1, where a smaller value means less correlation between predict label and ground truth label. Another quantitative metric is the adjusted Rand index (ARI), which is scaled between -1 and 1. The larger the ARI, the better the clustering performance.

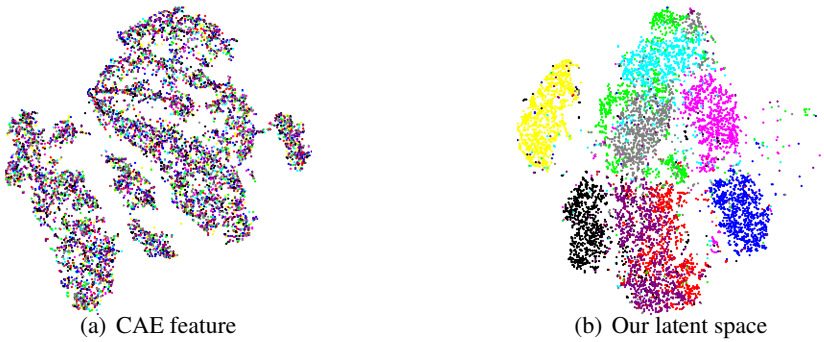
5.3.0.3 Implementation

We build our CAE in a bottle-neck structure, meaning we decrease the number of channels and the size of feature maps layer by layer. We design a six layer convolutional auto-encoder, where the kernel size in the first layer is 5 and in the last two layers of the encoder is 3×3. We set the number of channels in each layer to $20 - 10 - 5$ for the encoder, and the reverse for the decoder since they are symmetric in structure. Between layers, we set the stride to 2 in both horizontal and vertical directions, and use rectified linear unit (ReLU) as the non-linear activations. We use the same structure for both MNIST and Fashion-MNIST datasets.

Instead of greedy layer-wise pre-training Yang et al. [2017]; Xie et al. [2016], we pre-trained our network end-to-end from random initialization, until the reconstructed images are similar to the input ones (200 epochs suffice for pre-training). For subspaces initialization, we randomly sampled 2000 images and use DSC network to generate the clusters and corresponding subspaces. We noticed that initialization by the DSC subspaces leads to a model that under-performs in the beginning as compared to the k-Means algorithm. Nevertheless, our algorithm successfully recovers from such an initialization in all the experiments. During the optimization we use Adam Kingma and Ba [2014] optimizer, an adaptive momentum based gradient descent method, to minimize the loss, where we set the learning rate to 1.0×10^{-3} in both our pre-training and fine-tuning stages. For different datasets, the only two parameters need tuning are the λ in (7.2) and the subspace ambient dimension n , since the subspace intrinsic dimension p is fixed by the number of feature map of CAE.

5.3.1 MNIST Dataset

In this section, we will report and discuss results on the MNIST dataset. To the best of our knowledge, existing subspace clustering methods, with raw images as input, have not achieved satisfactory results on this dataset. As far as we know, the best performance reported in Peng et al. [2017] is in the range $58\% - 65\%$, where the DSIFT features are employed.



(a) CAE feature (b) Our latent space

Figure 5.3: Visualization using t-SNE on the latent space generated by projecting the testing set images on pre-trained CAE and our network. Points marked with the same color belong to the same class.

On MNIST, we fix our subspace dimension as 7, which means each subspace lies on a Grassmannian manifold $\mathcal{G}(80,7)$. The λ is set to 0.08, which balances between subspace clustering and CAE data reconstruction. Table (5.1) reports the results of all the baselines, including both subspace clustering algorithms and generic clustering algorithms. kSCN-S is to update the subspaces by employing the SVD decomposition, and kSCN-G stands for updating the subspaces by the Grassmannian gradients, which empirically is not as stable as the SVD updating scheme, probably due to the stochastic nature of each gradient step.This Grassmannian update, however, runs faster and takes less time to converge. We run our methods 15 times and report the average. The results of DEC are taken from the original paper. We tune the parameters for DCN very carefully and report the best results.

Among all the algorithms, our algorithm achieves the best performance in ACC and ARI. Especially for ACC, ours is 3% higher than the second best, namely DEC. From the results, it is not difficult to conclude that the DEC and DCN perform only marginally better than SAE-KM, which is the initialization for DEC and DCN. Specifically, DEC improvements over the initialization are around 3% and DCN only boosts around 1.5% over SAE-KM. By contrast, our method starts from CAE-KM (with 51% ACC), and improves it by 36.14% to 87.14% ACC. The improvement can be visualized by Fig (5.3), which shows the projections of CAE feature space and the latent space of our network in a two-dimensional space. Compared to CAE features, which are all mixed up, our latent space are well separated even though the two-dimensional space is not suitable for visualizing subspace structure as they reside in high-dimensional ambient space.

For traditional subspace clustering algorithms, around 37 Gigabytes of memory is required to store the affinity matrix, which is computationally prohibitive. Therefore, we contrast our algorithm against SSC, LRR, KSSC, SSC-OMP and Deep Subspace Clustering Networks on a smaller experiment, namely only using the 10000 test images of the MNIST dataset (see Table. (5.2) for results). Note that SSC-OMP completely fails in dealing with feature generated by SAE and CAE, achieving around 12% ACC and 2% NMI. Generally speaking, with more samples, better accuracies are expected. We can see that all the subspace clustering algorithms using the SAE feature perform better compared to using CAE feature. To some extent, it

Table 5.1: Results on MNIST (70000 samples).

	SAE-KM	CAE-KM	K-means	PCA-KS	DEC	DCN	kSCN-G	kSCN-S
ACC	81.29%	51 %	53%	68.53%	84.3%	83.31%	82.22%	**87.14%**
NMI	73.78 %	44.87%	50 %	64.17%	80%	**80.86**%	73.93%	78.15%
ARI	67%	33.52 %	37 %	54.17%	75%	74.87%	71.10%	**75.81** %

Table 5.2: The results of subspace clustering algorithms on the test sets of the MNIST and Fashion-MNIST datasets, the best two are in bold

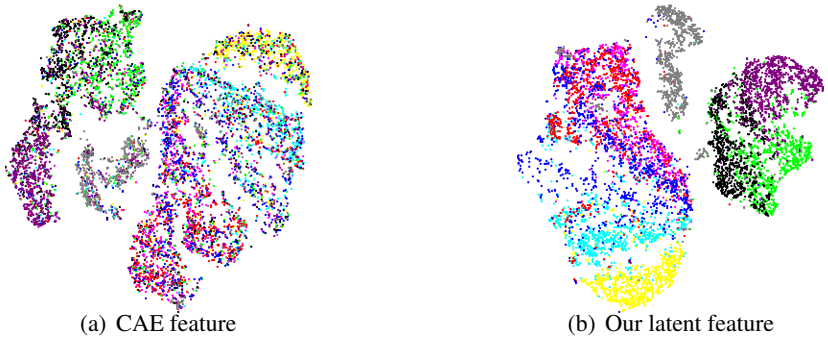
	MNIST		Fashion-MNIST	
	ACC	NMI	ACC	NMI
SSC-SAE	75.49%	66.26%	52.33 %	51.26%
SSC-CAE	43.03 %	56.81%	35.31%	18.10%
LRR-SAE	74.09%	66.97%	**58.09**%	59.19%
LRR-CAE	51.37%	66.59%	34.43%	18.57%
KSSC-SAE	**81.53**%	**84.53**%	57.10%	**60.40**%
KSSC-CAE	56.42%	65.66%	35.41%	18.18%
DSC-Net	53.20%	47.90%	55.81%	54.80 %
kSCN-S	**83.30**%	**77.38**%	**60.02**%	**62.30**%

proves that there exists a nonlinear mapping which is more favorable to subspace clustering. At the same time, our algorithm still achieves the best results within all subspace clustering algorithms, even higher that DSC-Net.

5.3.2 Fashion-MNIST

Unlike MNIST dataset, which only contains simple digits, every class in Fashion-MNIST has different styles and come from different gender groups: men, women, kids and neutral. In Fashion-MNIST, there are 60000 training images and 10000 test images. In our case, we pre-trained and fine-tuned the network using the whole dataset. On Fashion-MNIST, we fix our subspace dimension to 11 and set λ to 0.11.

Consistent with the MNIST dataset, the DCN sightly improves upon its initialization (SAE-KM) in terms of ACC and NMI. Moreover, we find out that the DCN algorithm works better with smaller learning rates, which in turn requires more epochs to converge properly. From Table (5.3), we can see that our method still improves the accuracy by 24% compared to our initialization, and outperforms other algorithms. The t-SNE maps in Fig. (5.4) show that there exists a subspace structure in our latent space even in two dimensional space. Table (5.2) shows that the subspace clustering algorithms also achieve acceptable results on the 10000 test sets, with our algorithm being the best among all. Compared to other subspace clustering algorithms, our algorithm runs much faster, only requiring less than 8 minutes. (including pre-training and fine tuning with subspace clustering) to generate final results, whereas the traditional algorithms need at least 40 minutes to process these 10000 samples even after the



(a) CAE feature (b) Our latent feature

Figure 5.4: Visualization using t-SNE for the latent space generated by pretrained CAE and our network on Fashion-MNIST. Points marked with the same color belong to the same class.

Table 5.3: Results on Fashion-Mnist

	SAE-KM	CAE-KM	K-means	PCA-KS	DCN	*k*SCN-G	*k*SCN-S
ACC	54.35%	39.84 %	47.58%	53.41%	56.14%	58.67%	**63.78**%
NMI	58.54 %	39.80%	51.24 %	57.5%	59.4%	52.88%	**62.04**%
ARI	41.86%	25.93 %	34.86 %	41.17%	43.04%	42%	**48.04**%

dimensionality reduction.

5.3.3 Further Discussion

Based on the above experiments, we observe that our algorithm consistently achieves higher accuracies as compared to DCN (even with the initialization using CAE). One may argue that the performance gain over DCN is due to the fact that unlike SAE, CAE can be trained easily[2]. To verify that this is not the case, we replace the SAE with the CAE in DCN to see whether DCN can generate competitive results. Table (5.4) demonstrates that even with the CAE, the DCN cannot boost the clustering results as much as ours. On MNIST, DCN-CAE can hardly improve the accuracy and NMI; on Fashion-MNIST, it can increase the accuracy more than 3 percent (and NMI around 1 percent).

This can be attributed to the loss introduced by k-means in DCN, compared to our k-subspace clustering loss which we believe is more robust.

In other words, the subspace structure could be more desirable than cluster centroids in high dimensional space. .

5.4 Summary

In this chapter, we proposed a scalable deep k-subspace clustering algorithm, which combined the k-subspace clustering and convolutional auto-encoder in a principle way. Our algorithm

[2]In our experiments, the number of parameters in SAE is 2600 times more than that of CAE.

Table 5.4: The performance of the DCN-CAE and its CAE initialization.

	MNIST		Fashion-MNIST	
	ACC	NMI	ACC	NMI
DCN-CAE	51.10 %	45.18 %	45.64 %	47.8%
Initilization	50.98%	44.87 %	42.38 %	46.75 %

makes it possible to scale subspace clustering algorithms to large datasets. Furthermore, we proposed two efficient and robust schemes to update the subspaces. These allow our k-SC networks to iteratively fit every sample into its corresponding subspace and update the subspaces accordingly, even from a bad initialization (as observed in our experiments).

Our extensive experiments on MNIST and Fashion-MNIST dataset demonstrated that our deep k-subspace clustering method provides significant improvements over various state-of-the-art subspace clustering solutions in terms of clustering accuracy and efficiency.

Chapter 6

Neural Collaborative Subspace Clustering

In this chapter, we further our step on making subspace clustering more scalable and accurate. Deep Subspace Clustering Networks (DSC-Net) in Chapter 4 are introduced to tackle the non-linearity arising in subspace clustering, where data is non-linearly mapped to a latent space with convolutional auto-encoders and a new *self-expressive layer* is introduced between the encoder and decoder to facilitate an end-to-end learning of the affinity matrix. Although DSC-Net outperforms traditional subspace clustering methods by large, its computational cost and memory footprint can become overwhelming even for mid-size problems. k-Subspace Clustering Networks (k-SCN) in Chpater 5 is proposed to make subspace clustering applicable to large datasets. However, it needs extra effort to take care of subspace explicitly, for example the dimension of subspace needs to choose heuristically.

Therefore, We propose a neural structure to improve the performance of subspace clustering while being mindful to the scalability issue. To this end, we first formulate subspace clustering as a classification problem, which in turn removes the spectral clustering step from the computations.

Our neural model is comprised of two modules, one for classification and one for affinity learning. Both modules collaborate during learning, hence the name "Neural Collaborative Subspace Clustering". During training and in each iteration, we use the affinity matrix generated by the subspace self-expressiveness to supervise the affinity matrix computed from the classification part. Concurrently, we make use of the classification part to improve self-expressiveness to build a better affinity matrix through collaborative optimization.

We evaluate our algorithm on three datasets , namely MNIST LeCun et al. [1998], Fashion-MNIST Xiao et al. [2017], and the Stanford Online Products dataset Oh Song et al. [2016] which exhibit different levels of difficulty. Our empirical study shows the superiority of the proposed algorithm over several state-of-the-art baselines including deep subspace clustering techniques.

6.1 Introduction

Subspace Clustering (SC) has achieved great success in various computer vision tasks, such as motion segmentation Kanatani [2001]; Elhamifar and Vidal [2009]; Ji et al. [2014a, 2016],

face clustering Ho et al. [2003]; Elhamifar and Vidal [2013b] and image segmentation Yang et al. [2008]; Ma et al. [2007].

Majority of the SC algorithms Yan and Pollefeys [2006]; Chen and Lerman [2009]; Elhamifar and Vidal [2013b]; Liu et al. [2013]; Wang et al. [2013]; Lu et al. [2012]; Ji et al. [2015]; You et al. [2016a] rely on the linear subspace assumption to construct the affinity matrix for spectral clustering. However, data do not naturally conform to linear models, which in turns results in the development of non-linear SC techniques. Kernel methods Chen et al. [2009]; Patel et al. [2013]; Patel and Vidal [2014]; Yin et al. [2016]; Xiao et al. [2016]; Ji et al. [2017a] can be employed to implicitly map data to higher dimensional spaces, hoping that data fit better to linear models in the resulting spaces. That said, it is not straightforward to identify the right kernel function and its parameters.

The use of deep neural networks as non-linear mapping functions to determine subspace friendly latent spaces has formed the latest developments in the field with promising results 4.

Despite significant improvements, SC algorithms still resort to spectral clustering which in hindsight requires constructing an affinity matrix. This step, albeit effective, hampers the scalability as it takes $O(n^2)$ memory and $O(kn^2)$ computations for storing and decomposing the affinity matrix for n data points and k clusters. There are several attempts to resolve the scalability issue. For example, You et al. [2016a] accelerate the construction of the affinity matrix using orthogonal matching pursuit; Zhang et al. [2018] resort to k-subspace clustering to avoid generating the affinity matrix. However, the issue is not fully addressed either due to the use of spectral clustering You et al. [2016a], or mitigates but at the cost of performance Zhang et al. [2018].

Here are a few attempts to tackle the scalability of subspace clustering. The SSC-Orthogonal Matching Pursuit (SSC-OMP) You et al. [2016b] replaces the large scale convex optimization procedure with the OMP algorithm to represent the affinity matrix. However, SSC-OMP sacrifice the clustering performance in favor of speeding up the computations, and it still may fail when the number of data points is very large. k-Subspace Clustering Networks (k-SCN) in Chpater 5 is proposed to make subspace clustering applicable to large datasets. This is achieved via bypassing the construction of affinity matrix and consequently avoiding spectral clustering, and introducing the iterative method of k-subspace clustering Tseng [2000]; Bradley and Mangasarian [2000] into a deep structure. Although k-SCN develops two approaches to update the subspace and networks, it still shares the same drawbacks as iterative methods, for instance, it requires a good initialization, and seems fragile to outliers.

