Deep Transform Learning

By

Jyoti Maggu

Under the supervision of

Dr. Angshul Majumdar

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

21st December, 2019

Deep Transform Learning

By

Jyoti Maggu

A Thesis

submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

21st December, 2019

Certificate

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

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

21st December, 2019

Dr. Angshul Majumdar

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Dedication

To,
My loving Parents– Every bit of me is little bit of you.

Acknowledgements

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

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

I am much indebted to Dr. Mayank Vatsa, Dr. Debarka Sengupta, (Asst. Prof. IIIT Delhi) and Dr. Victor Sanchez (Associate Professor, University of Warwick) for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives. I want to express my immense gratitude for the invaluable feedback to the external examiners Dr. Tanaya Guha, Dr. Pawel Wawrzyfiski and Dr. Sonya Coleman.

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

I have been blessed to have worked and guided by some of the eminent researchers in the field. I want to express my sincere thanks to Dr. Emilie Chouzenoux and Dr. Giovanni Chierchia for research collaboration and internship opportunity at A3SI, LIGM, Universite de Paris.

This thesis would be incomplete without the mention of my support system from the Salsa lab. I thank all my friends and colleagues in this journey, Hemant, Megha, Shikha, Shalini, Pooja, Priyadarshini, Smriti, Kanchan, Tanya, Ruchi, Divya, Prerna and many more. Special thanks to Vanika for always helping me. It is the endless cups of coffee and night long conversations with her that gave me a new perspective to look at everything from work to life. Staying together in Paris, chatting, walking and cooking together was fantastic. Aanchal has been the closest of my friends; my one stop destination for sharing it all – happiness and sorrows. Thanks for everything.

This thesis and my PhD itself would not have seen culmination if not for my family. My parents have given me the freedom to choose, the confidence to decide and a faith that no matter what, I have them watching my back. My gratitude to my family is incomplete without a very special mention for my mother, for always believing in me, for the endless love and supporting me throughout my life. Another person who bore the highs and lows of this journey almost as much as I did is my brother, Himanshu. He has borne all my mood swings, stress and lack of time and yet stood strong by my side through it all. I cannot thank enough my childhood friend Nishu for the unconditional support.

I owe a special thanks to my husband, Deepak Budhiraja, for supporting me and encouraging me throughout my Ph.D. To my beloved son Rudra, I would like to express my thanks for always cheering me up.

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

Abstract

Conventional dictionary learning is a synthesis formulation; it learns a dictionary to generate/synthesize the data from the learned coefficients. Transform learning is its analysis equivalent. The transform analyzes the data to generate the coefficients. Dictionary learning had been popular in both signal processing and machine learning communities. However, transform learning is largely unknown outside the signal processing research community. So far, transform learning has been primarily used for solving inverse problems.

The objective of the thesis is to build a completely new machine learning framework out of transform learning. It has already been shown how the basic transform learning has been used as an unsupervised feature extraction tool.

This work aims at proposing a supervised version of transform learning with a plug-and-play approach. The supervised version is general enough to perform classification without the need for any external classifier. The kernelized version of supervised transform learning and stochastic regularization on transform learning are also proposed. Based on the proposed supervised transform learning framework, problems on computer vision, bioinformatics, hyperspectral image classification, and arrhythmia classification are solved.

This work also focuses on an unsupervised greedy deep transform learning problem, where each of the layers was solved separately. This was a solution for unsupervised feature extraction using deep transform learning. But the greedy solution for deep transform learning was sub-optimal. Then work has been done on proposing an optimal solution to learn all the layers jointly. It was used to solve classification, clustering and inverse problems.

Another problem discussed in this work is the supervised version of deep transform learning. The supervised version is general enough to perform single-label classification and multi-label classification. Proposed supervised deep transform learning for multi-label classification has been used for solving a practical problem of non-intrusive load monitoring.

Another contribution of this work is to propose a deeply transformed subspace clustering framework. In this work, two techniques are introduced: transformed locally linear manifold clustering and transformed sparse subspace clustering. Next, a deeper architecture for the same is proposed.

Then, the idea of convolutional transform learning is introduced. Here, a set of independent convolutional filters are learned that operate on the images to produce representations (one corresponding to each filter). The kernels learned from this method have a close relationship with that of convolutional neural networks.

Finally, a semi-coupled transform learning framework is introduced. Given training data in two domains (source and target), it learns a transform in each of the domains such that the corresponding coefficients are (linearly) mapped from the source to the target. Since the mapping is in one direction (source to target) but not the other way round, It is called semi-coupled. This work is the analysis equivalent of (semi) coupled dictionary learning.

Contents

List of Tables

List of Figures

Acronyms

DeblurGAN	generative adversarial network-based deblurring
TL	transform learning
DL	dictionary learning
MOD	method of optimal directions
DDL	deep dictionary learning
CoDL	convolutional dictionary learning
CDL	coupled dictionary learning
SVD	singular value decomposition
MRI	magnetic resonance imaging
STL	supervised transform learning
KDL	kernel dictionary learning
RBM	restricted boltzmann machines
CSTL	class-sparse transform learning
LRTL	low-rank transform learning
ISTA	iterative soft thresholding algorithm
KTL	kernel transform learning
KMP	kernel matching pursuit
LCTL	label consistent transform learning
DBDL	discriminative bayesian dictionary learning
MTDL	multimodal task driven dictionary learning

DADL	discriminative analysis dictionary learning
SEDL	sparse embedded dictionary learning
NDL	non linear dictionary learning
DiTL	discriminative transform learning
DO	drop out
DC	drop connect
K-CSTL	kernel class-sparse transform learning
K-LRTL	kernel low-rank transform learning
K-DiTL	kernel discriminative transform learning
K-LCTL	kernel label consistent transform learning
CNN	convolutional neural network
AUC	area under curve
SSLR	spectral-spatial shared linear regression
HSSF	hierarchical spectral-spatial features
GSLR	group-sparse low-rank representation
OA	overall accuracy
AA	average accuracy
DBN	deep belief network
SAE	stacked auto encoder
SVM	support vector machine
DTL	deep transform learning
GDTL	greedy deep transform learning
JDTL	jointly learned deep transform learning
ADMM	alternating direction method of multipliers
CSSAE	class sparse stacked autoencoder

CSDBN	class sparse deep belief network
KNN	k-nearest neighbour
DSC	deep subspace clustering
DSIFT	dense scale-invariant feature transform
HOG	histogram of oriented gradients
PCA	principal component analysis
NMI	normalized mutual information
ARI	adjusted rand index
LASSO	least absolute selection and shrinkage operator
CS	compressed sensing
ReLU	rectified linear unit
AL	augmented lagrangian
PSNR	peak signal to noise ratio
SSIM	structural similarity index
GBD	graph-based deblurring
RCSR	row-column sparse representation
CRN	convolutional recurrent network
KLR	kernel low-rank method
LASSI	low-rank adaptive sparse signal model
NMSE	normalized mean squared error
NILM	non intrusive load monitoring
LCDL	label consistent dictionary learning
LCDTL	label consistent deep transform learning
RAKEL	random K label set
MLKNN	multi-label K-nearest neighbor

REDD	reference energy disaggregation dataset
LLMC	locally linear manifold clustering
SSC	Sparse subspace clustering
LRR	low-rank representation
DTSC	deeply transformed subspace clustering
SVD	singular value decomposition
OMP	orthogonal matching pursuit
DSC	deep sparse subspace clustering
DMF	deep matrix factorization
DKM	deep K-means clustering
TLLMC	transfomed locally linear manifold clustering
TSSC	transformed sparse subspace clustering
TLRR	transformed low-rank representation
DTSSC	deep transformed sparse subspace clustering
DTLLMC	deep transfomed locally linear manifold clustering
CoTL	convolutional transform learning
CTL	coupled transform learning
SCTL	semi-coupled transform learning
DCoTL	deep convolutional transform learning
OPCA	oriented principal component analysis
CPLSA	coupled probabilistic latent semantic analysis
MRR	mean reciprocal rank
SDAE	stacked denoising autoencoder
SGSA	stacked group sparse auto encoder
DDBN	discriminative deep belief network

LCKSVD label consistent KSVD

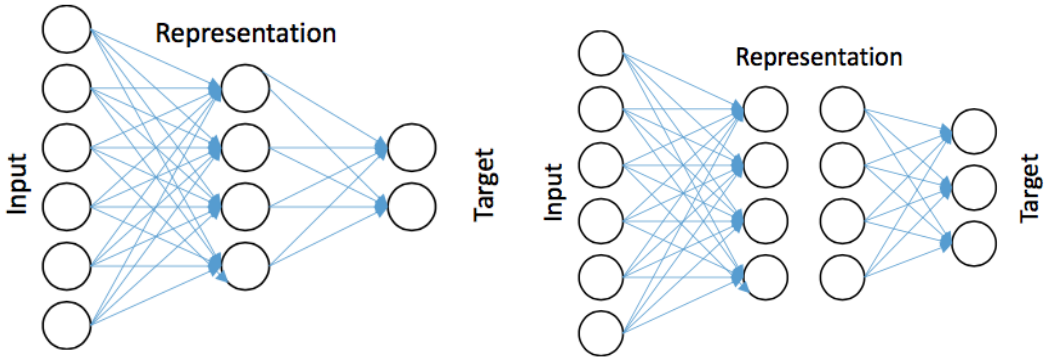
LCDDL label-consistent deep dictionary learning

SLCA stacked label consistent autoencoder

Chapter 1

Introduction

1.1 Introduction



(a) (b)

Figure 1.1: (a)Neural network (b)Segregated representation

In a typical neural network, there is an input layer where samples are presented, a hidden or representation layer and an output layer with the target values (figure 1.1a). The network is learned using back-propagation.

As shown in figure 1.1b, learning of such a network can be perceived as two sub-problems: learning the weights between input and the representation layer and between representation and output layer. If the representa-

tions are known, learning the weights between the hidden and output layer is trivial. It is a simple regression problem since both the input (representations of the input data) and output are known. Learning the weights between input and representation layer is a challenging task since both weights and output are unknown. This is called representation learning. There are four well-known frameworks for deep representation learning: stacked auto encoder, deep belief network, convolutional neural network and dictionary learning.

CNN gives excellent results in every perceivable image analysis task. However, the applicability of CNN is limited largely to images in the visible range. There is literature on the application of CNN to areas outside this range (hyperspectral imaging, radar imaging, etc.), but the success has been limited. This is largely because CNN need humongous volumes of supervised data to train; this is only available in the visible range of the spectrum. The pre-training fine-tuning paradigm does not generalize well outside this range; hence the applicability of CNN has been largely restricted in scientific imaging modalities. This also precludes their applicability in unsupervised tasks like clustering.

A deep belief network uses the restricted boltzmann machines (RBM) as the building block. Mathematically the cost function for RBM is cumbersome and cannot be solved efficiently; at best, it can be solved only approximately. This is a serious impediment that precludes any serious mathematical modification to the learning model. Besides, the inputs to

RBM / deep belief network (DBN) need to be either binary or in the range between 0 and 1 (gaussian Bernoulli RBM); this is highly restrictive. For any signal where the dynamic range is high, when the input is normalized (between 0 and 1), the small numbers reach the limits of machine precision and are treated as zeroes or garbage values.

Stacked autoencoder overcomes the limitations of DBN; it has a mathematically tractable cost function and can handle any input value. However, the problem of this model is that it requires learning twice the number of parameters/connections (encoders and decoders) than other neural networks. This makes the stacked auto encoder (SAE) susceptible to over-fitting.

Dictionary Learning has been used to solve inverse problems in imaging and as an unsupervised feature extraction tool in vision. The main disadvantage of DL for applications in vision is the relatively long feature extraction time during testing; owing to the requirement of solving an iterative optimization problem (l_0-minimization). Thus, the test feature generation is a time-consuming task and cannot be real-time.

The general idea that led the thesis is to introduce a new method for representation learning called transform learning, a technique developed in the field of signal processing, and apply it to problems considered in machine learning. Transform learning is an analysis framework [2, 3, 4]. It has been recently introduced and is not yet popular outside the signal processing community. It is a tool to find effective regularizer in ill-posed

inverse problems. As long as the underlying target (e.g., image, video, etc.) to be recovered is sparse, transform learning can help improve the reconstruction quality. In this work, it is shown that it can be used as an alternative to dictionary learning as well. The newly developed analysis framework of transform learning does not suffer from the slow feature extraction problem. In transform learning a basis/transform is learned to analyze the data to generate features/coefficients. For transform learning, the test features can, therefore, be generated by a matrix-vector product. This is fast and can be real-time.

1.2 Research contributions

This thesis focuses on building a completely new machine learning framework out of transform learning. It aims at a few inter-related problems. From the machine learning perspective, classification and clustering problems are solved. From a signal processing perspective, inverse problems are solved. The idea here is not to compete with the best dictionary learning techniques in computer vision, but to show that transform learning yields result at par (or better) than dictionary learning is and is computationally cheaper (faster) at run time. The proposed framework is implemented for different databases, comparison methods, and applications. The applications range from face recognition to MRI reconstruction to document retrieval. This work does not want to focus on one application and wants

to show the generic strength of the proposed work. The research contributions are summarized as follows:

- This work proposes supervised formulations of transform learning. Four different types of supervision penalties are proposed. They enforce (i) class- sparsity, (ii) similarity among intra-class features in terms of low-rank constraint, (iii) discriminative transform learning, (iv) label-consistency. Also, the work introduces the kernelized version of transform learning and stochastic regularization techniques drop out (DO) and drop connect (DC) into the transform learning formulation. The problems on computer vision, bioinformatics, hyperspectral image classification, and arrhythmia classification are solved to show the generic strength of the work.

- Then, a deep version of unsupervised transform learning is proposed. Two solutions are presented here: (i) greedy one that learns layers one after another, and (ii) one in which the whole structure is trained simultaneously in a single optimization run. It was used to solve classification, clustering and inverse problems.

- Another work discussed in this thesis is a deep version of supervised transform learning. Supervision is introduced by adding a label consistency penalty to the previous unsupervised formulation. An optimization algorithm is synthesized for training. It is based on proximal variable splitting, augmented Lagrangians, and alternating direction

method of multipliers. Proposed supervised deep transform learning for multi-label classification has been used for solving a practical problem of non-intrusive load monitoring.

- Another contribution is the incorporation of locally linear manifold clustering and sparse subspace clustering into the transform learning formulation. The method is introduced to perform transform analysis and clustering jointly through the formulation of a coupled minimization problem.

- Then the introduction of convolutions into transform learning is made. Convolutions are here determined in an unsupervised fashion based on the gauss-seidel algorithm.

- Finally, semi-coupled transform learning is proposed. Given are training data in two domains (source and target), it learns to transform each of the domains into lower-dimensional spaces between which there is a linear mapping. Then, it enables the reconstruction of target data from new source data.

The research outcomes have been disseminated through publications in journals and conferences.

1.3 Dissertation organization

Chapter 2 describes some important basic concepts that are building blocks of this thesis. It includes a brief description of stacked autoencoder, deep belief network, convolutional neural network and dictionary learning. Concepts related to deep dictionary learning supervised dictionary learning, coupled dictionary learning and coupled representation learning are also discussed along with an introduction to basic TL. Chapter 3 discusses supervised TL framework, its kernelized version, and stochastic regularization. Four types of supervision penalties are introduced into TL framework: class-sparse transform learning, low-rank transform learning, discriminative transform learning, and label consistent transform learning. The implementation of stochastic regularization is done through drop connect and drop out. Chapter 4 describes unsupervised deep TL frameworks. Two approaches are discussed- greedy and joint learning. It solves classification and clustering problems. Chapter 5 is based on solving inverse problems using unsupervised deep TL. First, the deblurring problem is solved. Then, it is used for solving the reconstruction problem. Chapter 6 introduces a deeper architecture of supervised TL. It is used for multilabel classification in a non-intrusive load monitoring application. Chapter 7 describes deep transformed subspace clustering for locally linear manifold clustering and sparse subspace clustering. Chapter 8 gives a description of convolutional TL. Convolutional filters are determined in an unsupervised fashion.

Chapter 9 describes semi-coupled TL. It is used for image super-resolution and cross-lingual document retrieval.

Chapter 2

Literature review

Deep learning methods have multiple levels of learning; in each layer, a more abstract representation of the raw data is learned via non-linear transformations. Using such non-linear modules of transformation, very complex functions can be determined. The critical idea stems from biology, where it is believed that cognition happens through several layers. The basic building blocks of deep learning are autoencoder, RBM, CNN, and DL. Each of them is discussed in this chapter. Introduction to basic concepts of deep dictionary learning, supervised DL, convolutional DL and coupled representation learning followed by basic transform learning are also given.

2.1 Autoencoder

The architecture of an autoencoder is shown in figure 2.1a. Autoencoder is a self-supervised network; it learns encoding and decoding weights between

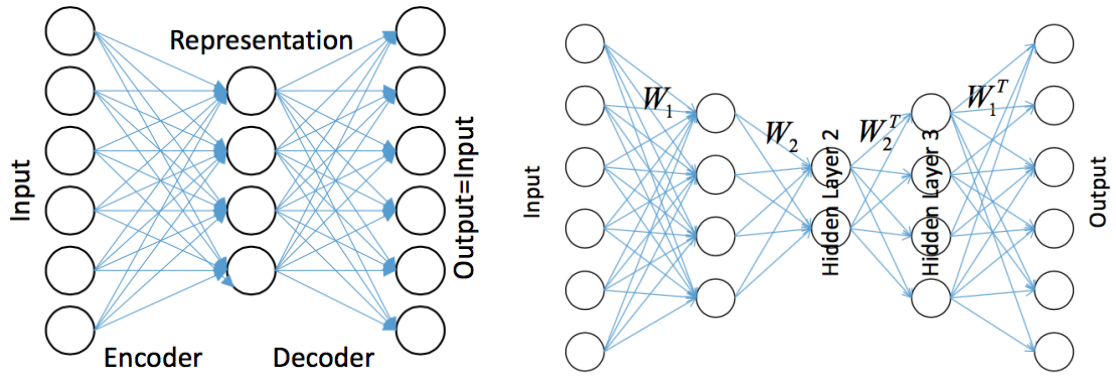


Figure 2.1: (a) Autoencoder (b) Stacked autoencoder

input and itself. It consists of two parts encoder and decoder. Encoder maps the input to a latent representation, and the decoder maps the encoded input back to the data. For a given input x the representations can be expressed as:

$$z = \phi(W_1 x) \tag{2.1}$$

Where, ϕ is the non-linearity and W_1 are the encoding weights. The decoder maps the representations back to the data space:

$$x = W_1{}'\phi(W_1 x) \tag{2.2}$$

The problem is to learn the encoding and decoding weights W_1 and $W_1{}'$. These are learned by minimizing the euclidean cost:

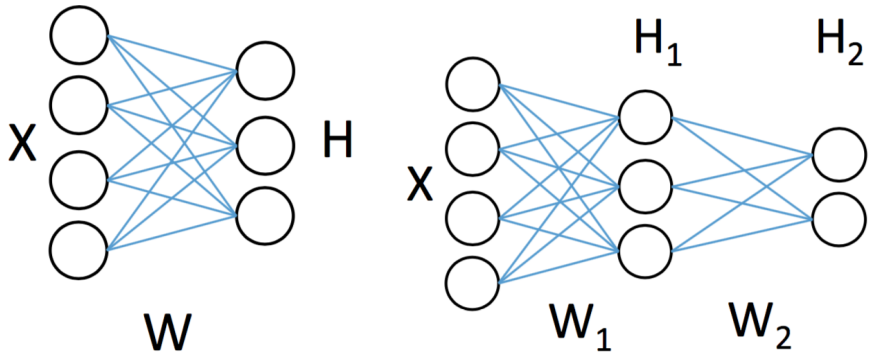
$$\min_{W_1, W_1{}'} \|x - W_1{}'\phi(W_1 x)\|_2^2 \tag{2.3}$$

The idea behind representation learning is to preserve the information content of the input in the representations. In autoencoder, information is

preserved at the representations in a euclidean sense. To form a neural network, the decoder part of the autoencoder is removed. The encoder forms the first part (input to representation layer) of the neural network. Targets are attached to the representations and backpropagation is used to train the network.

Deep networks can be formed using autoencoders by nesting them one inside the other. These networks are called stacked autoencoders (figure 2.1b). Once this network is learned, the decoder part is removed, and targets are attached to representations of the deepest layer. And the network is fine-tuned using backpropagation.

2.2 Restricted Boltzmann machine



(a) (b)

Figure 2.2: (a) Restricted Boltzmann machine (b) Deep belief network

The second approach to representation learning is the restricted Boltzmann machine (figure 2.2a). RBM learns by maximizing the similarity between the projection of the data and representations. As it has no output, backpropagation cannot be used for training. It is solved by contrastive

divergence [5]. Once the RBM is learned, targets are attached to its output to form a complete neural network.

Deep belief network [6] is formed by stacking one RBM unit after the other. The architecture is shown in figure 2.2b. The targets are attached to the final representation layer, and network weights are tunned using backpropagation.

2.3 Convolutional neural network

Convolutional neural networks are a special kind of multi-layer neural networks. Like other neural networks, CNN are trained with backpropagation, but the architecture is different. CNN is a sequence of layers. Mainly there are three types of layers: convolutional layer, pooling layer, and fully connected layer. These layers are stacked one after the other to form a complete convolutional neural network architecture. Each layer of CNN converts one layer of activations to another through a differentiable function. CNN takes an input image, process it and classify it under certain categories. Each input pass through convolutional layers with filters, pooling layers, fully connected layer, and a softmax function is applied for classification.

figure 2.3 shows the diagram of Lenet-5 architecture introduced by Lecun et al in 1998 [7]. It was designed to recognize handwritten and machine-printed digits. Lenet-5 is 7 layer CNN, among which 3 are convolutional

layers (C_1, C_2 and C_3). The convolutional layer consists of 5 by 5 filters with stride 1. There are two 2 by 2 average pooling layers (S_2 and S_4). The layer F_6 denotes the fully connected layer. Tanh activation function is used throughout the network.

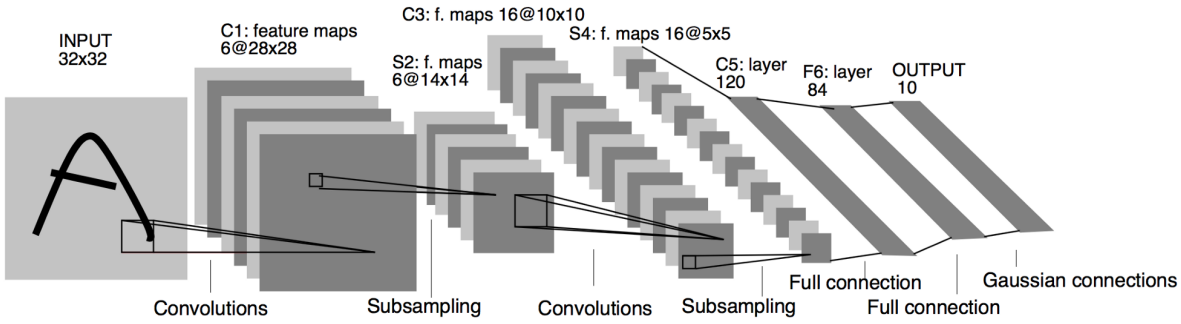


Figure 2.3: Lenet-5 architecture

2.4 Dictionary learning

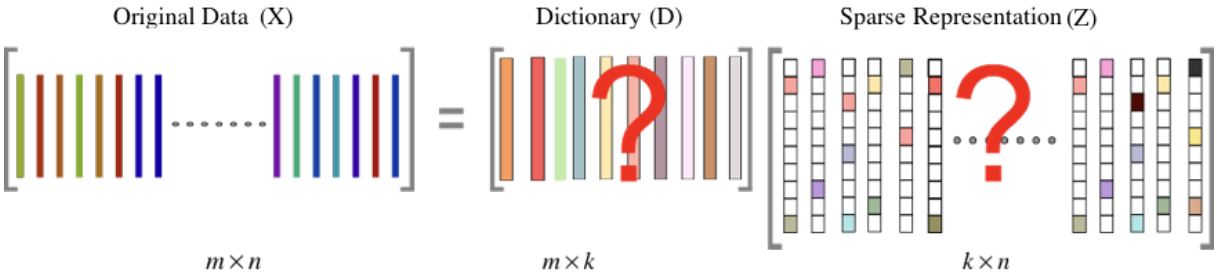


Figure 2.4: Dictionary learning

A dictionary is an over-complete basis, i.e. it has more columns than rows. The columns of a dictionary are often referred to as atoms. These atoms may be the linear combinations of other atoms in the dictionary. The benefit of a dictionary over the basis is that a dictionary may result in even sparser representation as compared to a fixed basis. However, it should be remembered that the dictionary is not ubiquitous, unlike the

fixed transforms. Dictionaries are learned to solve a problem for a particular class of signals; they are not intended to be generalizable to others. DL learns a dictionary/ basis to synthesize the data from the latent representation. Neural network interpretation of DL is shown in figure 2.5.

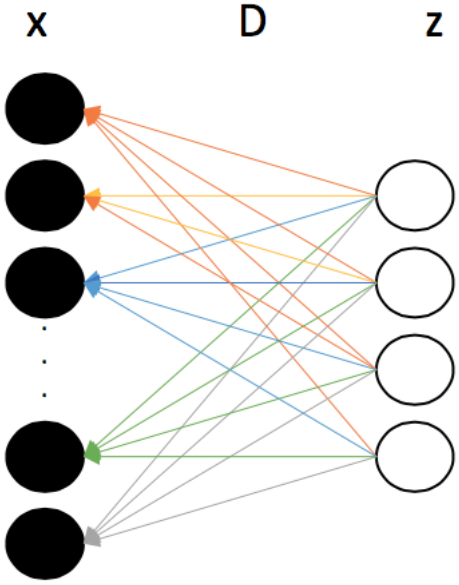


Figure 2.5: Neural network interpretation of DL

Suppose we are given the data $X \in \mathbb{R}^{m \times n}$ such that each column x_i represents a training sample of size m, thus we have total n training samples. Dictionary learning aims at learning a dictionary and the sparse representation of these signals, as shown in figure 2.4. The dictionary learning problem can be expressed as

$$\min_{D \in C, Z \in \mathbb{R}^{k \times n}} \frac{1}{n} \sum_{i=1}^{n} (\frac{1}{2}||x_i - Dz_i||_2^2 + \lambda ||z_i||_1) \tag{2.4}$$

where the constraint set C is defined as

$$C = \{D \in R^{m \times k} s.t \forall j = 1 \dots k, d_j^T d_j \leq 1\}$$

Various algorithms have been proposed in literature to solve the dictionary learning problem. The algorithms like method of optimal directions (MOD) solve for dictionary and sparse representation alternately. The K-SVD algorithm takes a total of K rank-one approximations to determine all K atoms of the dictionary sequentially. The rank-one approximations are done using singular value decomposition (SVD).

The euclidean cost function 2.5 of DL is given by:

$$\min_{D,Z} \|X - DZ\|_F^2 \tag{2.5}$$

2.5 Greedy deep dictionary learning

In deep learning, the idea is to learn multiple levels of dictionaries. It proposes to extend the shallow DL problem into multiple levels. The idea of forming a deep architecture using DL stems from the success of deep learning. Mathematically, a deep dictionary learning (DDL) problem with two levels as in figure 2.6 can be formulated as given in the equation below.

$$X = D_1 \phi(D_2 Z) \tag{2.6}$$

Where, X is the input data, D_1 and D_2 are dictionaries and Z are the

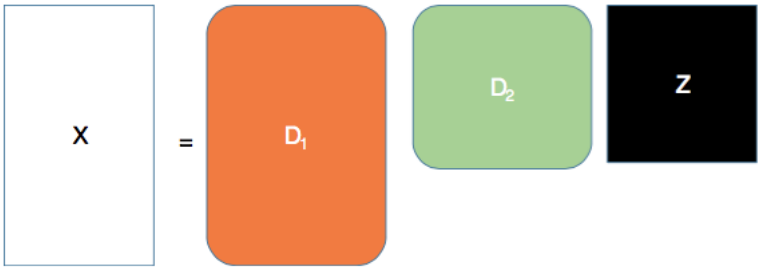


Figure 2.6: Schematic diagram: deep dictionary learning

coefficients. ϕ is the non-linearity. Greedy approach to DDL learns one layer at a time. First layer of the dictionary learns from the training data. The representations and dictionaries are learned for the first layer. in the second layer, representations learned in the first layer act as the input and it learns the dictionary for the second layer. Same concept can be extended to multiple layers. Formulation for DDL with non-linear activations can be expressed as:

$$X = D_1\phi(D_2\phi(...\phi(D_N Z))) \tag{2.7}$$

Where, D_1 to D_N are N level dictionaries and Z are the final level representations. Following optimization problem need to be solved.

$$\min_{D_1, D_2 ... D_N, Z} ||X - D_1\phi(D_2\phi(...\phi(D_N Z)))||_F^2 \tag{2.8}$$

2.5.1 Supervised dictionary learning

To incorporate supervision, a linear map is learned between the coefficients of the last layer and the actual targets. The supervised deep dictionary

learning can be formulated as:

$$\min_{D_N,Z,W} ||\phi^{-1}(Z_{N-1}) - D_N Z||_F^2 + \mu||Q - WZ||_F^2 + \lambda||Z||_1 \tag{2.9}$$

Here, Q are the targets and W is the linear map between the targets and coefficients.

2.6 Convolutional dictionary learning

The reconstruction of a signal x from a sparse representation z with respect to dictionary matrix D is linear, i.e. $x \approx Dz$. The convolutional dictionary learning (CoDL) [8] approach replaces unstructured dictionary D by a set of linear filters d_m. In this case, the reconstruction of signal x from representations z_m is $x \approx \Sigma_m d_m * z_m$. Where x is an image. Mathematically, CoDL can be formulated as:

$$\begin{aligned} &\arg\min_{\{d_m\},\{z_{m,k}\}} \frac{1}{2}\sum_k ||\sum_m d_m * z_m - x_k||_2^2 + \lambda \sum_{m,k} ||z_{m,k}||_1 \\ &s.t. ||d_m||_2 = 1 \forall m \end{aligned} \tag{2.10}$$

The constraint on the norms of filters d_m is required to avoid the scaling ambiguity between filters and coefficients, the training images x_k are considered to be N dimensional vectors, where N is the number of pixels in each image, and the number of filters and the number of training images are M and K respectively.

2.7 Coupled representation learning

The central concept of coupled representation learning is schematically shown in figure 2.7. There are two domains: source and target; a model learns representations from each domain, and a linear map is learned from the representation of the source to that of the target. It has been used in several problems arising in domain adaptation.

So far, there are two popular models for coupled representation learning; based on - dictionary learning and autoencoder. The formulation for CDL [9, 10] is given by -

$$\begin{aligned} &\min_{D_S, D_T, Z_S, Z_T} ||X - D_S Z_S||_F^2 + ||Y - D_T Z_T||_F^2 \\ &+\gamma ||Z_T - M Z_S||_F^2 + \lambda_S ||Z_S||_1 + \lambda_T ||Z_T||_1 \end{aligned} \tag{2.11}$$

Here X and Y are the data for the source and target domains respectively. D_S and the D_T are models (dictionaries) learnt for each of the two domains; their corresponding representations are Z_S and Z_T. M is the linear map coupling the representations. The dictionaries learn a representation in each domain, and there is a map between the representations of the two domains.