6.2 Our Formulation

To design a scalable SC algorithm, our idea is to identify whether a pair of points lies on the same subspace or not. Upon attaining such knowledge (for a large-enough set of pairs), a deep model can optimize its weights to maximize such relationships (being on subspaces or not). This can be nicely cast as a binary classification problem. However, since ground-truth labels are not available to us, it is not obvious how such a classifier should be built and trained.

In this work, we propose to make use of two confidence maps (see Fig. 6.1 for a conceptual visualization) as a supervision signal for SC. To be more specific, we make use of the concept of self-expressiveness to identify positive pairs, i.e., , pairs that lie on the same subspaces.

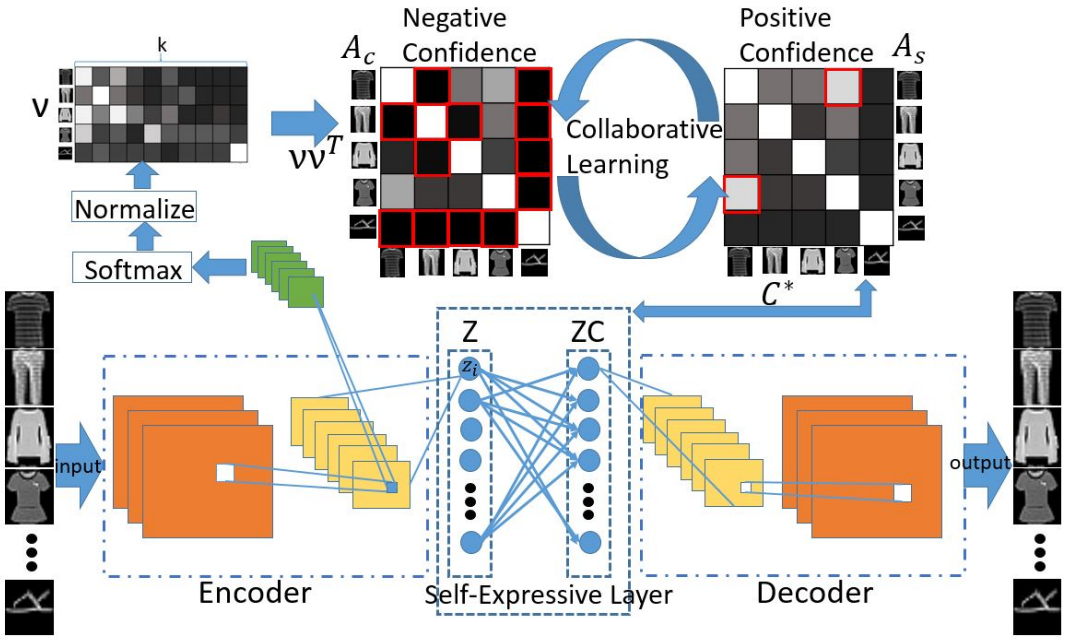


Figure 6.1: The Neural Collaborative Subspace Clustering framework. The affinity matrix generated by self-expressive layer, $\mathbf{A}_s$, and classifier, $\mathbf{A}_c$, supervise each other by selecting the high confidence parts for training. Red squares in $\mathbf{A}_s$ highlight positive pairs (belonging to the same subspace). Conversely, red squares in $\mathbf{A}_c$ highlight the negative pairs (belonging to different subspace). Affinities are coded with shades, meaning that light gray denotes large affinity while dark shades representing small affinities.

To identify negative pairs, pairs that do not belong to the same subspace, we benefit from a negative confidence map. This, as we will show later, is due to the fact that the former can confidently mine positive pairs (with affinity close to 1) while the latter is good at localizing negative pairs (with affinity close to 0). The two confidence maps, not only provide the supervision signal to optimize a deep model, but act collaboratively as partial supervisions for each other.

6.2.1 Binary Classification

Given a dataset with n points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ from k clusters, we aim to train a classifier to predict class labels for data points without using any ground-truth labels. To this end, we propose to use a multi-class classifier which consists of a few convolutional layers (with non-linear rectifiers) and a softmax output layer. We then convert the resulting outputs to an affinity-based binary classifier by

$$\mathbf{A}_c(i,j) = \circ_i \circ_j^\top, \tag{6.1}$$

where $\circ_i \in \mathbb{R}^k$ is a k dimensional prediction row vector after ℓ_2 normalization. Ideally, when $\circ_i$ is one-hot, $\mathbf{A}_c$ is a binary matrix encoding the confidence of data points belong-

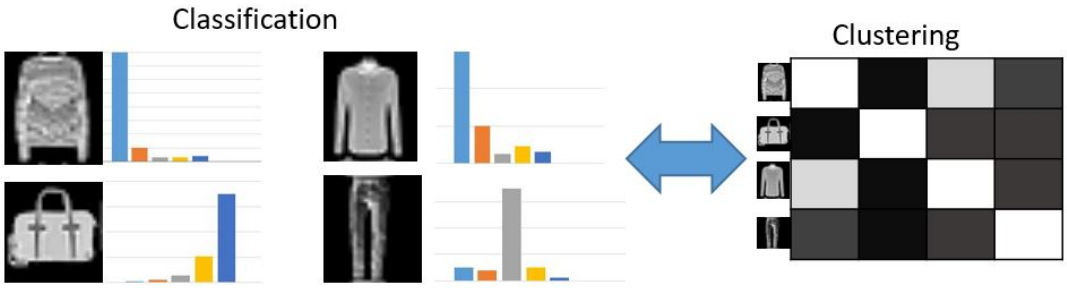


Figure 6.2: By normalizing the feature vectors after softmax function and computing their inner product, an affinity matrix can be generated to encode the clustering information.

ing to the same cluster. So if we supervise the classifier using $\mathbf{A}_c$, we will end up with a binary classification problem. Also note that $\mathbf{A}_c(i,j)$ can be interpreted as the cosine similarity between softmax prediction vectors of $\mathbf{x}_i$ and $\mathbf{x}_j$, which has been widely used in different contexts Nguyen and Bai [2010]. However, unlike the cosine similarity which lies in $[-1,1]$, $\mathbf{A}_c(i,j)$ lies within $[0,1]$, since the vectors are normalized by softmax and ℓ_2 norm. We illustrate this in Fig. 6.2.

6.2.2 Self-Expressiveness Affinity

Subspace self-expressiveness can be worded as: one data point $\mathbf{x}_i$ drawn from linear subspaces $\{\mathcal{S}_i\}_{i=1}^k$ can be represented by a linear combination of other points from the same subspace. Stacking all the points into columns of a data matrix $\mathbf{X}$, the self-expressiveness can be simply described as $\mathbf{X} = \mathbf{XC}$, where $\mathbf{C}$ is the coefficient matrix.

It has been shown (e.g., , Elhamifar and Vidal [2009]; Ji et al. [2014b]) that by minimizing certain norms of coefficient matrix $\mathbf{C}$, a block-diagonal structure (up to permutations) on $\mathbf{C}$ can be achieved. This translates into $c_{ji} \neq 0$ only if data points coming from the same subspace. Therefore, the loss function of learning the affinity matrix can be written as:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_p \quad \text{s.t.} \quad \mathbf{X} = \mathbf{XC},\ \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{6.2}$$

where $\|\cdot\|_p$ is a matrix norm. For example, Sparse Subspace Clustering (SSC) Elhamifar and Vidal [2009] uses the ℓ_1 norm, Low Rank Representation (LRR) models Liu and Yan [2011]; Vidal and Favaro [2014] opt for the nuclear norm, and Efficient Dense Subspace Clustering Ji et al. [2014b] utilizes the ℓ_2 norm. To handle data corruption, a relaxed version can be derived as:

$$\begin{aligned} \mathbf{C}^* = \arg\min_{\mathbf{C}} \|\mathbf{C}\|_p + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{XC}\|_F^2 \\ \text{s.t.} \quad \text{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \tag{6.3}$$

Here, λ is a weighting parameter balancing the regularization term and the data fidelity term.

To handle subspace non-linearity, one can employ convolutional auto-encoders to non-linearly map input data $\mathbf{X}$ to a latent space $\mathbf{Z}$, and transfer the self-expressiveness into a linear layer (without non-linear activation and bias parameters) named *self-expressive layer* Ji et al. [2017b] (see the bottom part of Fig. 6.1). This enables us to learn the subspace affinity $\mathbf{A}_s$ in an end-to-end manner using the weight parameters $\mathbf{C}^*$ in the self-expressive layer:

$$\mathbf{A}_s(i,j) = \begin{cases} (|c^*_{ij}| + |c^*_{ji}|)/2c_{\max} & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases} \tag{6.4}$$

where $c_{\max}$ is the maximum absolute value of off-diagonal entries of the current row. Note that $\mathbf{A}_s(i,j)$ then lies within $[0,1]$.

6.2.3 Collaborative Learning

The purpose of collaborative learning is to find a principle way to benefit from the strengths of different modules. The classification module and self-expressive module distill different information in the sense that the former tends to extract more abstract and discriminative features while the latter focuses more on capturing the pairwise correlation between data samples. As alluded to earlier, ideally, the subspace affinity $\mathbf{A}_s(i,j)$ is nonzero only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from the same subspace, which means that $\mathbf{A}_s$ can be used to mine similar pairs (i.e., , positive samples). On the other hand, if the classification affinity $\mathbf{A}_c(i,j)$ is close to zero, it indicates strongly that $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar (i.e., , negative sample). Therefore, we carefully design a mechanism to let both modules collaboratively supervise each other.

Given $\mathbf{A}_c$ and $\mathbf{A}_s$, we pick up the high-confidence affinities as supervision for training. We illustrate this process in Fig. 6.1. The "positive confidence" in Fig. 6.1 denotes the ones from the same class, and the "negative confidence" represents the ones from different classes. As such, we select high affinities from $\mathbf{A}_s$ and small affinities from $\mathbf{A}_c$, and formulate the collaborative learning problem as:

$$\begin{gathered} \min_{\mathbf{A}_s,\mathbf{A}_c} \mathbf{\Omega}(\mathbf{A}_s,\mathbf{A}_c,l,u) = \\ L_{pos}(\mathbf{A}_s,\mathbf{A}_c,u) + \alpha L_{neg}(\mathbf{A}_c,\mathbf{A}_s,l), \end{gathered} \tag{6.5}$$

where the $L_{pos}(\mathbf{A}_s,\mathbf{A}_c,u)$ and $L_{neg}(\mathbf{A}_c,\mathbf{A}_s,l)$ denote the cross-entropy function with sample selection process, which can be defined as follows:

$$\begin{gathered} L_{pos}(\mathbf{A}_s,\mathbf{A}_c,u) = H(\mathbf{M}_s||\mathbf{A}_c) \\ \text{s.t} \quad \mathbf{M}_s = \mathbb{1}(\mathbf{A}_s > u), \end{gathered} \tag{6.6}$$

and

$$\begin{gathered} L_{neg}(\mathbf{A}_c,\mathbf{A}_s,l) = H(\mathbf{M}_c||(\mathbf{1}-\mathbf{A}_s)) \\ \text{s.t} \quad \mathbf{M}_c = \mathbb{1}(\mathbf{A}_c < l), \end{gathered} \tag{6.7}$$

where $\mathbb{1}(\cdot)$ is the indicator function returning 1 or 0, $\{u,l\}$ are thresholding parameters, and

H is the entropy function, defined as $H(\mathbf{p}||\mathbf{q}) = \sum_j p_j \log(q_j)$.

Note that the cross-entropy loss is a non-symmetric metric function, where the former probability serves a supervisor to the latter. Therefore, in Eqn. (6.6), the subspace affinity matrix $\mathbf{A}_s$ is used as the "teacher" to supervise the classification part (the "student"). Conversely, in Eqn. (6.7), the classification affinity matrix $\mathbf{A}_c$ works as the "teacher" to help the subspace affinity learning module to correct negative samples. That said, to better facilitate gradient back-propagation between two modules, we can approximate indicator function by replacing $\mathbf{M}_s$ with $\mathbf{M}_s\mathbf{A}_s$ in Eqn. (6.6) and $\mathbf{M}_c$ with $\mathbf{M}_c(1-\mathbf{A}_c)$ in Eqn. (6.7). The weight parameter α in Eqn. 6.5, called collaboration rate, controls the contributions of L_{pos} and L_{neg}. It can be set as the ratio of the number of positive confident pairs and the negative confident pairs, or tuned for better performance.

6.2.4 Overall Model

After introducing all the building blocks of this work, we now explain how to jointly organize them in a network and train it with a carefully defined loss function. As shown in Fig. 6.1, our network is composed of four main parts: (i) a convolutional encoder that maps input data $\mathbf{X}$ to a latent representation $\mathbf{Z}$; (ii) a linear self-expressive layer which learns the subspace affinity through weights $\mathbf{C}$; (iii) a convolutional decoder that maps the data after self-expressive layer, i.e., , $\mathbf{ZC}$, back to the input space $\hat{\mathbf{X}}$; (iv) a multi-class classifier that outputs k dimensional prediction vectors, with which a classification affinity matrix can be constructed. Our loss function consists of two parts, i.e., , collaborative learning loss and subspace learning loss:

$$\begin{aligned}\mathbb{L}(\mathbf{X};\Theta) = L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s)\\ + \lambda_{cl}\mathbf{\Omega}(\mathbf{X},u,l;\Theta,\mathbf{A}_s,\mathbf{A}_c).\end{aligned} \tag{6.8}$$

Here, Θ denotes the parameters of the neural network and λ_{cl} is the weight of the collaborative learning loss. The $L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s)$ is the loss to train the affinity matrix through self-expressive layer. Combining Eqn. (6.3) and the reconstruction loss of the convolutional auto-encoder, we arrive at:

$$\begin{aligned}L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s) = \|\mathbf{C}\|_F^2 + \frac{\lambda_1}{2}\|\mathbf{Z}-\mathbf{ZC}\|_F^2\\ + \frac{1}{2}\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2\\ \text{s.t.}\quad \text{diag}(\mathbf{C}) = \mathbf{0},\end{aligned} \tag{6.9}$$

where $\mathbf{A}_s$ is a function of $\mathbf{C}$ as defined in (6.4). After the training stage, we no longer need to run the decoder and self-expressive layer to infer the labels. We can directly infer the cluster labels through the classifier output ν:

$$s_i = \arg\max_h \nu_{ih}, \quad h = 1,\ldots,k, \tag{6.10}$$

where s_i is the cluster label of image $\mathbf{x}_i$.

Algorithm 7 Neural Collaborative Subspace Clustering

Input: dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, number of clusters k, sample selection threshold u and l , learning rate of auto-encoder η_{ae}, and learning rate of other parts η
Initialization: Pre-train the Convolutional Auto-encoder by minimizing the reconstruction error.
repeat
For every mini-batch data
Train auto-encoder with *self-expressive layer* to minimize loss function in Eqn. (6.9) to update $\mathbf{A}_s$.
Forward the batch data through the classifier to get $\mathbf{A}_c$.
Do sample selection and collaborative learning through minimizing Eqn. (6.5) to update the classifier.
Jointly update all the parameters by minimizing Eqn. (6.8).
until reach the maximum epochs
Output: Get the cluster s_i for all samples by Eqn. (6.10)

6.3 Training

In this section, we provide more details about how training will be done. Similarly to other auto-encoder based clustering methods, we pre-train the auto-encoder by minimizing the reconstruction error to get a good initialization of latent space for subspace clustering.

According to Elhamifar and Vidal [2009], the solution to (6.2) is guaranteed to have a block-diagonal structure (up to certain permutations) under the assumption that the subspaces are independent. To account for this, we make sure that the dimensionality of the latent space ($\mathbf{Z}$) is greater than (the subspace intrinsic dimension) $\times$ (number of clusters) [1]. In doing so, we make use of the stride convolution to down-sample the images while increasing the number of channels over layers to keep the dimensionality of the latent space large. Since we have pre-trained the auto-encoder, we use a smaller learning rate in the auto-encoder when the collaborative learning is performed. Furthermore, compared to DSC-Net or other spectral clustering based methods which require to perform involved techniques to post process the affinity matrix, we only need to compute $\mathbf{A}_s = (|\mathbf{C}^*| + |\mathbf{C}^{*T}|)/2$ and normalize it (divided by the largest value in each row and assign 1.0 to the diagonal entries) to ensure the subspace affinity matrix lies in the same range with the classification affinity matrix.