For synthesis tasks, once the training is complete, the dictionaries for the source and target are preserved along with the learned linear map; the coefficients are not of any further use. Once a new sample in the source domain arrives, it's the representation. It is obtained by the learned

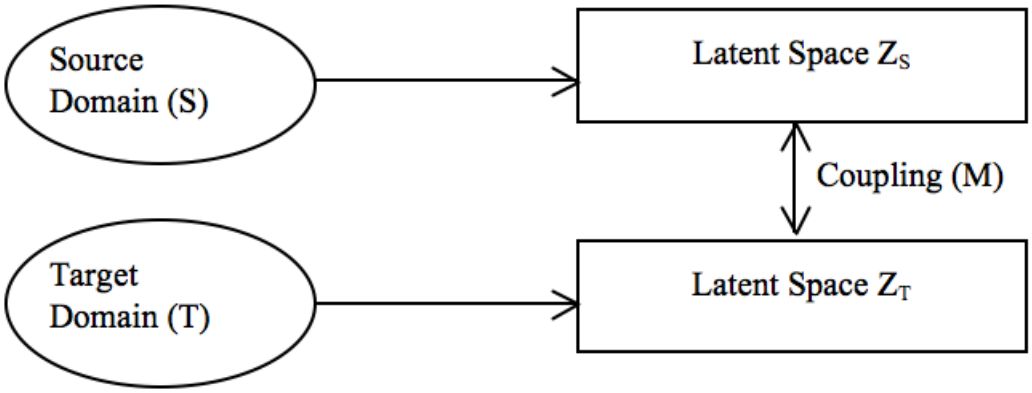


Figure 2.7: Coupled representation learning

dictionary of the source domain. This is further mapped onto the target domain by the learned linear map. The thus obtained representation in the target domain is synthesized into the corresponding signal by the target domain dictionary. For inverse problems, the source domain is usually the corrupted image, and the target domain is the clean image. This approach has been used for super-resolution [9], denoising [10], deblurring [11] and, reconstruction [12].

2.8 Analysis and synthesis formulation

To understand the difference between analysis and synthesis formulation, we need to digress and talk about compressed sensing; it studies the problem of solving an under-determined system of linear equations when the solution is known to be sparse. In general, a problem of the following form has infinitely many solutions;

$$y_{m\times 1} = A_{m\times n} y_{n\times 1}, m < n \tag{2.12}$$

The practical way to solve this problem when the solution is known to be sparse is by minimizing the l1-norm.

$$\begin{aligned} &\min_x ||x||_1 \\ &s.t \\ &y = Ax \end{aligned} \tag{2.13}$$

The l1-norm is defined as the sum of absolute values in the vector. In practice, the system is corrupted by white gaussian noise; the noisy version of equation 2.12 is expressed as,

$$\begin{aligned} &y = Ax + \eta \\ &\eta \in N(0, \sigma^2) \end{aligned} \tag{2.14}$$

In such a situation, the equality constraint of equation 2.13 is relaxed by a quadratic constraint,

$$\begin{aligned} &\min_x ||x||_1 \\ &s.t \\ &||y - Ax||_2^2 \leq \varepsilon \\ &\varepsilon = m\sigma^2 \end{aligned} \tag{2.15}$$

Natural signals are almost never sparse in their physical domain, e.g. biomedical signals like EEG, ECG, MEG and speech are not sparse in time domain and images are never sparse in pixel domain. However most natural signals have an approximately sparse representation in a transform domain, viz. speech is sparse in short time fourier transform, images are sparse in wavelets, biomedical signals are sparse in Gabor, etc. In com-

pressed sensing, we are mostly interested in orthogonal, and tight-frame transforms; both of these follow the analysis-synthesis equations:

- Synthesis:

$$x = T'Z$$

- Analysis:

$$Z = Tx$$

Here x is the signal of interest (dense), T is the sparsifying transform and Z the transform coefficients are assumed to be sparse. Using the synthesis form, the system of equations 2.14 can be expressed as,

$$y = AT'Z + \eta \tag{2.16}$$

Thus, we are back to the sparse regime and the solution can be recovered by,

$$\begin{aligned} &\hat{Z} = \min_{Z} ||Z||_1 \\ &s.t. \\ &||y - AT'Z||_2^2 \leq \varepsilon \end{aligned} \tag{2.17}$$

Once the sparse coefficients are obtained, the signal of interest can be recovered by applying the synthesis equation. This kind of solution 2.17 is called the synthesis formulation; it solves for the sparse coefficients from which the signal is 'synthesized'. Notice that the synthesis formulation is quite stringent; it allows for only a few transforms that are either orthogonal or tight-frame. This precludes many useful transforms like Gabor

or finite-difference. There is an alternate formulation called the co-sparse analysis prior. There is little theoretical understanding of this topic. In lay person's terms, unlike the synthesis prior formulation, the analysis prior formulation solves for the signal itself and not the sparse coefficients. The recovery is expressed as,

$$\hat{x} = \min_{x} ||Hx||_1 s.t. ||y - Ax||_2^2 \leq \varepsilon \tag{2.18}$$

Here H is the analysis operator; it can be any linear operator; it need not be orthogonal or tight-frame. One can see that the analysis and the synthesis prior are the same for orthogonal transforms but not for tight-frames (the proof is trivial). Notice that the analysis prior formulation is more generic. The synthesis prior is a special case of the analysis prior formulation $H = I$ and $A = AT'$. More than 95% of work on compressed sensing and sparse recovery including theoretical studies, algorithms and application areas are based on the synthesis prior formulation.

2.9 Transform learning

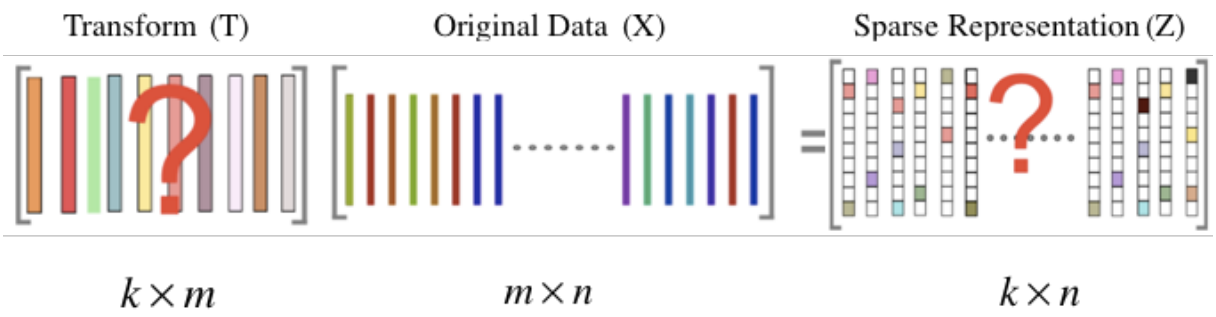


Figure 2.8: Transform learning

TL is the analysis equivalent of dictionary learning. It learns an analysis dictionary / transform (T) such that it operates on the data (X) to generate the coefficients (Z) (see figure 2.8). It's neural network representation is given as figure 2.9. Suppose we are given the data $X \in \mathbb{R}^{m \times n}$ such that each column x_i represents a training sample of size m, thus we have total n training samples. Mathematically this is represented as,

$$\min_{T \in \mathbb{R}^{k \times m}, Z \in \mathbb{R}^{k \times n}} \frac{1}{n} \sum_{i=1}^{n} (\frac{1}{2} ||Tx_i - z_i||_2^2 + \lambda ||z_i||_1) \tag{2.19}$$

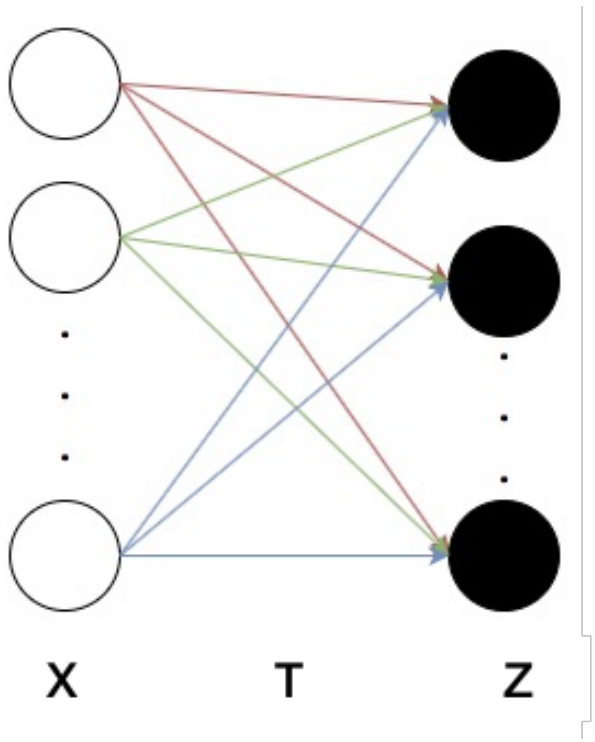


Figure 2.9: Neural network interpretation of TL

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

$$\min_{T,Z} ||TX - Z||_F^2 + \mu ||Z||_0 \tag{2.20}$$

Unfortunately such a formulation would lead to degenerate solutions; it is

easy to verify the trivial solution $T = 0$ and $Z = 0$. In order to ameliorate this the following formulation was proposed in [2].

$$\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu||Z||_0 \tag{2.21}$$

The factor $-\log\det$[1] imposes a full rank on the learned transform; this prevents the degenerate solution. The additional penalty $||T||_F^2$ is to balance scale; without this $-\log\det$ can keep on increasing; producing degenerate results in the other extreme. Note that the sparsity constraint on the coefficients is not mandatory for machine learning problems. It is useful for solving inverse problems in signal processing. In [3], an alternating minimization approach was proposed to solve the TL problem equation 2.21, and the same will be used throughout the thesis.

$$\begin{aligned} T &\leftarrow \min_T ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) \\ Z &\leftarrow \min_Z ||TX - Z||_F^2 + \mu||Z||_0 \end{aligned} \tag{2.22}$$

Updating the coefficients (Z) is straightforward. It can be updated via one step of hard thresholding [13], [14] . This is expressed as,

$$^2 Z \leftarrow (abs(TX) \geq \mu) \bigotimes TX \tag{2.23}$$

Here $\otimes$ represents element-wise product.

For updating the transform, one can notice that the gradients for dif-

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

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

ferent terms in equation 3.1 are easy to compute. Ignoring the constants, this is given by:

$$\begin{aligned} \nabla||TX - Z||_F^2 &= X^T(TX - Z) \\ \nabla||T||_F^2 &= T \\ \nabla \log\det T &= T^{-T} \end{aligned} \tag{2.24}$$

In the initial paper on transform learning [3], a non-linear conjugate gradient based technique was proposed to solve the transform update. In the second paper [4], with some linear algebraic tricks they were able to show that a closed form update exists for the transform.

$$\begin{aligned} XX^T + \lambda\epsilon I &= LL^T \\ L^{-1}YX^T &= QSR^T \\ T &= 0.5R(S + (S^2 + 2\lambda I)^{1/2})Q^T L^{-1} \end{aligned} \tag{2.25}$$

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

The main motivation that can be carried from TL is to use it beyond signal processing. TL has not been used for solving machine learning problems. We explore if TL features can be general enough to solve machine

learning problems, and we have computational cost and run time advantages.

Chapter 3

Supervised transform learning

This chapter introduces certain supervised formulations to transform learning. Four different types of supervision penalties are proposed. The first one is class-sparsity, which imposes common sparse support within representations of each class. The second one imposes similarity among intra-class features in terms of a low-rank constraint (high cosine similarity). The third penalty enforces features of the same class to be nearby each other and features of different classes to be far apart. The final formulation is the well known label-consistency formulation [15, 16], which learns a linear map from the feature space to the class targets. For the first time, we show how TL (and its supervised versions can be kernelized). It also introduces stochastic regularization techniques like drop out and drop connect into the TL formulation. First, this chapter describes the proposed methodology in section 3.1. Then section 3.2 presents the experiments and results. Experiments have been carried out on four different problems: computer vision, bioinformatics, hyperspectral imaging, and ECG based arrhythmia

classification.

3.1 Proposed formulation

Several variants of supervised transform learning (STL) are proposed. Supervision will appear as regularization terms on top of the basic TL formulation equation 3.1. To improve the results further, the kernelized versions of TL are proposed. This is motivated by the success of kernel dictionary learning (KDL) [17, 18, 19]. Finally we discuss how stochastic regularization techniques like DO [20] and DC [21] can significantly improve performance. The basic TL formulation is as follows:

$$\begin{aligned} T &\leftarrow \min_T ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) \\ Z &\leftarrow \min_Z ||TX - Z||_F^2 + \mu||Z||_0 \end{aligned} \tag{3.1}$$

Where $\mu, \lambda, \varepsilon$ are hyperparameters.

In STL, four different types of supervision penalties are proposed. The first one is class-sparsity, which imposes common sparse support within representations of each class. The second one imposes similarity among intra-class features in terms of a low-rank constraint (high cosine similarity). The third penalty enforces features of the same class to be nearby each other and features of different classes to be far apart. The final formulation is the well known label-consistency formulation, which learns a linear map from the feature space to the class targets.

3.1.1 Class-sparse transform learning

The basic formulation for TL is unsupervised, i.e. the class/label information is not required. In recent papers, class-sparsity has been proposed as a viable supervision term. In [22, 23] class-sparse autoencoders have been proposed, in [24] class-sparsity was incorporated in RBM. The main idea of class-sparsity is to impose a common sparse signature across all features of the same class. For example, consider a problem where the number of features is 10, and there are two classes. Class-sparsity would impose that the first-class should have the non-zero values at exactly the same positions, e.g. in say 1, 5 and 7; similarly the second class would have non-zero values at say 2, 3, 6 and 10. Class-sparsity shows excellent improvement in results. The reason for introducing class sparsity can be attributed to the traditional interpretation of the neural network. In a human brain, a class of input activates a given bunch of neurons. Class-wise sparsity tries achieving the same in a neural network. A transform can also be viewed as a neural network; Instead of interpreting the elements as a basis, they can be thought of as connections between the input and the representation. Class-sparsity would enforce representations from the same class to have common support. Assume there are n training images from C distinct classes. The supervised training samples can be represented as:

$$X = [\underbrace{x_{1,1}|\dots|x_{1,n_1}}_{X_1;class1}\underbrace{x_{2,1}|\dots|x_{2,n_2}}_{X_2;class2}\cdots\underbrace{x_{C,1}|\dots|x_{C,n_C}}_{X_C;classC}]$$

where $\{x_{1,1}|\ldots|x_{1,n_1}\}$ represents $X_1; class1$, $\{x_{2,1}|\ldots|x_{2,n_2}\}$ represents $X_{2;class2}$ and $\{x_{C,1}|\ldots|x_{C,n_C}\}$ represents $X_{C;classC}$. We propose to learn the features such that they will have the same sparsity signature across the class, i.e., they will have a common sparse support and Z_C will be row sparse. This is achieved by incorporating $l_{2,1}$-norm regularization [25] as follows: Mathematically class-sparse transform learning (CSTL) is expressed as follows:

$$\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu \sum_c ||Z_c||_{2,1} \qquad (3.2)$$

Here c denotes the class index for C number of classes and $Z = [Z_1|...|Z_c|...|Z_C]$. $[Z_1|...|Z_c|...|Z_C]$ denotes all $Z_i's$ stacked together column-wise.

$$||Z||_{2,1} = \sum_j ||Z^{j\rightarrow}||_2$$

is the sum of l_2-norms of the rows (indicated by j). The inner l_2-norm promotes a dense (non-zero) solution within the selected rows, but the outer l_1-norm (sum) enforces sparsity in selecting the rows. The proposed formulation shown in Equation3.2 enforces row-sparsity within each group. This makes the optimization supervised, i.e., the information regarding the class labels is required to formulate Equation3.2. The formulation enforces supervision by constraining that the features from the same group should have the same sparsity signature. The $l_{2,1}$-norm is a well-known penalty for imposing row-sparsity within the c^{th} class so that all the features have the same common support. The same has been used in [22, 23, 24].

Solving equation 3.2 is straightforward. Alternating minimization leads to:

$$\begin{aligned} T &\leftarrow \min_{T} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) \\ Z_c &\leftarrow \min_{Z_c} ||TX - Z||_F^2 + \mu||Z_c||_{2,1} \end{aligned} \tag{3.3}$$

Update of T is a standard transform update as shown in equation 2.24 and equation 2.25 having a closed form update.

Z can be decoupled for each class,

$$\min_{Z_c} ||TX_c - Z_c||_F^2 + \mu||Z_c||_{2,1} \tag{3.4}$$

The equation 3.4 is easily solved using one step of modified iterative soft thresholding algorithm (ISTA).

This is our first supervised formulation. We call it class sparse transform learning (CSTL). This can be used either for supervised or for unsupervised representation learning. For the unlabeled samples in semi-supervised learning, the $l_{2,1}-$norm will boil down to the sparsity promoting l_1-norm.

3.1.2 Low-rank transform learning

A good representation learning method would enforce similarity within the features of the same class. This would mean that the features would have a high cosine similarity. If all the features from the same class are stacked as columns of a matrix, the resultant would be rank-deficient. This is because the features would be similar to each other, and hence linearly dependent.

To achieve this explicitly, a low-rank penalty needs to be imposed on the class-wise feature matrix. This leads to the following formulation:

$$\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu \sum_c ||Z_c||_* \tag{3.5}$$

The nuclear norm [26] acts as the tightest convex surrogate of rank and has been used profusely in signal processing and machine learning to obtain low-rank solutions.

The solution for equation 3.5 can be obtained by alternating minimization. The transform update would remain exactly the same as before equation 2.25. The update for the coefficients is given by the following problem,

$$\begin{aligned} &\min_{Z} ||TX - Z||_F^2 + \mu \sum_c ||Z_c||_* \\ &\min_{Z_c} ||TX_c - Z_c||_F^2 + \mu||Z_c||_* \end{aligned} \tag{3.6}$$

The features for each class can be updated by one step of singular value shrinkage. This technique is called as low-rank transform learning (LRTL). The disadvantage of this technique is, if there are too few samples in any class, the low-rank assumption does not hold. This, in turn, might degrade the performance. This technique works best when there is an approximately even distribution of large numbers (at least having the same order as the feature size) of samples in each class. Furthermore, this technique is not as directly amenable to semi-supervised representation learning.

3.1.3 Discriminative transform learning

The aforesaid formulations, although supervised, are not discriminative, i.e. they enforce similarity between representations of the same class but do not discriminate between classes. In the next formulation, discrimination is enforced by enforcing intra-class similarity and inter-class discrimination. For each class 'c' it enforces the features of the same class to be clustered near each other. This is achieved by minimizing the total distance between each sample of the class with its class mean, i.e. $||Z_c - \bar{Z}_c||_F^2$. Here is the class mean repeated the same number of times as the number of samples in the class 'c'. Thus, it is arranged as a matrix. This reduces intra-class distance. To enforce discrimination, it also needs to enforce maximize the inter-class distances. This is achieved by maximizing the distances between all pairs of class means. Combining the two, the formulation is given as follows:

$$\begin{aligned} &\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu||Z||_1 \\ &+\eta_1 \sum_c ||Z_c - \bar{Z}_c||_2 - \eta_2 \sum_c \sum_{j \neq c} ||\bar{Z}_c - \bar{Z}_j||_2 \end{aligned} \tag{3.7}$$

Here $\sum_c \sum_{j \neq c} ||\bar{Z}_c - \bar{Z}_j||_2$ is for maximizing the class distances from each other. The term $\sum_c ||Z_c - \bar{Z}_c||_2$ minimizes the distance within class. This is called discriminative transform learning (DiTL). With little simplification, it can be shown that the term $||Z_c - \bar{Z}_c||_2$ can be represented as $||M_c Z_c||_2$ where

$$M_c = I - \frac{1_c}{n_c}$$

Here n_c is the number of samples in the class and 1_c denotes a matrix of all 1_c of dimensionality $n_c \times n_c$.

Similarly, we can express

$$\bar{Z}_c = Z_C \frac{1_c}{n_c}$$

where ${1'}_c$ denotes a matrix of dimension $n_c \times \nu$. ν being the dimensionality of the representation. Therefore

$$||\bar{Z}_c - \bar{Z}_j||_2$$

can be represented as

$$|\bar{Z}_c - \bar{Z}_j||_2 = ||Z_c \frac{{1'}_c}{n_c} - Z_j \frac{{1'}_j}{n_j}||_2 \tag{3.8}$$

With these simplifications, equation 3.7 can be expressed as,

$$\begin{aligned} &\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu||Z||_1 \\ &+\eta_1 \sum_c ||M_c Z_c||_2 - \eta_2 \sum_c \sum_{j \neq c} ||Z_c \tfrac{{1'}_c}{n_c} - Z_j \tfrac{{1'}_j}{n_j}||_2 \end{aligned} \tag{3.9}$$

As before, it is solved using alternate minimization. The update for the TL step remains unchanged. The update for features for each class (after

decoupling) can be expressed as,

$$\begin{aligned} &\min_{Z_c} ||TX_c - Z_c||_F^2 + \mu||Z_c||_1 \\ &+\eta_1 \sum_c ||M_c Z_c||_2 - \eta_2 \sum_c \sum_{j \neq c} ||Z_c \frac{1'_c}{n_c} - Z_j \frac{1'_j}{n_j}||_2 \end{aligned} \tag{3.10}$$

This is a $l1-$regularized least-squares problem. It can be solved using ISTA [14]. This concludes the steps of the training algorithm. We name it discriminative transform learning (DTL). Note that this formulation is naturally amenable to semi-supervised learning. For samples that do not have class information, only the sparsity penalty is imposed (without the discriminative terms); for labelled samples, both the sparsity penalty and the discriminative penalties are imposed.

3.1.4 Label consistent transform learning

Label consistency penalties proved to be immensely successful in machine learning tasks. It was introduced in [27] as label consistent K-SVD. A slightly varied formulation was proposed in [28] with the title 'discriminative K-svd'. However, the equivalence between the two has been shown in [29]. Later label-consistent formulation has been used in autoencoder [30] as well. The basic idea in label consistency is to learn a linear map that projects the sparse features into the target variables. The same idea is used here. The label consistent transform learning (LCTL) formulation

is expressed as follows,

$$\begin{aligned}&\min_{T,Z} ||TX - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) + \mu||Z||_1 \\ &+\eta||Q - MZ||_2\end{aligned} \tag{3.11}$$

Here M is the linear classifying map and Q are the target (binary) class labels. We call this the LCTL.

The LCTL formulations can be solved using alternating minimization. The transform update step remains the same as before. But in this formulation we need to learn an additional linear map M; this is obtained by

$$\min_{M} ||Q - MZ||_2 \tag{3.12}$$

This has a closed-form solution in the form of pseudo-inverse. For the LCTL formulation, the update for the sparse coefficients remains the same as that of modified soft thresholding as in the CSTL formulation.

3.1.5 Testing

For the first three techniques, supervised representation is learned. The features obtained from the training stage is used to learn a separate classifier. During testing, the sparse feature is obtained using the learned transform. This requires solving,

$$\min_{Z_{test}} ||TX_{test} - Z_{test}||_2 + \mu||Z_{test}||_1 \tag{3.13}$$

This has a closed form update via soft thresholding. The thus generated sparse feature is input to the learnt classifier for class assignment.

Note that the test phase of transform learning is much faster than that of dictionary learning. We only need a soft thresholding operation following a matrix-vector product to solve (22) [31]. In dictionary learning, one needs to solve an iterative optimization problem for generating the sparse codes (l1-minimization). For the LCTL techniques, one can use the generated features for training a separate classifier or follow the technique in label consistent K-SVD [31]. After generating the sparse features, the linear map (M) is used to obtain the target labels: $\hat{q} = Mz_{test}$. The generated target is not a binary vector. But the label can be assigned from the position of the highest value in $\hat{q}$.

3.1.6 Kernel transform learning

The idea of kernel transform learning (KTL) gets motivation from the success of KDL [17, 18], which in turn extends from the concept of double sparsity [30]. Instead of learning a dense dictionary, it learns a dictionary that is produced as a sparse combination of elements from a known basis (e.g. wavelet, DCT). Let $\varphi : \mathbb{R}^N \rightarrow F$ be a non-linear mapping from $\mathbb{R}^N$ into a higher dimensional feature space F. Since the feature space, F can be very high dimensional, in the kernel methods, kernels are usually employed to carry out the mapping implicitly. A kernel is a function that for all data x_i gives rise to a positive semi-definite matrix $K(x_i, x_j)$. It corresponds to

mapping the data with some mapping φ into a feature space F instead of the dot product in input space.

The formulation is expressed as,

$$X = \Phi A Z$$

Here Φ is the fixed basis, ΦA is the dictionary expressed as a sparse combination (A) of basis from Φ. The idea of decomposing the dictionary into a fixed portion and a learned variable stems from the concept of double sparsity. The formulation for kernel dictionary learning is given by,

$$\varphi(X) = \underbrace{\varphi(X)A}_{Dictionary} Z \tag{3.14}$$

The non-linear transformation on the data $\varphi(X)$ is synthesized from the coefficients (Z) from a dictionary formed by itself and a combination of its elements $\varphi(X)A$. The formulation for learning is,

$$\min_{A,Z} ||\varphi(X) - \varphi(X)AZ||_F^2 \qquad s.t. \qquad ||Z||_0 \leq \tau \tag{3.15}$$

The problem is solved using alternate minimization. The update step for A is actually independent of the data, since

$$A \leftarrow \min_A ||\varphi(X)(I - AZ)||_F^2$$
$$A = Z^T(Z^T Z)^{-1}$$

The update for the sparse coding stage is a solved problem via kernel matching pursuit (KMP) [32]. A more efficient solution via the Nystrom method

is proposed in [24].

Using the kernel method, we describe how the TL approach can be made nonlinear. It is shown that nonlinear TL approach can provide better discrimination compared to its linear counterpart, especially when the data is corrupted by noise.

Let us rewrite the basic transform learning formulation $TX = Z$ in kernelized form. The transform can be written as $B\varphi(X)^T$ and data in higher dimensional feature space can be expressed as $\varphi(X)$. Hence, the KTL equation can be written as:

$$\underbrace{B\varphi(X)^T}_{Transform}\varphi(X) = Z \tag{3.16}$$

The advantage of our proposed formulation is that, one can define the kernel upfront,

$$K(X, X) = \varphi(X)^T\varphi(X)$$

This allows expressing equation 3.16 as,

$$BK(X, X) = Z \tag{3.17}$$

Comparison between transform learning equation 2.19 and our proposed formulation equation 3.17 is that instead of the data matrix, we have the kernelized data matrix. The usual constraints of TL will apply. We formu-

late the learning as,

$$\min_{B,Z} ||BK(X,X) - Z||_F^2 + \lambda(\varepsilon||B||_F^2 - \log\det B) + \mu||Z||_0 \tag{3.18}$$

One can see that the update for B is the same as that of the transform and the update for Z remains the same as that of sparse coefficients. This concludes the training phase. For testing, the corresponding expression will be:

$$B\varphi(X)^T\varphi(x_{test}) = z_{test} \tag{3.19}$$

The kernel is automatically defined as:

$$K(X, x_{test}) = \varphi(X)^T\varphi(x_{test})$$

This allows expressing equation 3.19 as follows,

$$BK(X, x_{test}) = z_{test}$$

Since, one looks for sparse features, one needs to solve,

$$\min_{z_{test}} ||BK(X, x_{test}) - z_{test}||_F^2 + \mu||z_{test}||_0$$

This is solved by just one step of hard thresholding. We have shown the kernelized version for the basic TL. Our supervised versions follow directly; one only needs to replace the raw samples by their kernels and the transform by B; the rest remains the same. In short, here are the

kernelized versions of the four different supervision penalties introduced in this chapter.

- Kernel CSTL:

$$\min_{B,Z} ||BK(X,X) - Z||_F^2 + \lambda(\varepsilon||B||_F^2 - \log\det B) + \mu \sum_c ||Z_c||_{2,1}$$

- Kernel LRTL:

$$\min_{B,Z} ||BK(X,X) - Z||_F^2 + \lambda(\varepsilon||B||_F^2 - \log\det B) + \mu \sum_c ||Z_c||_*$$

- Kernel DiTL:

$$\begin{aligned}&\min_{B,Z} ||BK(X,X) - Z||_F^2 + \mu||Z_c||_1 \\ &+\lambda(\varepsilon||B||_F^2 - \log\det B) \\ &+\eta_1 \sum_c ||M_c Z_c||_2 - \eta_2 \sum_c \sum_{j \neq c} ||Z_c \frac{1'_c}{n_c} - Z_j \frac{1'_j}{n_j}||_2\end{aligned}$$

- Kernel LCTL:

$$\begin{aligned}&\min_{B,Z,M} ||BK(X,X) - Z||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) \\ &+\mu||Z||_1 + \eta||Q - MZ||_F^2\end{aligned}$$

3.1.7 Stochastic regularization

The idea of stochastic regularization (in neural networks) is fairly new. These techniques do not have the firm mathematical backing of deterministic regularization but have heuristic understanding. We discuss and adopt two different stochastic regularization techniques that have been recently

proposed in neural networks.

The main idea in DO [20] is to randomly drop units (along with their connections) from the network during training. This prevents nodes from co-adapting too much. Suppose we have training data X; in every iteration of DO some randomly chosen output units along with their connection weights are set to zero, as shown in figure 3.1. Here, out of three output neurons, z (selected randomly) is dropped. This idea can be adopted to

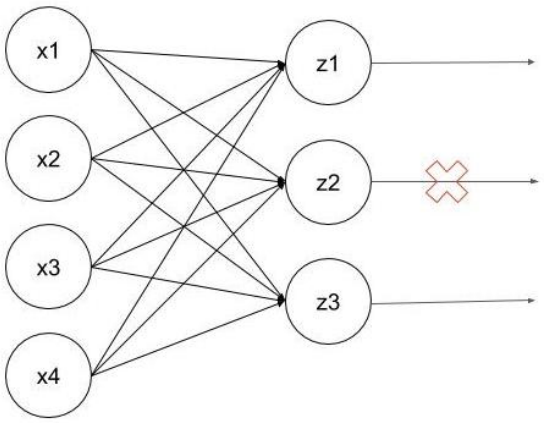


Figure 3.1: Drop out

TL. For that, we would need to interpret TL in a slightly different manner. The usual interpretation of TL is to think of the transform as an analysis basis which operates on the data to produce coefficients: this is shown in the left of figure 2.9. Instead of thinking of the transform elements as a basis, they can alternately be interpreted as connections between the input and the representation. This is more akin to a neural network interpretation shown in the right of figure 2.9. With the neural network interpretation, DO would require randomly putting some representations

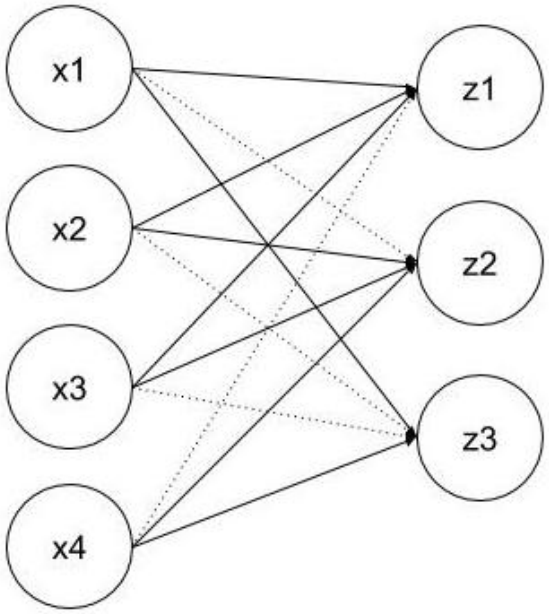


Figure 3.2: Drop connect

to zero. This can be easily achieved. The updates for the coefficients and transform are,

$$\begin{aligned} Z_k &\leftarrow \min_Z ||T_{k-1}X - Z||_F^2 + \mu||Z||_0 \\ T_k &\leftarrow \min_T ||TX - Z_k||_F^2 + \lambda(\varepsilon||T||_F^2 - \log\det T) \end{aligned} \tag{3.20}$$

In every iteration (k), before using Z_k to update the transform T_k, some of the elements in Z_k are randomly put to zero. This yields the desired effect of DO.

We have shown it here for the unsupervised formulation. But it is trivial to extend it to the supervised and kernelized formulations. The other approach to stochastic regularization is DC, as shown in figure 3.2. In this, the connections between successive layers of nodes are dropped. In our neural network type interpretation of TL, it would mean that some of

the elements in the transform are dropped after every iteration. This can be achieved by putting some of the elements T_k in equation 3.20 to zero before using the transform to update Z_{k+1} in equation 3.20 in the next iteration.

There is a heuristic interpretation of drop out and drop connect. By dropping the connections or representations, the architecture of the neural network changes. Over iterations, an average of many such architectures is apparently learned. This prevents over-fitting.