We adopt a three-stage training strategy: first, we train the auto-encoder together with the self-expressive layer using the loss in (6.9) to update the subspace affinity $\mathbf{A}_s$; second, we train the classifier to minimize Eqn. (6.5); third, we jointly train the whole network to minimize the loss in (6.8). All these details are summarized in Algorithm 7.

[1] Note that our algorithm does not require specifying subspace intrinsic dimensions explicitly. Empirically, we found a rough guess of the subspace intrinsic dimension would suffice, e.g., , in our experiments, we set it to 9.

6.4 Experiments

We implemented our framework with Tensorflow-1.6 Abadi et al. [2016] on an Nvidia TITAN X GPU. We mainly evaluate our method on three standard datasets, i.e., , MNIST, Fashion-MNIST and a subset of the Stanford Online Products dataset. All of these datasets are considered challenging for subspace clustering as it is hard to perform spectral clustering on datasets of this scale.

The number of clusters k is set to 10 as input to all competing algorithms. For all the experiments, we pre-train the convolutional auto-encoder for 60 epochs with a learning rate 1.0×10^{-3}.

The hyper parameters in our loss function are easy to tune. λ_1 in Eqn. (6.9) controls self-expressiveness, and it also affects the choice of u and l in Eqn. (6.8). If λ_1 is set large, the coefficient in the affinity matrix will become larger, and in that case the u should be increased. The other parameter λ_{cl} balances the cost of subspace clustering and collaborative learning, and we usually set it to keep these two terms in the same scale to treat them equally. We keep the $\lambda_1 = 10$ in all the experiments, and slightly change the l and u for each dataset. In all experiments, we employ a special residual convolutional block He et al. [2016a], which doesn't contain batch normalization layer. This is because we empirically observed that batch normalization would negatively affect the subspace structure in the latent space. We use the Rectified Linear Unit (ReLu) as the non-linear activation in the blocks. Hence, each residual block has two convolutional kernels and two ReLu layers as activation function.

Since no ground truth label is available to the algorithm, we chose to use a larger batch size as compared to the practice in supervised learning. This makes the training stable and robust. Specifically, we set the batch size to 5000, and used Adam Kingma and Ba [2014], an adaptive momentum based gradient descent method to minimize the loss in all our experiments. We set the learning rate to 1.0×10^{-5} for the auto-encoder and 1.0×10^{-3} for other parts of the network in all training stages. In each iteration, we train the self-expressiveness layer for 50 iterations, followed by 50 iterations' training along with classifier, and fine-tune the whole network by 10 iterations.

Baseline Methods. We use various clustering methods as the baseline methods including the classic clustering methods, subspace clustering methods, deep clustering methods, and GAN based methods. Specifically, we have the following baselines:

- classic methods: K-Means Lloyd [1982] (KM), K-Means with our CAE-feature (CAE-KM) and SAE-feature (SAE-KM);

- subspace clustering algorithms: sparse subspace clustering (SSC) Elhamifar and Vidal [2013b], Low Rank Representation (LRR) Liu et al. [2013], Kernel Sparse Subspace Clustering (KSSC) Patel and Vidal [2014], Deep Subspace Clustering Network (DSC-Net) Ji et al. [2017b], and k-Subspace Clustering Network (k-SCN) Zhang et al. [2018];

- deep clustering methods: Deep Embedded Clustering (DEC) Xie et al. [2016], Deep Clustering Network (DCN) Yang et al. [2017], and Deep Adaptive image Clustering (DAC) Chang et al. [2017];

- GAN based clustering methods: Info-GAN Chen et al. [2016] and ClusterGAN Mukherjee et al. [2018].

Evaluation Metric. For all quantitative evaluations, we make use of the unsupervised clustering accuracy rate, defined as

$$\text{ACC } \% = \max_{\Pi} \frac{\sum_{i=1}^{n} \mathbb{1}(y_i = \Pi(c_i))}{n} \times 100\% \,. \tag{6.11}$$

where y_i is the ground-truth label, c_i is the subspace assignment produced by the algorithm, and Π ranges over all possible one-to-one mappings between subspaces and labels. The mappings can be efficiently computed by the Hungarian algorithm.

We also use normalized mutual information (NMI) as the additional quantitative standard. NMI scales from 0 to 1, where a smaller value means less correlation between predicted labels and ground truth labels. Another quantitative metric is the adjusted Rand index (ARI), which is scaled between 0 and 1. It computes a similarity between two clusters by considering all pairs of samples and counting pairs that are assigned to the same or different clusters. The larger the ARI, the better the clustering performance.

6.4.1 MNIST

MNIST consists of 70000 hand-written digit images of size 28×28. Subspace non-linearity arises naturally for MNIST due to the variance of scale, thickness and orientation among all the images of each digit. We thus apply our method on this dataset to see how well it can handle this type of subspace non-linearity.

In this experiment, we use a convolutional layer followed by three pre-activation residual blocks He et al. [2016c] without batch normalization as encoder and a self-expressive layer in between encoder and decoder for the subspace affinity learning module. The kernel size is fixed as 3 and the number of channel is 20, 30 and 40 for each block. For the classification module, we connect three more convolutional layers after the encoder layers with kernel size 2, and one convolutional layer with kernel size 1 to output the feature vector. Meanwhile, we also tried simple version auto-encoder (same as DSC) and find that the classification module can not share the layers with the encoder, but need to use an independent small network because of its low capacity. For the threshold parameters u and l, we set them to 0.7 and 0.1 respectively in the first epoch of training, and increase u to 0.9 afterwards.

We report the clustering results of all competing methods in Table 6.1. Since spectral clustering based methods (i.e., , SSC-CAE, LRR-CAE, KSSC-CAE, DSC-Net) can not apply on the whole dateset (due to memory and computation issues), we only use the 10000 samples to show how they perform. As shown in Table 6.1, subspace algorithms do not perform very well even on 10000 samples. Although the DSC-Net is trapped by training the self-expressive layer, it outperforms other subspace clustering algorithm, which shows the potential of learning subspace structure using neural networks. On the other hand, DEC, DCN, k-SCN and our algorithm are all based on auto-encoder, which learn embeddings with different metrics to help clustering. However, our classification module boosts the performance by making the latent space of auto-encoder more discriminative. Therefore, our algorithm incorporates the advantage of different classes, e.g., , self-expressiveness, nonlinear mapping and discriminative

features, and achieves the best results among all the algorithms thanks to the collaborative learning paradigm.

	ACC(%)	NMI(%)	ARI(%)
CAE-KM	51.00	44.87	33.52
SAE-KM	81.29	73.78	67.00
KM	53.00	50.00	37.00
DEC	84.30	80.00	75.00
DCN	83.31	80.86	74.87
SSC-CAE	43.03	56.81	28.58
LRR-CAE	55.18	66.54	40.57
KSSC-CAE	58.48	67.74	49.38
DSC-Net	65.92	73.00	57.09
k-SCN	87.14	78.15	75.81
Ours	**94.09**	**86.12**	**87.52**

Table 6.1: Clustering results of baseline methods and our collaborative scheme on the MNIST dataset. For all the metrics, the larger value is better. The best results are shown in bold.

6.4.2 Fashion-MNIST

Same as in MINIST, Fashion-MNIST also has 70000 images of size 28×28. It consists of various types of fashion products. Unlike MNIST, every class in Fashion-MNIST has different styles with different gender groups (e.g., , men, women, kids and neutral). As shown in Fig. 6.3, the high similarity between several classes (such as { Pullover, Coat, Shirt}, { T-shirt, Dress }) makes the clustering more difficult. Compared to MNIST, the Fashion-MNIST clearly poses more challenges for unsupervised clustering.

On Fashion-MNIST, we employ a network structure with one convolutional layer, followed by pre-activation residual blocks without batch normalization in the encoder, and with a symmetric structure in the decoder. As the complexity of dataset increases, we also raise the dimensionality of ambient space to better suit self-expressiveness, and increase capacity for the classification module. For all convolutional layers, we keep kernel size as 3 and set the number of channels to 10-20-30-40 in the encoder, where three residual blocks are employed for 20, 30 and 40 repsectively.

We report the clustering results of all methods in Table 6.2, where we can clearly see that our framework outperforms all the baselines by a large margin including the best-performing baseline k-SCN. Specifically, our method improves over the second best one (i.e., , k-SCN) by 8.36%, 4.2% and 9% in terms of accuracy, NMI and ARI. We can clearly observe from Fig. 6.4 that the latent space of our framework, which is collaboratively learned by subspace and classification modules, has strong subspace structure and also keeps each subspace discriminative. For subspace clustering methods and similar to the previous experiment, we used only 10000 samples. DSC-Net does not drop a lot while the performance of other subspace clustering algorithms decline sharply compared with their performance on MNIST.

Figure 6.3: Examples of the Fashion-MNIST dataset

6.4.3 Stanford Online Products

The Stanford Online Products dataset is mainly used for supervised metric learning, and it is thus considered to be difficult for unsupervised tasks. Compared to the previous two datasets, the challenging aspects of this dataset include: (i) the product images contain various backgrounds, from pure white to real world environments; (ii) each product has various shapes, colors, scales and view angles; (iii) products across different classes may look similar to each other. To create a manageable dataset for clustering, we manually pick 10 classes out of 12 classes, with around 1000 images per class (10056 images in total), and then re-scale them to 32×32 gray images, as shown in Fig. 6.5.

Our networks for this dataset has one convolutional layer with 10 channels, followed by three pre-activation residual blocks without batch normalization, which have 20, 30 and 10 channels respectively. Table 6.3 shows the performance of all algorithms on this dataset. Due to the challenging nature of the dataset, most deep learning based methods fail to produce reasonable results. For example, DEC and DCN perform below their initialization, and DAC cannot self-supervise its model to achieve a better result. Similarly, infoGAN also fails to find clustering pattern. Along with KSSC and DSC-Net, our algorithm achieves the best results compared, especially in comparison to the deep learning based algorithms, hinting a better handling of non-linearity.

We can also observe that subspace based clustering algorithms perform better than clustering methods. This illustrates the benefits of the underlying subspace assumption in high dimensional regimes.

In summary, compared to other deep learning methods, our framework is not sensitive to

Figure 6.4: The visualization of the latent space of our collaborative scheme through dimensionality reduction by PCA.

Figure 6.5: Examples of the Stanford Online Products Dataset

the architecture of the neural network, as long as the dimensionality meets the requirement of subspace self-expressiveness. Furthermore, the two modules in our network progressively improve the performance in a collaborative way, which is both effective and efficient.

	ACC(%)	NMI(%)	ARI(%)
SAE-KM	54.35	58.53	41.86
CAE-KM	39.84	39.80	25.93
KM	47.58	51.24	34.86
DEC	59.00	60.10	44.60
DCN	58.67	59.4	43.04
DAC	61.50	63.20	50.20
ClusterGAN	63.00	64.00	50.0
InfoGAN	61.00	59.00	44.20
SSC-CAE	35.87	18.10	13.46
LRR-CAE	34.48	25.41	10.33
KSSC-CAE	38.17	19.73	14.74
DSC-Net	60.62	61.71	48.20
k-SCN	63.78	62.04	48.04
Ours	**72.14**	**68.60**	**59.17**

Table 6.2: Clustering results of baseline methods and our collaborative scheme on the Fashion-MNIST dataset. For all the metrics, the larger value is better. The best results are shown in bold.

	ACC (%)	NMI (%)	ARI (%)
DEC	22.89	12.10	3.62
DCN	21.30	8.40	3.14
DAC	23.10	9.80	6.15
InfoGAN	19.76	8.15	3.79
SSC-CAE	12.66	0.73	0.19
LRR-CAE	22.35	**17.36**	4.04
KSSC-CAE	26.84	15.17	7.48
DSC-Net	26.87	14.56	**8.75**
k-SCN	22.91	16.57	7.27
Ours	**27.5**	13.78	7.69

Table 6.3: Clustering results of baseline methods and our collaborative scheme on the Stanford product dataset.

6.5 Summary

In this chapter, we have introduced a novel collaborative learning for unsupervised subspace clustering. To this end, we have made use of the complementary property of the classifier-induced affinities and the subspace-based affinities to train a deep model. Our network can be trained in stochastic manner and can directly predict the clustering labels (once trained) without the need of performing spectral clustering. The experiments have shown that the proposed method outperforms the-state-of-art algorithms by a large margin on image clustering tasks, validating the proposed framework.

Chapter 7

Deep Unsupervised Saliency Detection: A Multiple Noisy Labeling Perspective

After introducing several chapters on subspace clustering, we bring the clustering concept on saliency detection. It is natural to connect the subspace clustering and saliency detection problem since the process of pursing the affinity matrix is quite similar to highlight the foreground region from the background. Both of these two algorithms aim to distinguish whether two samples belong to the same class, the difference is that saliency detection is applied on each pixel and subspace clustering build affinity matrix for each image. Along with different salient priors, we design a mechanism to build our framework which allows the neural network to be able to refine the prior information and improve the performance.

7.1 Introduction

Saliency detection aims at identifying the visually interesting objects in images that are consistent with human perception, which is intrinsic to various vision tasks such as context-aware image editing Zhang et al. [2009], image caption generation Xu et al. [2015]. Depending on whether human annotations have been used, saliency detection methods can be roughly divided as: unsupervised methods and supervised methods. The former ones compute saliency directly based on various priors (e.g., , center prior Goferman et al. [2012], global contrast prior Cheng et al. [2011], background connectivity prior Zhu et al. [2014] and etc.), which are summarized and described with human knowledge. The later ones learn direct mapping from color images to saliency maps by exploiting the availability of large-scale human annotated database.

Prior to the deep learning revolution, saliency methods mainly relied on different priors and handcrafted features Zhu et al. [2014]; Cheng et al. [2013b, 2011]; Goferman et al. [2012]. We refer interested readers to Borji et al. [2014] and Borji et al. [2015] for surveys and benchmark comparisons. Color contrast prior has been exploited at superpixel level in Cheng et al. [2011]. Shen and Wu Shen and Wu [2012] formulated saliency detection as a low-rank matrix decomposition problem by exploiting the sparsity prior for salient objects. Objectness, which highlights the object-like regions, has also been used in Jiang et al. [2013c] to mark the regions that have higher possibilities of being an object. Zhu *et al*. Zhu et al. [2014] presented a robust

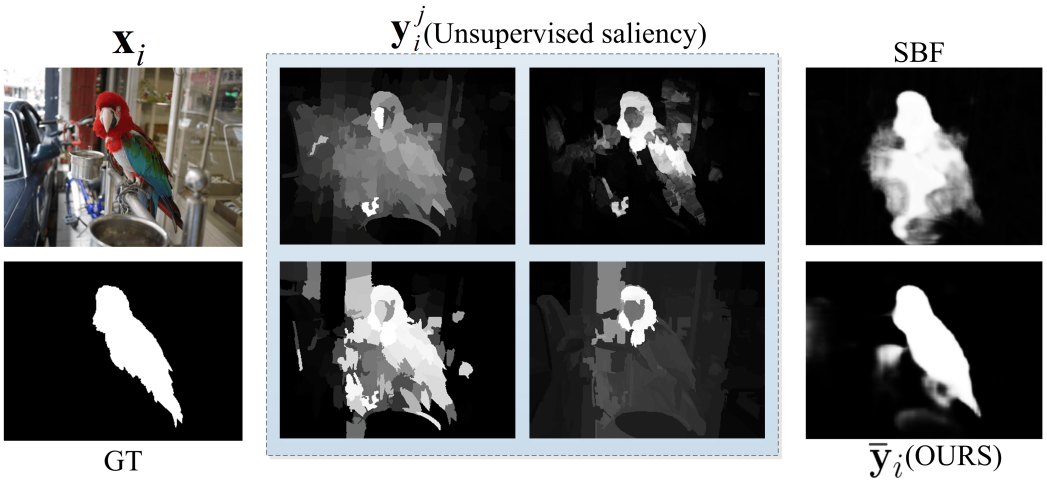


Figure 7.1: Unsupervised saliency learning from weak "noisy" saliency maps. Given an input image $\mathbf{x}_i$ and its corresponding unsupervised saliency maps $\mathbf{y}_i^j$, our framework learns the latent saliency map $\bar{\mathbf{y}}_i$ by jointly optimizing the saliency prediction module and the noise modeling module. Compared with SBF Zhang et al. [2017a] which also learns from unsupervised saliency but with different strategy, our model achieves better performance.

background measure, namely "boundary connectivity" along with an optimization framework to measure backgroundness of each superpixel. Building upon the center prior, Goferman et al. [2012] detects the image regions that represent the scene, especially those that are near image center.

In this chapter, we present a novel end-to-end deep learning framework for saliency detection that is free from human annotations, thus "unsupervised" (see Fig. 7.1 for a visualization). Our framework is built upon existing efficient and effective unsupervised saliency methods and the powerful capacity of deep neural network. The unsupervised saliency methods are formulated with human knowledge and different unsupervised saliency methods exploit different human designed priors for saliency detection. They are noisy (compared with ground truth human annotations) and could have method-specific bias in predicting saliency maps. By utilizing existing unsupervised saliency maps, we are able to remove the need of labor-intensive human annotations, also by jointly learn different priors from multiple unsupervised saliency methods, we are able to get complementary information of those unsupervised saliency.

To effectively leverage these noisy but informative saliency maps, we propose a novel perspective to the problem: *Instead of removing the noise in saliency labeling from unsupervised saliency methods with different fusion strategies Zhang et al. [2017a], we explicitly model the noise in saliency maps*. As illustrated in Fig. 7.2, our framework consists of two consecutive modules, namely a saliency prediction module that learns the mapping from a color image to the "latent" saliency map based on current noise estimation and the noisy saliency maps, and a noise modeling module that fits the noise in noisy saliency maps and updates the noise estimation in different saliency maps based on updated saliency prediction and the noisy saliency maps. In this way, our method takes advantages of both probabilistic methods and deterministic methods, where the latent saliency prediction module works in a deterministic way while the noise modeling module fits the noise distribution in a probabilistic manner. Experiments

suggest that our strategy is very effective and it only takes several rounds till convergence. Note that an epoch means a complete pass through all the training data, an iteration means a complete pass through a batch, and a round means an update on noise module.

7.2 Our Framework

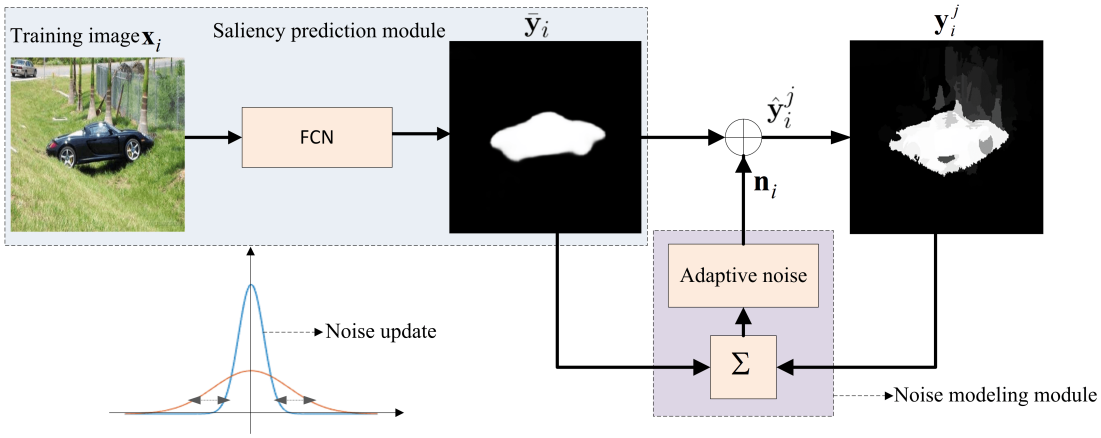


Figure 7.2: Conceptual illustration of our saliency detection framework, which consists of a "latent" saliency prediction module and a noise modeling module. Given an input image, noisy saliency maps are generated by handcrafted feature based unsupervised saliency detection methods. Our framework jointly optimizes both modules under a unified loss function. The saliency prediction module targets at learning latent saliency maps based on current noise estimation and the noisy saliency maps. The noise modeling module updates the noise estimation in different saliency maps based on updated saliency prediction and the noisy saliency maps. In our experiments, the overall optimization converges in several rounds.

Targeting at achieving deep saliency detection without human annotations, we propose an end-to-end noise model integrated deep framework, which builds upon existing efficient and effective unsupervised saliency detection methods and the powerful capacity of deep neural networks.

Given a color image $\mathbf{x}_i$, we would like to learn a better saliency map from its M noisy saliency maps $\mathbf{y}_i^j, j = 1, \cdots, M$ using different unsupervised saliency methods Yan et al. [2013a]; Jiang et al. [2013a]; Li et al. [2013]; Zhu et al. [2014]. A trivial and direct solution would be using the noisy saliency maps as "proxy" human annotations and train a deep model with these noisy saliency maps as supervision. However, it is well-known that the network training is highly prone to the noise in supervision signals. A simple fusion of the multiple labels (training with averaging, treating as multiple labels) will also not work due to the strong inconsistency between labels. While there could be many other potentials in utilizing the noisy saliency maps, they are all based on human-designed pipelines, thus cannot effectively exploit the best manner. Instead, we propose a principled way to infer the saliency maps from using multiple noisy labels and simultaneously estimate the noise.

7.2.1 Joint Saliency Prediction and Noise Modeling

By contrast to existing manually designed procedures and deep learning based pipeline Zhang et al. [2017a], we propose a new perspective toward the problem of learning from unsupervised saliency. As illustrated in Fig. 7.2, our framework consists of two consecutive modules, namely a saliency prediction module that learns the mapping from a color image to the "latent" saliency map, and a noise modeling module that fits the noise. These two modules work collaboratively toward fitting the noisy saliency maps. By explicitly modeling noise, we are able to train a deep saliency prediction model without any human annotations and thus achieve unsupervised deep saliency detection.

7.2.2 Loss Function

We start with a set of training images, denoted as $\mathbf{X} = \{\mathbf{x}_i, i = 1, \ldots, N\}$ and a set of M different saliency maps of these images, denoted as $\mathbf{Y} = \{\mathbf{y}_i^j, i = 1, \ldots, N; j = 1, \ldots, M\}$, where N is number of training images. These are precomputed by applying M different handcrafted "labellers". Throughout this discussion, i indexes the training image and j indexes the handcrafted labeller. We propose a neural network with parameter Θ for saliency detection, which computes a saliency map $\bar{\mathbf{y}}_i = f(\mathbf{x}_i, \Theta)$ of each image. Our idea is to model each of the handcrafted labellers as the sum of $\bar{\mathbf{y}}_i$ plus noise: $\mathbf{y}_i^j = \bar{\mathbf{y}}_i + \mathbf{n}_i^j$, where $\mathbf{n}_i^j$ is a sample chosen from some probability ("noise") distribution q_i, which is to be estimated. For simplicity in this work, it is assumed that the distribution q depends on $\mathbf{x}_i$, and not on the labeller j[1]. We assume a simple model for the noise distributions q_i, namely that it a zero-mean Gaussian, independent for each pixel of each image $\mathbf{x}_i$. Thus, the total distribution $\mathbf{q} = \{q_1, q_2, \ldots, q_N\}$ is assumed independent for all i and pixel (m, n), and is parametrized by a parameter set $\Sigma = \{\sigma_{mn}^i\}$, where i indexes the training image and (m, n) are pixel coordinates. Sometimes, distribution $\mathbf{q}$ will be denoted as $\mathbf{q}(\Sigma)$ to emphasize the role of the parameters Σ. With this simple parameterization it is easy to generate noise samples $\mathbf{n}_i^j$ for any i and j.

Given Θ, Σ, and an input image $\mathbf{x}_i$, one generates saliency map $\hat{\mathbf{y}}_i^j$ according to:

$$\hat{\mathbf{y}}_i^j = f(\mathbf{x}_i; \Theta) + \mathbf{n}_i^j = \bar{\mathbf{y}}_i + \mathbf{n}_i^j, \tag{7.1}$$

where each $\mathbf{n}_i^j$ is a sample drawn from distribution $q_i(\Sigma)$. In the training process, the parameters Θ of the network and Σ of the noise model are updated to minimize an appropriate loss function. The loss function has two parts:

$$\mathcal{L}(\Theta, \Sigma) = \mathcal{L}_{\text{pred}}(\Theta, \Sigma) + \lambda \mathcal{L}_{\text{noise}}(\Theta, \Sigma), \tag{7.2}$$

where λ is the regularizer to balance these two terms. Under our optimization framework, increasing the variance in noise modeling will make the prediction loss $\mathcal{L}_{\text{pred}}$ large and decrease the $\mathcal{L}_{\text{noise}}$. Meanwhile, keeping the variance lower will decrease the cross-entropy loss $\mathcal{L}_{\text{pred}}$ but increase $\mathcal{L}_{\text{noise}}$. Thus our model balances between these two losses and converges to the state minimizing the overall loss. These two losses are described below:

[1] Assuming that distribution q is also dependent on the labeller j was observed not to improve results

Saliency Prediction: For the latent saliency prediction module, we use a fully convolutional neural network (FCN) due to its superior capability in feature learning and feature representation. We use the conventional cross-entropy loss and compute the loss function element-wisely across the whole training images.

The predictive loss $\mathcal{L}_{\text{Pred}}$ is designed to measure the agreement of the predicted labellings $\hat{\mathbf{y}}_i^j$ with handcrafted labellings $\mathbf{y}_i^j$. Cross-entropy loss is used for this purpose, and the cross-entropy loss between modeled value $\hat{y}$ and "ground truth" value y (noisy label) is given by:

$$L_{\text{CE}} = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y})). \tag{7.3}$$

This is applied to all pixel (m,n), all labellers j and all the test images $\mathbf{x}_i$ to give the total prediction loss.

$$\mathcal{L}_{\text{pred}}(\Theta, \mathbf{\Sigma}) = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{m,n} L_{\text{CE}}(\mathbf{y}_{i,mn}^j, \hat{\mathbf{y}}_{i,mn}^j), \tag{7.4}$$

where $\hat{\mathbf{y}}_{i,mn}^j$ is our noisy saliency map prediction at pixel (m,n) which can be easily computed by (7.1) element-wise, and $\hat{\mathbf{y}}_{i,mn}^j$ is truncated to lie in the range of $[0,1]$.

Noise Modeling To effectively handle noisy saliency maps from different unsupervised saliency map labelers, we build a probabilistic model to approximate the noise, and connect it with our deterministic part (latent saliency prediction model as shown in Fig. 7.2). In this way, our entire model can be trained in an end-to-end manner to minimize the overall loss function Eq. (7.2).

The noise loss $\mathcal{L}_{\text{noise}}$ measures (for each training image $\mathbf{x}_i$) the agreement of the noise distribution $q_i(\mathbf{\Sigma})$ with the empirical variance of the measurements $\mathbf{y}_i^j$ with respect to the output $\bar{\mathbf{y}}_i = f(\mathbf{x}_i;\Theta)$ of the network. More precisely, given an input $\mathbf{x}_i$, define $\hat{\mathbf{n}}_i^j = \mathbf{y}_i^j - \bar{\mathbf{y}}_i$, the empirical error of each $\mathbf{y}_i^j$ with respect to the network prediction. For each pixel location (m,n), this provides M samples from a zero-mean Gaussian probability distribution p_i, and its variance on every pixel can be written as $\hat{\sigma}_{i,mn}$. The complete set of parameters for p_i is denoted as $\hat{\mathbf{\Sigma}} = \{\hat{\sigma}_{i,mn}\}$.

Since it is intractable to estimate the true posterior distribution of $\hat{\mathbf{n}}_i^j$, thus we propose to approximate it by sequentially optimizing the parameters of prior. We assume that the noise is generated by some random process, involving an unobserved continuous random variable set $\mathbf{\Sigma}$. From an encoder perspective, the unobserved variable $\mathbf{n}$ can be interpreted as a latent representation. Here, we model $\hat{\mathbf{y}}_i^j$ as a probabilistic encoder, since given an image $\mathbf{x}_i$ and network parameters Θ it produces a distribution (e.g., a Gaussian) over possible values of the code $\mathbf{n}$. The process consists of two steps: (1) a noise map $\mathbf{n}_i$ is generated from some prior distribution $q(\mathbf{\Sigma}^*)$; (2) a noise map $\hat{\mathbf{n}}_i^j$ is produced and estimating the corresponding parameter $\hat{\sigma}_i$

The corresponding noise loss is defined to be the KL divergence between distribution p_i

and q_i.

$$\mathcal{L}_{\text{noise}}(\Theta, \Sigma) = \sum_{i}^{N} \mathbf{KL}(q(\mathbf{\Sigma}_i) \| \, p(\hat{\mathbf{\Sigma}}_i)). \tag{7.5}$$

Since we employ the Gaussian distribution as the prior distribution for our noise model, the KL divergence has a closed-form solution as:

$$\mathbf{KL}(q(\mathbf{\alpha}) \| \, p(\hat{\mathbf{\alpha}})) = \log(\hat{\sigma}/\sigma) + \frac{\sigma^2 + (\mu - \hat{\mu})^2}{2\hat{\sigma}^2} - \frac{1}{2}, \tag{7.6}$$

Based on this equation, we can update σ_i^2 for every coordinate (m, n) as

$$(\sigma_i^{t+1})^2 = (\sigma_i^t)^2 + \alpha((\hat{\sigma}_i^t)^2 - (\sigma_i^t)^2), \tag{7.7}$$

by differentiating Eq. (7.6) with respect to $\sigma_{i,mn}^2$, where α is the step size, and we set $\alpha = 0.01$ for training in all experiments.

For different images we have the corresponding noise maps, which follows i.i.d. Gaussian distributions with different variance. Thus, it is hard to converge if simultaneously optimizing the FCN parameters Θ and noise parameters Σ. In order to train the whole network smoothly, we update the parameters of noise module after the prediction loss converges. Noise maps of a given image are sampled from the same distribution in a round, but they are updated in every round. At the first round, we initialize noise variance to be zero, and train the FCN until it converges. Based on the variance of the saliency prediction and noisy labels, we then update the noise variance for each image and retrain the network. Through minimizing the loss function Eq. (7.2) with this procedure, We can train the network and estimate the corresponding noise maps.

7.2.3 Deep Noise Model based Saliency Detector

Network Architecture We build our latent saliency prediction module upon the DeepLab network Chen et al. [2017], where a deep CNN (ResNet-101 He et al. [2016b] in particular) originally designed for image classification is re-purposed by 1) transforming all fully connected layers to convolutional layers and 2) increasing feature resolution through dilated convolution Chen et al. [2017]. Figure 7.2 shows the whole structure of our framework. Specifically, our model takes a rescaled image $\mathbf{x_i}$ of 425×425 as input. For training, the noise model is used to iteratively update saliency prediction $\hat{\mathbf{y}}_i^j$, and it's excluded in testing stage, where the latent saliency prediction output $\bar{\mathbf{y}}_i$ in Fig. 7.2 is our predicted saliency map.

Implementation details: We trained our model using Caffe Jia et al. [2014] with maximum epoch of 20. We initialized our model by using the Deep Residual Model trained for image classification He et al. [2016b]. We used the stochastic gradient descent method with momentum 0.9 and decreased learning rate 90% when the training loss did not decrease. Base learning rate is initialized as 1e-3 with the "poly" decay policy Jia et al. [2014]. For validation, we set "test_iter" as 500 (test batch size 1) to cover the full 500 validation images. The training took 4 hours for one round with training batch size 1 and "iter_size" 20 on a PC with an NVIDIA Quadro M4000 GPU.