3.2 Experiments and results

3.2.1 Computer vision

We carry out classification on some benchmark computer vision datasets: face Recognition (YaleB and AR), object categorization (Caltech 101) and scene categorization (Scene 15). All these datasets are well known. The extended yale face database B (YaleB) includes $2,414$ face images of 38 persons under 64 illumination conditions, which is challenging due to plentiful expressions and varying illumination conditions. All the original images are cropped to 192×168 pixels and then projected onto $504-$dimensional vectors with a randomly generated matrix (i.i.d gaussian) to obtain random-face features. Following the common settings for this database, we chose one-half of the images for training, and the remaining samples were used for testing. The AR face database contains more than $4,000$ colour face

images of 126 people. Each person has 26 frontal face images which are taken during two sessions. This database includes frontal views of faces with different facial expressions, lighting conditions, and occlusion conditions (sunglasses and scarves). All the images are cropped and scaled to 165×120. We followed a common evaluation protocol in our experiments for this database, in which we used a subset of 2600 images pertaining to 50 males and 50 female subjects. For each subject, we randomly chose 20 samples for training and the rest for testing. The Caltech 101 database comprises of $9,144$ images from 102 classes. Each category has 31 to 800 images. We use the standard bag-of-features (BoF) + spatial pyramid matching (SPM) frame for feature extraction. 30 images per category are randomly selected for training and the remaining for testing. The number of samples in each category of the fifteen scene dataset (Scene 15) ranges from 200 to 400, and the average image size is around 250×300 pixels. This database contains 15 scenes, such as kitchen, bedroom, and country scenes. The feature extraction scheme remains the same as in Caltech 101. Following the common experimental settings, 100 images per category are randomly chosen as training data with the rest as testing data. We compare our proposed techniques with discriminative bayesian dictionary learning (DBDL) [33], multimodal task driven dictionary learning (MTDL) [34], discriminative analysis dictionary learning (DADL) [35], sparse embedded dictionary learning (SEDL) [36] and non linear dictionary learning (NDL) [37] and label consistent dictionary learning (LCDL)

[38]. For the proposed supervised dictionary learning techniques, we show results with the basic formulations, with the kernelized formulation, and with stochastic regularization. The parametric values used in this work are shown in table 3.1. These values have been obtained on a validation dataset (CIFAR−10); they have not been tuned for the datasets used in this work.

Table 3.1: Parameter values

Method	**Parameters**
CSTL	$\lambda = 0.1; \mu = 0.05$
LRTL	$\lambda = 0.1; \mu = 0.05$
DiTL	$\lambda = 0.1; \mu = 0.05; \eta_1 = 0.25; \eta_2 = 0.1$
LCTL	$\lambda = 0.1; \mu = 0.05; \eta = 1$

The experimental results are shown in Tables 3.2. Our proposed methods: CSTL, LRTL and DiTL require separate classifiers. In this work, we use a simple nearest neighbour, classifier. For all the kernelized versions we have used a polynomial kernel of order 3. In the following tables, DO and DC refer to stochastic regularizations and 'K' refers to the kernelized version. The kernel class-sparse transform learning (K-CSTL), kernel low-rank transform learning (K-LRTL), kernel discriminative transform learning (K-DiTL) require separate classifiers but kernel label consistent transform learning (K-LCTL) doesn't require any external classifier. For both DO and DC, 5% dropping was used. For all our formulations, the number of basis elements used was half the dimensionality of the input data.

The results show certain trends. First, kernelization helps; we always

Table 3.2: Classification accuracy

Method	YALEB	AR faces	Caltech-101	Scene-15
DBDL[33]	97.2	97.4	74.6	98.7
MTDL[34]	97.0	97.1	73.2	97.1
DADL[35]	97.7	98.7	74.6	98.3
SEDL[36]	96.6	94.2	81.9	96.2
NDL[37]	91.8	92.1	62.8	83.7
LCDL[38]	92.7	94.6	64.4	86.7
CSTL	96.2	97.1	73.6	96.8
CSTL-DO	96.4	97.3	73.7	97.0
CSTL-DC	96.5	97.3	73.9	97.1
K-CSTL	97.1	97.6	74.1	97.5
K-CSTL-DO	97.2	97.8	74.2	97.6
K-CSTL-DC	97.4	97.8	74.2	97.6
LRTL	96.0	96.6	72.8	96.2
LRTL-DO	96.2	96.8	73.0	96.4
LRTL-DO	96.3	96.9	73.1	96.6
K-LRTL	96.8	97.4	73.6	97.0
K-LRTL-DO	97.0	97.5	73.8	97.1
K-LRTL-DC	97.0	97.6	73.8	97.3
DiTL	97.0	98.0	**74.6**	97.9
DiTL-DO	97.1	98.2	**74.8**	98.1
DiTL-DC	97.2	98.4	**74.8**	98.2
K-DiTL-DO	**97.9**	**98.9**	**75.6**	**98.9**
K-DiTL-DC	**97.9**	**98.9**	**75.7**	**98.9**
LCTL	**97.8**	**98.8**	**75.1**	98.6
LCTL-DO	**98.0**	**98.9**	**75.2**	**98.8**
LCTL-DC	**98.1**	**99.0**	**75.4**	**98.9**
K-LCTL	**98.4**	**99.2**	**75.9**	**99.1**
K-LCTL-DO	**98.5**	**99.4**	**76.1**	**99.3**
K-LCTL-DC	**98.6**	**99.4**	**76.2**	**99.3**

get better results after kernelization compared to its linear counterpart. Second, the stochastic regularization techniques improve results for both the linear and kernel formulations. Third, of the two stochastic regularization techniques, DC is always as good or better than DO. The LRTL formulation yields the worst results among all the proposed formulations. The class sparsity formulation improves upon LRTL but is not able to produce better results than the prior techniques. The DiTL formulation

improves further, but the linear formulation is not always able to beat the previous best. But with the kernelized version of the DiTL, we are able to surpass prior results. The LCTL formulation almost always yields the best results (except for scene 15). With kernelization and stochastic regularization, LCTL improves even further. In computer vision, deep learning is popular these days. But note that the results of deep learning on these datasets are not very high. As the article [39] reveals, well known CNN architectures perform sub-par (less than the methods compared here) Yale B and AR face datasets. This is mainly due to the non-availability of too many training samples. Similarly, [40] reports results on several CNN based architectures for scene 15; all of them report accuracies lower than the ones here.

3.2.2 Bioinformatics

We apply our techniques to two well-known problems in bioinformatics. The first one is the prediction of DNA methylation state using genome topological features. A recent study applied deep learning (stacked autoencoder) to address the said problem [41]. The work described above is dubbed DeepMethyl; it predicts the methylation state of DNA CpG dinucleotides using features inferred from three−dimensional genome topology (based on Hi−C) and DNA sequence patterns. It is a binary classification problem (methylated or unmethylated). The experimental data are from immortalized myelogenous leukaemia ($K562$) and healthy lymphoblastoid

($GM12878$) cell lines. Following [42], DeepMethyl [41] used sequential features generated within a window of the genome and features generated from the three-dimensional topology of a genome indicated by the Hi−C experiment. Two sets of experiments have been proposed in the aforesaid study: Benchmark 1. the methylation level of sequential neighbouring regions is included as features. Benchmark 2. Excluding the methylation level as features to increase difficulty. The details of the prior experiments and feature extraction are given in [41]. The results are shown in Table 3.3. Taking cues from the previous set of experiments, we do not show results for all the different techniques. We only show results for the kernelized versions along with stochastic regularization using DC. An RBF kernel has been used throughout. The dropping rate for DC has been fixed at 5%. For CSTL, LRTL and DiTL; an SVM classifier is used (since it is a binary classification problem). As a benchmark, basic TL is used as a benchmark.

Table 3.3: Classification accuracy compared with DeepMethyl

Method	**Benchmark−1**		**Benchmark−2**	
	GM12878	**K562**	**GM12878**	**K562**
DeepMethyl[41]	89.7	88.6	84.8	72.0
TL	87.9	86.6	83.0	69.9
KTL	88.2	87.2	83.3	70.6
KTL-DC	88.8	87.6	84.0	71.3
K-CSTL	**90.1**	**89.2**	**85.1**	**72.0**
K-CSTL-DC	**90.8**	**90.5**	**85.9**	**72.7**
K-LRTL	88.9	88.0	83.1	71.1
K-LRTL-DC	89.3	88.3	83.8	71.4
K-DiTL	**91.7**	**91.6**	**86.0**	**72.6**
K-DiTL-DC	**92.4**	**92.1**	**86.8**	**74.3**
K-LCTL	**92.6**	**92.0**	**87.0**	**75.1**
K-LCTL-DC	**93.2**	**92.8**	**87.6**	**75.8**

For all of our formulations, 100 transform basis was used. The results

show that the basic TL formulation (linear or kernelized) does not outperform the existing stacked autoencoder based formulation. The LRTL formulation improves slightly upon the basic formulation but still does not beat the prior method. The LCTL formulation improves over DeepMethyl. The DiTL and LCTL formulations improve upon further.

Our second experiment is carried out with DeepChrome [43]. This problem takes histone modifications as input and predicts gene expression as a classification task on 56 different cell- types from REMC database [44]. In [43], the $10,000$ basepair (bp) DNA region (+/- 5000 bp) around the transcription start site (TSS) of each gene was divided into bins of length 100 bp. Each bin includes 100 bp long adjacent positions flanking the TSS of a gene. In total, five core histone modification marks from the REMC database were considered [44]. These five histone modifications were selected as they are uniformly profiled across all cell-types considered in the study. This makes the input for each gene a 5×100 matrix, where columns represent different bins and rows represent histone modifications. This constitutes the input feature. The aforesaid work used a CNN for prediction. For the parameter settings and experimental protocol, we followed [43]. In this work, we employ TL after vectorizing the 5×100 dimensional input feature. DeepChrome [43] reports average AUC as the evaluation metric. The results are shown in the following table. In prior studies, we have seen that DO always performs better than DC; hence, in this final experiment, we only use the later.

Table 3.4: DeepChrome: AUC comparison

SVM	0.75	
DeepChrome[43]	0.80	
	Without DC	**With DC**
KTL	0.72	0.75
K-CSTL	0.79	0.82
K-LRTL	0.75	0.77
K-DiTL	**0.82**	**0.83**
K-LCTL	**0.83**	**0.84**

In [43], DeepChrome was compared with SVM. The results are the same as we obtained. We find that the KTL formulation does not improve results; in fact, with SVM, the results are exactly the same as obtained on raw data. When supervision is added, results improve. As before, LRTL only improves slightly over the basic TL formulation. CSTL shows further improvement. With DC, it beats DeepChrome. Results improve even further with the DiTL formulation. The best results are obtained with the LCTL formulations.

3.2.3 Hyperspectral image classification

We evaluate our proposed technique on the problem of hyperspectral image classification; the datasets are Indian Pines which has 200 spectral reflectance bands after removing bands covering the region of water absorption and 145×145 pixels of sixteen categories, and the Pavia University scene which has 103 bands of 340×610 pixels of nine categories. In this work, we follow the standard evaluation protocol on these datasets. For both the first datasets, we follow the standard protocol; we randomly

select 5% of the labelled data as the training set and rest as the testing set. This is repeated 100 times, and the average accuracies are reported. In this work we have compared with several state-of-the-art techniques: spectral-spatial shared linear regression (SSLR) [45], hierarchical spectral-spatial features (HSSF) [46], CNN [47] and group-sparse low-rank representation (GSLR) [48]. We compare with these techniques since they are the latest ones and comprise an even mix of shallow and deep techniques. For our proposed technique, we input the raw pixel values: there is no requirement of extracting spatial and spectral features as needed in prior techniques [45, 46, 47]. For tuning the parameters, Salinas has been used as a validation set. We found that the same parameter values, as shown in table 3.1 yields outstanding results. The number of transform basis used in these experiments were 50. For the CSTL, LRTL and DiTL, we need a separate classifier. Here the group sparse representation-based classifier is used [49]. In all cases (CSTL, LRTL, DiTL and LCTL) an RBF kernel has been used along with DC regularization (with 5% dropping).DO gives worse results than DC and hence has not been shown here. The experimental results are shown in tables 3.5 and 3.6. The overall accuracy (OA), average accuracy (AA) and Kappa are used for evaluation metrics.

One cannot compare the results obtained here from the CNN based technique [47] and SSLR [45] with the corresponding papers since the volume of the training data used there is much higher compared to ours. Therefore it is reasonable to expect a drop in the accuracy. Even for other techniques,

Table 3.5: Classification accuracy on Indian Pines

Class	K-CSTL-DC	K-LRTL-DC	K-DiTL-DC	K-LCTL-DC	GSLR[48]	SSLR[45]	HSSF[46]	CNN[47]
1	75.00	95.83	**97.92**	95.83	96.08	82.58	81.32	86.58
2	96.43	96.67	97.21	**98.15**	95.30	90.73	80.51	82.68
3	95.47	90.93	96.67	**96.93**	96.72	87.84	79.75	84.36
4	86.19	85.71	93.33	91.14	**95.05**	85.13	68.57	90.33
5	96.42	93.74	**95.75**	91.21	90.25	80.92	79.25	93.88
6	98.66	97.32	**99.55**	91.11	97.74	92.24	95.18	95.99
7	82.61	69.57	60.87	**100.00**	91.67	85.00	95.20	88.67
8	97.95	98.41	**100.00**	99.97	98.92	98.14	93.08	96.49
9	50.00	55.56	50.00	**100.00**	84.21	80.00	89.47	97.00
10	93.80	93.80	94.60	**96.70**	90.64	88.86	77.85	87.35
11	95.54	94.37	**99.28**	97.20	95.22	82.13	85.62	74.90
12	91.85	93.66	95.65	96.73	91.60	81.57	69.95	92.82
13	**100.00**	99.47	**100.00**	**100.00**	**100.00**	95.68	96.36	95.59
14	95.96	99.14	**99.83**	99.48	97.80	95.26	96.28	95.55
15	88.01	87.43	91.81	**98.37**	83.10	80.46	83.16	84.41
16	88.24	**100.00**	91.76	95.29	92.22	97.67	96.06	95.06
OA	95.10	94.86	97.83	97.97	94.75	88.19	84.70	84.81
AA	92.55	90.73	91.85	97.81	93.53	87.76	85.48	90.12
Kappa	0.94	0.94	0.96	0.97	0.94	0.88	0.83	0.82

Table 3.6: Classification accuracy on Pavia

Class	K-CSTL-DC	K-LRTL-DC	K-DiTL-DC	K-LCTL-DC	GSLR[48]	SSLR[45]	HSSF[46]	CNN[47]
1	89.56	82.23	94.40	**99.64**	84.47	89.21	98.79	89.06
2	79.98	72.47	**95.91**	92.68	93.02	98.48	99.85	89.80
3	85.45	82.26	**91.71**	90.06	75.65	84.34	87.10	80.67
4	98.66	98.56	96.22	**98.94**	97.05	94.16	93.99	90.85
5	99.91	99.82	99.10	**100.00**	99.73	98.86	99.61	95.91
6	95.76	93.92	94.45	**98.77**	92.32	82.66	98.34	91.17
7	97.96	92.46	95.32	**98.81**	94.80	80.31	92.62	89.68
8	96.43	78.98	95.37	**98.54**	92.95	88.69	93.97	86.09
9	98.49	96.98	96.48	96.48	**99.37**	90.18	94.20	94.84
OA	90.65	81.07	94.27	97.89	91.45	90.16	97.53	89.28
AA	93.58	88.61	95.44	97.10	92.15	89.65	95.38	89.55
Kappa	0.91	0.84	0.95	0.98	0.88	0.89	0.97	0.87

the results may not be directly comparable, since they handpick the spectral bands; in here all the spectral bands are used. The results show that our proposed label consistent formulation yields the best results as always. It also has the highest number of class-wise accuracies. Closely following is the discriminative formulation. The class-wise sparse regularization yields better results than the ones compared against but falls short of the DiTL and LCTL formulations. The LRTL formulation yields the worse results among the proposed techniques. Here we do not report the standard

deviations in each class (some studies indicate that). But we performed simple statistical t-tests between mutual pairs of close performing methods to see (at 99% confidence interval) if they are significantly different. The summarized results are as follows: The results of the t-test reveal that LCTL and DiTL are considerably different for both the datasets. Apart from GSLR nothing comes close by our proposed techniques for the Indian Pines dataset. GSLR is significantly better than LRTL but is worse than the rest of the proposed methods (LCTL, DiTL and CSTL). On the Pavia dataset, CNN is significantly better than LRTL. Both are worse than all others. CSTL is significantly better than SSLR but worse than all others. It is even worse than GSLR. The existing technique HSSF is significantly better than our proposed DiTL but is worse than our LCTL.

3.2.4 Arrhythmia classification

Here, five types of beat classes of arrhythmia as recommended by association for the advancement of medical instrumentation (AAMI) were analyzed from electrocardiogram (ECG) signals namely: non-ectopic beats, supraventricular ectopic beats, ventricular ectopic beats, fusion beats and unclassifiable and paced beats. The classification experiments are carried out on the MIT-BIH arrhythmia dataset from www.physionet.org. The MIT-BIH arrhythmia database contains 48 half-hour recordings of two-channel ambulatory ECG, obtained from 47 subjects in the year 1975 and 1979 by the beth-Israel hospital arrhythmia laboratory at Boston. Twenty-

four-hour ambulatory ECG recordings were collected from a mixed population of size 4000 having inpatients (around 60%) and outpatients (around 40%). The recordings were digitized at 360 samples per second per channel with an 11-bit resolution over a 10 mV range. Two or more cardiologists independently annotated each record; the consensus was made to obtain the computer-readable reference annotations for each beat. For classification experiments, the MIT-BIH protocol is converted to the AAMI / ANSI standard. This leads to 5 classes - non-ectopic beat (N), supra-ventricular ectopic beats (S), ventricular ectopic beats (V), fusion beat (F) and unknown beat (Q). Owing to the relative sparsity of samples in the F and Q class, they are merged with V; this is following the AAMI2 protocol proposed in [50]. For the experimental protocol we follow [50]; this is repeatable protocol. The division into test set and training set is shown in Table I. The record number # of the patient used for training are: 101, 114, 112, 207, 223, 106, 115, 124, 208, 230, 108, 116, 201, 209, 109, 118, 203, 215, 112, 119, 205, 220; for testing are: 100, 117, 210, 221, 233, 103, 121, 212, 222, 234, 105, 123, 213, 228, 111, 200, 214, 231, 113, 202, 219, 232.

Table 3.7: Train and test set details

Dataset	**N**	**S**	**V**	**F**	**Q**	**Total**	**# Rec**
Train	45844	943	3788	415	8	50998	22
Test	44238	1836	3221	388	7	49690	22
Total	90082	2779	7009	803	15	100688	44

The proposed technique is compared with two recent works [41, 42]. In

[41] deep neural networks built upon stacked autoencoder and deep belief networks are used. In [42], deep 1D CNN are used. The said studies are known to yield the best-known results; they excel overall state of the art shallow techniques. We follow the standard preprocessing step [41]. All ECG signals are first preprocessed using a 200ms width median filter to remove the P wave and QRS complex, then a 600ms width median filter to remove the T wave. The resulted signals are subtracted from the original signals to yield the baseline−corrected ECG signals. Then a 12−order low-pass filter with a 35Hz cut−off frequency is applied to remove power−line and high−frequency noise. For feature extraction, we follow exactly the same technique as outlined in [41]. The processing is done using the ecgpuwave toolbox. In all our formulations, 50 transform basis was used. Since our proposed technique requires specification of certain parameters, we have tuned them on the European society of cardiology ST−T database. The parameters obtained are shown in table 3.8.

Table 3.8: Parameter values

Method	**Parameters**
CSTL	$\lambda = 0.5; \mu = 0.05$
LRTL	$\lambda = 0.5; \mu = 0.05$
DTL	$\lambda = 0.5; \mu = 0.05; \eta_1 = 0.25; \eta_2 = 0.1$
LCTL	$\lambda = 0.5; \mu = 0.05; \eta = 0.8$

For our proposed methods: LCTL, LRTL and DiTL; the nearest neighbour classifier is used. Our LCTL formulation does not require an external classifier. We have shown the results for a linear and polynomial kernel of order 2. Other kernels show mark degradation in results. DC regulariza-

tion has been used. The comparisons are made with DBN, SAE and CNN. The results are shown in Table 3.9. The customary measures of overall accuracy, sensitivity, and specificity of each class are reported.

Table 3.9: ECG classification accuracy: AAMI 2 protocol

	Accuracy	**F**		**S**		**V**	
Method		**Sens.**	**Spec.**	**Sens.**	**Spec.**	**Sens.**	**Spec.**
DBN[41]	94.8	95.7	66.5	23.9	**100**	83.9	**100**
SAE[41]	93.8	96.9	62.8	20.8	**100**	83.2	**100**
CNN[42]	97.2	**100**	66.2	21.0	**100**	85.2	98.5
CSTL	88.6	95.8	31.9	14.1	97.2	41.8	92.8
K-CSTL	90.1	98.8	52.2	15.1	97.6	45.7	95.4
LRTL	86.5	91.2	39.8	11.0	97.0	52.4	90.8
K-LRTL	89.2	95.6	56.8	13.6	97.4	60.2	95.2
DiTL	92.0	96.8	54.5	13.6	**100**	48.6	93.6
K-DiTL	96.9	98.8	56.6	24.5	**100**	85.2	96.2
LCTL	93.8	94.6	55.3	15.9	97.0	48.9	89.6
K-LCTL	**100.00**	**100**	**70.5**	**28.0**	**100**	**94.1**	**100**

The proposed methods use DC regularizations. One cannot compare the reported results with the ones in [51, 52]. This is because the division of test and training samples used here are different. We follow a repeatable protocol while most others used a randomized one. The results in Table 3.9 show that with kernelization, there is an improvement in results. The kernelized versions of DiTL and LCTL improve upon the previous bests.

3.3 Discussion

There are three major contributions to this chapter. First, this work proposes supervised versions of transform learning; four different supervision techniques have been proposed. Second, this chapter also introduces Ker-

nelization to transform learning. Third, it shows how stochastic regularization techniques like drop out and drop connect can be incorporated into transform learning. Experiments have been carried out on four different kinds of problems: computer vision, bioinformatics, hyperspectral imaging, and biomedical signal analysis. In such a diverse set of experiments, we outperform state-of-the-art techniques.

Chapter 4

Unsupervised deep transform learning - classification and clustering problems

The concept of deep transform learning (DTL) is motivated by the success of deep learning techniques like SAE, DBN, CNN and DDL. In DTL, multiple levels of transforms are learned to represent the data in terms of coefficients. Usually, such deep architectures are used for classification problems leading to deep neural networks, but this is not mandatory. It can be used for other problems where class information is not required; for example in clustering or reconstruction. The layers of single level transforms have been stacked one after the other leading to the framework of DTL; it has been shown in there that DTL indeed is a more powerful tool than conventional ones like SAE, DBN and DDL. In this chapter, we will discuss greedy deep transform learning (GDTL), followed by jointly

learned deep transform learning (JDTL) in section 4.1. Then experiments are performed for classification and clustering problems in section 4.2.

4.1 Proposed formulation

4.1.1 Greedy deep transform learning

Deep learning methods have multiple levels of learning; in each layer, a more abstract representation of the raw data is learned via non-linear transformations. Using such non-linear modules of transformation, very complex functions can be determined. Deeper representations are learned by stacking one transform after another. The learning is done greedily. The first layer determines the transform and features from the input training samples. The subsequent layers use the features(after activation) from the previous layer as training input.

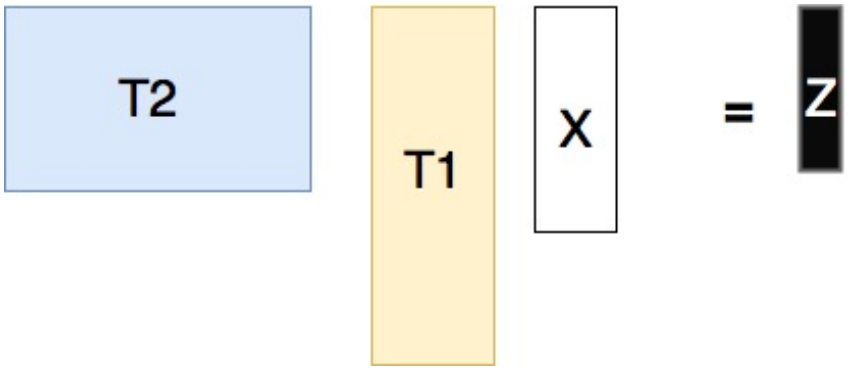


Figure 4.1: Deep transform learning

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

This is expressed as,

$$T_N\varphi...(T_2\varphi(T_1X)) = Z \tag{4.1}$$

Here T_1 operates on the data X to produce the first level of coefficients. T_2 analyzes the first level of the coefficient to produce the second level. Finally, T_N operates the second level of coefficients to generate Z. Here φ denotes the activation function, without which all the transforms will collapse into the single one. Following the greedy paradigm, we solve it one layer at a time. With the substitution,

$$\varphi(T_{N-1}\varphi...(T_2\varphi(T_1X))) = Z_{N-1} \tag{4.2}$$

It can be expressed as, $T_NZ_{N-1} = Z$ This can be alternatively expressed as,

$$(T_{N-1}\varphi...(T_2\varphi(T_1X))) = \varphi^{-1}(Z_{N-1}) \tag{4.3}$$

With the substitution $\varphi(T_{N-2}\varphi...(T_2\varphi(T_1X))) = Z_{N-2}$, we have for the next layer,

$$(T_{N-1}Z_{N-2}) = \varphi^{-1}(Z_{N-1}) \tag{4.4}$$

Continuing the substitution in this fashion, till final layer, we have,

$$T_1X = \varphi^{-1}(Z_1) \tag{4.5}$$

Note that, for all layers it is easy to invert the activation function, since φ operates element-wise.

We start solving the different layers of transforms in backward direction. Starting from equation 4.5, this is easily solved using the standard transform learning formulation.

$$\min_{T_1, Z_1} ||T_1 X - Z_1||_F^2 + \lambda(||T_1||_F^2 - \log\det T_1) \tag{4.6}$$

In each iteration, $Z_1 = T_1 X$. Once, Z_1 is solved, it acts as input to second layer for solving T_2 and Z_2; as shown below:

$$\min_{T_2, Z_2} ||T_2 Z_1 - Z_2||_F^2 + \lambda(||T_2||_F^2 - \log\det T_2) \tag{4.7}$$

Now, Z_2 acts as input to the third layer and so on. This is continued till the final layer of transform T_N. This completes the training process. The advantage of such a greedy training paradigm is that for each level, we only need solving a shallow transform learning problem which has algorithms with convergence guarantees.

During testing, the objective is to generate the test feature (Z_{test}) given the input test sample (X_{test}). This is expressed as

$$T_N \varphi ... (T_2 \varphi (T_1 X_{test})) = Z_{test} \tag{4.8}$$

The multiple layers of transforms have already been learnt during the training phase. Therefore during testing, one needs to apply them one after the other.

4.1.2 Jointly learned deep transform learning

The greedy approach is sub-optimal since there is no flow of information from deeper to shallower layers. In deep learning, this issue is addressed by back-propagation during the fine-tuning stage for supervised learning problems. We are proposing an unsupervised representation learning technique; there are no outputs to back-propagate from. Hence, we need to derive an algorithm for solving all the layers of transforms via a joint optimization framework called JDTL. The deep transform learning formulation is given in equation 4.9.

$$T_N\phi...(T_2\phi(T_1X)) = Z \tag{4.9}$$

The joint optimization problem, we intend to solve is as follows,

$$\min_{T_i's,Z} ||T_N(\phi...(T_2(\phi(T_1X)))) - Z||_F^2 + \lambda \sum_i (\mu||T_i||_F^2 - \log\det T_i) \tag{4.10}$$

This will be solved using the variable splitting approach. Using the same substitutions as in the greedy approach, i.e.,

$$\phi(T_{N-1}...(T_2\phi(T_1X))) = Z_{N-1}$$

We can express the augmented Lagrangian for equation 6.1 as,

$$\begin{aligned} &\min_{T_i's,Z,Z_{N-1}} ||T_NZ_{N-1} - Z||_F^2 + \lambda \sum_{i=1}^{N} (\mu||T_i||_F^2 - \log\det T_i) \\ &+\mu||(T_{N-1}...(T_2(\phi(T_1X)))) - \phi^{-1}(Z_{N-1})||_F^2 \end{aligned} \tag{4.11}$$

As in the greedy solution, we have used the fact that inverting the activation function is trivial since it operates element-wise. In the second, step we substitute, $\phi(T_{N-2}...(T_2\phi(T_1X))) = Z_{N-2}$,this allows expressing equation 4.11 as follows (in terms of augmented lagrangian),

$$\begin{aligned}&\min_{T_i's,Z,Z_{N-1},Z_{N-2}} ||T_NZ_{N-1} - Z||_F^2 + \mu||T_{N-1}Z_{N-2} - \phi^{-1}(Z_{N-1})||_F^2 \\ &+\lambda \sum_i (\mu||T_i||_F^2 - \log\det T_i) \\ &+\mu||\phi(T_{N-2}(\phi...(T_2(\phi(T_1X))))) - Z_{N-2}||_F^2\end{aligned} \tag{4.12}$$

Continuing in this fashion, with the final substitution $\phi(T_1X) = Z_1$,we get the complete augmented lagrangian formulation,

$$\begin{aligned}&\min_{T_i's,Z,Z_i's} ||T_NZ_{N-1} - Z||_F^2 + \lambda \sum_{i=1}^{N} (\mu||T_i||_F^2 - \log\det T_i) \\ &+\mu||T_1X - \phi^{-1}(Z_1)||_F^2 + \mu \sum_{i=2}^{N-1} ||T_iZ_{i-1} - \phi^{-1}(Z_i)||_F^2\end{aligned} \tag{4.13}$$

The alternating direction method of multipliers (ADMM) [53] allows equation 4.13 to be segregated into the following sub-problems:

$$S_1 : \min_{T_N} ||T_NZ_{N-1} - Z||_F^2 + \lambda(||T_N||_F^2 - \log\det T_N)$$

$$S_2 : \min_{T_{N-1}} \mu||T_{N-1}Z_{N-2} - \phi^{-1}(Z_{N-1})||_F^2 + \lambda(||T_{N-1}||_F^2 - \log\det T_{N-1})$$

.

.

.

$$S_N : \min_{T_1} \mu||T_1 X - \phi^{-1}(Z_1)||_F^2 + \lambda(||T_1||_F^2 - \log\det T_1)$$

$$S_{N+1} : \min_{Z} ||T_N Z_{N-1} - Z||_F^2 \Rightarrow T_N Z_{N-1} = Z$$

$$S_{N+2} : \begin{array}{l} \min\limits_{Z_{N-1}} ||T_N Z_{N-1} - Z||_F^2 + \mu||T_{N-1} Z_{N-2} - \phi^{-1}(Z_{N-1})||_F^2 \\ \equiv \min\limits_{Z_{N-1}} ||T_N Z_{N-1} - Z||_F^2 + \mu||\phi(T_{N-1} Z_{N-2}) - Z_{N-1}||_F^2 \end{array}$$

.

.

.

$$S_{2N} : \min_{Z_1} ||T_2 Z_1 - \phi^{-1}(Z_2)||_F^2 + \mu||\phi(T_1 X) - Z_1||_F^2$$

We see that the sub-problems S_1 to S_N are all standard transform updates. We already know how to solve them (solving equation 2.21). Solving for the final/deepest representation is simple; follows from S_{N+1}. The solution of the intermediate representations is from S_{N+2} to S_{2N}. All of them are least-squares problems in their equivalent form; hence has a closed-form solution: pseudo-inverse. Notice that we have kept the Lagrangian multiplier μ constant for all the layers. Moreover, we must give equal importance to all the layers, and hence, we keep $\mu = 1$ throughout. This is usually

not the case for generic optimization problems. However, in this work, we argue that each of the layers has equal importance, μ equaling unity is a logical choice. This concludes the derivation of the joint optimization algorithm. The testing stage remains the same as the greedy technique. One only needs to apply one transform after another in the correct order with the activation functions to generate the representation of a test sample.

4.2 Experiments and results

Unlike most deep learning techniques where all the layers are initialized randomly, we only have to initialize T_1, that too for solving $T_1 X = \phi^{-1}(Z_1)$ from the training data. After solving the transform learning problem, the obtained value of T_1 becomes the initial value for the ensuing iterations. Once the first layer of coefficient Z_1 is learned, it is used to initialize the second layer of transform by solving $T_2 Z_1 = \phi^{-1}(Z_2)$. The second layer representation is used to initialize the transform for the third layer, and so on.