7.3 Experimental Results

In this section, we report experimental results on various saliency detection benchmarking datasets.

.

Table 7.1: Performance of mean F-measure (F_β) and MAE for different methods including ours on seven benchmark datasets.

	MSRA-B		ECSSD		DUT		SED2		PASCALS		THUR		SOD	
Methods	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE
BL1	.7905	.0936	.7205	.1444	.5825	.1369	.7773	.1112	.6714	.2206	.5953	.1339	.6306	.1870
BL2	.6909	.1710	.6542	.2170	.4552	.2951	.7232	.1406	.6776	.2409	.5119	.2545	.5928	.2566
BL3	.8879	.0587	.8717	.0772	.7253	.0772	.8520	.0819	.8264	.1525	.7368	.0749	.7922	.1231
OURS	.8770	.0560	.8783	.0704	.7156	.0860	.8380	.0881	.8422	.1391	.7322	.0811	.7976	.1182

Table 7.2: Performance of mean F-measure (F_β) and MAE for different methods including ours on seven benchmark datasets (Best ones in bold). From DSS to DC are deep learning based supervised methods, from DRFI to HS are the handcrafted feature based unsupervised methods, SBF and OURS are deep learning based unsupervised saliency detection methods.

	MSRA-B		ECSSD		DUT		SED2		PASCALS		THUR		SOD	
Methods	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE
DSS&.8941	.0474	.8796	.0699	.7290	.0760	.8236	.1014	.8243	.1546	.7081	.1142	.8048	.1118	
NLDF	.8970	.0478	**.8908**	.0655	**.7360**	.0796	-	-	.8391	.1454	-	-	**.8235**	**.1030**
Amulet	-	-	.8825	**.0607**	.6932	.0976	**.8745**	**.0629**	.8371	**.1292**	.7115	.0937	.7729	.1248
UCF	-	-	.8521	.0797	.6595	.1321	.8444	.0742	.8060	.1492	.6920	.1119	.7429	.1527
SRM	.8506	.0665	.8260	.0922	.6722	.0846	.7447	.1164	.7766	.1696	.6894	.0871	.7246	.1369
DMT	-	-	.7589	.1601	.6045	**.0758**	.7778	.1074	.6657	.2103	.6254	.0854	.6978	.1503
RFCN	-	-	.8426	.0973	.6918	.0945	.7616	.1140	.8064	.1662	.7062	.1003	.7531	.1394
DeepMC	.8966	.0491	.8061	.1019	.6715	.0885	.7660	.1162	.7327	.1928	.6549	.1025	.6862	.1557
MDF	.7780	.1040	.8097	.1081	.6768	.0916	.7658	.1171	.7425	.2069	.6670	.1029	.6377	.1669
DC	**.8973**	**.0467**	.8315	.0906	.6902	.0971	.7840	.1014	.7861	.1614	.6940	.0959	.7603	.1208
DRFI	.7282	.1229	.6440	.1719	.5525	.1496	.7252	.1373	.5745	.2556	.5613	.1471	.5440	.2046
RBD	.7508	.1171	.6518	.1832	.5100	.2011	.7939	.1096	.6581	.2418	.5221	.1936	.5927	.2181
DSR	.7227	.1207	.6387	.1742	.5583	.1374	.7053	.1452	.5785	.2600	.5498	.1408	.5500	.2133
MC	.7165	.1441	.6114	.2037	.5289	.1863	.6619	.1848	.5742	.2719	.5149	.1838	.5332	.2435
HS	.7129	.1609	.6234	.2283	.5205	.2274	.7168	.1869	.5948	.2860	.5157	.2178	.5383	.2729
SBF	-	-	.7870	.0850	.5830	.1350	-	-	.7780	.1669	-	-	.6760	.1400
OURS	.8770	.0560	.8783	.0704	.7156	.0860	.8380	.0881	**.8422**	.1391	**.7322**	**.0811**	.7976	.1182

7.3.1 Setup

Dataset: We evaluated performance of our proposed model on 7 saliency benchmarking datasets. 3,000 images from the MSRA-B datasetLiu et al. [2007] are used to get the noisy labels (where 2,500 images for training and 500 images for validation) and the remaining 2,000 images are kept for testing. Most of the images in MSRA-B dataset only have one salient object. The ECSSD dataset Yan et al. [2013a] contains 1,000 images of semantically meaningful but structurally complex images. The DUT dataset Yang et al. [2013] contains 5,168 images. The SOD saliency dataset Jiang et al. [2013b] contains 300 images, where many images contain multiple salient objects with low contrast. The SED2 Alpert et al. [2012] dataset contains 100 images with each image contains two salient objects. The PASCAL-S Li et al. [2014] dataset is generated from the PASCAL VOC Everingham et al. [2015] dataset and contains

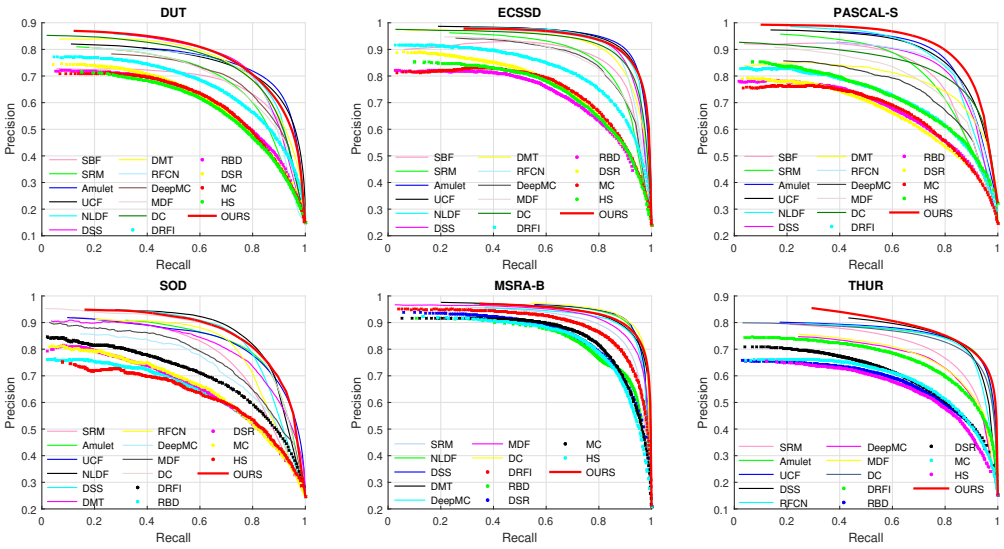


Figure 7.3: PR curves on six benchmark datasets (DUT, ECSSD, PASCAL-S, SOD, MSRA-B, THUR). Best Viewed on Screen.

850 images. The THUR dataset Cheng et al. [2014] contains 6,232 images of five classes, namely "butterfly","coffee mug","dog jump","giraffe" and "plane".

Unsupervised Saliency Methods: In our framework, we learn unsupervised saliency from existing unsupervised saliency detection methods. In our experiment, we choose RBD Zhu et al. [2014], DSR Li et al. [2013], MC Jiang et al. [2013a] and HS Yan et al. [2013a] due to their effectiveness and efficiency as illustrated in Borji et al. [2015].

Competing methods: We compared our method against 10 state-of-the-art deep saliency detection methods (with clean labels): DSS Hou et al. [2017], NLDF Luo et al. [2017], Amulet Zhang et al. [2017c], UCF Zhang et al. [2017d], SRM Zhang et al. [2017a], DMT Li et al. [2016], RFCN Wang et al. [2016], DeepMC Zhao et al. [2015], MDF Li and Yu [2015] and DC Li and Yu [2016], 5 conventional handcrafted feature based saliency detection methods: DRFI Jiang et al. [2013b], RBD Zhu et al. [2014], DSR Li et al. [2013], MC Jiang et al. [2013a], and HS Yan et al. [2013a], which were proven in Borji et al. [2015] as the state-of-the-art methods before the deep learning revolution, and the very recent unsupervised deep saliency detection method SBF Zhang et al. [2017a].

Evaluation metrics: We use 3 evaluation metrics, including the mean absolute error (MAE), F-measure, as well as the Precision-Recall (PR) curve. MAE can provide a better estimate of the dissimilarity between the estimated and ground truth saliency map. It is the average per-pixel difference between the ground truth and the estimated saliency map, normalized to [0, 1], which is defined as:

$$MAE = \frac{1}{W \times H} \sum_{x=1}^{W} \sum_{y=1}^{H} |S(x,y) - GT(x,y)|, \tag{7.8}$$

where W and H are the width and height of the respective saliency map S, GT is the ground

truth saliency map.

The F-measure (F_β) is defined as the weighted harmonic mean of precision and recall:

$$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 Precision + Recall}, \tag{7.9}$$

where $\beta^2 = 0.3$, *Precision* corresponds to the percentage of salient pixels being correctly detected, *Recall* is the fraction of detected salient pixels in relation to the ground truth number of salient pixels. The PR curves are obtained by thresholding the saliency map in the range of [0, 255].

7.3.2 Baseline Experiments

As there could be different ways to utilize the multiple noisy saliency maps, and for fair comparisons with straightforward solutions for our task, we run the following three baseline methods and the results are reported in Table 7.1.

Baseline 1: using noisy unsupervised saliency as pseudo ground truth: For a given input image $\mathbf{x}_i$ and its M handcrafted feature based saliency map $\mathbf{y}_i^j, j = 1, ..., M$, we get M image pairs with noisy label $\{\mathbf{x}_i, \mathbf{y}_i^j, j = 1, ..., M)$. Then we train a deep model He et al. [2016b] based on those noisy labels directly, and the results are shown as "BL1" in Table 7.1.

Baseline 2: using averaged unsupervised saliency as pseudo ground truth: Instead of using all the four unsupervised saliency as ground truth, we use the averaged saliency map of those unsupervised saliency as pseudo ground truth, and trained another baseline model "BL2" in Table 7.1.

Baseline 3: supervised learning with ground truth supervision: Our proposed framework consists of the saliency prediction module and the noise modeling module to effectively leverage the noisy saliency maps. To illustrate the best performance our model can achieve as well as to provide a baseline comparison for our framework, we train our latent saliency module directly with clean labels, which naturally gives an upper bound of the saliency detection performance. The results "BL3" are reported in Table 7.1.

Analysis: In Table 7.1, we compare our unsupervised saliency method with the above baseline configurations. Our method clearly outperforms both BL1 and BL2 with a wide margin, demonstrating the superiority of our end-to-end learning framework. As illustrated in Table 7.1, the performance of BL1 is better than the performance of BL2. This is because: 1) For BL1, we have 12,000 training image pairs (four unsupervised saliency methods), while for BL2, we have 3,000 averaged noisy labels; 2) as those unsupervised saliency methods tend to prefer different priors for saliency detection, and their saliency maps can be complementary or controversial to some extent. Simply averaging those saliency maps results in even worse proxy saliency map supervision. Compared with BL3, which is trained with ground truth clean labels and without noise, our unsupervised method achieves highly comparable results. This demonstrates that by jointly learning the latent saliency maps and modeling the noise in a unified framework, we are able to learn the desired reliable saliency maps even without any human annotations.

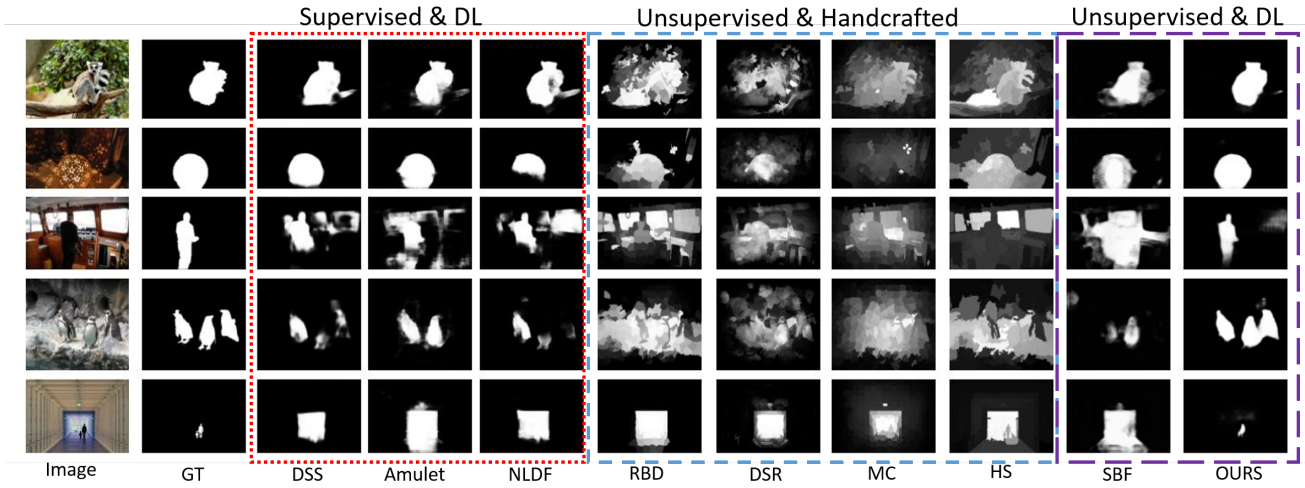


Figure 7.4: Visual comparison between our method and other competing methods.

7.3.3 Comparison with the State-of-the-art

Quantitative Comparison We compared our method with eleven most resent deep saliency methods and five conventional methods. Results are reported in Table 7.2 and Fig. 7.3, where "OURS" represents the results of our model. Table 7.2 shows that on those seven benchmark datasets, deep supervised methods significantly outperform traditional methods with 2%-12% decrease in MAE, which further proves the superiority of deep saliency detection.

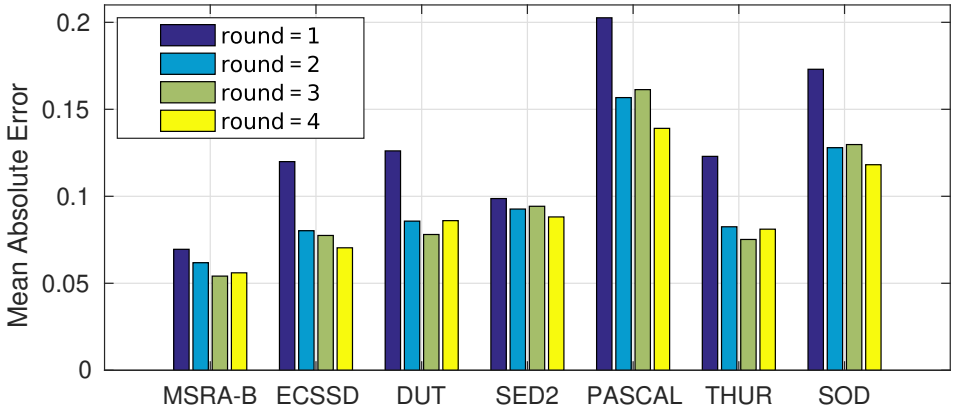
MSRA-B is a relatively simple dataset, where most salient objects dominate the whole image. The most recent deep supervised saliency methods Hou et al. [2017] Luo et al. [2017] Zhang et al. [2017c] can achieve the highest mean F-measure of 0.8970, and our unsupervised method without human annotations can achieve a mean F-measure of 0.8770, which is only a slight worse. The DUT dataset has more than 25% of images with saliency occupation less than 4%. Small salient object detection is quite challenging which increase the difficulty of this dataset. We achieve the third highest mean F-measure compared with all the competing methods. The THUR dataset is the largest dataset we used in the experiments, and most of the images have complex background. The state-of-the-art competing method achieves a mean F-measure/MAE as 0.7115/0.0854, while our method achieves the best mean F-measure and MAE as 0.7322/0.0811. SBF Zhang et al. [2017a] uses inter- and intra-image confidence map as pseudo ground truth to train an unsupervised deep model based on unsupervised saliency, which is quite different from our formulation of predicting saliency from unsupervised saliency as learning from noisy labels. Table 7.2 shows that our framework leads to better performance, with 10% mean F-measure improvement and 3% decrease of MAE on average. Fig. 7.3 shows comparison between PR curves of our method and the competing methods on four benchmarking datasets. For the PASCAL-S and THUR dataset, our method ranks almost the 1st, and for the other three datasets, our method achieves competitive performance compared with the competing deep supervised methods. These experiments altogether proves the effectiveness our proposed unsupervised saliency detection framework.

Qualitative Comparison Figure 7.4 demonstrates several visual comparisons, where our method consistently outperforms the competing methods, especially those four unsupervised saliency we used to train our model. The first image is a simple scenario, and most of the competing methods can achieve good results, while our method achieves the best result with most of the background region suppressed. Background of the third image is very complex, and all the competing methods fail to detect salient object. With proper noisy labels, we achieve the best results compared with both unsupervised saliency methods and deep saliency methods. The fourth image is in very low-contrast, where most of the competing methods failed to capture the whole salient objects with the last penguin mis-detected, especially for those unsupervised saliency methods. Our method captures all the three penguins properly. The salient objects in the last row are quite small, and the competing methods failed to capture salient regions, while our method capture the whole salient region with high precision.

Ablation Studies: In this chapter, we propose to iteratively update the noise modeling module and the latent saliency prediction model to achieve accurate saliency detection. As the two modules work collaboratively to optimize the overall loss function, it is interesting to see how the saliency prediction results evolves with respect to the increase of updating round. In Fig. 7.5, we illustrate both the performance metric (MAE) with respect to updating round and an example saliency detection results. Starting with the zero noise initialization, our method consistently improves the performance of saliency detection with the updating of noise modeling. Also, only after several updating rounds, our method convergences to desired state as shown in Fig. 7.5.

7.4 Summary

In this chapter, we propose an end-to-end saliency learning framework without the need of human annotated saliency maps in network training. We represent unsupervised saliency learning as learning from multiple noisy saliency maps generated by various efficient and effective conventional unsupervised saliency detection methods. Our framework consists of a latent saliency prediction module and an explicit noise modeling models, which work collaboratively. Extensive experimental results on various benchmarking datasets prove the superiority of our method, which not only outperforms traditional unsupervised methods with a wide margin but also achieves highly comparable performance with current state-of-the-art deep supervised saliency detection methods. In the future, we plan to investigate the challenging scenarios of multiple saliency object detection and small salient object detection under our framework. Extending our framework to dense prediction tasks such as semantic segmentation Lu et al. [2017] and monocular depth estimation Li et al. [2015] could be interesting directions.



(a) MAE of each round on 7 datasets

(b) Input (c) GT (d) 1st

(e) 2nd (f) 3rd (g) 4th

Figure 7.5: Performance of each round. Top: MAE of each dataset. Bottom: an example image, ground-truth and intermedia results generated by each updating round.

Chapter 8

Conclusion and Future Work

Unsupervised learning is one of the most promising and difficult topics in computer vision and machine learning, which is applied and benefits a broad range of applications. The study of unsupervised learning has been becoming more popular in recent years since the intensive human labeling and a large amount of data are paramount for training deep learning models. However, there is still a big gap between current learning methods and the way of human learning, unsupervised learning is gaining more popular nowadays as many challenging problems yet to be solved. In this concluding chapter, we summarize the research contributions of this dissertation and discuss the important directions of future work.

8.1 Contributions

This thesis contributes new ideas and frameworks to unsupervised learning.

1. We present a computationally efficient framework that employs multi-resolution residual maps for dictionary learning and sparse coding in order to address the shortcomings of previous methods. It allows dictionary atoms to access larger context, at the same time keeps descriptive capacity in Chapter 3;

2. Chapter 4 presents the first framework for deep subspace clustering, it opens a new era for subspace clustering and boosts the clustering accuracy.

3. Chapter 5 and chapter 6 present two different ways of bypassing the process of building an affinity matrix and applying spectral clustering to make the subspace clustering scalable to large datasets.

4. Chapter 7 builds a mechanism to connect deep clustering and saliency detection, it extends the unsupervised learning to dense prediction. Below, we summarize the contributions in each chapter.

Chapter 3. We build a two-pass dictionary learning and sparse coding algorithm. The computational efficiency stems from encoding at the coarsest resolution and encoding the residuals that are significantly sparse. This enables our cascade to go as deep as needed without any compromise. All the atoms in a dictionary have the same dimensionality while their receptive fields vary depending on the layer. Compared to existing multi-scale approaches operating

indiscriminately on image pyramids or wavelets, our dictionary comprises atoms that adapt to the information available at each layer. The details learned from residual images progressively refine our reconstruction objective. This allows our method to generate a flexible image representation using a much smaller number of coefficients. Our extensive experiments demonstrate that our method applies favorably in image coding, denoising, inpainting and artifact removal tasks.

Chapter 4. We introduce a novel deep neural network architecture to learn (in an unsupervised manner) an explicit non-linear mapping of the data that is well-adapted to subspace clustering. To this end, we build our deep subspace clustering networks (*DSC-Nets*) upon deep auto-encoders, which non-linearly map the data points to a latent space through a series of encoder layers. Our key contribution then consists of introducing a novel *self-expressive* layer – a fully connected layer without bias and non-linear activations – at the junction between the encoder and the decoder. This layer encodes the "self-expressiveness" property Rao et al. [2008]; Elhamifar and Vidal [2009] of data drawn from a union of subspaces, that is, the fact that each data sample can be represented as a linear combination of other samples in the same subspace. To the best of our knowledge, our approach constitutes the first attempt to directly learn the affinities (through combination coefficients) between all data points within one neural network. Furthermore, we propose effective pre-training and fine-tuning strategies to learn the parameters of our DSC-Nets in an unsupervised manner and with a limited amount of data.

Chapter 5. Our contributions in this chapter are three-folds: 1. We bypass the steps of constructing an affinity matrix and performing spectral clustering, which has been used in mainstream subspace clustering algorithms and accelerates the computation by using a variant of k-subspace clustering. As a result, our method can handle datasets that are orders of magnitudes larger than those considered in traditional methods. 2. In order to address non-linearity, we equip deep neural networks with subspace priors. This in return enables us to learn an explicit non-linear mapping of the data that is well-suited for subspace clustering. 3. We propose novel strategies to update subspace bases. When the size of the dataset at hand is manageable, we update subspaces in closed-form using Singular Value Decomposition (SVD) with a simple mechanism to rule out outliers. For large datasets, we update subspaces by making use of the stochastic optimization methods on the Grassmann manifolds.

Empirically, evaluations on relatively large datasets such as MNIST and Fashion-MNIST dataset show that our proposed method achieves the state-of-the-art results in terms of clustering accuracies and speed.

Chapter 6. In this chapter, we continue working on how to extend the scalability of deep subspace models. Unlike Chapter 5, which explicitly build the subspace for each cluster, we propose a neural structure to improve the performance of subspace clustering while being mindful to the scalability issue. To this end, we first formulate subspace clustering as a classification problem, which in turn removes the spectral clustering step from the computations. Our neural model is comprised of two modules, one for classification and one for affinity learning. Both modules collaborate during learning, hence the name "Neural Collaborative Subspace Clustering". During training and in each iteration, we use the affinity matrix generated by the subspace self-expressiveness to supervise the affinity matrix computed from the classification part. Concurrently, we make use of the classification part to improve self-expressiveness to build a better affinity matrix through collaborative optimization.We evaluate our algorithm

on three datasets, namely MNIST, Fashion-MNIST and the Stanford Online Products dataset which exhibit different levels of difficulty. Our empirical study shows the superiority of the proposed algorithm over several state-of-the-art baselines including deep subspace clustering techniques.

Chapter 7. In this chapter, we bridge the clustering problem and saliency detection problem. We present a novel perspective to unsupervised deep saliency detection and learn saliency maps from multiple noisy unsupervised saliency methods. We formulate the problem as a joint optimization of a latent saliency prediction module and a noise modeling module. Meanwhile, our deep saliency model is trained in an end-to-end manner without using any human annotations, leading to an extremely cheap solution. Extensive performance evaluation on seven benchmarking datasets show that our framework outperforms existing unsupervised methods with a wide margin while achieving comparable results with state-of-the-art deep supervised saliency detection methods.

We have addressed the unsupervised learning problem from shallow to deep structure and apply it in different applications. In general, the Chapter 3 extend single-layer linear dictionary learning to multi-layer cascaded nonlinear dictionary learning. Inspired by Chapter 4, we also extend the non-linear feature learning idea to subspace clustering and achieve very promising results. Chapter 5 and Chapter 6 show that it is possible to apply subspace clustering to large dataset especially when the data volume is growing exponentially. Chapter 7 is a good example to connect unsupervised learning to the supervised task to remove human labeling.

8.2 Future Work

After finishing all of these works, we definitely would like to continue our contribution in this area. Besides, there are many difficulties for the community and us to employ unsupervised learning methods to wide applications. We conclude this dissertation with some suggestions for future research.

Subspace clustering with complicated datasets. The assumption of subspace clustering is based on the self-expressiveness or the distance from a sample to its corresponding subspace. However, when it comes to complicated images such as ImageNet, Cifar10, where an image not only contains the object itself but also abuntdant texture information, the assumption will collapse. Therefore, it will be a breakthrough in the subspace clustering community to find a better cost function to describe the affinity instead of self-expressiveness or distance-based. On the other hand, it is also possible to find a method to extract good feature which only contains the information of corresponding subspace information. Representation learning frameworks such as Deep InfoMax Hjelm et al. [2018] could be good candidates to combine with subspace clustering.

Subspace in other fields. Subspace is not only widely used in clustering but also in other fields, such as action recognition and anomaly detection Wang and Cherian [2019], adversarial attacking Ma et al. [2018] and few-shot learning Motiian et al. [2017]. Compared to the one-hot vector style representation, the subspace has more freedom degree to train, which increases the difficulty of training but brings more possibilities. The most popular discriminators are all linear classifier, as a result, the high dimensional boundaries are subspaces. Hence, it is worth

working on how subspace affects the classification problem, for example, Tramèr et al. [2017] pointed out that adversarial subspaces with higher dimensionality are more likely to intersect.

Dictionary learning and Neural Networks. Dictionary learning and sparse coding were the most popular framework before the dominance of CNNs. There are very few works combines these two frameworks. Although CNNs are the deeper and nonlinear version of dictionary learning, dictionary learning has its own strengths. CNNs update their parameters through backpropagation, which leads CNNs to neglect the class with fewer samples. When the number of samples in each class is not even, the neural network will fail to recognize samples from a smaller group, which causes the "long-tail" problem Liu et al. [2019]. Because of the different updating mechanisms, dictionary learning treats the sample from every class equally. It would be promising to employ dictionary learning as a memory cell to help CNNs work better in these scenarios.

Bibliography

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; ET AL., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, (2016). (cited on pages 58, 68, and 82)

AGARWAL, P. K. AND MUSTAFA, N. H., 2004. K-means projective clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 155–165. ACM. (cited on pages 8 and 63)

AHARON, M.; ELAD, M.; AND BRUCKSTEIN, A., 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54, 11 (2006), 4311–4322. doi:10.1109/TSP.2006.881199. (cited on pages 2, 15, 39, and 45)

AHMED, N.; NATARAJAN, T.; AND RAO, K. R., 1974. Discrete cosine transform. *IEEE transactions on Computers*, 100, 1 (1974), 90–93. (cited on page 15)

ALPERT, S.; GALUN, M.; BRANDT, A.; AND BASRI, R., 2012. Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34, 2 (Feb 2012), 315–327. doi:10.1109/TPAMI.2011.130. (cited on page 95)

ASIF, M. S. AND ROMBERG, J., 2013. Fast and accurate algorithms for re-weighted ℓ_1-norm minimization. *IEEE Transactions on Signal Processing*, 61, 23 (2013), 5905–5916. (cited on page 19)

BALDI, P. AND HORNIK, K., 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2, 1 (1989), 53–58. (cited on page 21)

BAO, C.; JI, H.; QUAN, Y.; AND SHEN, Z., 2016. Dictionary learning for sparse coding: Algorithms and convergence analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38, 7 (2016), 1356–1369. (cited on page 44)

BASRI, R. AND JACOBS, D. W., 2003. Lambertian reflectance and linear subspaces. *TPAMI*, 25, 2 (2003), 218–233. (cited on page 25)

BECK, A. AND TEBOULLE, M., 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2, 1 (2009), 183–202. (cited on page 20)

BENGIO, Y.; LAMBLIN, P.; POPOVICI, D.; AND LAROCHELLE, H., 2007. Greedy layer-wise training of deep networks. In *NIPS*, 153–160. (cited on pages 23 and 69)

BORJI, A.; CHENG, M.; HOU, Q.; JIANG, H.; AND LI, J., 2014. Salient object detection: A survey. *CoRR*, abs/1411.5878 (2014). (cited on pages 32 and 89)

BORJI, A.; CHENG, M.; JIANG, H.; AND LI, J., 2015. Salient object detection: A benchmark. *IEEE Trans. on Image Processing*, 24, 12 (2015), 5706–5722. doi:10.1109/TIP.2015.2487833. (cited on pages 32, 89, and 96)

BORJI, A.; FRINTROP, S.; SIHITE, D. N.; AND ITTI, L., 2012. Adaptive object tracking by learning background context. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 23–30. IEEE. (cited on page 31)

BORJI, A. AND ITTI, L., 2011. Scene classification with a sparse set of salient regions. In *2011 IEEE International Conference on Robotics and Automation*, 1902–1908. IEEE. (cited on page 31)

BOUREAU, Y.-L.; CUN, Y. L.; ET AL., 2008. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, 1185–1192. (cited on pages 2 and 22)

BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; AND ECKSTEIN, J., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3, 1 (2011), 1–122. (cited on page 20)

BRADLEY, P. S. AND MANGASARIAN, O. L., 2000. K-plane clustering. *Journal of Global Optimization*, 16, 1 (2000), 23–32. (cited on pages 8, 63, and 76)

BRIGHAM, E. O. AND BRIGHAM, E. O., 1988. *The fast Fourier transform and its applications*, vol. 448. prentice Hall Englewood Cliffs, NJ. (cited on page 15)

BURT, P. J. AND ADELSON, E. H., 1983. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31, 4 (1983), 532–540. doi:10.1109/TCOM.1983.1095851. (cited on page 20)

CAI, D.; HE, X.; HAN, J.; AND HUANG, T., 2011. Graph regularized nonnegative matrix factorization for data representation. *TPAMI*, 33, 8 (2011), 1548–1560. (cited on page 60)

CAI, D.; HE, X.; HU, Y.; HAN, J.; AND HUANG, T., 2007. Learning a spatially smooth subspace for face recognition. In *CVPR*, 1–7. IEEE. (cited on page 59)

CANDES, E. AND TAO, T., 2005. Decoding by linear programming. *arXiv preprint math/0502327*, (2005). (cited on page 14)

CANDES, E. J. AND DONOHO, D. L., 2000. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, DTIC Document. (cited on page 20)

CANDES, E. J.; ROMBERG, J. K.; AND TAO, T., 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59, 8 (2006), 1207–1223. (cited on page 14)

CHANG, J.; WANG, L.; MENG, G.; XIANG, S.; AND PAN, C., 2017. Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 5880–5888. IEEE. (cited on page 82)

CHEN, G.; ATEV, S.; AND LERMAN, G., 2009. Kernel spectral curvature clustering (KSCC). In *ICCV Workshops*, 765–772. IEEE. (cited on pages 53 and 76)

CHEN, G. AND LERMAN, G., 2009. Spectral curvature clustering (SCC). *IJCV*, 81, 3 (2009), 317–330. (cited on pages 26, 53, and 76)

CHEN, L. C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; AND YUILLE, A. L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PP, 99 (2017), 1–1. doi:10.1109/TPAMI.2017.2699184. (cited on page 94)

CHEN, S. S.; DONOHO, D. L.; AND SAUNDERS, M. A., 2001. Atomic decomposition by basis pursuit. *SIAM review*, 43, 1 (2001), 129–159. (cited on page 20)