4.2.1 Classification using jointly learned deep transforms

Classification is carried out on five benchmark deep learning datasets: MNIST, 20-newsgroup, GTZAN, YaleB and AR faces.

- MNIST dataset [54]

  It consists of 28×28 images of handwritten digits ranging from 0 to

9. The dataset has 60000 training images and 10000 testing images.

- 20-newsgroup dataset [55]

  The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics split into two subsets: one for training (or development) and the other one for testing (or for performance evaluation). The split between the train and test set is based upon messages posted before and after a specific date. The training set consists of 11269 samples, and the test set contains 7505 examples. We have used 5000 most frequent words for the binary input features. We follow the same protocol as outlined in [56].

- GTZAN [57]

  It is a music genre dataset. It contains 10000 three-second audio clips, equally distributed among 10 musical genres: blues, classical, country, disco, hip-hop, pop, jazz, metal, reggae, and rock. Each example in the set is represented by 592 Mel-phon coefficient (MPC) features. These are a simplified formulation of the Mel-frequency cepstral coefficients (MFCCs) that are shown to yield better classification performance. Since there is no predefined standard split and fewer examples, we have used 10-fold cross-validation, where each fold consisted of 9000 training examples and 1000 test examples.

- E-YALE-B [58]:

  The Extended Yale B database contains 2432 images with 38 subjects

under 64 illumination conditions. Each image is cropped to 192-by-168 pixels and downscaled to 48-by-42 pixels. For our experiments, we shuffled all the samples, took 70% for training and 30% for testing.

- AR-Faces [59]:

  This database contains more than 4000 images of 126 different subjects (70 male and 56 female). The images have various facial expressions, the lighting varies, and some of the images are partially occluded by sunglasses and scarves. For our experiments, we selected 2600 images of 100 individuals (50 males and 50 females), which is 26 different images for each subject. The train set contains 2000 images, and 600 images are kept in the test set. Each image has 540 features.

We compare our proposed jointly learned deep transform learning with three state-of-the-art representation learning techniques; two of them are supervised: they are class sparse stacked autoencoder (CSSAE) [60] and class sparse deep belief network (CSDBN) [24] . The third one is unsupervised; it is DDL[61]. For all these techniques, we found that reducing the number of nodes in each layer to half that of the previous layer yields the best results consistently. In the CSSAE and the CSDBN formulations one needs specifying the sparsity parameter; for CSSAE a value of 0.1 yields the best results and for CSDBN the corresponding value is 0.02. There is no parameter required for the DDL technique.

Our proposed joint unsupervised DTL uses a simple approach for fixing

the number of nodes in each layer. It uses a three-layer architecture where the number of bases are halved in subsequent layers, i.e. for MNIST since the input of length 784, the number basis in the first layer is 392, in the second it is 196, and in the third, it is 98. For other datasets, the values change accordingly. All the techniques compared against can only learn a representation; they do not have in-built classifiers. Hence, we employ two off-the-shelf classifiers: KNN and SVM with RBF kernel. The parameters of SVM have been tuned via grid search for each technique.

Table 4.1: Nearest neighbour classification (accuracy: joint unsupervised DTL)

Dataset	CSSAE[60]	CSDBN[24]	DDL[61]	GDTL	Proposed
MNIST	97.33	97.05	97.75	97.62	**97.91**
20-Newsgroup	70.48	70.09	70.48	70.98	**72.64**
GTGAN	83.31	80.99	83.31	83.31	**83.89**
YALEB	84.27	84.17	91.28	92.12	**95.33**
AR Faces	82.14	81.35	93.11	92.91	**96.40**

Table 4.2: SVM classification (accuracy): joint unsupervised DTL

Dataset	CSSAE	CSDBN	DDL	GDTL	Proposed
MNIST	98.50	98.53	98.64	98.52	**98.71**
20-Newsgroup	71.29	71.18	71.97	72.40	**73.19**
GTGAN	83.42	81.83	84.92	83.68	**85.18**
YALEB	85.21	84.97	92.66	93.01	**97.67**
AR Faces	82.22	82.11	93.35	93.76	**96.80**

The results from CNN for GTZAN is 59.20; for YaleB is 61.18; for AR faces is 57.9; for MNIST, CNN yields an accuracy of 98.40 with a modified LeNet architecture; for the 20-newsgroup data, CNN is not applicable.

Table 4.3: Going deep: joint unsupervised DTL with SVM classifier (accuracy)

Dataset	1-layer	2-layers	3-layers	4-layers
MNIST	96.93	97.05	**97.75**	97.19
20-Newsgroup	68.98	70.19	**70.48**	70.12
GTGAN	81.11	82.99	**83.31**	83.20
YALEB	95.11	97.41	**97.67**	96.36
AR Faces	94.98	95.87	**96.80**	96.24

4.2.2 Clustering using jointly learned deep transforms

We compare with three major ones: DDL, SAE[62] and deep sparse subspace clustering (DSC)[63]. Experiments were carried out on the COIL20 (object recognition) [64] and Extended YaleB (face recognition) [65] datasets. The COIL20 database contains 1,440 samples distributed over 20 objects, where each image is with the size of 32×32. The used YaleB consists of 2,414 samples from 38 individuals, where each image is with a size of 192×168. For both, the datasets dense scale-invariant feature transform (DSIFT) and histogram of oriented gradients (HOG) features were extracted. They were further reduced by principal component analysis (PCA) to the dimensionality of 300. Since the ground truths (class labels) for these datasets are available, clustering accuracy was measured in terms of normalized mutual information (NMI), ARI and F-score.

The architectures used for comparison have been obtained from the published studies since they were the best-performing ones according to the authors. Both SAE and DSC uses a five-layer architecture with 300-200-150-200-300 nodes; both of them use the tanh activation function. Prior

study on DDL used a four-tier architecture with 600-300-150-75 nodes; the activation function is tanh.

For our proposed DTL, a three-layer architecture (tanh activation) with a 300-150-75 basis in each layer was used. The value of the hyper-parameter μ is fixed at unity. The value of the parameter λ has been fixed to 0.1 throughout; we checked that the results are stable for values ofλ between 0.05 and 0.5. In DDL, SAE and our proposed technique, after obtaining the coefficients from the final layer, a simple K-means clustering is used. We are showing results for DSIFT and HOG features.

Table 4.4: Clustering on COIL-20: joint unsupervised DTL

	HOG			**DSIFT**		
Method	**NMI**	**ARI**	**F-Score**	**NMI**	**ARI**	**F-Score**
SAE[62]	89.26	74.25	75.70	77.09	56.59	59.07
DSC[63]	91.19	81.92	82.86	91.19	84.80	85.58
DDL[61]	90.12	80.20	81.30	91.04	84.60	83.54
GDTL	91.51	81.88	82.59	92.36	85.08	84.96
Proposed	**93.37**	**84.24**	**85.82**	**94.38**	**86.92**	**88.56**

Table 4.5: Clustering on YALEB: joint unsupervised DTL

	HOG			**DSIFT**		
Method	**NMI**	**ARI**	**F-Score**	**NMI**	**ARI**	**F-Score**
SAE[62]	93.43	82.57	83.07	87.54	75.82	76.50
DSC[63]	96.91	90.25	89.46	90.85	83.00	83.45
DDL[61]	96.82	88.97	89.13	90.20	81.83	83.42
GDTL	97.12	90.56	90.48	91.27	82.51	84.66
Proposed	**98.93**	**93.43**	**92.06**	**93.26**	**85.62**	**85.86**

4.3 Discussion

A new framework for deep learning called deep transform learning has been proposed recently. The idea is to represent the training data as a non-linear combination of several layers of transforms. The first work solves DTL problem greedily. It was a sub-optimal approach since there was no flow of information from deeper layers to the shallower layers. In the second work, all the transforms and coefficients are solved simultaneously in a single optimization problem. It is an unsupervised method, and any classifier can be used on the learned features to make predictions. Results are computed on benchmark classification and clustering datasets. The performance is compared with state-of-art methods. In both cases, the proposed approach produced significantly better results than the existing methods.

Chapter 5

Unsupervised deep transform learning - inverse problems

This chapter addresses the issue of solving a linear inverse problem. Many problems in imaging can be generalized to a linear inverse problem, for example, image denoising, deblurring, reconstruction, inpainting, and image super-resolution. Conventional inversion techniques are transductive; solving an inverse problem with only some prior knowledge about the solution. The advent of deep learning led the way for inductive (trained) inversion techniques. The main issue with inductive inversion is that unless the unseen signal (to be inverted) is similar to the training data, the learned model fails to generalize rendering poor inversion results. A recent study on DDL has shown how it can combine the best of both worlds: deep learning with transductive inversion. In this chapter, we show how the analysis counterpart of DL, called transform learning, can be extended deeper for transductive inversion. First, a brief literature review is described in

section 5.1, followed by the proposed methodology in section 5.2. Then section 5.3 presents the experiments and results. Experiments have been carried out on deblurring and reconstruction.

5.1 Literature review

Many problems in machine learning and signal processing such as unmixing, regression, denoising, reconstruction, source separation, etc. solve the linear inverse problem. Mathematically it is represented as follows,

$$y = Ax \tag{5.1}$$

Where y is the observation, A is the linear system of equations, and x is the unknown. Problems differ in the nature of A. For example, in denoising, A is an identity, for regression, it is the system of explanatory variables, and for magnetic resonance imaging (MRI) reconstruction, it is a Fourier operator. In most practical scenarios, the linear system is noisy. In machine learning problems such as regression, the 'noise′ is the modelling error. In other problems, such as reconstruction, the noise is usually generated by the physical process, e.g. thermal noise in MRI. Therefore, a more practical model of our interest is the noisy version of equation 5.1 given by,

$$y = Ax + \eta \tag{5.2}$$

Here η is the noise and its statistical properties are assumed to be known. The most straightforward approach to solve equation 5.2 is to find a minimum variance solution. For the commonly assumed Gaussian noise, this turns out to be the pseudo-inverse. For other types of noise, the solution is more sophisticated. In this work, we will assume that the noise is Gaussian.

Later techniques, instead of just solving for the minimum variance solution, assumed some prior. The simplest prior can be the minimum energy solution, which reduces to Tikhonov regularization. This is expressed as follows,

$$\arg\min_x ||y - Ax||_2^2 + \lambda||x||_2^2 \tag{5.3}$$

More modern approaches assumed the solution to be sparse. This led to regularization via the l1-norm leading to the following,

$$\arg\min_x ||y - Ax||_2^2 + \lambda||x||_1 \tag{5.4}$$

Perhaps the most famous applications of sparse recovery in machine learning are least absolute selection and shrinkage operator (LASSO) regression [66] and relevance vector machine [67]. In signal processing, the field of compressed sensing (CS) [68] started from the idea of sparse recovery. In this work, we are mainly interested in signal processing aspects of linear inverse problems. CS-based techniques became popular in this domain because a large class of signals can be represented in a sparse fashion in some fixed transform domain (wavelet, DCT, Gabor, etc.). Many

such transforms are either orthogonal or tight-framed. After incorporating transform (Ψ) domain sparsity, typically 5.5 is expressed as follows:

$$\arg\min_{\alpha} ||y - A\Psi^T\alpha||_2^2 + \lambda||\alpha||_1 \tag{5.5}$$

Here α is the sparse representation of the image in Ψ. Usually l_1 norm minimization is used to recover sparse coefficients. This allows expressing the signal via analysis-synthesis equations. Therefore signal recovery could be framed as a sparse synthesis prior problem. In this formulation, the signal is recovered by applying the synthesis equation on the recovered coefficients. The majority of studies in CS are based on the synthesis prior formulation. It is theoretically well understood, and there are many efficient algorithms to solve it. However, in practice, the synthesis prior is restrictive; it can accommodate only transforms that follow the analysis-synthesis equations. This precludes many powerful priors such as total variation in image processing tasks. Therefore, in practice, the co-sparse analysis prior formulation [69] is known to yield better results. This is expressed in the following fashion.

$$\arg\min_{x} ||y - Ax||_2^2 + \lambda||\Psi x||_1 \tag{5.6}$$

In CS, the quality of reconstruction is directly proportional to the sparsity of the signal in the transform domain; sparser the representation better is the recovery. The fixed transforms used in CS are mathematically well defined and generic in nature; they can sparsely represent a wide class

of signals. But it is well known in signal processing that, to get the best (sparsest) representation, the basis needs to be adaptively learned from the signal itself. This is the reason, DL based inversion techniques [70] eventually surpassed CS. The formulation for DL based inversion is expressed as follows,

$$\min_{x,D,Z} ||y - Ax||_2^2 + \lambda(\sum_i ||P_i x - Dz_i||_2^2 s.t ||z_i||_0 \leq \tau) \tag{5.7}$$

The first term is the standard data fidelity term assuming gaussian noise. The term within the brackets corresponds to DL. Given the patches of the signal $P_i x$, a dictionary D is learnt such that the corresponding coefficients z_i is sparse. Instead of the standard CS sparsity prior in a fixed transform, equation 5.7 learns both the basis (D) and the coefficients (z_i) adaptively from the signal. In a very recent work [71], it was shown that instead of learning only a single layer of the dictionary, better results could be obtained if multiple layers are learned. The following formulation is given in [71],

$$\begin{aligned} &\min_{x,D_1,D_2,D_3,Z} ||y - Ax||_2^2 \\ &+\lambda(\sum_i ||P_i x - D_1\phi(D_2\phi(D_3 z_i))||_2^2 + \gamma||z_i||_1) \end{aligned} \tag{5.8}$$

Here $D1$, $D2$, $D3$ are three layers of dictionaries. The non-linear activation function ϕ prevents collapsing of the three dictionaries into a single one. Note that instead of the $l0-$norm DDL employs the $l1-$norm to promote sparsity. DL is a synthesis formulation; it learns a basis (dictionary) from the signals such that one can generate the signals from the learnt

coefficients. Just as there is an analysis version of CS, there is an analysis version of DL called TL [2, 3]. TL learns an analysis basis (transform) that operates on the signals to generate the coefficients. The TL based inversion formulation is as follows,

$$\begin{aligned} &\min_{x,T,Z} ||y - Ax||_2^2 + \lambda(\sum_i ||TP_ix - z_i||_2^2 + \mu(||T||_F^2 - \log\det T) \\ &s.t||z_i||_0 \leq \tau \end{aligned} \tag{5.9}$$

Instead of applying transform on full image, the small sub-images of 8×8 are extracted from it. We call the small sub-image a patch of the image. The term in brackets correspond to TL. Here T operates on the patches of the image P_ix to produce sparse coefficients z_i. A is an inversion operator which is used differently for different problems. For deblurring, A is convolution operator and for reconstruction, it is projection operator.

A very recent work [72] proposed a sparse autoencoder based denoising formulation. This is an interesting formulation since autoencoders have never been used for transductive inversion. The mathematical expression is as follows,

$$\begin{aligned} &\min_{\hat{x},W,W^T} ||y - x||_2^2 + \lambda(\sum_i ||P_ix - W^T\varphi(WP_ix)||_2^2 \\ &+\gamma||\varphi(WP_ix)||_1) \end{aligned} \tag{5.10}$$

The term within the brackets corresponds to sparse autoencoder learning. W and W^T are the encoding and decoding weights. The $l1-$norm on the features promotes sparsity. Note that the autoencoder based formulation

has been used for denoising, and not for generic inversion. But it can be easily extended to solve general inversion problems.

5.2 Proposed formulation

We propose an adaptive deep learning-based approach for solving inverse problems. It is based on DTL. In standard (shallow) TL, learned basis (T) is applied over each patch ($P_i x$) of the image to obtain sparse coefficients (z_i).

$$TP_i x = z_i \qquad \forall i \tag{5.11}$$

The formulation has already been mentioned in equation 5.9 but it is repeated here for convenience.

$$\begin{aligned} &\min_{x,T,Z} ||y - Ax||_2^2 + \lambda(\sum_i ||TP_i x - z_i||_2^2 + \mu(||T||_F^2 - \log\det T) \\ &s.t ||z_i||_0 \leq \tau \end{aligned} \tag{5.12}$$

We extend from single level to multi-level deep transforms. Therefore each patch is operated by multiple levels of transforms.

$$T_1\phi(T_2\phi(...\phi(T_N)))P_i x = z_i \tag{5.13}$$

We are showing the derivation for three layers, but the approach we follow is easily extendable to fewer or more layers. Here ϕ denotes the activation function; without which all the transforms will collapse into a single one.

In this work, we will be using rectified linear unit (ReLU) type activations. Before going into the formulation, we discuss the reason for the deep extension. The shallow inversion formulation is linear. It is known in neural network theory about the function approximation capability of non-linear networks with ReLU [73]. The approximation capacity improves when one goes deeper with ReLU [74]. This is the prime reason behind the extension to deeper layers of transform with ReLU activation. We extend the basic TL based inversion formulation equation 5.9 to accommodate multiple layers of transforms. Mathematically this is expressed as follows,

$$\begin{aligned} &\min_{x,T_1,T_2,T_3,Z} ||y - Ax||_2^2 + \lambda(\sum_i ||T_3T_2T_1P_ix - z||_F^2 \\ &+\mu \sum_{j=1}^{3} (||T_i||_F^2 - \log\det T_i) + \gamma||z_i||_1) \end{aligned} \tag{5.14}$$

Z is formed by stacking the $z_i's$ as columns. Also, the coefficients in equation 5.14 after application of each layer of transform should be non-negative to impose ReLU activation. This means that

$$T_1P_ix > 0$$

and

$$T_2T_1P_ix > 0$$

P_iX is the i^{th} patch of the image X. T_1 is the transform being applied on this patch and $T_1P_ix > 0$ suggests ReLU activation. To solve equation 5.14 we resort to the variable splitting technique [15]. We introduce two

sets of proxy variables

$$w_i = T_1 P_i x$$

and

$$h_i = T_2 T_1 P_i x$$

This leads to the following augmented lagrangian (AL) formulation.

$$\begin{aligned} &\min_{x,T_1,T_2,T_3,H,W,Z} ||y - Ax||_2^2 + \mu \sum_{j=1}^{3} (||T_i||_F^2 - \log\det T_i) + \gamma||z_i||_1) \\ &+\lambda(\sum_i ||T_3 h_i - z_i||_F^2 + ||T_2 w_i - h_i||_F^2 + ||T_1 P_i x - w_i||_F^2) \end{aligned} \tag{5.15}$$

H and W are formed by stacking the $h_i's$ and $w_i's$ as columns. In this formulation equation 5.15 the proxy variables w_i and h_i need to be greater than 0. Ideally we would require to have two multiplicative hyper-parameters corresponding to the two newly introduced terms $||T_2 w_i - h_i||_F^2$and $||T_1 P_i x - w_i||_F^2$. However, we argue that since these two terms actually correspond to two intermediate layers of deep transform learning, there is no reason to weigh them differently. Hence we keep the multiplicative hyper-parameter to be unity. We employ the ADMM [75] to solve equation 5.15. Basically, we update each of the variables as sub-problems:

$$S_1 : \min_{T_1} \sum_i ||T_1 P_i x - w_i||_F^2 + \mu(||T_1||_F^2 - \log\det T_1)$$

$$S_2 : \min_{T_2} \sum_i ||T_2 w_i - h_i||_F^2 + \mu(||T_2||_F^2 - \log\det T_2)$$

$$S_3 : \min_{T_3} \sum_i ||T_3 h_i - z_i||_F^2 + \mu(||T_3||_F^2 - \log\det T_3)$$

$$S_4 : \min_x ||y - Ax||_2^2 + \lambda \sum_i ||T_1 P_i x - w_i||_F^2$$

$$S_5 : \min_{h_i} ||T_3 h_i - z_i||_F^2 + ||T_2 w_i - h_i||_F^2 \forall i$$

$$S_6 : \min_{w_i} ||T_2 w_i - h_i||_F^2 + ||T_1 P_i x - w_i||_F^2 \forall i$$

$$S_7 : \min_z ||T_3 h_i - z_i||_F^2 + \gamma ||z_i||_1 \forall i$$

All the sub-problems have closed-form solutions. S_1 to S_3 are standard transform updates. S_4 to S_6 are least-square problems that have a closed-form solution in the form of pseudo-inverse. However, here we solve it using a conjugate gradient since A may not always be available as an explicit matrix. S_7 is a sparse coefficient update. It must be noted that for the updates of S_4 and S_5, one must need to ensure that the solution is non-negative; this is easily enforced by putting all the negative values of h_i and w_i as zeroes after the pseudo-inverse solution. The problem is non-smooth and non-convex. There is recent work that shows the convergence of ADMM (to local minima) for such class of problems [76] especially when each of the sub-problems has a closed-form update. In practice, we stop the iterations when the objective function does not change much over

consequent iterations. The computational complexity of each iteration is mainly dictated by the transform updates. Since they require computing singular value decompositions, their complexity is $O(n^3)$. The complexity for solving the least square problems is $O(n^2)$ by the conjugate gradient. The last sub-problem (sparse update) costs $O(n)$.

5.3 Experiments and results

5.3.1 Deblurring

Blur is a common image degradation. Blurring is, at least locally, a linear operation resulting from the convolution of a sharp image with a filter. When the support of the blur kernel is small compared to the patch size, one can assume a linear relationship between the blurry and sharp patches. Thus, a simple approach to the deblurring problem consists of learning how to invert this linear transform. The formulation for deblurring can be formulated as 5.12, considering A as a blur kernel which is block circulant. For comparison we have used some of the latest techniques in image deblurring: GBD [77], RCSR [78] and DeblurGAN [79]. Note that the first two are transductive approaches, while the last one is an inductive approach. For comparative denoising performance, we have followed the protocol outlined in [11]. The training image set consists of 91 images for all the problems. The testing dataset is comprised of Set5 and Set14 composed of 5 and 14 images. Note that only [79] requires training; the others do not

have a training phase since they are transductive in nature. The images were synthetically blurred using known motion blur kernel. Our proposed approach requires the specification of three parameters: λ, μ and γ. These parameters have been tuned on separate validation images (not used in the experiments). The values obtained are $\lambda = .2$,$\mu = .1$ and $\gamma = .05$. We have used a three-layer network for the deblurring experiments. We also tried going deeper, but the results deteriorated owing to over-fitting. The number of transform basis used in each layer are 256-128-64. Overlapping patches of size 32×32 are used here. The comparative results are shown in the following table. The metrics used here are PSNR and SSIM.

Table 5.1: Comparative deblurring performance: PSNR

	Blurry	**RCSR**[78]	**GBD** [77]	**DeblurGAN**[79]	**Proposed**
Images		**SET-14**			
Baboon	20.29	20.21	21.30	23.54	**23.91**
Barbara	23.91	22.86	24.13	26.49	**26.96**
Bridge	21.06	20.93	22.97	24.51	**25.05**
Coastguard	22.25	22.19	23.83	**25.70**	**25.70**
Comic	18.59	18.41	20.44	22.26	**22.28**
Face	28.02	27.79	29.53	31.07	**31.65**
Flower	21.43	21.25	23.74	25.21	**25.73**
Foreman	23.56	23.10	26.53	26.98	**27.40**
Lena	26.62	26.36	28.81	30.22	**30.65**
Man	22.58	22.39	24.28	26.04	**26.49**
Monarch	21.83	21.66	24.80	25.39	**25.91**
Pepper	26.10	25.70	28.44	29.42	**29.96**
Ppt3	18.97	18.69	21.49	**23.00**	**23.00**
Zebra	19.61	19.40	22.52	**23.45**	**23.45**
		SET-5			
Baby	27.17	26.71	29.58	31.06	**31.69**
Bird	25.13	24.69	28.01	29.06	**29.63**
Butterfly	16.32	16.12	19.51	20.04	**20.88**
Head	28.09	27.82	29.60	**31.77**	31.43
Woman	22.32	21.98	25.36	26.80	**27.24**

The results in table 5.1 and table 5.2 show that our proposed method yields the best aggregate performance. It is better than other transductive

Table 5.2: Comparative deblurring performance: SSIM

	Blurry	RCSR[78]	GBD [77]	DeblurGAN[79]	Proposed
Images		**SET-14**			
Baboon	0.37	0.37	0.49	0.73	**0.76**
Barbara	0.67	0.64	0.70	0.82	**0.85**
Bridge	0.44	0.44	0.59	0.75	**0.76**
Coastguard	0.38	0.38	0.52	**0.73**	**0.73**
Comic	0.45	0.44	0.60	0.70	**0.74**
Face	0.66	0.65	0.72	0.82	**0.87**
Flower	0.60	0.59	0.70	0.73	**0.79**
Foreman	0.72	0.70	0.78	0.80	**0.85**
Lena	0.76	0.75	0.80	0.83	**0.87**
Man	0.58	0.56	0.66	0.68	**0.78**
Monarch	0.77	0.76	0.82	0.84	**0.86**
Pepper	0.75	0.73	0.79	0.80	**0.84**
Ppt3	0.72	0.70	0.77	0.78	**0.78**
Zebra	0.48	0.47	0.67	0.74	**0.77**
		SET-5			
Baby	0.78	0.76	0.85	0.86	**0.89**
Bird	0.76	0.74	0.83	0.84	**0.85**
Butterfly	0.48	0.47	0.62	0.63	**0.65**
Head	0.66	0.65	0.72	**0.84**	**0.84**
Woman	0.73	0.71	0.80	0.80	**0.82**

Figure 5.1: Man Left to Right: original, blurred image,RCSR, GBD, DeblurGAN and proposed

Table 5.3: Run-time comparison

Method	Time in sec
RCSR[78]	301
GBD [77]	31
DeblurGAN[79]	1
Proposed 2 layer	125
Proposed 3 layer	216
Proposed 4 layer	399

techniques like RCSR and GBD. Ours is also better than state-of-the-art learning-based technique DeblurGAN; on an average, the improvement is over .5dB of PSNR and .2 SSIM. For qualitative evaluation, we show

one of the images in figure 5.1. Next, we show the run-times of different algorithms. Experiments were run on a 64 bit Windows 10 PC running on i7 @ 3.1 GHz with 16 GB of RAM. The transductive techniques were run on MatlabTM R2015a, and the inductive technique was run on python. We show the average (per image) run-time from the different techniques in table 5.3.

The results show that DeblurGAN is the fastest; this is obvious since it has a long training time but very fast operational time with only a few matrix products. The RCSR is the most time consuming (however we were able to achieve faster speed than reported in [78] possibly owing to a better environment). Of the transductive techniques, GBD is the fastest. Our method is much slower than GBD; moreover, as we go deeper, the run-time increases because of the requirement of solving more sub-problems.

5.3.2 Reconstruction

We address the problem of dynamic MRI reconstruction. Transforms for different patches are trained; each patch can be sparsely represented by the corresponding transform. Then, obtain the samples by a sensing projection matrix which is incoherent with transform. Finally, solve an optimization problem to reconstruct the image.

Experiments are conducted on two publicly available datasets. The two sequences will be called the cardiac perfusion sequence 1 and 2. The data was collected on a 3T Siemens scanner. In this work, we simulated a ra-

dial sampling with 24 lines that were acquired for each time frame; this corresponds to an under-sampling ratio of 0.21. The full resolution of the dynamic MR images is 128 $\times$ 128. About 6.7 samples were collected per second. The scanner parameters for the radial acquisition were transmission rate = 2.5-3.0 msec, TE = 1.1 msec, flip angle = 12° and slice thickness = 6 mm. The reconstructed pixel size varied between $1.8mm^2$ and $2.5mm^2$. Each image was acquired in a $\sim$ 62-msec read-out, with the radial field of view (FOV) ranging from 230 to 320 mm.

We have compared our method with a few recent reconstruction techniques. The first one is based on a LASSI [80]. The second one is a KLR [81]. Both are transductive approaches. The last one is an inductive deep learning-based approach that combines dynamic modelling via recurrent neural networks with spatial modelling via a CNN; this is called CRN [82]. As before, for our proposed method, we tuned the parameters using a grid search on a separate dynamic MRI data not used here. The obtained parametric values are: $\lambda = .2$, $\mu = .5$ and $\gamma = .2$. 3D patches of size 16 $\times$ 16 $\times$ 4 were used. We obtained the best results for four layers; the number of basis elements in each layer is 256-128-128-64. The experimental results are shown in table 5.4. MRI reconstruction quality is usually measured by NMSE. We use the same metric. From the numerical results, we find that our method yields the best results. It is much better than the learned technique CRN. The other transductive techniques, LASSI, and KLR are much worse.

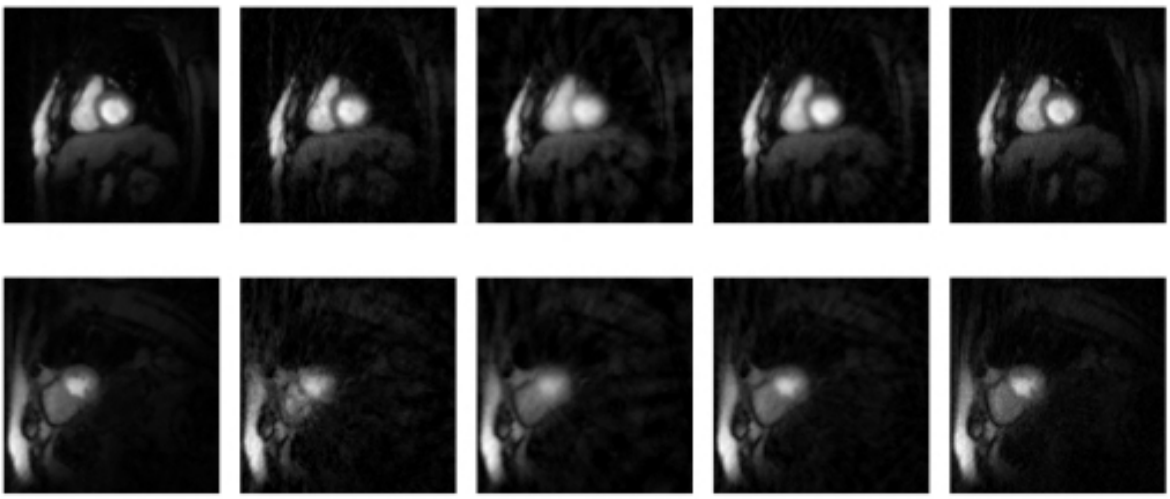


Figure 5.2: Reconstructed image. Top: cardiac perfusion 1; Bottom: cardiac perfusion 2. Left to right: ground-truth, LASSI, KLR, CRN and proposed

The numerical results do not give the complete picture for MRI reconstruction. Therefore it is customary to show reconstructed and difference (between ground-truth and original) images. We will only show the results with four layers from our proposed deep transform learning because it yields the best results.

Table 5.4: Reconstruction performance in terms of NMSE

Method	**LASSI**[80]	**KLR**[81]	**CRN**[82]	**DTL 2-layer**	**DTL 3-layer**	**DTL 4-layer**	**DTL 5-layer**
Cardiac 1	0.0586	0.0412	0.0356	0.0408	0.0219	**0.0184**	0.0357
Cardiac 2	0.0474	0.0401	0.0312	0.0400	0.0202	**0.0149**	0.0286

One frame each from the two reconstructed sequences are shown in figure 5.2. One can see that even though LASSI shows poor NMSE, its reconstruction quality is actually at par with KLR and CRN; LASSI shows lot of reconstruction artifacts but is able to preserve the edges. KLR and CRN on the other hand overtly smooths the tissue boundaries. Our proposed technique preserves the tissue boundaries with minimal artifacts.