CHEN, X.; DUAN, Y.; HOUTHOOFT, R.; SCHULMAN, J.; SUTSKEVER, I.; AND ABBEEL, P., 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2172–2180. (cited on page 83)

CHENG, M.; MITRA, N. J.; HUANG, X.; AND HU, S., 2014. Salientshape: group saliency in image collections. *The Visual Computer*, 30, 4 (2014), 443–453. doi:10.1007/s00371-013-0867-4. (cited on page 96)

CHENG, M.; ZHANG, G.; MITRA, N.; HUANG, X.; AND HU, S.-M., 2011. Global contrast based salient region detection. In *CVPR*, 409–416. doi:10.1109/CVPR.2011.5995344. (cited on pages 6, 31, 32, and 89)

CHENG, M.-M.; WARRELL, J.; LIN, W.-Y.; ZHENG, S.; VINEET, V.; AND CROOK, N., 2013a. Efficient salient region detection with soft image abstraction. In *Proceedings of the IEEE International Conference on Computer vision*, 1529–1536. (cited on page 32)

CHENG, M.-M.; WARRELL, J.; LIN, W.-Y.; ZHENG, S.; VINEET, V.; AND CROOK, N., 2013b. Efficient salient region detection with soft image abstraction. In *ICCV*, 1529–1536. doi:10.1109/ICCV.2013.193. (cited on page 89)

COMON, P., 1994. Independent component analysis, a new concept? *Signal processing*, 36, 3 (1994), 287–314. (cited on page 2)

COSTEIRA, J. AND KANADE, T., 1998. A multibody factorization method for independently moving objects. *IJCV*, 29, 3 (1998), 159–179. (cited on page 26)

DABOV, K.; FOI, A.; KATKOVNIK, V.; AND EGIAZARIAN, K., 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16, 8 (2007), 2080–2095. (cited on pages 4 and 47)

DAUBECHIES, I.; DEFRISE, M.; AND DE MOL, C., 2004. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57, 11 (2004), 1413–1457. (cited on page 20)

DO, M. N. AND VETTERLI, M., 2005. The contourlet transform: an efficient directional multiresolution image representation. *Image Processing, IEEE Transactions on*, 14, 12 (2005), 2091–2106. (cited on page 20)

DONG, W.; ZHANG, L.; LUKAC, R.; AND SHI, G., 2013. Sparse representation based image interpolation with nonlocal autoregressive modeling. *Image Processing, IEEE Transactions on*, 22, 4 (2013), 1382–1394. (cited on page 21)

DONOHO, D. L., 1995. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41, 3 (1995), 613–627. (cited on pages 18 and 19)

DONOHO, D. L. ET AL., 2006. Compressed sensing. *IEEE Transactions on information theory*, 52, 4 (2006), 1289–1306. (cited on page 14)

EFRON, B.; HASTIE, T.; JOHNSTONE, I.; TIBSHIRANI, R.; ET AL., 2004. Least angle regression. *The Annals of statistics*, 32, 2 (2004), 407–499. (cited on pages 17, 19, and 20)

ELAD, M.; FIGUEIREDO, M. A.; AND MA, Y., 2010. On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, 98, 6 (2010), 972–982. (cited on page 14)

ELHAMIFAR, E. AND VIDAL, R., 2009. Sparse subspace clustering. In *CVPR*, 2790–2797. (cited on pages 8, 25, 26, 27, 53, 54, 63, 75, 78, 81, and 102)

ELHAMIFAR, E. AND VIDAL, R., 2013a. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35, 11 (2013), 2765–2781. (cited on pages 5, 25, and 27)

ELHAMIFAR, E. AND VIDAL, R., 2013b. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35, 11 (2013), 2765–2781. (cited on pages 53, 54, 57, 58, 63, 69, 76, and 82)

ENGAN, K.; AASE, S. O.; AND HUSOY, J. H., 1999. Method of optimal directions for frame design. *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99*, 5 (1999), 2443–2446. doi:10.1109/ICASSP.1999.760624. (cited on page 15)

EVERINGHAM, M.; ESLAMI, S. M. A.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; AND ZISSERMAN, A., 2015. The pascal visual object classes challenge: A retrospective. *Int. Journal of Computer Vision*, 111, 1 (2015), 98–136. doi:10.1007/s11263-014-0733-5. (cited on page 95)

FAVARO, P.; VIDAL, R.; AND RAVICHANDRAN, A., 2011. A closed form solution to robust subspace estimation and clustering. In *CVPR*, 1801–1807. IEEE. (cited on page 54)

FENG, J.; LIN, Z.; XU, H.; AND YAN, S., 2014. Robust subspace segmentation with block-diagonal prior. In *CVPR*, 3818–3825. (cited on page 26)

FIGUEIREDO, M. A. AND NOWAK, R. D., 2003. An em algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12, 8 (2003), 906–916. (cited on page 20)

FIGUEIREDO, M. A. AND NOWAK, R. D., 2005. A bound optimization approach to wavelet-based image deconvolution. In *IEEE International Conference on Image Processing 2005*, vol. 2, II–782. IEEE. (cited on page 18)

GIRSHICK, R., 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

GLOROT, X.; BORDES, A.; AND BENGIO, Y., 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323. (cited on page 2)

GOFERMAN, S.; ZELNIK-MANOR, L.; AND TAL, A., 2012. Context-aware saliency detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34, 10 (Oct 2012), 1915–1926. doi:10.1109/TPAMI.2011.272. (cited on pages 6, 31, 32, 89, and 90)

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; AND BENGIO, Y., 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680. (cited on page 24)

GORODNITSKY, I. F. AND RAO, B. D., 1997. Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45, 3 (1997), 600–616. doi:10.1109/78.558475. (cited on page 20)

HARTIGAN, J. A. AND WONG, M. A., 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28, 1 (1979), 100–108. (cited on page 4)

HE, K.; GKIOXARI, G.; DOLLÁR, P.; AND GIRSHICK, R., 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. (cited on page 82)

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016b. Deep residual learning for image recognition. In *CVPR*, 770–778. doi:10.1109/CVPR.2016.90. (cited on pages 94 and 97)

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016c. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer. (cited on page 83)

HINTON, G. E., 2009. Deep belief networks. *Scholarpedia*, 4, 5 (2009), 5947. (cited on page 2)

HINTON, G. E. AND SALAKHUTDINOV, R. R., 2006. Reducing the dimensionality of data with neural networks. *Science*, 313, 5786 (2006), 504–507. (cited on page 23)

HJELM, R. D.; FEDOROV, A.; LAVOIE-MARCHILDON, S.; GREWAL, K.; BACHMAN, P.; TRISCHLER, A.; AND BENGIO, Y., 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, (2018). (cited on pages 2 and 103)

HO, J.; YANG, M.-H.; LIM, J.; LEE, K.-C.; AND KRIEGMAN, D., 2003. Clustering appearances of objects under varying illumination conditions. In *CVPR*, vol. 1, 11–18. IEEE. (cited on pages 25 and 76)

HOU, Q.; CHENG, M.-M.; HU, X.; BORJI, A.; TU, Z.; AND TORR, P. H. S., 2017. Deeply supervised salient object detection with short connections. In *CVPR*, 3203–3212. (cited on pages 9, 33, 96, and 98)

JI, P.; LI, H.; SALZMANN, M.; AND DAI, Y., 2014a. Robust motion segmentation with unknown correspondences. In *ECCV*, 204–219. Springer. (cited on page 75)

JI, P.; LI, H.; SALZMANN, M.; AND ZHONG, Y., 2016. Robust multi-body feature tracker: a segmentation-free approach. In *CVPR*, 3843–3851. (cited on page 75)

JI, P.; REID, I. D.; GARG, R.; LI, H.; AND SALZMANN, M., 2017a. Adaptive low-rank kernel subspace clustering. In *arXiv preprint arXiv:1707.04974v4*. (cited on page 76)

JI, P.; SALZMANN, M.; AND LI, H., 2014b. Efficient dense subspace clustering. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 461–468. IEEE. (cited on pages 25, 26, 28, 54, 57, 58, 63, and 78)

JI, P.; SALZMANN, M.; AND LI, H., 2015. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *ICCV*, 4687–4695. (cited on pages 26, 53, and 76)

JI, P.; ZHANG, T.; LI, H.; SALZMANN, M.; AND REID, I., 2017b. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, 23–32. (cited on pages 2, 70, 79, and 82)

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; AND DARRELL, T., 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM Int. Conf. Multimedia* (Orlando, Florida, USA, 2014), 675–678. doi:10.1145/2647868.2654889. (cited on page 94)

JIANG, B.; ZHANG, L.; LU, H.; YANG, C.; AND YANG, M., 2013a. Saliency detection via absorbing markov chain. In *ICCV*, 1665–1672. doi:10.1109/ICCV.2013.209. (cited on pages 91 and 96)

JIANG, H.; WANG, J.; YUAN, Z.; WU, Y.; ZHENG, N.; AND LI, S., 2013b. Salient object detection: A discriminative regional feature integration approach. In *CVPR*, 2083–2090. doi:10.1109/CVPR.2013.271. (cited on pages 32, 95, and 96)

JIANG, P.; LING, H.; YU, J.; AND PENG, J., 2013c. Salient region detection by UFO: Uniqueness, focusness and objectness. In *ICCV*, 1976–1983. doi:10.1109/ICCV.2013.248. (cited on pages 32 and 89)

KANATANI, K.-I., 2001. Motion segmentation by subspace separation and model selection. In *ICCV*, vol. 2, 586–591. IEEE. (cited on pages 25, 26, and 75)

KIM, J.; HAN, D.; TAI, Y.-W.; AND KIM, J., 2014. Salient region detection via high-dimensional color transform. In *CVPR*, 883–890. doi:10.1109/CVPR.2014.118. (cited on page 32)

KINGMA, D. AND BA, J., 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*, (2014). (cited on pages 57, 70, and 82)

KINGMA, D. P. AND WELLING, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on pages 2 and 24)

KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105. (cited on pages 1 and 56)

LE PENNEC, E. AND MALLAT, S., 2005. Sparse geometric image representations with bandelets. *IEEE Transactions on Image Processing*, 14, 4 (2005), 423–438. doi:10.1109/TIP.2005.843753. (cited on page 20)

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; AND HAFFNER, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11 (1998), 2278–2324. (cited on pages 68 and 75)

LEE, H.; GROSSE, R.; RANGANATH, R.; AND NG, A. Y., 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, 609–616. ACM. (cited on page 2)

LEE, K.-C.; HO, J.; AND KRIEGMAN, D. J., 2005. Acquiring linear subspaces for face recognition under variable lighting. *TPAMI*, 27, 5 (2005), 684–698. (cited on pages 53 and 58)

LI, B.; SHEN, C.; DAI, Y.; VAN DEN HENGEL, A.; AND HE, M., 2015. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *CVPR*, 1119–1127. doi:10.1109/CVPR.2015.7298715. (cited on page 99)

LI, C.-G. AND VIDAL, R., 2015. Structured sparse subspace clustering: A unified optimization framework. In *CVPR*, 277–286. (cited on page 26)

LI, G. AND YU, Y., 2015. Visual saliency based on multiscale deep features. In *CVPR*, 5455–5463. doi:10.1109/CVPR.2015.7299184. (cited on pages 33 and 96)

LI, G. AND YU, Y., 2016. Deep contrast learning for salient object detection. In *CVPR*, 478–487. doi:10.1109/CVPR.2016.58. (cited on pages 33 and 96)

LI, X.; LU, H.; ZHANG, L.; RUAN, X.; AND YANG, M., 2013. Saliency detection via dense and sparse reconstruction. In *ICCV*, 2976–2983. doi:10.1109/ICCV.2013.370. (cited on pages 91 and 96)

LI, X.; ZHAO, L.; WEI, L.; YANG, M. H.; WU, F.; ZHUANG, Y.; LING, H.; AND WANG, J., 2016. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE Trans. on Image Processing*, 25, 8 (Aug 2016), 3919–3930. doi:10.1109/TIP.2016.2579306. (cited on pages 33 and 96)

LI, Y.; HOU, X.; KOCH, C.; REHG, J. M.; AND YUILLE, A. L., 2014. The secrets of salient object segmentation. In *CVPR*, 280–287. doi:10.1109/CVPR.2014.43. (cited on page 95)

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; AND ZITNICK, C. L., 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer. (cited on page 1)

LIU, G.; LIN, Z.; YAN, S.; SUN, J.; YU, Y.; AND MA, Y., 2013. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35, 1 (2013), 171–184. (cited on pages 53, 54, 58, 63, 69, 76, and 82)

LIU, G.; LIN, Z.; AND YU, Y., 2010. Robust subspace segmentation by low-rank representation. In *Int. Conf. on Machine Learning (ICML)*, 663–670. (cited on pages 26, 27, and 54)

LIU, G. AND YAN, S., 2011. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, 1615–1622. IEEE. (cited on page 78)

LIU, T.; SUN, J.; ZHENG, N.-N.; TANG, X.; AND SHUM, H.-Y., 2007. Learning to detect a salient object. In *CVPR*, 1–8. doi:10.1109/CVPR.2007.383047. (cited on page 95)

LIU, Y.; LIU, S.; AND WANG, Z., 2015a. A general framework for image fusion based on multi-scale transform and sparse representation. *Information Fusion*, 24 (2015), 147–164. (cited on page 21)

LIU, Z.; LUO, P.; WANG, X.; AND TANG, X., 2015b. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*. (cited on page 45)

LIU, Z.; MIAO, Z.; ZHAN, X.; WANG, J.; GONG, B.; AND YU, S. X., 2019. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2537–2546. (cited on page 104)

LLOYD, S., 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28, 2 (1982), 129–137. (cited on pages 69 and 82)

LU, C.-Y.; MIN, H.; ZHAO, Z.-Q.; ZHU, L.; HUANG, D.-S.; AND YAN, S., 2012. Robust and efficient subspace segmentation via least squares regression. In *ECCV*, 347–360. Springer. (cited on pages 26, 53, 54, and 76)

LU, Z.; FU, Z.; XIANG, T.; HAN, P.; WANG, L.; AND GAO, X., 2017. Learning from weak and noisy labels for semantic segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 39, 3 (Mar 2017), 486–500. doi:10.1109/TPAMI.2016.2552172. (cited on page 99)

LUO, Z.; MISHRA, A.; ACHKAR, A.; EICHEL, J.; LI, S.; AND JODOIN, P.-M., 2017. Non-local deep features for salient object detection. In *CVPR*. (cited on pages 32, 33, 96, and 98)

MA, X.; LI, B.; WANG, Y.; ERFANI, S. M.; WIJEWICKREMA, S.; SCHOENEBECK, G.; SONG, D.; HOULE, M. E.; AND BAILEY, J., 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, (2018). (cited on page 103)

MA, Y.; DERKSEN, H.; HONG, W.; AND WRIGHT, J., 2007. Segmentation of multivariate mixed data via lossy data coding and compression. *TPAMI*, 29, 9 (2007). (cited on pages 25 and 76)

MAIRAL, J.; BACH, F.; PONCE, J.; AND SAPIRO, G., 2009a. Online dictionary learning for sparse coding. *Proceedings of the 26th International Conference on Machine Learning*, (2009), 1–8. doi:10.1145/1553374.1553463. http://dl.acm.org/citation.cfm?id=1553463. (cited on pages 15, 39, and 45)

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G.; AND ZISSERMAN, A., 2009b. Non-local sparse models for image restoration. *Proceedings of the IEEE International Conference on Computer Vision*, (2009), 2272–2279. doi:10.1109/ICCV.2009.5459452. (cited on page 2)

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G.; AND ZISSERMAN, A., 2009c. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2272–2279. IEEE. (cited on page 47)

MAIRAL, J.; SAPIRO, G.; AND ELAD, M., 2008. Learning multiscale sparse representations for image and video restoration. *Multiscale Modeling & Simulation*, 7, 1 (2008), 214–241. (cited on pages 20 and 45)

MAKHZANI, A.; SHLENS, J.; JAITLY, N.; GOODFELLOW, I.; AND FREY, B., 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, (2015). (cited on page 24)