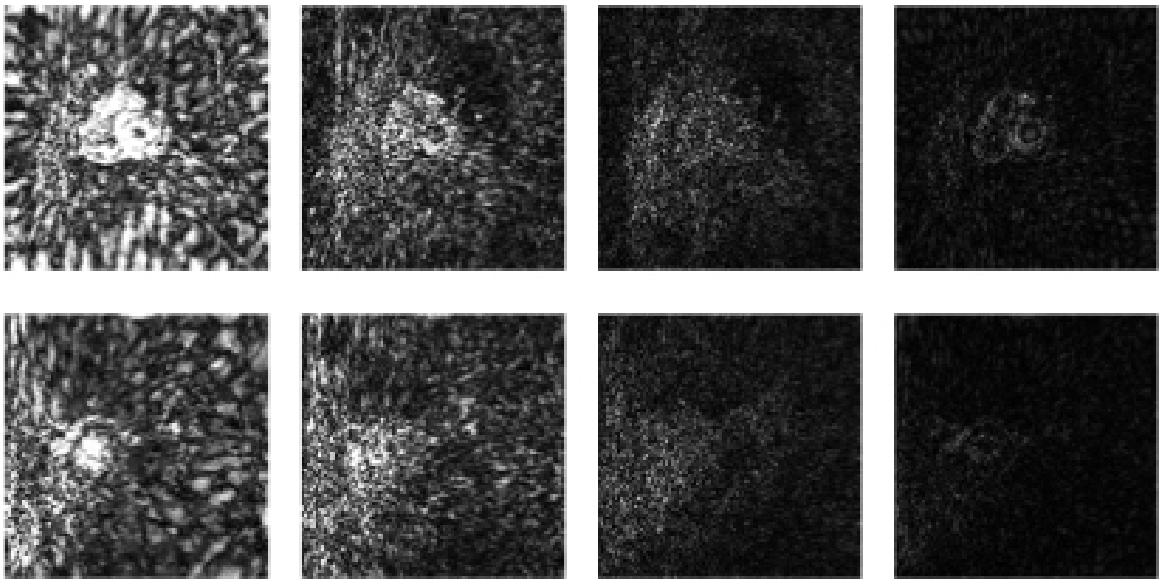


Figure 5.3: Difference image. Top: cardiac perfusion 1; Bottom: cardiac perfusion 2. Left to right: LASSI, KLR, CRN and proposed

The difference images are shown in figure 5.3. These are obtained by taking the absolute difference between the fully sampled ground-truth and the reconstructed images. The thus obtained difference images are contrast-enhanced uniformly for visual clarity. From these difference images, we can see that LASSI indeed considerable reconstruction artefacts; these are much less pronounced in KLR. CRN improves over KLR, but our method yields the best reconstruction; the artefacts are negligible.

5.4 Discussion

This is the first work that shows that proposes a generic inversion approach based on deep transform learning. In particular, we have addressed two problems in this chapter: deblurring and MRI reconstruction. For both

problems, we have compared with the state-of-the-art in each area. We improve upon the rest by a considerable margin for both problems. In recent times, there are a plethora of studies proposing deep learning-based inductive solutions to inversion problems. The main criticism against this class of approach is that it does not take into account the inversion model [83]. There are a few recent studies like [84] that introduce deep learning as a prior in the inversion process; such approaches are likely to be more consistent in terms of quality and reproducibility. Our work is a step in this direction.

Chapter 6

Supervised deep transform learning

In the previous chapters, We have discussed DTL in an unsupervised fashion; which needed a third party classifier. The unsupervised DTL based formulations are optimal for solving problems like denoising, reconstruction and impainting but it may not lead to optimal solution in classification tasks. In such tasks the goal is to have the transforms and coefficients with high discriminative capabilities. This motivates to learn DTL in a supervised framework that can classify directly without any need of an external classifier. The basic idea in label consistency is to learn a linear map that projects the sparse features into the target variables as discussed in Chapter 2. The formulation can be generalized to perform both single label and multi-label classification. In this chapter, a deeper version of LCTL is proposed in section 6.1. Experiments are performed on benchmark single label and multi label classification problems. The proposed algorithm is applied on a practical application of non intrusive load monitoring (NILM) in section 6.2.

6.1 Proposed formulation

In a binary classification problem, each sample belongs to only one output class whereas, in multi-label classification, a single sample may belong to more than one output class. NILM is a multi-label classification problem since at a given point of time; it is likely that multiple appliances are running. The proposed approach has been used to predict appliance states at a particular time instance.

In this work, we extend DTL to its supervised version with multi-label consistency terms. We solve the problem in an optimal fashion using the variable splitting augmented lagrangian approach followed by the ADMM. Basically, we learn a linear map such that the coefficients from the final level maps to the multi-label targets. The complete formulation is as follows,

The formulation for DTL is given by

$$\min_{T_i's,Z} ||T_N(\phi...(T_2(\phi(T_1X)))) - Z||_F^2 + \lambda \sum_i (\mu||T_i||_F^2 - \log\det T_i) \quad (6.1)$$

We add a label consistency term to the formulation equation 6.1 to incorporate supervision. In LCDL [85], a penalty term is added that minimizes the error between the mappings of the learned representations and target binary class labels. For shallow TL [85], the formulation is given by:

$$\min_{T_1,Z,W} \|T_1X - Z\|_F^2 + \lambda\|Q - WZ\|_F^2 \quad (6.2)$$

Here, the first term is the standard TL term. In the second term, W is the linear map between the learned representations Z to the target binary class labels Q. λ is a regularization parameter. In, Q positions of the corresponding class labels is 1 and rest are 0's.

In [86] it is shown that the label consistent formulation can be extended to the multi-label classification problem. In such problems, a single sample may belong to multiple output classes, which is the case in most standard classification problems.

In our proposed formulation, we add a label consistency term on top of the deep transform learning problem and call it label consistent deep transform learning (LCDTL) 6.1, given by:

$$\begin{aligned} &\min_{T_i's,Z,W} ||T_N(\phi...(T_2(\phi(T_1X)))) - Z||_F^2 + \lambda||Q - \phi(WZ)||_F^2 \\ &+\lambda \sum_i (\mu||T_i||_F^2 - \log\det T_i) \end{aligned} \tag{6.3}$$

In order to jointly learn all the layers, we use a variable splitting method [87] to solve DTL problem. For three layers, the formulation becomes:

$$\begin{aligned} &\min_{T_1,T_2,T_3,Z,W} ||T_3\phi(T_2\phi(T_1X))) - Z)||_F^2 + \lambda||Q - \phi(WZ)||_F^2 \\ &+\varepsilon \sum_i (||T_i||_F^2 - \log\det T_i) \end{aligned} \tag{6.4}$$

We substitute $Z_1 = \phi(T_1X)$ and $Z_2 = \phi(T_2Z_1)$. This leads to following

AL,

$$\min_{T_1,T_2,T_3,Z_1,Z_2,Z,W} ||T_3Z_2 - Z)||_F^2 + \lambda||Q - \phi(WZ)||_F^2 + \varepsilon \sum_i (||T_i||_F^2 - \log\det T_i)$$
$$+\mu(||Z_2 - \phi(T_2Z_1)||_F^2 + \mu||Z_1 - \phi(T_1X)||_F^2 \tag{6.5}$$

The multiplicative terms $\mu's$ indicate the relative weight age of each layer. Since there is no reason to give one layer more importance than others, we have set all of them as unity. ADMM [53] has been used to solve equation 6.5. Following the ADMM we can segregate equation 6.5 into following sub-problems:

$$S_1 : \begin{aligned} &\min_{T_1} ||Z_1 - \phi(T_1X))||_F^2 + \varepsilon||T_1||_F^2 - \log\det T_1 \\ &\equiv \min_{T_1} ||\phi^{-1}(Z_1) - T_1X)||_F^2 + \varepsilon||T_1||_F^2 - \log\det T_1 \end{aligned}$$

$$S_2 : \begin{aligned} &\min_{T_2} ||Z_2 - \phi(T_2Z_1))||_F^2 + \varepsilon||T_2||_F^2 - \log\det T_2 \\ &\equiv \min_{T_2} ||\phi^{-1}(Z_2) - T_2Z_1)||_F^2 + \varepsilon||T_2||_F^2 - \log\det T_2 \end{aligned}$$

$$S_3 : \min_{T_3} ||T_3Z_2 - Z||_F^2 + \varepsilon||T_3||_F^2 - \log\det T_3$$

$$S_4 : \min_{Z} ||T_3Z_2 - Z||_F^2 + \lambda||Q - WZ||_F^2$$

$$S_5 : \begin{aligned} &\min_{Z_1} \mu(||Z_2 - \phi(T_2Z_1))||_F^2 + ||Z_1 - \phi(T_1X))||_F^2) \\ &\equiv \min_{Z_1} \mu(||\phi^{-1}(Z_2) - T_2Z_1)||_F^2 + ||Z_1 - \phi(T_1X))||_F^2) \end{aligned}$$

$$S_6 : \min_{Z_2} ||T_3 Z_2 - Z||_F^2 + \mu(||Z_2 - \phi(T_2 Z_1))||_F^2)$$

$$S_7 : \min_{W} ||Q - WZ||_F^2$$

During the training phase, the above sub-problems are solved using alternating minimization. Solving for $T's$ is a standard transform update. All these sub-problems of solving $Z's$ are simple least-square problems having closed-form solutions in Moore-Penrose pseudo-inverse. The activation function ϕ is unitary applied element-wise. Hence, it is trivial to invert. We have used two stopping criteria for the iterations. The first one is a limit on the maximum number of iterations. The second one is the local convergence of the objective function. The testing phase is similar to the one discussed in section 3.1.5.

6.2 Experiments and results

6.2.1 Single-label classification

To evaluate the performance of proposed method, experiments are performed on benchmark face recognition datasets: AR face [88] and YaleB face [65]. Brief description of both the datasets is given in 4.2.1. The proposed architecture is compared against stacked denoising autoencoder (SDAE)[89], stacked group sparse auto encoder (SGSA)[22], stacked label

Table 6.1: Classification accuracy on face recongnition

Method	AR	yaleB
SDAE[89]	37.60	42.81
SGSA[22]	32.50	66.27
SLCA[90]	85.21	86.22
DBN[6]	34.91	43.62
DDBN[91]	38.20	60.34
LCKSVD[92]	87.67	90.80
DDL[93]	42.66	63.35
LCDDL[94]	96.50	94.57
Proposed 1-layer	97.80	98.80
Proposed 2-layer	97.91	98.87
Proposed 3-layer	**98.89**	**98.65**
Proposed 4-layer	96.16	97.24

consistent autoencoder (SLCA)[90], DBN[6], discriminative deep belief network (DDBN)[91], label consistent KSVD (LCKSVD)[92], DDL[93] and label-consistent deep dictionary learning (LCDDL)[94]. For all the deep techniques, three layers are used and in each layer the number of nodes are reduced by half from the previous layer. All the techniques apart from DDL have their in-built classifiers. For DDL, a separate SVM classifier with radial basis function was used.

For the proposed LCDTL, we show the results from one to four layers. Here we halve the number of atoms in each subsequent layer. The classification accuracies are shown in the table 6.1.

6.2.2 Multilabel classification

In the multi-label classification problem, one sample can belong to multiple classes. In [86] it is shown that label consistent KSVD [85] can be used for multi-label classification problems. Our work is a deeper extension and

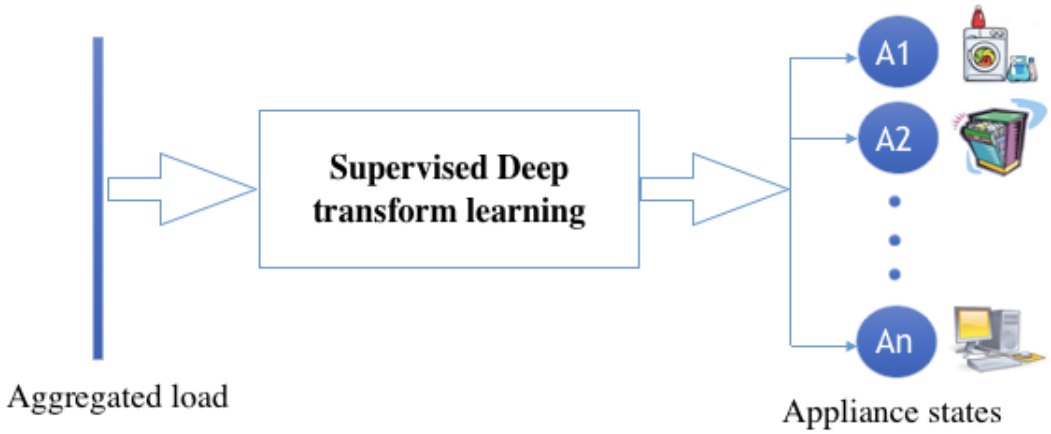


Figure 6.1: NILM as multi-label classification problem

hence can be used for multi-label classification as discussed in the previous section. A practical example of multi-label classification can be found in NILM, as shown in figure 6.1. In NILM, one tries to predict the states of individual appliances from the aggregated power readings [95].

Ideally, NILM should be able to desegregate the load from only the smart-meter reading. However, this is a challenging problem (especially at low sampling rates); in practice, a learning-based paradigm is followed where the training stage is intrusive, but the testing/operation stage is not. The building is instrumented during the training stage to gather data, from which machine learning models are learned. During operation, the sensors are removed, and the learned models are used to predict the consumption of each device.

This paradigm is not entirely non-intrusive. The training stage is intrusive requiring deployment of multiple sensors. In recent times a multi-label classification approach provides an entirely non-intrusive alternative. It does not require any instrumentation; it only requires the recording of the

state-of-the-appliance, i.e. whether it is ON or OFF. During the training stage, given the smart-meter reading and the recorded states of the appliances, a machine learning model learns multi-label classification; here each appliance is treated as a label - since several appliances can be ON at the same time, it turns out to be a multi-label classification problem. During the operational stage, the learned model is used to predict the state of the appliances given the smart-meter readings. To estimate the actual power consumption, the state is multiplied by the average power consumption of the device.

$$\min_{T_i's,Z,W} ||T_N(\phi...(T_2(\phi(T_1X)))) - Z||_F^2 + \lambda||Q - \phi(WZ)||_F^2$$
$$+\lambda \sum_i (\mu||T_i||_F^2 - \log\det T_i)$$

Here, X is aggregated power meter data, and Q are the targets. Each target has the same length as the number of the appliance; the appliances are in order. If an appliance is "ON", the corresponding value is 1 or else it is 0. The map W projects the coefficients Z to the multi-label target labels Q.

For simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep transform learning, We have used two traditional techniques as benchmarks- random K label set (RAKEL) [96] and multi-label K-nearest neighbor (MLKNN) [97] for comparison.

Experiments have been carried out on two benchmark datasets: REDD

[98] and Pecan Street dataset [99]. A brief description of both the datasets is as follows:

- **REDD dataset**

  The REDD dataset consists of both aggregated and appliance level power data from six houses at 1Hz. However, we do not make use of the appliance level consumption data; we only need its state. To emulate real-world conditions, the samples are averaged over a time period of 1 minute for our experiments. Four high power-consuming devices used in our experiments are the dishwasher, kitchen outlet, lighting, and washer dryer.

- **Pecan street dataset**

  Pecan street dataset consists of a one-minute appliance level and building level electricity data from 240 houses. For experiments, subsets of 28 houses have been used. For this work, 4 most power-consuming devices (site meter, air conditioner, electric furnace, and sockets) are used for experiments. To prepare training and testing data, aggregated and sub-metered data are averaged over a period of 1 minute. Each training sample contains power consumed by a particular device in one day while each testing sample contains total power consumed in one day in a particular house. 80% of the houses are assigned to the training set and 20% to the test set.

6.2.3 Evaluation metrics

Metrics used for single label classification cannot be used for multi-label classification. Prior work [96] proposed to used F_1 macro and F_1 micro evaluation measures. To evaluate the performance, three measures are used: F_1 macro, F_1 micro and energy error. Given the number of true positives (TP), false positives (FP), and false negatives (FN) the F_1 measure is define as:

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{6.6}$$

F_1 macro and F_1 micro are the measures derived from F_1 score. These are label based evaluation measures that depend on the averaging method (macro or micro) used [96]. F_1 micro measure is computed by averaging the F1 scores for each label. Whereas, F_1 macro is computed after summing true positives, false positives and false negatives across all labels.

$$F1_{macro} = F1\left(\sum_{i=1}^{N} TP_i, \sum_{i=1}^{N} FP_i, \sum_{i=1}^{N} FN_i\right) \tag{6.7}$$

$$F1_{micro} = \frac{1}{N}\sum_{i=1}^{N} F1(TP_i, FP_i, FN_i) \tag{6.8}$$

Here, TP_i, FP_i and FN_i denote the number of true positives, false positive and false negative for label i. N is the number of labels in the dataset.

The energy error is defined in [100]. It is defined as:

$$error = \frac{|\left(\sum_{i=1}^{N} Avergae\_power_i\right) - \left(\sum_{i=1}^{N} Actual\_power_i\right)|}{\left(\sum_{i=1}^{N} Actual\_power_i\right)} \tag{6.9}$$

Here, N is the number of appliances. We compare with MLKNN and RAKEL as benchmarks which are used in prior studies [101, 102]. These techniques use thresholded wavelet coefficients as input features.

Table 6.2: Performance evaluation on REDD

Method	F1-macro measure	F1-micro measure	Average energy error
MLKNN[97]	0.5931	**0.6034**	0.1067
RAKEL[96]	0.5334	0.5749	0.9949
Proposed (one layer)	0.5838	0.5884	0.0983
Proposed (two layers)	0.5857	0.5905	0.0892
Proposed (three layers)	**0.5981**	0.6001	**0.0766**
Proposed (four layers)	0.5951	0.5914	0.0827

Table 6.3: Appliance level evaluation on REDD

	LCTL(1-layer)		LCTL(3-layer)		MLKNN[97]		RAKEL[96]	
Device	ERROR	F1-score	ERROR	F1-score	ERROR	F1-score	ERROR	F1-score
Dishwasher	**0.0250**	0.5353	0.0786	**0.5722**	0.1250	0.4937	0.9964	0.3413
Kitchen Outlet	0.1492	0.5660	**0.0556**	0.5731	0.0647	0.6202	0.9952	**0.6645**
Lighting	0.1141	0.6959	0.1241	0.6768	**0.1105**	0.7384	0.9943	**0.6975**
Washer Dryer	**0.0250**	0.5379	0.0982	**0.5702**	0.0743	0.4304	0.9964	0.4302

Table 6.4: Performance evaluation on Pecan street

Method	F1-macro measure	F1-micro measure	Average energy error
MLKNN[97]	0.6227	0.6263	0.0989
RAKEL[96]	**0.6620**	**0.6663**	0.9995
Proposed (one layer)	0.6079	0.6079	0.0236
Proposed (two layers)	0.6089	0.6082	0.0223
Proposed (three layers)	0.6104	0.6104	**0.0115**
Proposed (four layers)	0.6087	0.6096	0.0228

In Tables 6.2 and 6.4, the aggregate results over all the houses are shown.

Table 6.5: Appliance level evaluation on Pecan street

	LCTL(1-layer)		LCTL(3-layer)		MLKNN[97]		RAKEL[96]	
Device	**ERROR**	**F1-score**	**ERROR**	**F1-score**	**ERROR**	**F1-score**	**ERROR**	**F1-score**
Site Meter	0.0239	0.8299	**0.0113**	0.8404	0.0696	0.8096	0.9995	**0.7072**
Air Conditioner	**0.0176**	0.5274	0.0125	0.5286	0.1381	0.5663	0.9995	**0.5841**
Electric Furnace	0.0194	0.5101	**0.0059**	0.5135	0.0899	0.5165	0.9995	**0.6568**
Socket	0.0302	0.5643	**0.0149**	0.5589	0.2696	0.5983	0.9995	**0.7071**

It shows that for the REDD dataset, we do better than most others, however for the Pecan Street dataset, we do not do as well as others in terms of the F1 scores. However, in terms of the energy error, we do much better. In practice, this is the error that is essential for NILM problems. We observe that our proposed deep methods significantly outperform all other shallow and deep techniques both in terms of F_1-score as well as energy error. The other observation is that, once we go deeper, the results improve from layers 1 to 3; but when we go even deeper, the problem of over-fitting arises, and the results deteriorate. Of the pre-existing shallow techniques, RAKEL performs decently in terms of F_1-score, but is very poor in terms of energy error; MLKNN yields balanced results.

In Tables 6.3 and 6.5, we show the performances at the appliance level. We have demonstrated results only for layer three since it yields the best results. We see that the conclusions remain the same. We generate the best results in terms of all metrics. RAKEL yields by far the worst results.

6.3 Discussion

In this chapter, we propose a supervised formulation for deep transform learning called LCDTL. This work proposes a joint approach to learn all the transforms simultaneously as a single optimization problem. Supervision is incorporated by adding a label consistency term on top of the unsupervised formulation. It is shown that the formulation can be generalized to perform multi-label classification without needing an external classifier. The comparison has been performed on state-of-art techniques. The proposed method is applied to a practical application of NILM. It outperforms the existing methods.

Chapter 7

Deep transformed subspace clustering

In this chapter, we propose to incorporate subspace clustering into the TL formulation. The core idea is to perform the clustering task in a transformed domain instead of processing the raw samples directly. The transform analysis step and the clustering are not done piecemeal but are performed jointly through the formulation of a coupled minimization problem. Subspace clustering assumes that the data is separable into separate subspaces; this assumption may not always hold. For such cases, we believe that, even if the raw data is not separable into subspaces, one can learn a deep representation such that the learned representation is separable into subspaces. To achieve the intended goal, we embed subspace clustering techniques (locally linear manifold clustering (LLMC), Sparse subspace clustering (SSC) and low-rank representation (LRR)) into DTL. The entire formulation is jointly learned; giving rise to a new class of methods

called deeply transformed subspace clustering (DTSC). The introduction and motivation for the problem are given in section 7.1. In section 7.2 literature review is discussed. Section 7.3 presents the proposed algorithm of deeply transformed subspace clustering (DTSC). The experimental effectiveness is demonstrated on two different image databases in section 7.4.

7.1 Introduction

The problem of clustering is well known. It studies how signals are naturally grouped together. One of the best-known applications of clustering is image segmentation, where there is no labelled data available, and one must distinguish between the background and foreground. Perhaps the simplest and most widely used clustering technique is the K-means [103]. It groups the samples such that the total distance of the data points within the cluster is minimized. The problem is NP-hard and hence is solved greedily. One of the limitations of K-means is that it operates on the raw data and hence fails to capture non-linear relationships. This can be simply fixed by resorting to the kernel K-means [104], where the standard euclidean distance between the samples typically used in K-means is replaced by a kernelized version of it. Spectral clustering [104], [105] extends the kernel K-means strategy by replacing the kernelized data matrix by a so-called affinity matrix. This allows generalizing the kernel metric to any similarity

measure. Subspace clustering techniques [106] is a particular class of spectral clustering approach which assumes that the samples from the same cluster lie in the same subspace. In practice, it requires to express each data point as a linear combination of the other data points. The associated linear weights then serve as inputs for creating the affinity matrix. In the past, it has been shown [107] that instead of applying subspace clustering on the raw data, a projection space can be learned such that the clustering is carried out in the projected domain as shown in figure 7.1. For instance, in [107], a tight-frame operator was learned from the data along with the subspace clustering formulation. In this work, we propose to adopt a similar concept as in [107], which is to perform subspace clustering in a transformed space as shown in figure 7.2, with the aim to obtain clusters with more useful features thanks to the transform step. Indeed, raw data have many irrelevant dimensions that could mask existing clusters in noisy data. Transform learning is thus expected to help in removing irrelevant and redundant dimensions of high-dimensional data. For improved versatility, we propose to replace the tight-frame transform from [107] by a more general linear transform operator, as it was done in [2]. A subspace clustering strategy based on LLMC is then incorporated in our transform learning framework, and the ensuing estimation problem is solved jointly by means of an alternating minimization algorithm.

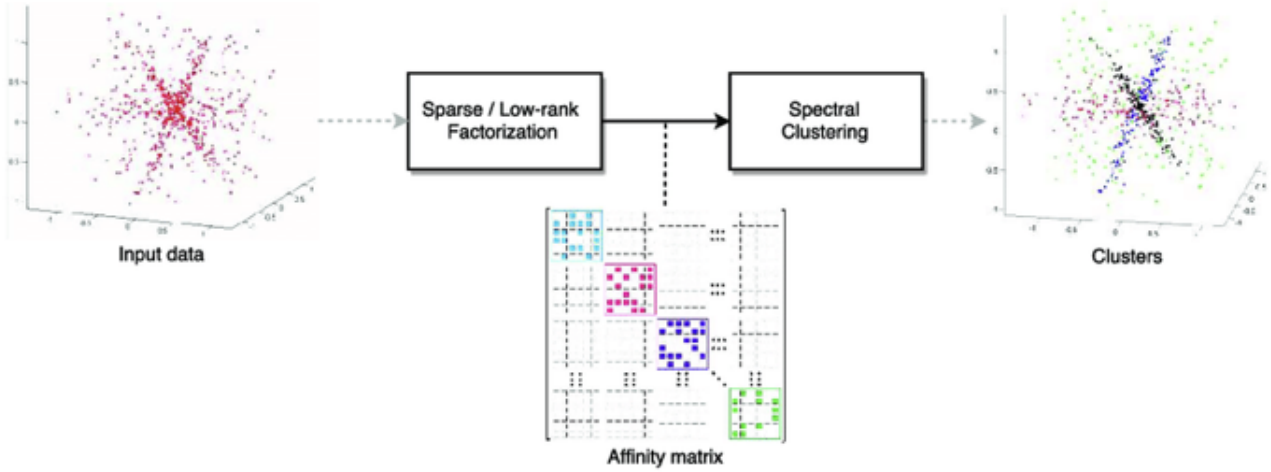


Figure 7.1: Illustration of the subspace clustering framework based on sparse and low-rank representation approaches for building the affinity matrix[1]

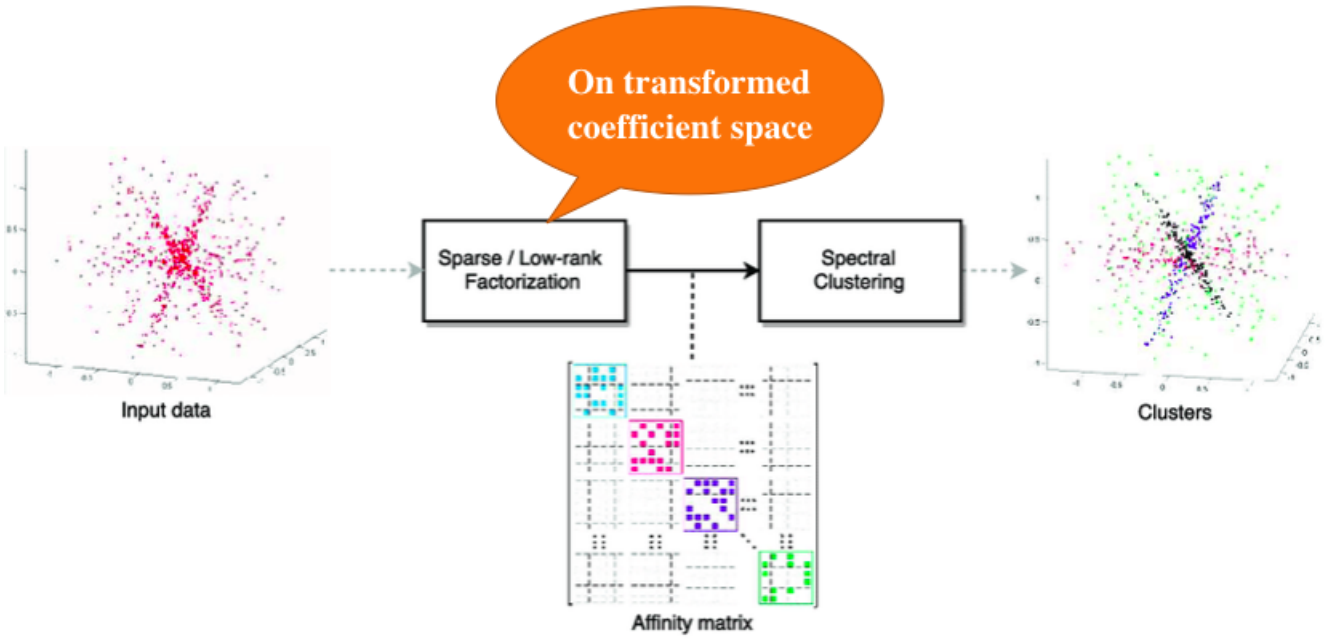


Figure 7.2: Illustration of the transformed subspace clustering framework based on sparse and low-rank representation approaches for building the affinity matrix

7.2 Literature review

Subspace clustering[108] is an extension to the basic clustering technique which clusters high dimensional data that lie in union of several low-dimensional subspaces. Let $\{S_l\}_{L=1}^{n}$ be an arrangement of n linear subspaces of $\mathbb{R}^D$ of dimensions $\{d_l\}_{L=1}^{n}$. Consider a given collection of N noise- free data points x_i, $i \in 1...N$ that lie in the union of the n subspaces. Denote the matrix containing all the data points as

$$X \triangleq [x_1...x_N] = [X_1...X_n]\Gamma$$

where $X_l \in \mathbb{R}^{D \times N_l}$ is a rank-d_l matrix of the $N_l > d_l$ points that lie in S_l and $\Gamma \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. Subspace clustering techniques such as LLMC [109], SSC [108] and LRR [110] express the samples as a linear combination of other samples. It tries to find clusters in different subspaces of the same dataset. Each data point is expressed as a linear combination of the other data points. This is expressed as,

$$x_i = X_{i^c} c_i, \; \forall i \; \in \; 1 \ldots N$$

Here above, for every , $i \in \mathbb{R}^m$ denotes the i^{th} sample, $X_{i^c} \in \mathbb{R}^{m \times n-1}$ gathers all the other samples column-wise and $c_i \in \mathbb{R}^{n-1}$ states for the corresponding linear weight vector.

In subspace clustering techniques, the general learning formulation is

expressed as follows,

$$\min_{c_i \in \mathbb{R}^{n-1}} ||x_i - X_{i^c} c_i||_2^2 + R(c_i) \forall i \in 1...N \tag{7.1}$$

Where R is a regularization function, depending on its nature, several formulations can be obtained. For LLMC, there is no regularization, i.e. $R = 0$. For SSC, R is a sparsity promoting penalty, such as l1-norm [108] or l0 pseudo-norm [111]. For LRR, R is a low-rank penalty, usually taking the form of a nuclear norm.

Let us define the coefficient matrix

$$C = [\zeta_1 | \zeta_2 | ... | \zeta_n] \in \mathbb{R}^{n \times n} \tag{7.2}$$

where, for every $i \in 1 \dots N$ $\zeta_i \in \mathbb{R}^n$, is a vector with the i^{th} entry equal to 0, and the remaining $n - 1$ entries equal to c_i. Once C is obtained for all the n samples by resolution of (equation), an affinity matrix $A \in \mathbb{R}^{n \times n}$ needs to be computed. Such matrix defines the similarity (inverse distance) between the samples and hence by applying some kind of cut (e.g., N-Cut), allows to segment the clusters. Several variants have been proposed for the definition of the affinity matrix [112]. For example, one option can be:

$$A = |C| + |C^T| \tag{7.3}$$

Where $|C|$ defines absolute value of each entry of the matrix C. Another option, retained in LRR, is to form the affinity matrix 7.1 from the scaled

left singular values of C. Since C is low rank, its skinny SVD reads $C = USV^T$. The affinity matrix is generated from scaling the left singular vectors by the corresponding square rooted singular values, such that For every $i \in 1 \dots n$ and $j \in 1 \dots n$,

$$A_{ij} = ([U_1 U_1^T]_{ij})^2 \tag{7.4}$$

where $U_1 = US^{1/2}$ Yet another way to generate the affinity matrix (usually for LLMC) is by:

$$A = |C| + |C^T| + |C^T C| \tag{7.5}$$

Once the affinity matrix is defined (by using any suitable formula), one needs to segment the clusters. Usually, the spectral clustering algorithm (normalized-cuts) is used for this purpose.

7.3 Proposed formulation

So far, there has been only a single work that incorporates subspace clustering into a deep learning framework. In [113], they incorporate the sparse subspace clustering formulation into the features from the deepest layer of a SAE. We propose a new framework called deeply transformed subspace clustering 7.2. Mathematically the formulation for DTL is as follows,

$$T_3 T_2 T_1 X = Z \tag{7.6}$$

Although we have not explicitly shown any activation function between the layers, we will implicitly impose ReLU type activation. We have shown the formulation for three layer. The generalization to more number of layers will be straightforward. The optimization problem is posed as:

$$\begin{aligned} &\min_{T_i,Z} ||T_3T_2T_1X - Z||_F^2 + \lambda \sum_{i=1}^{3} (||T_i||_F^2 - \log\det T_i) \\ &s.t \\ &T_1X \geq 0 \\ &T_2T_1X \geq 0 \end{aligned} \tag{7.7}$$

Note that we have dropped the sparsity promoting term on the coefficients. Usually, in deep learning, the dimensionality of the coefficients reduce in each layer; therefore, the representation is naturally compact. In this work, our goal is to embed the subspace clustering formulation into the DTL to jointly learn the representation and clustering. The notion is similar to prior studies [114] [115] [116], where the goal is to learn a deep subspace that is conducive to clustering. The difference between our work and the prior studies lies both in the choice of the clustering technique and the deep architecture. In subspace clustering, it is assumed that the data is naturally separated into certain subspaces (a selection of one or more dimensions). This may not always hold. Here, we are assuming that even if the original data is not separable into subspaces, by learning a non-linear representation of it (via DTL), the coefficients will fall into separate

subspaces. Mathematically our formulation can be expressed as,

$$\begin{aligned}
&\min_{T_3,T_2,T_1,Z,C} \underbrace{||T_3T_2T_1X - Z||_F^2 + \lambda \sum_{i=1}^{3} (||T_i||_F^2 - \log\det T_i)}_{Deep_transform} \\
&\underbrace{+\gamma \sum_i ||z_i - Z_{i^c} c_i||_2^2 + R(C)}_{subspace_clustering} \\
&s.t. \\
&T_1X \geq 0 \\
&T_2T_1X \geq 0
\end{aligned} \tag{7.8}$$

To solve equation 7.8, we follow the popular variable splitting strategy. After introducing the proxy variable, the AL is solved via ADMM. In our case equation 7.8, we introduce two proxy variables

$$T_2T_1X = X_3$$

and

$$T_1X = X_2$$

. The AL becomes,

$$\begin{aligned}
&\min_{T_1,T_2,T_3,X_2,X_3,Z,C} ||T_3X_3 - Z||_F^2 + \mu_1||T_2X_2 - X_3||_F^2 + \mu_2||T_1X - X_2||_F^2 \\
&+\lambda \sum_{i=1}^{3} (||T_i||_F^2 - \log\det T_i) + \gamma \sum_i ||z_i - Z_{i^c} c_i||_2^2 + R(C) \\
&s.t. \\
&X_3 \geq 0, X_2 \geq 0
\end{aligned} \tag{7.9}$$

In equation 7.9, the hyper-parameters μ_1 and μ_2 correspond to the representation in shallower layers. We argue that there is no reason to prefer one layer over the other. Therefore we assign $\mu_1 = \mu_2 = 1$. With this slight simplification, we have:

$$\begin{aligned} &\min_{T_1,T_2,T_3,X_2,X_3,Z,C} ||T_3X_3 - Z||_F^2 + ||T_2X_2 - X_3||_F^2 + ||T_1X - X_2||_F^2 \\ &+\lambda \sum_{i=1}^{3} (||T_i||_F^2 - \log\det T_i) + \gamma \sum_i ||z_i - Z_{i^c} c_i||_2^2 + R(C) \\ &s.t. \\ &X_3 \geq 0, X_2 \geq 0 \end{aligned} \tag{7.10}$$

The problem 7.10 can be solved using ADMM [117]. Each of the variables is updated separately from the following sub-problems.

- S_1 :

$$\min_{T_1} ||T_1X - X_2||_F^2 + \lambda(||T_1||_F^2 - \log\det T_1)$$

- S_2 :

$$\min_{T_2} ||T_2X_2 - X_3||_F^2 + \lambda(||T_2||_F^2 - \log\det T_2)$$

- S_3 :

$$\min_{T_3} ||T_3X_3 - Z||_F^2 + \lambda(||T_3||_F^2 - \log\det T_3)$$

- S_4 :

$$\min_{X_3} ||T_3X_3 - Z||_F^2 + ||T_2X_2 - X_3||_F^2 s.t. X_3 \geq 0$$

- S_5 :

$$\min_{X_2} ||T_2X_2 - X_3||_F^2 + ||T_1X - X_2||_F^2 s.t. X_2 \geq 0$$

- S_6 :

$$\min_{Z} ||T_3 X_3 - Z||_F^2 + \gamma \sum_i ||z_i - Z_{i^c} c_i||_2^2$$

- S_7 :

$$\min_{C} \sum_i ||z_i - Z_{i^c} c_i||_2^2 + R(C)$$

Sub-problems S_1 to S_3 are standard transform updates whose closed-form solution is given in [118]. S_4 and S_5 are least-square problems with closed-form updates: one first needs computing the pseudo-inverse followed by imputing all the negative values to zero. S_6 is a simple least-squares problem. The solution to S_7 will depend on the regularization used. With no regularization (locally linear manifold clustering), it will have a closed-form update via the pseudo-inverse. With $l1-$norm regularization S_7 will have to solved via some kind of ISTA such as [119]; this case pertains to SSC. When the regularizer in S_7 is a nuclear norm, one needs to solve it via singular value shrinkage [120]. This concludes the derivation of the main algorithm. Once, equation 7.8 is solved, our work proceeds in the same fashion as standard subspace clustering. Given C, we compute the affinity matrix using equation 7.5, which is then segmented / clustered by Normalized cut (more sophisticated techniques like optimized cuts [121] can also be used).

7.4 Experiments and results

Experiments were carried out on the COIL20 (object recognition) [64] and Extended YaleB (face recognition) [65] datasets. The COIL20 database contains 1,440 samples distributed over 20 objects, where each image is with the size of 32×32. The used YaleB consists of 2,414 samples from 38 individuals, where each image is with a size of 192×168. For our proposed method, we do not require any feature extraction technique. However, when we applied the orthogonal matching pursuit (OMP), DSC, TL-LLMC and TL-SSC algorithms on the raw data, very poor results were obtained. We thus chose to feed them with extracted features, based on DSIFT and HOG, reduced by PCA to the dimensionality of 300. Since the ground truth (class labels) for both datasets is available, clustering accuracy was measured in terms of accuracy, NMI, ARI, precision and F-score. The parameters are selected using the grid search.

Figure 7.3: Example images from Coil-20 dataset

We compare our methods deep transfomed locally linear manifold clus-

Figure 7.4: Example images from YaleB dataset

tering (DTLLMC) and DTSSC with three deep clustering benchmarks: DSC [63], deep K-means clustering (DKM) [114] and deep matrix factorization (DMF) [122]. The said studies have been published recently and have compared with traditional clustering techniques like matrix factorization, spectral clustering, subspace clustering, hierarchical clustering etc. Therefore, we do not compare with the traditional ones. We follow the experimental protocol from [113]. The results are shown in table 7.1 (COIL20) and table 7.2(YaleB). Since the last stage of all the clustering algorithms involves K-means, we ran the experiments 100 times and reported the mean and standard deviations. The parametric settings for the methods compared against have been taken from the respective papers. For our proposed technique, we have kept $\lambda = 0.1$ and $\gamma = 1$. transfomed locally linear manifold clustering (TLLMC) does not require the specification of any other parameter. transformed sparse subspace clustering (TSSC) has $\tau = 0.1$ as the sparsity promoting term and transformed low-rank representation (TLRR) has $\tau = 0.01$ as the rank deficiency term. The algorithms

are robust to these parametric values; changes by order of magnitude to either side do not affect the results statistically. We show the results of variation in the number of layers in table 7.3 and the results of the joint versus greedy solution in table 7.4. In the greedy solution, we learn the deep transform separately and feed the features into subspace clustering. Since the SSC yields better results than LLMC, we are showing the layer-wise comparison results and comparison with the greedy framework on SSC formulation only.

Table 7.1: comparison with benchmarks on COIL 20

Method	**DSC**[63]	**DKM**[114]	**DMF**[122]	**DTLLMC**	**DTSSC**
Accuracy	85.00	88.00	86.00	92.27	**99.01**
NMI	0.91	**0.94**	0.92	0.89	0.91
ARI	0.84	0.86	0.85	0.90	**0.96**
Precision	0.82	0.85	0.84	0.93	**0.98**
F-Measure	0.85	0.87	0.84	0.93	**0.99**

Table 7.2: Comparison with benchmarks on Extended Yale B

Method	**DSC**[63]	**DKM**[114]	**DMF**[122]	**DTLLMC**	**DTSSC**
Accuracy	88.00	91.00	89.00	93.13	**99.26**
NMI	0.90	0.92	0.90	0.92	**0.95**
ARI	0.83	0.90	0.83	0.91	**0.97**
Precision	0.79	0.91	0.80	0.94	**0.99**
F-Measure	0.83	0.90	0.84	0.94	**0.97**

Table 7.3: Results with number of layers on EYALEB: DTSSC

Method	**DTSSC-1layer**	**DTSSC-2layers**	**DTSSC-3layers**
Accuracy	99.22	99.23	**99.26**
NMI	0.9448	0.9451	**.9476**
ARI	0.9656	0.9663	**.9666**
Precision	0.9887	0.9900	**0.9912**
F-Measure	0.9567	0.9610	**0.9667**

Table 7.4: Comparison of joint vs greedy solution on Coil-20: DTSSC

Method	Greedy DTSSC-3layers	Proposed DTSSC-3layers
Accuracy	94.17	**99.01**
NMI	0.7874	**0.9087**
ARI	0.8115	**0.9553**
Precision	0.9082	**0.9771**
F-Measure	0.8576	**0.9886**

7.5 Discussion

From the results, we see that our proposed methods are considerably better (on average) than the rest in terms of every clustering metric. The main reason behind this to work is using relevant features from high dimensional dataset jointly by searching the appropriate cluster in some low-dimensional subspace of the dataset. The joint problem solving reduces the chances of masking relevant clusters by noisy features. We find that the results improve from one to three layers, but once we go beyond three layers, the results deteriorate. This is because, with more layers the number of parameters to learn increases; with a limited volume of training data (as is the case), this leads to over-fitting and subsequent deterioration of results. Between the joint and greedy formulations, the joint formulation yields better results. This is expected because this formulation learns the weights with the goal of clustering. The greedy unsupervised formulation does not have this advantage.

Chapter 8

Convolutional transform learning

In this chapter, a new representation learning technique called convolutional transform learning is discussed. In standard transform learning, a dense basis is learned that analyses the image to generate the representation from the image. Here, we learn a set of independent convolutional filters that operate on the images to produce representations (one corresponding to each filter). The significant advantage of our proposed approach is that it is entirely unsupervised; unlike CNN where labeled images are required for training. Moreover, it relies on a well-sounded minimization technique with established convergence guarantees.

The chapter is organized as follows. First a brief introduction of convolutional transform learning is given in section 8.1. The proposed algorithm is presented in section 8.2. The experiments are performed on standard datasets against the state-of-the-art solution for each problem. Experiments and results are discussed in section 8.4.

8.1 Introduction

The convolutional transform learning (CoTL) can be considered as a fully connected neural network. Usually, a neural network requires the learning of a lot of parameters (connection weights/transform basis). When training data is limited, this leads to over-fitting. CNN [123, 124] reduces the number of connections to be learned drastically by learning only a few convolutional filters. This automatically leads to improved generalization performance. Today the success of CNN has become very pervasive. However, there are some stark shortcomings in CNN. First, they cannot be learned without supervision since they are based on backpropagation. Getting large volumes of labeled data is a challenge in many application fields, e.g. medical imaging and remote sensing. Secondly, there is no guarantee in CNN that the filters learned will be mutually different; CNN just initializes them randomly and depends on the non-convergence of the backpropagation algorithm to maintain the mutual difference.

In recent times, there has been an increasing amount of works dealing with CoDL [125, 126, 127]. A brief overview of CoDL is given in section 2.11. However, the field is nascent, and the performance of such techniques has yet to reach those of CNN. Another issue with the DL formulation is its synthesis nature; in neural network terms, this would correspond to a feed-backward neural network. On the other hand, transform learning-based techniques are interpretable as a feed-forward neural network. Therefore,

in this work, we propose a convolutional version of TL, expecting improved performance in terms of analysis metrics.

8.2 Proposed formulation

The concept of the transform model can be easily interpreted as a neural network architecture in figure 2.9. Rethinking the transform T as connections from data X to learned features Z, which is a set of different convolutional filters. The different colors show different kernels/filters. The unsupervised learning of filters and coefficients using this model generates more general, unique and near to orthogonal filters that in turn generates good features to be used for solving many machine learning problems.

Let us consider a dataset $\left\{x^{(m)}\right\}_{1\leq m\leq M}$ with M entries in $\mathbb{R}^{\mathbb{N}}$. Our CoTL formulation relies on the key assumption that matrix T gathers a set of K kernels $t_1, \ldots, t_K$ with K entries, i.e.

$$T = [t_1 \mid \ldots \mid t_K] \in \mathbb{R}^{K\times K}. \tag{8.1}$$

The proposed model then reads:

$$(\forall m \in \{1, \ldots, M\}) \quad X^{(m)}T \approx Z_m. \tag{8.2}$$

Hereabove, $\left(X^{(m)}\right)_{1\leq m\leq M} \in \mathbb{R}^{N\times K}$ are Toeplitz matrices associated to

$(x^{(m)})_{1\leq m\leq M}$ such that:

$$\begin{aligned}(\forall m \in \{1,\dots,M\}) \quad X^{(m)}T &= \left[X^{(m)}t_1 \mid \dots \mid X^{(m)}t_K\right] \\ &= \left[t_1 * x^{(m)} \mid \dots \mid t_K * x^{(m)}\right]\end{aligned} \tag{8.3}$$

where $*$ is a discrete convolution operator with suitable padding, and

$$(\forall m \in \{1,\dots,M\}) \quad Z_m = \left[z_1^{(m)} \mid \dots \mid z_K^{(m)}\right], \tag{8.4}$$

contains the coefficients associated to each entry $m \in \{1,\dots,M\}$ of the dataset. Let us denote:

$$Z = [Z_1^\top \mid \dots \mid Z_M^\top]^\top \in \mathbb{R}^{NM\times K}. \tag{8.5}$$

The goal is then to estimate (T, Z) from $\left\{x^{(m)}\right\}_{1\leq m\leq M}$. To this aim, we propose to solve the following optimization problem generalizing equation (2.21) to our convolutional learning framework:

$$\min_{T\in\mathbb{R}^{K\times K}, Z\in\mathbb{R}^{NM\times K}} F(T, Z) \tag{8.6}$$

where the objective function F is defined, for every $T \in \mathbb{R}^{K\times K}$ and every

$Z \in \mathbb{R}^{NM \times K}$ as:

$$
\begin{aligned}
F(T, Z) = & \frac{1}{2} \sum_{k=1}^{K} \sum_{m=1}^{M} \|t_k * x^{(m)} - z_k^{(m)}\|_2^2 \\
& + \sum_{k=1}^{K} \sum_{m=1}^{M} \left(\beta \|z_k^{(K)}\|_1 + \iota_{[0,+\infty[}(z_k^{(K)}) \right) \\
& + \mu \|T\|_F^2 - \lambda \log\det T \qquad (8.7) \\
= & \frac{1}{2} \sum_{m=1}^{M} \|X^{(m)} T - Z_m\|_F^2 + \mu \|T\|_F^2 \\
& - \lambda \log\det T + \beta \|Z\|_1 + \iota_{[0,+\infty[^{NM \times K}}(Z). \qquad (8.8)
\end{aligned}
$$

Hereabove, function $\iota_{[0,+\infty[}$ denotes the indicator function of the positive orthant [1], equals to 0 for nonnegative entries, $+\infty$ elsewhere. Moreover, $(\lambda, \mu, \beta) \in]0, +\infty[^3$ are regularization parameters.

8.2.1 Optimization algorithm

The resolution of problem equation (8.6) requires an efficient algorithm for dealing with nonsmooth functions and hard constraints. In the optimization literature, proximal algorithms constitute one of the most efficient approaches to tackle such problems [128, 129, 130]. The key tool in those methods is the proximity operator [131, 132] of a proper, lower semi-continuous, convex function $\psi : \mathbb{R}^N \mapsto]-\infty, +\infty]$ defined as:[2]

$$
(\forall \tilde{z} \in \mathbb{R}^N) prox_{\psi}(\tilde{z}) = \arg\min_{z \in \mathbb{R}^N} \psi(z) + \frac{1}{2} ||z - \tilde{z}||^2. \qquad (8.9)
$$

[1] set of positive real numbers
[2] See also http://proximity-operator.net/

Problem equation (8.6) fits nicely into the framework provided by the alternating proximal algorithm from [130],[133]. For any initialization $T^{[0]} \in \mathbb{R}^{K\times K}$ and $Z^{[0]} \in \mathbb{R}^{NM\times K}$, its iterations are as follows:

$$\begin{aligned} &\text{For} \quad i = 0, 1, \dots \\ &\left\lfloor \begin{aligned} T^{[i+1]} &= \operatorname{prox}_{\gamma_1 F(\cdot, Z^{[i]})}\left(T^{[i]}\right) \\ Z^{[i+1]} &= \operatorname{prox}_{\gamma_2 F(T^{[i+1]}, \cdot)}\left(Z^{[i]}\right) \end{aligned} \right. \end{aligned} \tag{8.10}$$

where γ_1 and γ_2 are some positive constants. The convergence of sequence $(T^{(i)}, Z^{(i)})_{i\in N}$ to a minimizer of F is guaranteed, as a consequence of the convergence properties of the proximal regularization of Gauss-Seidel method algorithm [3] established in [130]. In the remaining of this section, we show that the updates on both variables T and Z have closed-form expressions, and thus can be computed with high precision in an efficient manner.

8.2.1.1 Update of T

Let $i \in \mathbb{N}$. Then, by definition,

$$T^{[i+1]} = \operatorname{prox}_{\gamma_1 F(\cdot, Z^{[i]})}\left(T^{[i]}\right) \tag{8.11}$$

$$\begin{aligned} &= \operatorname{argmin}_{T\in\mathbb{R}^{K\times K}} \frac{1}{2}\sum_{m=1}^{M} \|X^{(m)}T - Z_m^{[i]}\|_F^2 \\ &+ \mu\|T\|_F^2 - \lambda \log\det T + \frac{1}{2\gamma_1}\|T - T^{[i]}\|_F^2. \end{aligned} \tag{8.12}$$

[3] Algorithm 8.10 is a particular case of the proximal regularization of the Gauss-Seidel method algorithm. The convergence of the latter is established in [120]. Thus, Algorithm 6.10 converges too.

Using [134], we deduce that:

$$T^{[i+1]} = \frac{1}{2}\Lambda^{-1/2} V \left(\Sigma + (\Sigma^2 + 2\lambda I_K)^{1/2}\right) U^\top, \tag{8.13}$$

with

$$\Lambda = \sum_{m=1}^{M} (X^{(m)})^\top X^{(m)} + \gamma_1^{-1} I_K + 2\mu I_K, \tag{8.14}$$

the singular value decomposition:

$$U\Sigma V^\top = \left(\sum_{m=1}^{M} (Z_m^{[i]})^\top X^{(m)} + \gamma_1^{-1} T^{[i]}\right) \Lambda^{-1/2}, \tag{8.15}$$

and I_K the identity matrix of $\mathbb{R}^K$.

8.2.2 Remark for rectangular T:

Let us emphasize that our approach and the above update can easily be generalized to the case when matrix T is rectangular, that is $T \in \mathbb{R}^{K_1 \times K_2}$ with not necessarily equality between K_1 and K_2. Then, the penalization term on T should be replaced by:

$$(\forall T \in \mathbb{R}^{K_1 \times K_2}) \quad R(T) = \begin{cases} \mu \|T\|_F^2 - \lambda \sum_{k=1}^{K} \log(\lambda_k) & \text{if} \quad T \in \mathcal{X}_K^{++}, \\ +\infty & \text{otherwise,} \end{cases} \tag{8.16}$$

with $K = \min(K_1, K_2)$, $(\lambda_k)_{1 \leq k \leq K}$ are the singular values of T and $\mathcal{X}_K^{++}$ indicates the set of matrices $T \in \mathbb{R}^{K_1 \times K_2}$ with strictly positive singular values (i.e. T has rank equals to K). The gradient of equation (8.16) on

its definition domain reads:

$$(\forall T \in \mathcal{X}_K^{++}) \quad \nabla R(T) = 2\mu T - \lambda T^{\dagger}, \tag{8.17}$$

with $(\cdot)^{\dagger}$ the pseudo-inverse operation (equivalent to inverse, when $K_1 = K_2 = K$). Using Proposition 24.68 from [135], we can determine the new update for variable T in our algorithm:

Let $i \in \mathbb{N}$. Then:

$$T^{[i+1]} = \text{prox}_{\gamma_1 F(\cdot, Z^{[i]})}\left(T^{[i]}\right) \tag{8.18}$$

$$= \text{argmin}_{T \in \mathbb{R}^{K_1 \times K_2}} \frac{1}{2} \sum_{m=1}^{M} \|X^{(m)} T - Z_m^{[i]}\|_F^2 + \mu \|T\|_F^2 + \lambda R(T) + \frac{1}{2\gamma_1} \|T - T^{[i]}\|_F^2 \tag{8.19}$$

$$= \frac{1}{2} \Lambda^{-1} U \text{Diag}\left(\left[\sigma_1 + (\sigma_1^2 + 2\lambda)^{1/2}, \ldots, \sigma_K + (\sigma_K^2 + 2\lambda)^{1/2}, 0, \ldots, 0\right]\right) V^{\top} \tag{8.20}$$

with

$$\Lambda^{\top} \Lambda = \sum_{m=1}^{M} (X^{(m)})^{\top} X^{(m)} + \gamma_1^{-1} I_{K_1} + 2\mu I_{K_1}, \tag{8.21}$$

and the singular value decomposition:

$$U \Sigma V^{\top} = \left(\sum_{m=1}^{M} (Z_m^{[i]})^{\top} X^{(m)} + \gamma_1^{-1} T^{[i]} \right) \Lambda^{-1}, \tag{8.22}$$

with $U \in \mathbb{R}^{K_1 \times K_1}$, $V \in \mathbb{R}^{K_2 \times K_2}$ orthogonal matrices and

$$\Sigma = \text{Diag}\left([\sigma_1, \ldots, \sigma_K, 0, \ldots, 0]\right).$$

The impact of log-det term in equation (8.7) is straight-forward. Such penalty allows ensuring that the kernels are diverse enough to capture good correlations and hence generate good features. Changing the penalty parameter associated with the log-det term has an important impact on the learned kernels. When the kernel size equals the number of its elements (i.e., square case), then a full rank property is enforced on T, and in the limit case when μ tends to infinity, the operator T is such that $T^{-1} = \frac{2\mu}{\lambda}T$.

8.2.2.1 Update of Z

Let $i \in \mathbb{N}$. Then, using the definition of the proximity operator,

$$Z^{[i+1]} = \text{prox}_{\gamma_2 F(T^{[i+1],\cdot})}\left(Z^{[i]}\right) \tag{8.23}$$

$$= \text{argmin}_{Z\in\mathbb{R}^{MN\times K}} \frac{1}{2}\sum_{m=1}^{M} \|X^{(m)}T^{[i+1]} - Z_m\|_F^2 + \beta\|Z\|_1 + \iota_{[0,+\infty[^{MN\times K}}(Z) + \frac{1}{2\gamma_2}\|Z - Z^{[i]}\|_F^2. \tag{8.24}$$

By relying on the useful properties of the proximity operator listed in [132], we obtain that, for every $m \in \{1,\ldots,M\}$,

$$Z_m^{[i+1]} = \max\left(\mathcal{X}_{\frac{\gamma_2\beta}{\gamma_2+1}}\left(\frac{Z_m^{[i]} + \gamma_2 Z^{(m)}T^{[i+1]}}{\gamma_2+1}\right), 0\right)^{4} \tag{8.25}$$

[4] max(A,0):We truncate to 0 all negative entries of the matrix A.

where $\mathcal{X}_\theta$ denotes the soft thresholding operator with parameter $\theta \geq 0$, i.e.:

$$(\forall u \in \mathbb{R}) \quad \mathcal{X}_\theta(u) = \begin{cases} u + \theta & \text{if} \quad u < -\theta \\ 0 & \text{if} \quad u \in [-\theta, \theta] \\ u - \theta & \text{if} \quad u > \theta. \end{cases} \tag{8.26}$$

Remark:

First, the function in 8.24 is fully separable, i.e. it can be written as a sum over all the entries of X. So we can resonate for the scalar function:

$$\frac{1}{2}([X^m T^{i+1}]_{p,q} - Z_{p,q,r})^2 + \beta|Z_{p,q,r}| + p_{[0,+\infty[}(Z_{p,q,r}) + \frac{1}{2}\gamma_2(Z_{p,q,r} - Z^i_{p,q,r})^2 \tag{8.27}$$

Where

$$\beta|Z_{p,q,r}| + p_{[0,+\infty[}(Z_{p,q,r})$$

is the function 'ix' of table 10.2 in [132]. The quadratic terms can be treated as in the case 'iv' of table 10.1 in [132].

8.3 Multi-layer case with a simplified 2D model

Instead of learning only a single layer of convolutional filters, better results can be obtained if multiple convolutional layers are learned. We consider three layers deep architecture.

8.3.1 Problem formulation

Let us now address the generalized case of learning L convolutional transform layers for learning convolutional filters in the deeper network. We need to introduce novel notations:

$$T_\ell = [t_{1,\ell} \ \dots \ t_{K,\ell}] \, , \forall \ell \in \{1, \dots, L\} \tag{8.28}$$

$$Z_{m,\ell} = [z_1^{(m,\ell)} \ \dots \ z_K^{(m,\ell)}] \in \mathbb{R}^{N \times K} \quad \forall m \in \{1, \dots, M\} \, , \forall \ell \in \{1, \dots, L\} \tag{8.29}$$

$$Z_\ell = [Z_{1,\ell}^\top \ \dots \ Z_{M,\ell}^\top]^\top \in \mathbb{R}^{NM \times K} \quad \forall \ell \in \{1, \dots, L\} \tag{8.30}$$

We want to solve:

$$\underset{(T_\ell)_{1 \leq \ell \leq L} \in \mathbb{R}^{K \times K}, (Z_\ell)_{1 \leq \ell \leq L} \in \mathbb{R}^{NM \times K}}{\text{minimize}} \quad F(T_1, \dots, T_L, Z_1, \dots, Z_L) \tag{8.31}$$

with, for all $(T_\ell)_{1 \leq \ell \leq L} \in \mathbb{R}^{K \times K}$ and $(Z_\ell)_{1 \leq \ell \leq L} \in \mathbb{R}^{NM \times K}$,

$$F(T, Z) = \sum_{\ell=1}^{L} \begin{bmatrix} \frac{1}{2} \sum\limits_{m=1}^{M} ||\mathcal{Z}_{m,\ell-1} T_\ell - Z_{m,\ell}||_F^2 + \mu ||T_\ell||_F^2 \\ -\lambda \log\det(T_\ell) + \beta ||Z_\ell||_1 + \iota_{[0,+\infty[}(Z_\ell) \end{bmatrix}, \tag{8.32}$$

where $\mathcal{Z}_{m,0} = X^{(m)}$ and, for every $\ell \in \{2, \dots, L\}$, $\mathcal{Z}_{m,\ell-1} \in \mathbb{R}^{N \times K}$ such that

$$\mathcal{Z}_{m,\ell-1} T_\ell = [\mathcal{Z}_{m,\ell-1} t_{1,\ell} \ \dots \ \mathcal{Z}_{m,\ell-1} t_{K,\ell}] \tag{8.33}$$

$$= \left[t_{1,\ell} * z_1^{(m,\ell-1)} \dots t_{K,\ell} * z_K^{(m,\ell-1)} \right]. \tag{8.34}$$

8.3.2 Optimization algorithm

Here again, we propose to use an alternating proximal minimization algorithm:

$$
\begin{array}{l}
\text{For} \quad n = 0, 1, 2, \ldots \\
\left| \begin{array}{l}
\text{For} \quad \ell = 1, 2, \ldots, L \\
\left| \begin{array}{ll}
T_\ell^{[i+1]} & = \text{prox}_{\gamma_1 F(T_1^{[i+1]}, \ldots, T_{\ell-1}^{[i+1]}, \cdot, T_{\ell+1}^{[i]}, \ldots, T_L^{[i]}, Z_1^{[i+1]}, \ldots, Z_\ell^{[i+1]}, Z_{\ell+1}^{[i]}, \ldots, Z_L^{[i]})} \left(T_\ell^{[i]} \right) \\
Z_\ell^{[i+1]} & = \text{prox}_{\gamma_2 F(T_1^{[i+1]}, \ldots, T_{\ell-1}^{[i+1]}, T_\ell^{[i+1]}, T_{\ell+1}^{[i]}, \ldots, T_L^{[i]}, Z_1^{[i+1]}, \ldots, Z_{\ell-1}^{[i+1]}, \cdot, Z_{\ell+1}^{[i]}, \ldots, Z_L^{[i]})} \left(Z_\ell^{[i]} \right)
\end{array} \right.
\end{array} \right.
\end{array}
\tag{8.35}
$$

with $(T_\ell^{[0]})_{1 \leq \ell \leq L} \in \mathbb{R}^{K \times K}$, $(Z_\ell^{[0]})_{1 \leq \ell \leq L} \in \mathbb{R}^{NM \times K}$ and γ_1 and γ_2 some positive constants.

8.3.2.1 Update of T

Let $i \in \mathbb{N}$ and $\ell \in \{1, \ldots, L\}$. Then:

$$
\begin{aligned}
T_\ell^{[i+1]} &= \text{prox}_{\gamma_1 F(T_1^{[i+1]}, \ldots, T_{\ell-1}^{[i+1]}, \cdot, T_{\ell+1}^{[i]}, \ldots, T_L^{[i]}, Z_1^{[i+1]}, \ldots, Z_\ell^{[i+1]}, Z_{\ell+1}^{[i]}, \ldots, Z_L^{[i]})} \left(T_\ell^{[i]} \right) && (8.36) \\
&= \text{argmin}_{T_\ell \in \mathbb{R}^{K \times K}} \frac{1}{2} \sum_{m=1}^{M} \| \mathcal{Z}_{m,\ell-1}^{[i+1]} T_\ell - Z_{m,\ell}^{[i]} \|_F^2 + \mu \| T_\ell \|_F^2 - \lambda \log \det(T_\ell) \\
&\qquad + \frac{1}{2\gamma_1} \| T_\ell - T_\ell^{[i]} \|_F^2 && (8.37) \\
&= \frac{1}{2} \Lambda^{-1} V \left(\Sigma + \left(\Sigma^2 + 2\lambda I_K \right)^{1/2} \right) U^\top && (8.38)
\end{aligned}
$$

with

$$
\Lambda^\top \Lambda = \sum_{m=1}^{M} (\mathcal{Z}_{m,\ell-1}^{[i+1]})^\top (\mathcal{Z}_{m,\ell-1}^{[i+1]}) + \gamma_1^{-1} I_K + 2\mu I_K, \tag{8.39}
$$

and

$$U\Sigma V^\top = \left(\sum_{m=1}^{M} (Z_{m,\ell}^{[i]})^\top (\mathcal{Z}_{m,\ell-1}^{[i+1]}) + \gamma_1^{-1} T_\ell^{[i]}\right) \Lambda^{-1}. \tag{8.40}$$

8.3.2.2 Update of Z

Let $i \in \mathbb{N}$ and $\ell \in \{2, \dots, L-1\}$. Then:

$$Z_\ell^{[i+1]} = \operatorname{prox}_{\gamma_2 F(T_1^{[i+1]},\dots,T_{\ell-1}^{[i+1]},T_\ell^{[i+1]},T_{\ell+1}^{[i]},\dots,T_L^{[i]},Z_1^{[i+1]},\dots,Z_{\ell-1}^{[i+1]},\cdot,Z_{\ell+1}^{[i]},\dots,Z_L^{[i]})}\left(Z_\ell^{[i]}\right) \tag{8.41}$$

$$\begin{aligned} &= \operatorname{argmin}_{Z_\ell \in \mathbb{R}^{MN\times K}} \frac{1}{2}\sum_{m=1}^{M} \|\mathcal{Z}_{m,\ell-1}^{[i+1]} T_\ell^{[i+1]} - Z_{m,\ell}\|_F^2 + \frac{1}{2}\sum_{m=1}^{M} \|\mathcal{Z}_{m,\ell} T_{\ell+1}^{[i+1]} - Z_{m,\ell+1}^{[i]}\|_F^2 \\ &\quad + \beta\|X_\ell\|_1 + \iota_{[0,+\infty[}(Z_\ell) + \frac{1}{2\gamma_2}\|Z_\ell - Z_\ell^{[i]}\|_F^2 \end{aligned} \tag{8.42}$$

When $\ell = 1$, since $\mathcal{Z}_{m,0} = X^{(m)}$, the minimization problem reads :

$$Z_1^{[i+1]} = \operatorname{prox}_{\gamma_2 F(T_1^{[i+1]},T_2^{[i]},\dots,T_L^{[i]},\cdot,Z_2^{[i]},\dots,Z_L^{[i]})}\left(Z_1^{[i]}\right) \tag{8.43}$$

$$\begin{aligned} &= \operatorname{argmin}_{Z_1 \in \mathbb{R}^{MN\times K}} \frac{1}{2}\sum_{m=1}^{M} \|X^{(m)} T_1^{i+1]} - Z_{m,1}\|_F^2 + \frac{1}{2}\sum_{m=1}^{M} \|\mathcal{Z}_{m,1} T_2^{[i]} - Z_{m,2}^{[i]}\|_F^2 \\ &\quad + \beta\|Z_1\|_1 + \iota_{[0,+\infty[}(Z_1) + \frac{1}{2\gamma_2}\|Z_1 - Z_1^{[i]}\|_F^2 \end{aligned} \tag{8.44}$$

When $\ell = L$, we have :

$$Z_L^{[i+1]} = \operatorname{prox}_{\gamma_2 F(T_1^{[i+1]},\dots,T_L^{[i+1]},Z_1^{[i+1]},\dots,Z_{L-1}^{[i+1]},\cdot)}\left(Z_L^{[i]}\right) \tag{8.45}$$

$$\begin{aligned} &= \operatorname{argmin}_{Z_L \in \mathbb{R}^{MN\times K}} \frac{1}{2}\sum_{m=1}^{M} \|\mathcal{Z}_{m,L-1}^{[i+1]} T_L^{[i+1]} - Z_{m,L}\|_F^2 \\ &\quad + \beta\|Z_L\|_1 + \iota_{[0,+\infty[}(Z_L) + \frac{1}{2\gamma_2}\|Z_L - Z_L^{[i]}\|_F^2 \end{aligned} \tag{8.46}$$

8.4 Experiments and results

To assess the performance of the proposed approach, we considered the following datasets of small-to-medium size, on which we performed feature extraction. The images very small images which are downsampled from original full-size images.