MALLAT, S., 1999. *A wavelet tour of signal processing*. Elsevier. (cited on pages 14 and 15)

MALLAT, S. G., 1989. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11, 7 (1989), 674–693. (cited on page 20)

MALLAT, S. G. AND ZHANG, Z., 1993. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41, 12 (1993), 3397–3415. (cited on pages 17 and 20)

MARTIN, D.; FOWLKES, C.; TAL, D.; AND MALIK, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, 416–423. (cited on page 45)

MO, Q. AND DRAPER, B. A., 2012. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *ECCV*, 402–415. Springer. (cited on page 26)

MOTIIAN, S.; JONES, Q.; IRANMANESH, S.; AND DORETTO, G., 2017. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, 6670–6680. (cited on page 103)

MUKHERJEE, S.; ASNANI, H.; LIN, E.; AND KANNAN, S., 2018. Clustergan : Latent space clustering in generative adversarial networks. *CoRR*, abs/1809.03627 (2018). http://arxiv.org/abs/1809.03627. (cited on page 83)

NAIR, V. AND HINTON, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814. (cited on page 23)

NEEDELL, D. AND TROPP, J. A., 2009. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26, 3 (2009), 301–321. (cited on page 18)

NEEDELL, D. AND VERSHYNIN, R., 2009. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9, 3 (2009), 317–334. (cited on page 18)

NENE, S. A.; NAYAR, S. K.; AND MURASE, H., 1996a. Columbia object image library (COIL-100). *Technical Report CUCS-006-96*, (1996). (cited on pages 53 and 60)

NENE, S. A.; NAYAR, S. K.; AND MURASE, H., 1996b. Columbia object image library (COIL-20). *Technical Report CUCS-005-96*, (1996). (cited on pages 53 and 60)

NG, A. Y.; JORDAN, M. I.; WEISS, Y.; ET AL., 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*, vol. 14, 849–856. (cited on pages 26, 30, and 57)

NGUYEN, H. V. AND BAI, L., 2010. Cosine similarity metric learning for face verification. In *Asian conference on computer vision*, 709–720. Springer. (cited on page 78)

OCHS, P. AND BROX, T., 2012. Higher order motion models and spectral clustering. In *CVPR*. (cited on page 26)

OH SONG, H.; XIANG, Y.; JEGELKA, S.; AND SAVARESE, S., 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4004–4012. (cited on page 75)

OLSHAUSEN, B. A. AND FIELD, D. J., 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37, 23 (1997), 3311–3325. (cited on page 2)

OPHIR, B.; LUSTIG, M.; AND ELAD, M., 2011. Multi-Scale Dictionary Learning Using Wavelets. *IEEE Journal of Selected Topics in Signal Processing*, 5, 5 (2011), 1014–1024. doi:10.1109/JSTSP.2011.2155032. (cited on pages 20 and 45)

PARK, H.-S. AND JUN, C.-H., 2009. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36, 2 (2009), 3336–3341. (cited on page 5)

PATEL, V. M.; VAN NGUYEN, H.; AND VIDAL, R., 2013. Latent space sparse subspace clustering. In *ICCV*, 225–232. (cited on pages 53, 54, and 76)

PATEL, V. M. AND VIDAL, R., 2014. Kernel sparse subspace clustering. In *ICIP*, 2849–2853. IEEE. (cited on pages 28, 53, 54, 58, 69, 76, and 82)

PATI, Y. C.; REZAIIFAR, R.; AND KRISHNAPRASAD, P., 1993. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. (1993), 40–44. (cited on pages 17 and 20)

PENG, X.; FENG, J.; XIAO, S.; LU, J.; YI, Z.; AND YAN, S., 2017. Deep sparse subspace clustering. *arXiv preprint arXiv:1709.08374*, (2017). (cited on page 70)

PENG, X.; XIAO, S.; FENG, J.; YAU, W.-Y.; AND YI, Z., 2016. Deep subspace clustering with sparsity prior. In *IJCAI*. (cited on pages 58, 59, and 61)

PEYRÉ, G., 2009. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34, November 2008 (2009), 17–31. doi:10.1007/s10851-008-0120-3. (cited on page 2)

POULTNEY, C.; CHOPRA, S.; CUN, Y. L.; ET AL., 2007. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, 1137–1144. (cited on page 22)

PURKAIT, P.; CHIN, T.-J.; ACKERMANN, H.; AND SUTER, D., 2014. Clustering with hypergraphs: the case for large hyperedges. In *ECCV*, 672–687. Springer. (cited on page 26)

RAO, S. R.; TRON, R.; VIDAL, R.; AND MA, Y., 2008. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *CVPR*, 1–8. IEEE. (cited on pages 53, 54, and 102)

ROTH, S. AND BLACK, M. J., 2009. Fields of experts. *International Journal of Computer Vision*, 82, 2 (2009), 205–229. doi:10.1007/s11263-008-0197-6. (cited on page 2)

RUBINSTEIN, R.; ZIBULEVSKY, M.; AND ELAD, M., 2008. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *CS Technion*, 40, 8 (2008), 1–15. (cited on pages 39 and 45)

SALAKHUTDINOV, R. AND HINTON, G., 2009. Deep boltzmann machines. In *Artificial intelligence and statistics*, 448–455. (cited on page 2)

SALAKHUTDINOV, R. AND LAROCHELLE, H., 2010. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 693–700. (cited on page 2)

SAMARIA, F. S. AND HARTER, A. C., 1994. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, 138–142. IEEE. (cited on pages 53 and 59)

SANDIN, F. AND MARTIN-DEL CAMPO, S., 2016. Dictionary learning with equiprobable matching pursuit. *arXiv preprint arXiv:1611.09333*, (2016). (cited on page 43)

SELESNICK, I., 2009. A derivation of the soft-thresholding function. *Polytechnic Institute of New York University*, (2009). (cited on page 19)

SELESNICK, I. W.; GRABER, H. L.; DING, Y.; ZHANG, T.; AND BARBOUR, R. L., 2014. Transient artifact reduction algorithm (tara) based on sparse optimization. *IEEE Transactions on Signal Processing*, 62, 24 (2014), 6596–6611. (cited on page 20)

SHAWE-TAYLOR, J. AND CRISTIANINI, N., 2004. *Kernel methods for pattern analysis*. Cambridge university press. (cited on pages 53 and 61)

SHEN, X. AND WU, Y., 2012. A unified approach to salient object detection via low rank matrix recovery. In *CVPR*, 853–860. doi:10.1109/CVPR.2012.6247758. (cited on pages 32 and 89)

SHI, J. AND MALIK, J., 2000. Normalized cuts and image segmentation. *TPAMI*, 22, 8 (2000), 888–905. (cited on pages 26 and 30)

SIMONCELLI, E. P. AND FREEMAN, W. T., 1995. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *icip*, 3444. IEEE. (cited on page 20)

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; AND SALAKHUTDINOV, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15, 1 (2014), 1929–1958. (cited on page 24)

STARCK, J.-L.; MURTAGH, F.; AND FADILI, J. M., 2010. *Sparse image and signal processing: wavelets, curvelets, morphological diversity*. Cambridge university press. (cited on page 14)

SUGANO, Y.; MATSUSHITA, Y.; AND SATO, Y., 2010. Calibration-free gaze sensing using saliency maps. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2667–2674. IEEE. (cited on page 31)

SULAM, J.; OPHIR, B.; AND ELAD, M., 2014. Image denoising through multi-scale learnt dictionaries. In *Image Processing (ICIP), 2014 IEEE International Conference on*, 808–812. IEEE. (cited on pages 2, 21, and 39)

TARQUINO, J.; RUEDA, A.; AND ROMERO, E., 2014. A multiscalesparse representation for diffusion weighted imaging (dwi) super-resolution. In *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*, 983–986. IEEE. (cited on page 21)

TIBSHIRANI, R., 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58, 1 (1996), 267–288. (cited on page 17)

TOLSTIKHIN, I.; BOUSQUET, O.; GELLY, S.; AND SCHOELKOPF, B., 2017. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, (2017). (cited on page 24)

TRAMÈR, F.; PAPERNOT, N.; GOODFELLOW, I.; BONEH, D.; AND MCDANIEL, P., 2017. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, (2017). (cited on page 104)

TROPP, J. A., 2004. Greed is good: Algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, 50, 10 (2004), 2231–2242. (cited on pages 17 and 40)

TROPP, J. A.; GILBERT, A. C.; AND STRAUSS, M. J., 2006. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal processing*, 86, 3 (2006), 572–588. (cited on pages 16 and 18)

TSENG, P., 2000. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105, 1 (2000), 249–252. (cited on pages 8, 63, and 76)

VIDAL, R., 2011. Subspace clustering. *IEEE Signal Processing Magazine*, 28, 2 (2011), 52–68. (cited on page 26)

VIDAL, R. AND FAVARO, P., 2014. Low rank subspace clustering (LRSC). *Pattern Recognition Letter*, 43 (2014), 47–61. (cited on pages 28, 54, 58, 63, and 78)

VIDAL, R.; MA, Y.; AND SASTRY, S., 2005. Generalized principal component analysis (gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27, 12 (2005), 1945–1959. (cited on page 15)

VIDAL, R.; TRON, R.; AND HARTLEY, R., 2008. Multiframe motion segmentation with missing data using powerfactorization and GPCA. *IJCV*, 79, 1 (2008), 85–105. (cited on page 26)

VINCENT, P.; LAROCHELLE, H.; BENGIO, Y.; AND MANZAGOL, P.-A., 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103. ACM. (cited on page 23)

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; AND MANZAGOL, P.-A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11, Dec (2010), 3371–3408. (cited on page 23)

VON LUXBURG, U., 2007. A tutorial on spectral clustering. *Statistics and computing*, 17, 4 (2007), 395–416. (cited on pages 28, 29, and 30)

WANG, J. AND CHERIAN, A., 2019. Gods: Generalized one-class discriminative subspaces for anomaly detection. *arXiv preprint arXiv:1908.05884*, (2019). (cited on page 103)

WANG, L.; WANG, L.; LU, H.; ZHANG, P.; AND RUAN, X., 2016. Saliency detection with recurrent fully convolutional networks. In *ECCV*, 825–841. doi:10.1007/978-3-319-46493-0_50. (cited on pages 33 and 96)

WANG, T.; BORJI, A.; ZHANG, L.; ZHANG, P.; AND LU, H., 2017. A stagewise refinement model for detecting salient objects in images. In *ICCV*. (cited on page 32)

WANG, Y.-X.; XU, H.; AND LENG, C., 2013. Provable subspace clustering: When LRR meets SSC. In *NIPS*, 64–72. (cited on pages 26, 53, and 76)

WOLD, S.; ESBENSEN, K.; AND GELADI, P., 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2, 1-3 (1987), 37–52. (cited on page 2)

XIAO, H.; RASUL, K.; AND VOLLGRAF, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. (cited on pages 69 and 75)

XIAO, S.; TAN, M.; XU, D.; AND DONG, Z. Y., 2016. Robust kernel low-rank representation. *IEEE transactions on neural networks and learning systems*, 27, 11 (2016), 2268–2281. (cited on pages 53, 54, and 76)

XIE, J.; GIRSHICK, R.; AND FARHADI, A., 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. (cited on pages 2, 23, 65, 69, 70, and 82)

XU, K.; BA, J.; KIROS, R.; CHO, K.; COURVILLE, A.; SALAKHUDINOV, R.; ZEMEL, R.; AND BENGIO, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Int. Conf. on Machine Learning (ICML)*, vol. 37, 2048–2057. (cited on page 89)

YAN, J. AND POLLEFEYS, M., 2006. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV*, 94–106. Springer. (cited on pages 26, 53, and 76)

YAN, Q.; XU, L.; SHI, J.; AND JIA, J., 2013a. Hierarchical saliency detection. In *CVPR*, 1155–1162. doi:10.1109/CVPR.2013.153. (cited on pages 91, 95, and 96)

YAN, R.; SHAO, L.; AND LIU, Y., 2013b. Nonlocal hierarchical dictionary learning using wavelets for image denoising. *Image Processing, IEEE Transactions on*, 22, 12 (2013), 4689–4698. (cited on pages 15 and 21)

YANG, A. Y.; RAO, S. R.; AND MA, Y., 2006. Robust statistical estimation and segmentation of multiple subspaces. In *null*, 99. IEEE. (cited on page 25)

YANG, A. Y.; WRIGHT, J.; MA, Y.; AND SASTRY, S. S., 2008. Unsupervised segmentation of natural images via lossy data compression. *CVIU*, 110, 2 (2008), 212–225. (cited on page 76)

YANG, B.; FU, X.; SIDIROPOULOS, N. D.; AND HONG, M., 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, vol. 70, 3861–3870. (cited on pages 2, 23, 65, 69, 70, and 82)

YANG, C.; ZHANG, L.; LU, H.; RUAN, X.; AND YANG, M., 2013. Saliency detection via graph-based manifold ranking. In *CVPR*, 3166–3173. doi:10.1109/CVPR.2013.407. (cited on page 95)

YIN, H., 2015. Sparse representation with learned multiscale dictionary for image fusion. *Neurocomputing*, 148 (2015), 600–610. (cited on page 21)

YIN, M.; GUO, Y.; GAO, J.; HE, Z.; AND XIE, S., 2016. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *CVPR*, 5157–5164. (cited on pages 53, 54, and 76)

YOU, C.; LI, C.-G.; ROBINSON, D. P.; AND VIDAL, R., 2016a. Oracle based active set algorithm for scalable elastic net subspace clustering. In *CVPR*, 3928–3937. (cited on pages 26, 53, and 76)

YOU, C.; ROBINSON, D.; AND VIDAL, R., 2016b. Scalable sparse subspace clustering by orthogonal matching pursuit. In *CVPR*, 3918–3927. (cited on pages 58, 65, 70, and 76)

ZHANG, D.; HAN, J.; AND ZHANG, Y., 2017a. Supervision by fusion: Towards unsupervised learning of deep salient object detector. In *ICCV*. (cited on pages xvii, 9, 90, 92, 96, and 98)

ZHANG, G.-X.; CHENG, M.-M.; HU, S.-M.; AND MARTIN, R. R., 2009. A shape-preserving approach to image resizing. *Computer Graphics Forum*, 28, 7 (2009), 1897–1906. (cited on page 89)

ZHANG, J.; DAI, Y.; AND PORIKLI, F., 2017b. Deep salient object detection by integrating multi-level cues. In *Proc. IEEE Winter Conference on Applications of Computer Vision*, 1–10. doi:10.1109/WACV.2017.8. (cited on page 33)

ZHANG, P.; WANG, D.; LU, H.; WANG, H.; AND RUAN, X., 2017c. Amulet: Aggregating multi-level convolutional features for salient object detection. In *ICCV*. (cited on pages 9, 32, 33, 96, and 98)

ZHANG, P.; WANG, D.; LU, H.; WANG, H.; AND YIN, B., 2017d. Learning uncertain convolutional features for accurate saliency detection. In *ICCV*. (cited on pages 32 and 96)

ZHANG, T.; JI, P.; HARANDI, M.; HARTLEY, R. I.; AND REID, I. D., 2018. Scalable deep k-subspace clustering. *CoRR*, abs/1811.01045 (2018). http://arxiv.org/abs/1811.01045. (cited on pages 76 and 82)

ZHANG, T.; SZLAM, A.; WANG, Y.; AND LERMAN, G., 2012. Hybrid linear modeling via local best-fit flats. *International journal of computer vision*, 100, 3 (2012), 217–240. (cited on page 65)

ZHAO, R.; OUYANG, W.; LI, H.; AND WANG, X., 2015. Saliency detection by multi-context deep learning. In *CVPR*, 1265–1274. doi:10.1109/CVPR.2015.7298731. (cited on pages 33 and 96)

ZHU, W.; LIANG, S.; WEI, Y.; AND SUN, J., 2014. Saliency optimization from robust background detection. In *CVPR*, 2814–2821. doi:10.1109/CVPR.2014.360. (cited on pages 6, 31, 32, 89, 91, and 96)