- YALE [136]: The Yale dataset contains 165 images of 15 individuals, downscaled to 32-by-32 pixels. There are 11 images per subject, one per different facial expression or configuration. For our experiments, we shuffled all the samples and took 70% for training and 30% for testing. Moreover, we generated different train/test splits YALE-2,..., YALE-8. In a YALE-p dataset, p images per subject are kept in the train set, and $11-p$ images are kept in the test set. So doing, the train set contains $15p$ images, and the test set contains $15(11-p)$ images.
- E-YALE-B [58]: The Extended Yale B database contains 2432 images with 38 subjects under 64 illumination conditions. Each image is cropped to 192-by-168 pixels and downscaled to 48-by-42 pixels. For our experiments, we shuffled all the samples, took 70% for training and 30% for testing.
- AR-Face [59]: This database contains more than 4000 images of 126 different subjects (70 male and 56 female). The images have various facial expressions, the lighting varies, and some of the images are par-

tially occluded by sunglasses and scarves. For our experiments, we selected 2600 images of 100 individuals (50 males and 50 females), which is 26 different images for each subject. The train set contains 2000 images, and 600 images are kept in the test set. Each image has 540 features.

8.4.1 Classification accuracy

We compared the proposed feature extraction approach CoTL with TL [137] and DL [70]. Since our method is unsupervised, it is only fair to compare with other unsupervised representation learning tools. As these are all unsupervised learning methods, we evaluated their performance by feeding the extracted features to a supervised classifier and then computing the classification accuracy. We also performed the classification directly on raw images (Raw). For the classification task, we used two popular techniques: KNN and SVM. Our algorithm was ran until convergence (typically 10 iterations are sufficient), with parameters $\gamma_1 = \gamma_2 = 1$. For every tested method, the hyper-parameters were cross-validated.

We found that the proposed method CoTL yields better results than regular TL for all the considered datasets and classifiers, while being better than dictionary learning (DL) on all the datasets when using the nearest neighbour classifier, and on YALE, E-YALE-B, YALE-2, YALE-6, YALE-7, and YALE-8 when using an SVM classifier. To complete our analysis, we also compared to a CNN trained on raw images through a standard

supervised classification procedure. We used a custom CNN composed of the following layers: $Conv[64 \times 3 \times 3] \rightarrow ReLU \rightarrow Pool[2 \times 2] \rightarrow BNorm \rightarrow Conv[128 \times 3 \times 3] \rightarrow ReLU \rightarrow Pool[2 \times 2] \rightarrow BNorm \rightarrow DO \rightarrow FC[256] \rightarrow ReLU \rightarrow FC[classes] \rightarrow Softmax$.

According to the results reported in Table, the proposed CoTL compares favourably with CNN. This may be related to the fact that CNN are known to require large training sets in order to achieve breakthrough performance, whereas the considered datasets are small.

Another important observation is that in most of our experiments on downsampled data, we have observed that SVM outperforms KNN. Intuitively, when we have a limited set of points in many dimensions, SVM tends to be very good because it should be able to find the linear separation that should exist. Moreover, SVM is expected to be robust to outliers since it only uses the most relevant points to find the linear separation (support vectors). In general, if we have large datasets in a low dimensional space, then KNN is probably a suitable choice. If we have few points in the dataset, lying in a high dimensional space, then a linear SVM is perhaps better.

Our classification accuracy is comparable to the one obtained with CNN. It should, however, be emphasized that the upvote for the proposed methodology is its unsupervised way of learning convolved features in contrast to CNN, where convolved features are learned in a supervised manner.

Table 8.1: Classification accuracy on benchmark datasets.

	Dataset	Raw	TL	DL	CoTL
KNN	**YALE**	58.00	68.00	54.00	**70.00**
	E-YALE-B	71.03	72.28	71.72	**84.00**
	AR-Faces	55.00	53.50	54.50	**56.00**
	YALE-2	43.40	49.63	43.70	**51.85**
	YALE-3	49.40	48.33	47.50	**55.83**
	YALE-4	52.38	50.48	44.76	**54.28**
	YALE-5	51.11	53.33	44.44	**54.44**
	YALE-6	53.33	50.67	50.67	**57.33**
	YALE-7	60.20	61.67	53.33	**66.67**
	YALE-8	63.60	57.78	57.78	**71.11**
SVM	**YALE**	68.00	78.00	80.00	**88.00**
	E-YALE-B	93.24	94.21	95.58	**97.38**
	AR-Faces	87.33	84.33	**97.67**	88.87
	YALE-2	58.52	51.11	58.52	**62.22**
	YALE-3	62.50	60.83	**66.67**	64.17
	YALE-4	60.95	53.33	**64.76**	64.52
	YALE-5	66.67	57.78	**68.89**	66.67
	YALE-6	73.33	61.33	81.33	**82.67**
	YALE-7	80.00	66.67	78.33	**83.33**
	YALE-8	80.00	71.11	80.00	**84.44**
CNN	**YALE**	84.00	-	-	-
	E-YALE-B	98.60	-	-	-
	AR-Faces	95.50	-	-	-
	YALE-2	62.96	-	-	-
	YALE-3	64.17	-	-	-
	YALE-4	67.60	-	-	-
	YALE-5	74.44	-	-	-
	YALE-6	76.00	-	-	-
	YALE-7	81.67	-	-	-
	YALE-8	82.22	-	-	-

The learned features by the proposed method are general enough to be used for other image processing tasks by making small changes in the formulation.

8.4.2 Computational time

The proposed method is tested on small size images, which are downsampled from the original full-size images. While the DL and TL methods take one to ten seconds for learning representations, the proposed approach takes around one minute. The difference in terms of computational time

is simply related to the fact that, in the case of TL and DL, the transform requires a matrix-vector product while in the proposed approach, convolution and deconvolution operations are needed.

8.4.3 Analysis of the learned kernels

A given number K_2 of kernels with $K_1 = K_2^2$ coefficients is learned to represent the dataset ideally. Each kernel t_k is convolved with the image x to generate a different feature vector z_k. The intra-kernel diversity is taken care of by the penalties in the proposed formulation.

The proposed algorithm is capable of learning nontrivial and nonidentical kernels, thanks to the regularization on T present in equation (8.6). In particular, the results reported in table 8.1 were obtained by fixing $K_2 = 5$, which corresponds to a good trade-off between model accuracy and complexity.

Since $K_1 > K_2$ here, the estimated T is rectangular and over-complete. The retrieved kernels are distinct from each other, as soon as $\mu > 0$. In contrast, if we had considered a large number of small size kernels (i.e., rectangular case with $K_1 < K_2$), T would have been under-complete and the number of distinct kernels would be equals to the smallest dimension of T, that is K_1; the others being some linear combination of each other.

Note that the initialization of T plays no role in the learning process since the optimization problem in equation (8.6) is convex.

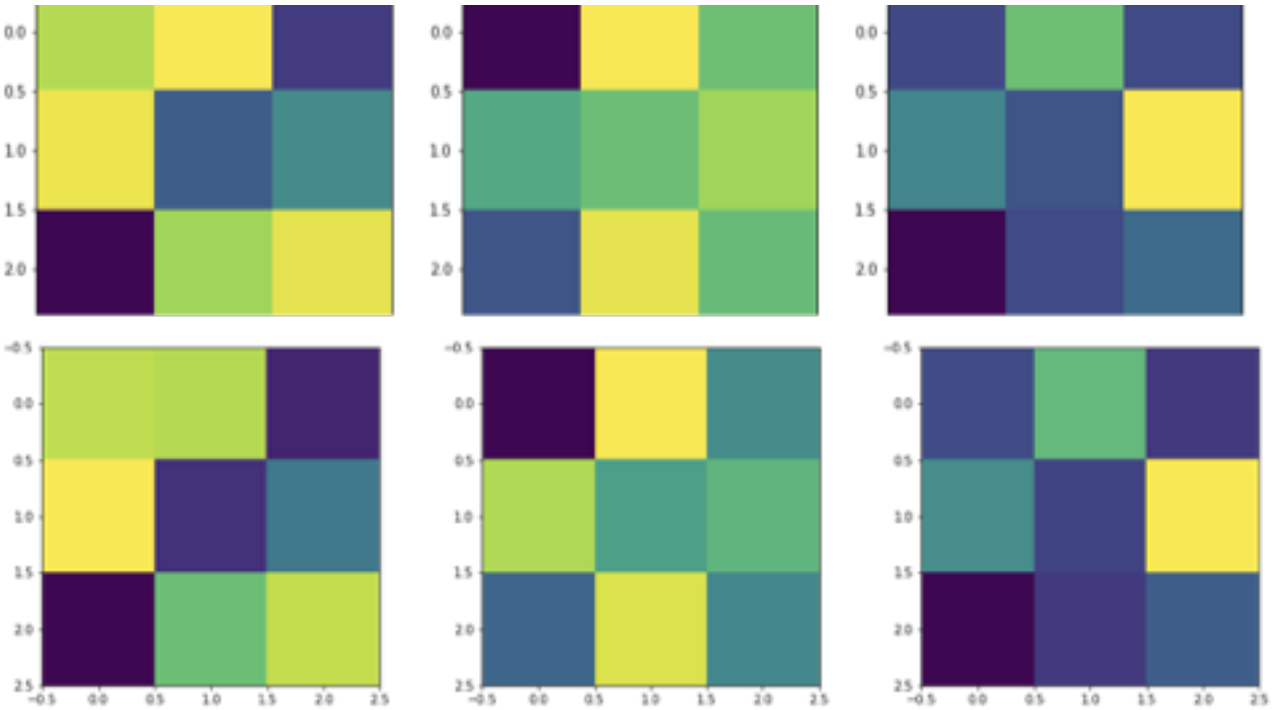
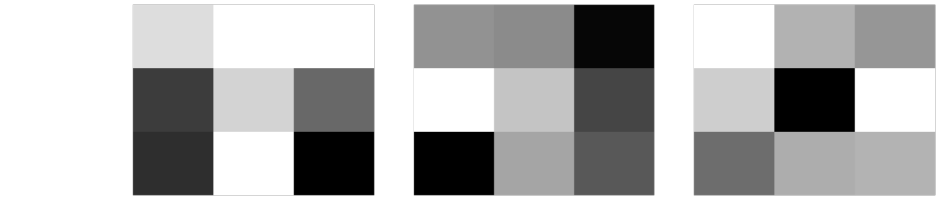


Figure 8.1: Kernels in CNN(top) and CTL(bottom)

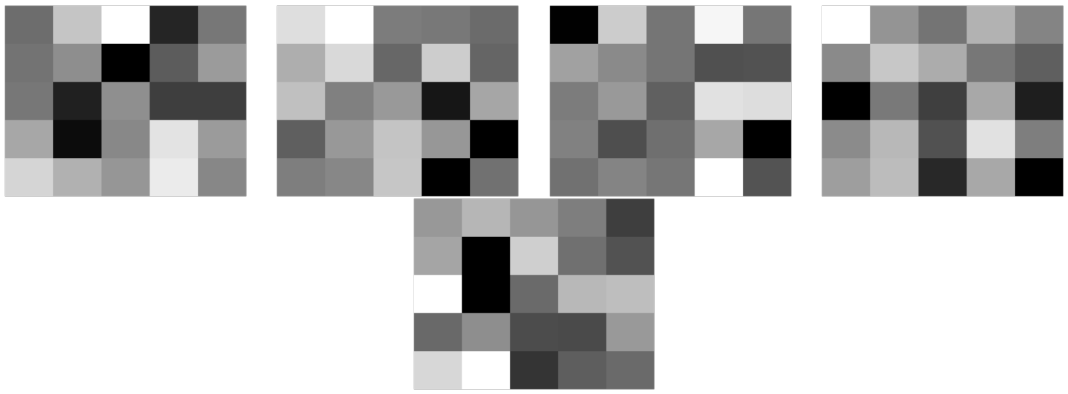
It can be easily observed from figure 8.1 that there is a close relationship between CNN and CTL.

8.4.4 Classification accuracy with deep convolutional transform learning

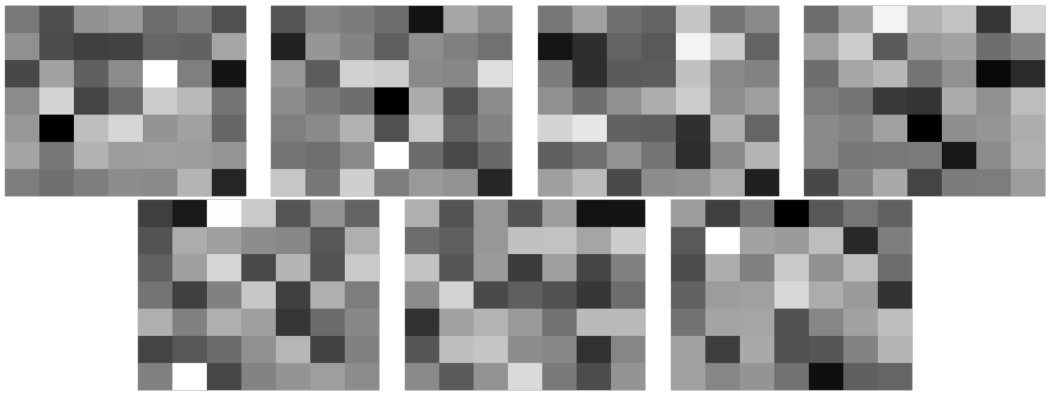
We show that the accuracy of deep transform learning indeed improves when one goes deeper. Going deep beyond three layers makes performance degrade as the model tends to overfit for the small training set. To elucidate, we have used a simple SVM classifier for deep convolutional transform learning (DCoTL). The results are shown in Table 8.2 for levels 1, 2, 3 and 4. It has already been shown that a single layer of convolutional transform learning yields better results than other single layer representation learning tools, including dictionary learning and transform learning. Therefore



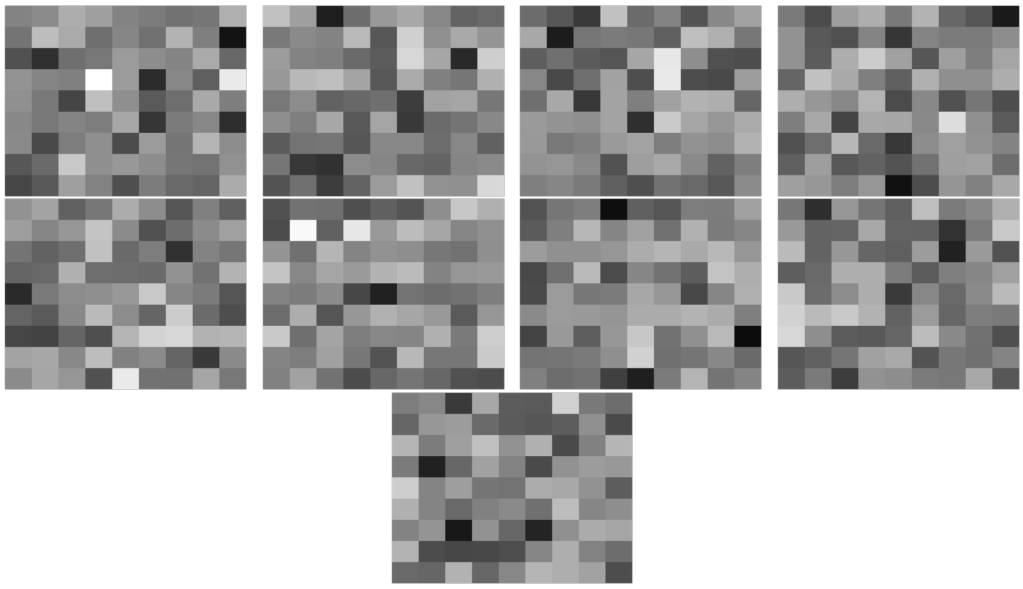
(a) $[M_2 = 3]$



(b) $[M_2 = 5]$



(c) $[M_2 = 7]$



(d) $[M_2 = 9]$

Figure 8.2: Kernels learned on YALE dataset.

Table 8.2: Accuracy on SVM with layers

Dataset	CoTL	DCoTL-2	DCoTL-3	DCoTL-4
YALE 150 × 150	94.00	94.28	**96.00**	92.21
YALE 32 × 32	88.00	89.11	**90.00**	87.73
E-YALE-B	97.38	97.00	**98.00**	94.44
AR-Faces	88.87	92.22	**97.67**	82.21

it is expected that by going deeper, we will improve upon their deeper counterparts. We do not repeat those baseline experiments here. We compare our proposed technique with raw features and CoTL (shallow). We skip comparison with CNN because of its supervised nature, whereas the proposed technique is unsupervised. We take extracted features from the proposed deep convolutional transform learning and perform classification using external classifiers KNN and SVM. The classification accuracy is shown in table 8.3 and table 8.4.

Table 8.3: Classification accuracy using KNN

Dataset	Raw features	CoTL	DCoTL
YALE 150 × 150	78.00	70.00	**80.00**
YALE 32 × 32	**60.00**	58.85	**60.00**
E-YALE-B	71.03	84.00	**85.00**
AR-Faces	55.00	**56.00**	58.00

Table 8.4: Classification accuracy using SVM

Dataset	Raw features	CoTL	DCoTL
YALE 150 × 150	93.00	94.00	**96.00**
YALE 32 × 32	68.00	88.00	**90.00**
E-YALE-B	93.24	97.38	**98.00**
AR-Faces	87.33	88.87	**97.67**

8.4.5 Clustering results with deep convolutional transform learning

Then we perform clustering on the extracted features of DCoTL and report the comparison of ARI in table 8.5. We also report clustering time on extracted features in table 8.6. The time to cluster extracted features from the proposed methodology is comparatively less than others.

Table 8.5: Convolutional transformed clustering: ARI

YALEB/Method	Raw features	DCoTL-2	DCoTL-3
K-means	0.785	0.734	**0.788**
Random	0.733	0.718	**0.738**
PCA-based	0.734	**0.791**	0.777

Table 8.6: Clustering time in sec

YALEB/Method	Raw features	DCoTL-2	DCoTL-3
K-means	2.28	0.45	**0.14**
Random	1.95	0.33	**0.08**
PCA-based	0.36	0.09	**0.03**

8.5 Discussion

This work proposes a new representation learning technique called convolutional transform learning. Here, we learn a set of independent convolutional filters that operate on the images to produce representations (one corresponding to each filter). The significant advantage of our proposed approach is that it is entirely unsupervised; unlike CNN, where labelled images are required for training. Moreover, it relies on a well-sounded minimization technique with established convergence guarantees. We have compared the proposed method with dictionary learning and transform

learning on standard image classification datasets. Results show that our approach improves over the rest by a considerable margin.

Chapter 9

Semi-coupled transform learning

In this chapter, semi-coupled transform learning is proposed. Given training data in two domains (source and target), it learns a transform in each of the domains such that the corresponding coefficients are (linearly) mapped from the source to the target. Since the mapping is in one direction (source to target) but not the other way round, we call it semi-coupled. Our work is the analysis equivalent of (semi) coupled dictionary learning. The proposed technique has been applied to two problems: the first being image super-resolution and the second, cross-lingual document retrieval.

The chapter is organized as follows. The chapter starts with an introduction to semi-coupled transform learning (SCTL) in section 9.1 followed by section 9.2, which discusses the existing work. The proposed algorithm is presented in section 9.3. The experiments are performed on standard datasets against the state-of-the-art solution for each problem. Experiments and results are discussed in section 9.4.

9.1 Introduction

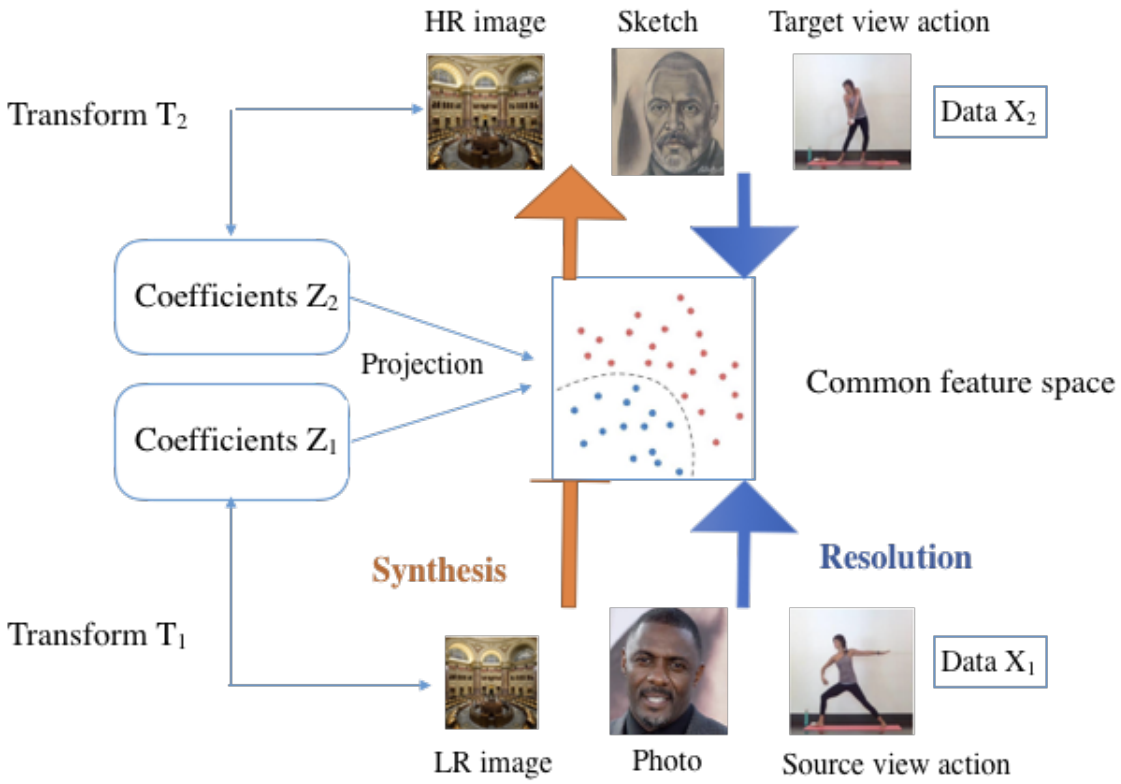


Figure 9.1: Semi-coupled TL

There are many problems in image processing and computer vision, which can be recast in the framework for transfer learning. For example, consider the example of a single image super-resolution; the objective is to create a high-resolution image from a low-resolution one. There are many signal processing (sparsity) based techniques to solve this problem [138, 139]. However, in recent times, dictionary learning-based approaches are preferred owing to their improved performance. For each of the domains (high resolution and low resolution), two dictionaries are learned, such that the coefficients of low-resolution dictionary can be linearly mapped onto the high-resolution dictionary [140, 141, 10, 142]. After the training phase,

when a new low-resolution image is an input, it learns the corresponding coefficients from the learned dictionary; the coefficients, in turn, are mapped to that of the high-resolution version by the learned linear map. From the thus formed high-resolution coefficients, the corresponding high-resolution image is synthesized. The transform learning equivalent is shown in figure 9.1. This formulation falls under the purview of CDL. Similar approaches have been applied to other problems, e.g. photo sketch synthesis [141, 10], RGB to Depth image matching [143], pose varying face matching [144] and visible (VIS) to near infra-red (NIR) face matching [144]. In photo sketch matching the problem is to match a person′s sketch to that of a digital photo: this is usually used in law enforcement. Similar problems exist in the other domains as well: e.g. matching between visible image collected during daylight and that of NIR image collected during the night [9]. Although one finds most applications of coupled dictionary learning in vision problems, it has been used in computational linguistics as well [10]. There in the problem is cross-lingual document retrieval. The query is in one language (source), and the problem is to find the documents from the other (target) language.

9.2 Literature review

9.2.1 Coupled dictionary learning

The idea of CDL was proposed in [140, 141, 10, 142, 143, 144]. Let there be two domains: 1 and 2. X_1 and X_2 are the training data for the two domains. CDL trains two dictionaries D_1 and D_2 (along with their coefficients Z1and Z2) and linear coupling maps from domain1 to 2: M_{12} and from 2 to 1: M_{21}. Mathematically this is expressed as,

$$\begin{aligned} &\min_{D_1,D_2,Z_1,Z_2,M_{12},M_{21}} ||X_1 - D_1Z_1||_F^2 + ||X_2 - D_2Z_2||_F^2 \\ &+\mu(||Z_2 - M_{21}Z_1||_F^2 + ||Z_1 - M_{12}Z_2||_F^2) + \eta(||Z_1||_0 + ||Z_2||_0) \end{aligned} \tag{9.1}$$

Here we have abused the notations slightly; the l0-norm is defined on the vectorized version of the $Z's$. Solving equation 9.1 may apparently be a daunting task. However, when segregated into separate sub-problems, they have well-known solutions. During testing, say the signal is available in domain 1, and the corresponding signal in domain 2 needs to be generated; such problems can arise in photo sketch synthesis and image super-resolution. The learned dictionary in domain 1 is used to generate the coefficients.

$$\min_{Z_1^{test}} ||X_1^{test} - D_1Z_1^{test}||_F^2 + \eta||Z_1^{test}||_1 \tag{9.2}$$

The generated coefficients of domain 1 are now transformed to domain 2 by the learnt linear map: $\hat{Z}_2^{test} = M_{21}Z_1^{test}$. For classification problems usually

a classifier is trained with the coefficients Z_1 and Z_2. During testing (e.g. RGB to NIR matching), $\hat{Z}_2^{test}$ is run through the classifier for domain 2. For synthesis problems (e.g. super-resolution), the high resolution image is synthesized by $\hat{x}_2^{test} = D_2\hat{z}_2^{test}$.

9.3 Proposed formulation

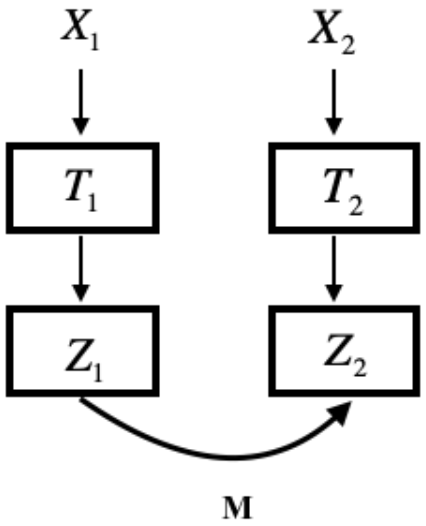


Figure 9.2: Semi-coupled TL: architectural diagram

This work proposes a semi-coupled formulation; we learn a single directional map (from source to target). Suppose there are domains: 1 and 2. Say X_1 and X_2 are the corresponding training data. Semi-coupled analysis sparse coding learns two transforms T_1 and T_2 (one for each domain) and their corresponding features Z_1 and Z_2 so that the features from one of the domains can be linearly mapped M into the other as shown in figure 9.2 and its neural network interpretation is given in figure 9.3. For example, in photo sketch identification, one needs to find the digital photograph

from a photo sketch: not the other way round. If there is a need for bi-directionality, e.g. in RGB and NIR matching, we can always learn two semi-coupled transforms from one domain to the other as shown in figure 9.1.

Mathematically our formulation is expressed as,

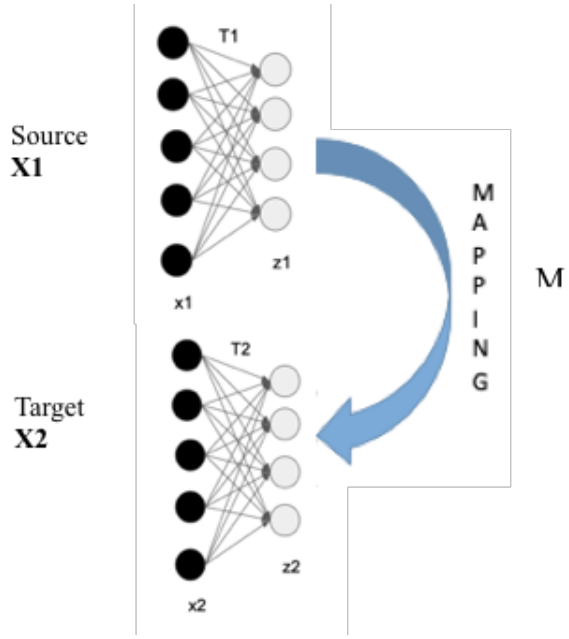


Figure 9.3: Semi-coupled TL: neural network interpretation

$$\min_{T_1,T_2,Z_1,Z_2,M} ||T_1X_1 - Z_1||_F^2 + ||T_2X_2 - Z_2||_F^2 + \mu||Z_2 - MZ_1||_F^2$$
$$+\eta(||Z_1||_1 + ||Z_2||_1) + \lambda(\varepsilon||T_1||_F^2 + \varepsilon||T_2||_F^2 - \log\det T_1 - \log\det T_2) \tag{9.3}$$

The alternating minimization approach is used for solving equation 9.4. It can be segregated into the following sub-problems.

- S_1 :

$$\min_{T_1} ||T_1X_1 - Z_1||_F^2 + \lambda(\varepsilon||T_1||_F^2 - \log\det T_1)$$

- S_2 :

$$\min_{T_2} ||T_2X_2 - Z_2||_F^2 + \lambda(\varepsilon||T_2||_F^2 - \log\det T_2)$$

- S_3 :

$$\min_{Z_1} ||T_1X_1 - Z_1||_F^2 + \mu||Z_2 - MZ_1||_F^2 + \eta||Z_1||_1$$
$$\equiv \min_{Z_1} || \begin{pmatrix} T_1X_1 \\ \sqrt{\mu}Z_2 \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu}M \end{pmatrix} Z_1||_F^2 + \eta||Z_1||_1$$

- S_4 :

$$\min_{Z_2} ||T_2X_2 - Z_2||_F^2 + \mu||Z_2 - MZ_1||_F^2 + \eta||Z_2||_1$$
$$\equiv \min_{Z_2} || \begin{pmatrix} T_2X_2 \\ \sqrt{\mu}MZ_1 \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu}I \end{pmatrix} Z_2||_F^2 + \eta||Z_2||_1$$

- S_5 :

$$\min_{M} ||Z_2 - MZ_1||_F^2$$

Sub-problems S_1 and S_2 are standard transform updates. We already know how to update them. Sub-problem S_3 and S_4 are regular updates for sparse transform coefficients; they just require one step of soft thresholding. Updating the map is easy since S_5 is a simple least square problem. This concludes the training phase.

Let us now consider fully CTL given as:

$$\begin{aligned} &\min_{T_1,T_2,Z_1,Z_2,M_{12},M_{21}} ||T_1X_1 - Z_1||_F^2 + ||T_2X_2 - Z_2||_F^2 \\ &+\mu||Z_2 - M_{12}Z_1||_F^2 + ||Z_1 - M_{21}Z_2||_F^2 \\ &+\eta(||Z_1||_1 + ||Z_2||_1) + \lambda(\varepsilon||T_1||_F^2 + \varepsilon||T_2||_F^2 - \log\det T_1 - \log\det T_2) \end{aligned} \tag{9.4}$$

In fully CTL, one would have to learn another linear map from domain 2 to domain 1. As mentioned at the onset, this is not required in most cases. Even if it is necessary, we can learn another semi-coupled transform from 2 to 1. But learning a fully coupled transform in a single problem means one more variable (linear map) to solve. Given the limited training data, solving more variables/parameters would lead to over-fitting. Hence we consciously avoid such a formulation. During testing, it can apply to two kinds of problems. In the first, one can carry out analysis in the feature domain. For such, the coefficients in domain 2 are used to learn a classifier. During testing the sample is given in domain 1. From which the corresponding feature is generated by sparse coding:

$$z_1^{test} \leftarrow signum(T_1 x_1^{test}) \cdot \max(0, abs(T_1 x_1^{test}) - \mu) \tag{9.5}$$

From the features of domain 1, the target domain features are generated by $\hat{z}_2^{test} = M z_1^{test}$. These features are input to the learnt classifier for final results. There can be a second possibility, where the analysis is carried out not on the transform features (z-domain), but on the samples itself (x-domain). In such a case, instead of stopping at $\hat{z}_2^{test}$, one needs to synthesize the corresponding sample. This is done by solving the inverse problem $T_2 \hat{x}_2^{test} = \hat{z}_2^{test}$. Once $\hat{x}_2^{test}$, one can carry out further analysis in the sample domain.

Figure 9.4: Original(left), CDL(mid), proposed(right)

9.4 Experiments and results

9.4.1 Image super-resolution

For image super-resolution, we train on the CIFAR-100 dataset. These are 32 × 32 images (HR: high resolution). Our interest is in 4 (2× 2)-fold super-resolution. During training, we blur and down-sample the CIFAR images to 16 × 16 (LR: low resolution). We follow a patch-based technique. The LR images from the source and the HR the corresponding targets. From the LR images, we extract 8 × 8 patches and their corresponding 16 × 16 patches from the target HR images. On this, our proposed SCTL formulation is run. The training is carried out on the 50K training images of CIFAR-100. The remaining 10K test images are used for validation. The

tuned parameter values we obtained are λ=0.1, ϵ=1, μ=0.5 and γ=0.05.

Table 9.1: PSNR for super-resolution

Image name		**Lena**	**Barbara**	**Pepper**	**Cameraman**
Color	**CDL**[145]	30.79	28.21	29.76	27.86
	Proposed	33.03	30.28	31.81	30.14
Gray scale	**CDL**[145]	31.27	28.98	30.46	28.70
	Proposed	34.55	31.17	32.68	30.85

9.4.2 Cross lingual document retrieval

In this work, we follow the exact evaluation protocol outlined in [146]. We test all algorithms on the Europarl data set of documents in English and Spanish, and a set of Wikipedia articles in English and Spanish that contain interlanguage links between them (i.e., articles that the Wikipedia community have identified as comparable across languages). For the Europarl data set, we use 52,685 documents as training, 11,933 documents as a development set, and 18,415 documents as a final test set. Documents are defined as speeches by a single speaker, as in [147]. For the Wikipedia set, we use 43,380 training documents, 8,675 development documents, and 8,675 final test. For both corpora, the terms are extracted by word breaking all documents, removing the top 50 most frequent phrases and keeping the next 20,000 most frequent terms. No stemming or folding is applied. We assess performance by testing each document in English against all possible documents in Spanish, and vice versa. We measure the top-1 accuracy (i.e., whether the true comparison is the closest in the test set), and the mean reciprocal rank (MRR) of the true comparable, and

report the average performance over the two retrieval directions. Ties are counted as errors. We have compared our method against oriented principal component analysis (OPCA) and coupled probabilistic latent semantic analysis (CPLSA)): two best-performing methods proposed in [146]; and CDL [145]. The dimensions for the projections are given in the respective papers. For our problem, λ=0.1, ϵ=1, μ=1 and η=0.05 are used for both semi coupled analysis sparse coding and symmetrically coupled analysis sparse coding. The number of projections used is 300. The final results are shown in Tables 9.2 and 9.3.

For Wikipedia experiments, we use the unpaired t-test with Bonferroni correction to determine the smallest set of algorithms that have statistically significantly better accuracy than the rest. The p-value threshold for significance is chosen to be 0.05.

Table 9.2: Comparable document retrieval on Europarl

Algorithm	**Accuracy**	**MRR**
OPCA[146]	97.42	0.9846
CPLSA[146]	97.16	0.9782
CDL[145]	98.12	0.9839
Proposed	**99.54**	**0.9896**

Table 9.3: Comparable document retrieval on Wikipedia

Algorithm	**Accuracy**	**MRR**
OPCA[146]	72.55	0.7734
CPLSA[146]	45.79	0.5130
CDL[145]	72.79	0.7742
Proposed	**78.68**	**0.8002**

9.5 Discussion

For image super-resolution, we have compared our method with the CDL formulation [10]. The authors of [10] have compared with other super-resolution techniques [139, 141] and have shown to supersede them; they also improve the baseline bicubic interpolation technique. Therefore it is enough to show that our proposed technique SCTL yields better results than CDL [10]. We have carried out experiments both on greyscale and RGB images. For RGB, the image was converted to YCbCr space. The super-resolution technique was only applied to the illuminance channel. For the others, simple bicubic interpolation is done. The results are shown in figure 9.4. If one concentrates on the sharp edges (for example Lena′s nose), one can see that CDL images are blurred compared to our proposed method.

The PSNR values are shown in Table 9.1. The results establish the superiority of our proposed method over CDL [10] and hence over [139, 141] (since [10] showed improvement upon them). In all cases we improve upon the state-of-the-art by more than 2dB: this is a significantly large improvement. To put it in context,[10] improves upon the prior works [139, 141] by 1 to 1.5 dB. Here we improve upon the best known [10] by more than 2 dB in every case. One notices that the performance (of both algorithms) for the grayscale image is always better than the colour counterpart. This is because, in colour imaging, only the illuminance channel is properly

super-resolved; the other channels are simply extrapolated using bicubic interpolation. In cross-lingual document retrieval experiments, for the Europarl, there is no statistically significant difference between OPCA and CPLSA. CDL is significantly better than them. Our proposed techniques are even better than CDL. There is no statistically significant difference between our two algorithms. For the Wikipedia dataset, OPCA and CDL are statistically similar; both of them are significantly better than CPLSA. Our proposed coupled analysis sparse coding techniques show significant improvement over OPCA and CDL. Even for this dataset, there is no statistically significant difference between semi-coupled analysis sparse coding and the symmetrically coupled counterpart.

Chapter 10

Conclusions

This thesis focuses on developing a new framework for deep learning called deep transform learning. By developing this new framework, we try to solve machine learning problems using transform learning. Currently, Transform learning has not been used outside the signal processing community. We work on solving problems like classification, clustering and inverse problems using transform learning.

The rest of the chapter is organized as follows. Section 10.1 presents chapter wise contribution of the thesis. In section 10.2, future work is discussed.

10.1 Summary of contribution

10.1.1 Supervised transform learning

This work introduces certain supervised formulations to transform learning. Four different types of supervision penalties are proposed. The first one is class-sparsity, which imposes common sparse support within representations of each class. The second one imposes similarity among intra-class features in terms of a low-rank constraint (high cosine similarity). The third penalty enforces features of the same class to be nearby each other and features of different classes to be far apart. The final formulation is the well known label-consistency formulation, which learns a linear map from the feature space to the class targets. For the first time, we show how transform learning (and its supervised versions can be kernelized). Finally, this work also introduces stochastic regularization techniques like drop out and drop connect into the transform learning formulation. Experiments have been carried out on four different problems: computer vision, bioinformatics, hyperspectral imaging, and ECG based arrhythmia classification. In such a diverse variety of problems, our method excels over all existing ones.

10.1.2 Deep transform learning - classification and clustering problem

DTL is a deeper version of transform learning. DTL is formed by stacking multiple layers of transforms one after the other. The first approach to solve DTL is greedy, where one layer is learned at a time. In the greedy

approach, the input to the first layer is the training data. Then the output coefficients learned in the previous layer forms the input to the next layer. The problem with this approach is that the data flows only in the forward direction. The deeper layers do not influence the learning of shallower layers. Hence no feedback. Then, we are addressing this shortcoming by proposing a solution where all the transforms and coefficients are learned simultaneously in a single optimization problem. This is the unsupervised method, and any classifier can be used to make predictions.

Experiments are performed on standard classification and clustering datasets. It is seen that in both the cases proposed approach performed better than the state-of-art methods.

10.1.3 Deep transform learning - inverse problem

This chapter addresses the problem of solving a linear inverse problem. Conventional inversion techniques are transductive in nature; solving an inverse problem with only some prior knowledge about the solution. The advent of deep learning led the way for inductive (trained) inversion techniques. The main issue with inductive inversion is that unless the unseen signal (to be inverted) is of similar nature as the training data, the learned model fails to generalize rendering poor inversion results. A recent study on deep dictionary learning has shown how it can combine the best of both worlds: deep learning with transductive inversion. In this work, we show how the analysis counterpart of dictionary learning, called transform

learning, can be extended deeper for transductive inversion. Results on two standard inversion problems: deblurring and reconstruction, show that the proposed techniques excel over the state-of-the-art.

10.1.4 Supervised deep transform learning

This work proposes to incorporate supervision into the deep transform learning framework. Supervision is introduced by adding a label consistency penalty to the previous unsupervised formulation. The derivation for solving the ensuing formulation is based on the state-of-the-art optimization paradigm that includes proximal variable splitting, augmented lagrangians and alternating direction method of multipliers. This formulation is flexible enough to handle a single label and multilabel classification problems. The experiments have been carried out for the multilabel problem. The proposed approach is applied to a real-time problem of energy disaggregation. The problem is modelled as a -multilabel classification problem, where the task is to predict the appliance's state (ON/OFF). The experiments are performed on REDD, and Pecan Street datasets and the proposed techniques excel over the state-of-the-art.

10.1.5 Deep transformed subspace clustering

This work incorporates the simplest subspace clustering formulation: locally linear manifold clustering, into the transform learning formulation. The core idea is to perform the clustering task in a transformed domain

instead of processing the raw samples directly. The transform analysis step and the clustering are not done piecemeal but are performed jointly through the formulation of a coupled minimization problem. Then, we embed subspace clustering techniques (locally linear manifold clustering, sparse subspace clustering, and low-rank representation) into deep transform learning. The entire formulation is jointly learned; giving rise to a new class of methods called deeply transformed subspace clustering. To test the performance of the proposed techniques, benchmarking is performed on image clustering problems. Comparison with state-of-the-art clustering techniques shows that our formulation improves upon them.

10.1.6 Convolutional transform learning

This work proposes a new representation learning technique called convolutional transform learning. In standard transform learning, a dense basis is learned that analyses the image to generate the representation from the image. Here, we learn a set of independent convolutional filters that operate on the images to produce representations (one corresponding to each filter). The major advantage of our proposed approach is that it is completely unsupervised; unlike CNN where labelled images are required for training. Moreover, it relies on a well-sounded minimization technique with established convergence guarantees. We have compared the proposed method with dictionary learning and transform learning on standard image classification datasets. Results show that our method improves over the

rest by a considerable margin.

10.1.7 Semi-coupled transform learning

This work introduces semi-coupled transform learning. Given training data in two domains (source and target), it learns a transform in each of the domains such that the corresponding coefficients are (linearly) mapped from the source to the target. Since the mapping is in one direction (source to target) but not the other way round, we call it semi-coupled. Our work is the analysis equivalent of (semi) coupled dictionary learning. We have showcased our results for two tasks. The first one is a synthesis problem where the task is to super-resolve from a low-resolution image. Previously coupled dictionary-based techniques have shown significant success in this problem. Our proposed transform learning-based formulation improves upon the state-of-the-art. The second problem is an analysis problem where the task is cross-lingual document retrieval. In this task, we have shown that the proposed method surpasses the previous state-of-the-art.

10.2 Future work

10.2.1 Coupled deep transform learning

Consider a problem of cross-style image synthesis, such as image super-resolution, artistic rendering, photo-sketch synthesis, and multi-modal biometrics, etc., Where we need to convert an image in one style into another

style for better visualization, interpretation, and recognition. For examples, up-convert a low-resolution image to a high resolution one, and convert a face sketch into a photo for matching, etc. Using coupled deep transform learning, a pair of transforms and a mapping function will be simultaneously learned. The transform pair can well characterize the structural domains of the two styles of images, while the mapping function can reveal the intrinsic relationship between the two styles domains. We propose to learn a deep network figure 10.1 after the success of a shallow semi-coupled transform learning network.

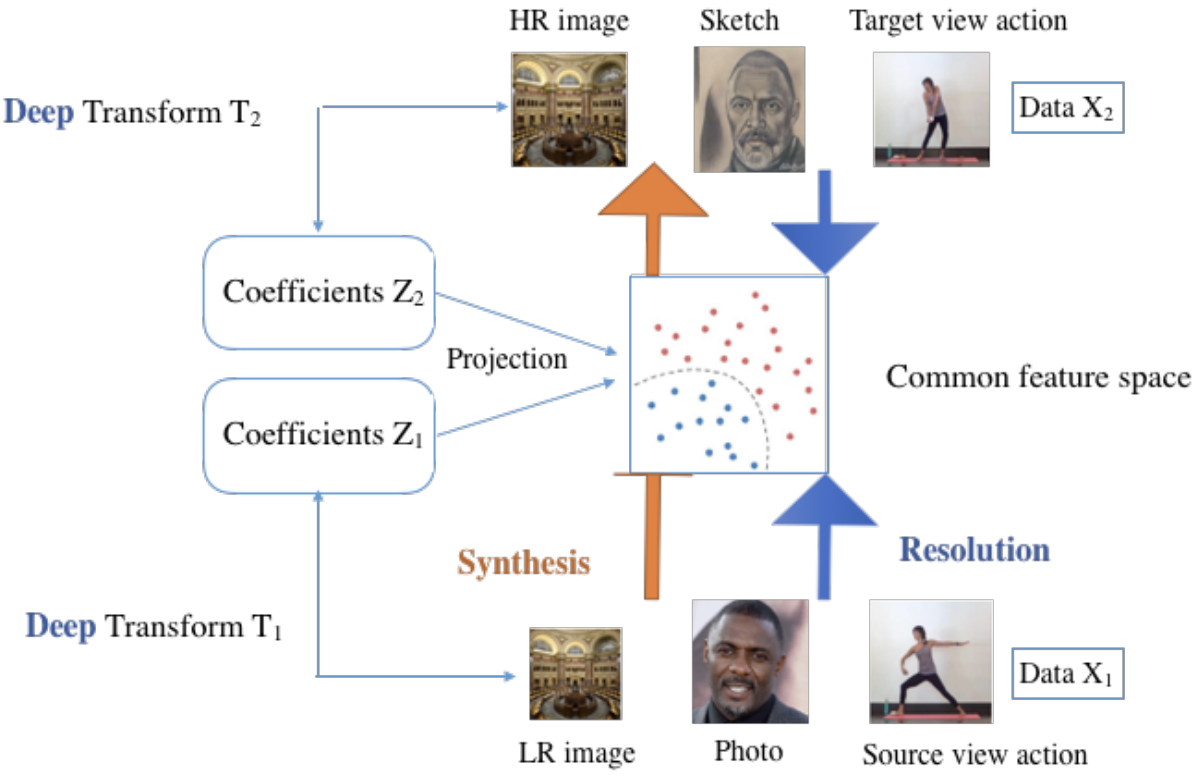
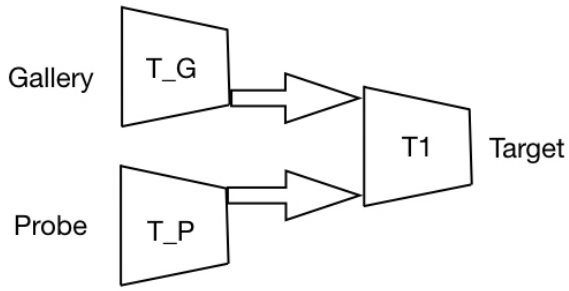


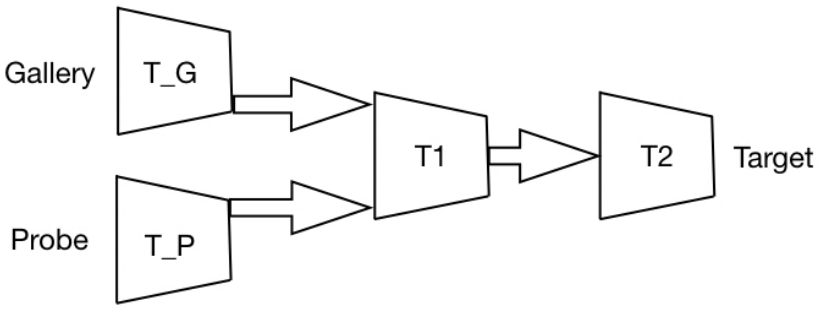
Figure 10.1: Deep coupled transform learning

10.2.2 Deep transform information fusion network

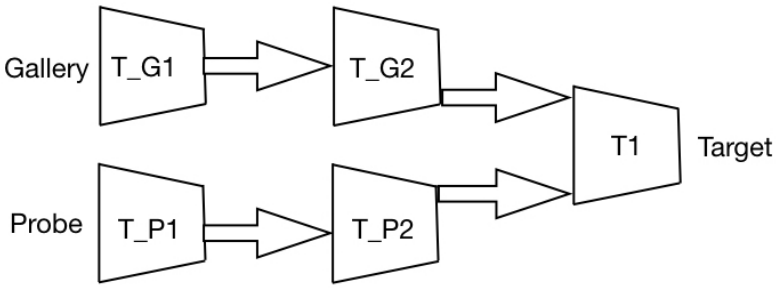
Considering the problem of disguise detection, there are two sets of images: gallery and probe. Gallery of images consisting of the genuine people, and the probe is an image - can be either an imposter or genuine. The task is to identify if the probe is genuine or imposter. In figure 10.2, we show some examples of the deep transform information fusion network for disguise detection. In figure 10.2a, we have one level of transform for processing the individual probe and gallery image and another level for fusing the two. In figure 10.2b, a single layer of transform processes the probe and gallery image, and two layers for fusion. In the final architecture figure 10.2c, there are two layers for processing the probes and gallery images and one layer for fusion. We propose to make a deep transform information fusion network architecture with the objective of classifying two inputs as a match or mismatch (binary classification).



(a) Architecture 1



(b) Architecture 2



(c) Architecture 3

Figure 10.2: Information fusion deep transform network

Bibliography

[1] A. Sobral, "Robust low-rank and sparse decomposition for moving object detection: From matrices to tensors," Ph.D. dissertation, 05 2017.

[2] S. Ravishankar and Y. Bresler, "Sparsifying transform learning for compressed sensing mri," in *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE, 2013, pp. 17–20.

[3] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning - Part I," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 625–636, 2015.

[4] S. Ravishankar and Y. Bresler, "Online sparsifying transform learning - Part II," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 637–646, 2015.

[5] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, 2009, pp. 1601–1608.

[6] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[7] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[8] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning," *CoRR*, vol. abs/1709.02893, 2017. [Online]. Available: http://arxiv.org/abs/1709.02893

[9] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2216–2223.

[10] D.-A. Huang and Y.-C. Frank Wang, "Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2496–2503.

[11] S. Xiang, G. Meng, Y. Wang, C. Pan, and C. Zhang, "Image deblurring with coupled dictionary learning," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 248–271, 2015.

[12] P. Dou, Y. Wu, S. K. Shah, and I. A. Kakadiaris, "Monocular 3d facial shape reconstruction from a single 2d image with coupled-dictionary learning and sparse coding," *Pattern Recognition*, vol. 81, pp. 515–527, 2018.

[13] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, Dec 2008. [Online]. Available: https://doi.org/10.1007/s00041-008-9035-z

[14] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[15] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1697–1704. [Online]. Available: https://doi.org/10.1109/CVPR.2011.5995354

[16] Z. Jiang, Z. Lin, and L. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2651–2664, 11 2013.

[17] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 2021–2024.

[18] A. Golts and M. Elad, "Linearized kernel dictionary learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 726–739, 2016.

[19] A. Shrivastava, V. M. Patel, and R. Chellappa, "Non-linear dictionary learning with partially labeled data," *Pattern Recognition*, vol. 48, no. 11, pp. 3283–3292, 2015.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, 2013, pp. 1058–1066.

[22] A. Sankaran, M. Vatsa, R. Singh, and A. Majumdar, "Group sparse autoencoder," *Image and Vision Computing*, vol. 60, pp. 64–74, 2017.

[23] A. Majumdar, R. Singh, and M. Vatsa, "Face verification via class sparsity based supervised encoding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1273–1280, 2016.

[24] A. Sankaran, G. Goswami, M. Vatsa, R. Singh, and A. Majumdar, "Class sparsity signature based restricted boltzmann machine," *Pattern Recognition*, vol. 61, pp. 674–685, 2017.

[25] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint 2, 1-norms minimization," in *Advances in neural information processing systems*, 2010, pp. 1813–1821.

[26] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[27] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 2691–2698.

[28] I. Kviatkovsky, M. Gabel, E. Rivlin, and I. Shimshoni, "On the equivalence of the lc-ksvd and the d-ksvd algorithms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 2, pp. 411–416, 2016.

[29] A. Gogna, A. Majumdar, and R. Ward, "Semi-supervised stacked label consistent autoencoder for reconstruction and analysis of biomedical signals," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2196–2205, 2017.

[30] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on signal processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[31] D. L. Donoho, "De-noising by soft-thresholding," *IEEE transactions on information theory*, vol. 41, no. 3, pp. 613–627, 1995.

[32] Y. B. P. Vincent, "Kernel matching pursuit," 2002.

[33] N. Akhtar, F. Shafait, and A. Mian, "Discriminative bayesian dictionary learning for classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 12, pp. 2374–2388, Dec 2016.

[34] N. M. R. A. J. W. K. Bahrampour, Soheil; Nasrabadi, "Multimodal task-driven dictionary learning for image classification," 2016.

[35] P. Zhu, Q. Hu, C. Zhang, and W. Zuo, "Coupled dictionary learning for unsupervised feature selection," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 2422–2428. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016237

[36] Y. Chen and J. Su, "Sparse embedded dictionary learning on face recognition," *Pattern Recogn.*, vol. 64, no. C, pp. 51–59, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.patcog.2016.11.001

[37] J. Hu and Y.-P. Tan, "Nonlinear dictionary learning with application to image classification," *Pattern Recogn.*, vol. 75, no. C, pp. 282–291, Mar. 2018. [Online]. Available: https://doi.org/10.1016/j.patcog.2017.02.009

[38] V. Singhal and A. Majumdar, "Supervised deep dictionary learning for single label and multi-label classification," *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2018.

[39] M. Mehdipour Ghazi and H. Kemal Ekenel, "A comprehensive analysis of deep learning based representation for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 34–41.

[40] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[41] Y. Wang, T. Liu, D. Xu, H. Shi, C. Zhang, Y.-Y. Mo, and Z. Wang, "Predicting dna methylation state of cpg dinucleotide using genome topological features and deep networks," *Scientific reports*, vol. 6, p. 19598, 2016.

[42] E. Yaffe and A. Tanay, "Probabilistic modeling of hi-c contact maps eliminates systematic biases to characterize global chromosomal architecture," *Nature genetics*, vol. 43, no. 11, p. 1059, 2011.

[43] R. Singh, J. Lanchantin, G. Robins, and Y. Qi, "Deepchrome: deep-learning for predicting gene expression from histone modifications," *Bioinformatics*, vol. 32, no. 17, pp. i639–i648, 2016.

[44] A. Kundaje, W. Meuleman, J. Ernst, M. Bilenky, A. Yen, A. Heravi-Moussavi, P. Kheradpour, Z. Zhang, J. Wang, M. J. Ziller *et al.*, "Integrative analysis of 111 reference human epigenomes," *Nature*, vol. 518, no. 7539, p. 317, 2015.

[45] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, pp. 849–856.

[46] Y. Zhou and Y. Wei, "Learning hierarchical spectral–spatial features for hyperspectral image classification," *IEEE transactions on cybernetics*, vol. 46, no. 7, pp. 1667–1678, 2015.

[47] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016.

[48] Z. He, L. Liu, S. Zhou, and Y. Shen, "Learning group-based sparse and low-rank representation for hyperspectral image classification," *Pattern Recognition*, vol. 60, pp. 1041–1056, 2016.

[49] A. Majumdar and R. K. Ward, "Robust classifiers for data reduced via random projections," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 5, pp. 1359–1371, 2010.

[50] E. J. D. S. Luz, T. M. Nunes, V. H. C. De Albuquerque, J. P. Papa, and D. Menotti, "Ecg arrhythmia classification based on optimum-path forest," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3561–3573, 2013.

[51] M. M. Al Rahhal, Y. Bazi, H. AlHichri, N. Alajlan, F. Melgani, and R. R. Yager, "Deep learning approach for active classification of elec-

trocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, 2016.

[52] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2015.

[53] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[56] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[57] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[58] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, 2005.

[59] M. M. and B. R., "The ar face database," CVC 24, Tech. Rep., Jun. 1998.

[60] A. Sankaran, M. Vatsa, R. Singh, and A. Majumdar, "Group sparse autoencoder," *Image and Vision Computing*, vol. 60, pp. 64–74, 2017.

[61] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.

[62] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised autoencoders," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2108–2118, Oct 2015.

[63] X. Peng, J. Feng, S. Xiao, J. Lu, Z. Yi, and S. Yan, "Deep sparse subspace clustering," *arXiv preprint arXiv:1709.08374*, 2017.

[64] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-20)," CUCS-005-96, Tech. Rep., Feb. 1996.

[65] Yale, "The extended yale face database b," 2001. [Online]. Available: http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html

[66] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[67] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[68] D. L. Donoho *et al.*, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[69] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse problems*, vol. 23, no. 3, p. 947, 2007.

[70] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[71] J. Lewis, V. Singhal, and A. Majumdar, "Solving inverse problems in imaging via deep dictionary learning," *IEEE Access*, vol. 7, pp. 37 039–37 049, 2018.

[72] A. Majumdar, "Blind denoising autoencoder," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 1, pp. 312–317, 2018.

[73] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep relu neural networks," *Neural Networks*, vol. 108, pp. 296–330, 2018.

[74] D. Yarotsky, "Error bounds for approximations with deep relu networks," *Neural Networks*, vol. 94, pp. 103–114, 2017.

[75] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2014.

[76] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[77] Y. Bai, G. Cheung, X. Liu, and W. Gao, "Graph-based blind image deblurring from a single photograph," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1404–1418, 2018.

[78] M. Tofighi, Y. Li, and V. Monga, "Blind image deblurring using row–column sparse representations," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 273–277, 2017.

[79] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8183–8192.

[80] S. Ravishankar, B. E. Moore, R. R. Nadakuditi, and J. A. Fessler, "Low-rank and adaptive sparse signal (lassi) models for highly ac-

celerated dynamic imaging," *IEEE transactions on medical imaging*, vol. 36, no. 5, pp. 1116–1128, 2017.

[81] U. Nakarmi, Y. Wang, J. Lyu, D. Liang, and L. Ying, "A kernel-based low-rank (klr) model for low-dimensional manifold recovery in highly accelerated dynamic mri," *IEEE transactions on medical imaging*, vol. 36, no. 11, pp. 2297–2307, 2017.

[82] C. Qin, J. Schlemper, J. Caballero, A. N. Price, J. V. Hajnal, and D. Rueckert, "Convolutional recurrent neural networks for dynamic mr image reconstruction," *IEEE transactions on medical imaging*, vol. 38, no. 1, pp. 280–290, 2018.

[83] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, 2017.

[84] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 2018.

[85] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[86] Y. Yankelevsky and M. Elad, "Graph-constrained supervised dictionary learning for multi-label classification," in *Science of Electrical*

Engineering (ICSEE), IEEE International Conference on the. IEEE, 2016, pp. 1–5.

[87] J. M. Bioucas-Dias, "A variable splitting augmented lagrangian approach to linear spectral unmixing," in *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2009. WHISPERS'09. First Workshop on.* IEEE, 2009, pp. 1–4.

[88] A. Martinez and R. Benavente, "The ar face database. cvc technical report 24," Tech. Rep., june 1998.

[89] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[90] A. Gogna, A. Majumdar, and R. Ward, "Semi-supervised stacked label consistent autoencoder for reconstruction and analysis of biomedical signals," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 9, pp. 2196–2205, 2017.

[91] Y. Liu, S. Zhou, and Q. Chen, "Discriminative deep belief networks for visual data classification," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2287–2296, 2011.

[92] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *Computer*

Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011, pp. 1697–1704.

[93] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, "Deep dictionary learning," *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.

[94] V. Singhal and A. Majumdar, "Supervised deep dictionary learning for single label and multi-label classification," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–7.

[95] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[96] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[97] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *Granular Computing, 2005 IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 718–721.

[98] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, vol. 25, no. Citeseer. Citeseer, 2011, pp. 59–62.

[99] O. Parson, G. Fisher, A. Hersey, N. Batra, J. Kelly, A. Singh, W. Knottenbelt, and A. Rogers, "Dataport and nilmtk: A building

data set designed for non-intrusive load monitoring," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2015, pp. 210–214.

[100] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Artificial Intelligence and Statistics*, 2012, pp. 1472–1482.

[101] K. Basu, V. Debusschere, S. Bacha, U. Maulik, and S. Bondyopadhyay, "Nonintrusive load monitoring: A temporal multilabel classification approach," *IEEE Transactions on industrial informatics*, vol. 11, no. 1, pp. 262–270, 2015.

[102] S. M. Tabatabaei, S. Dick, and W. Xu, "Toward non-intrusive load monitoring via multi-label classification," *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 26–40, 2017.

[103] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: http://www.jstor.org/stable/2346830

[104] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 551–556. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014118

[105] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, pp. 849–856. [Online]. Available: http://dl.acm.org/citation.cfm?id=2980539.2980649

[106] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, March 2011.

[107] V. M. Patel, H. Van Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 225–232.

[108] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[109] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

[110] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *CoRR*, vol. abs/1010.2955, 2010. [Online]. Available: http://arxiv.org/abs/1010.2955

[111] Y. Chen, G. Li, and Y. Gu, "Active orthogonal matching pursuit for sparse subspace clustering," *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 164–168, 2017.

[112] R. Vidal, "Subspace clustering," vol. 28, pp. 52 – 68, 04 2011.

[113] J. F. W. Y. Y. X. Peng, S. Xiao and Z. Yi, "Deep sub-space clustering with sparsity prior." IJCAI, 2016, pp. 1925–1931.

[114] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3861–3870.

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

[116] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang, "Learning a task-specific deep architecture for clustering," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 369–377.

[117] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[118] S. Ravishankar and Y. Bresler, "Closed-form solutions within sparsifying transform learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 5378–5382.

[119] J. Zeng, S. Lin, and Z. Xu, "Sparse solution of underdetermined linear equations via adaptively iterative thresholding," *Signal Processing*, vol. 97, pp. 152–161, 2014.

[120] X. Lin and G. Wei, "Accelerated reweighted nuclear norm minimization algorithm for low rank matrix recovery," *Signal Processing*, vol. 114, pp. 24–33, 2015.

[121] X. Li and Z. Tian, "Optimum cut-based clustering," *Signal Processing*, vol. 87, no. 11, pp. 2491–2502, 2007.

[122] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 3, pp. 417–429, 2017.

[123] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[124] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[125] F. Huang and A. Anandkumar, "Convolutional dictionary learning through tensor factorization," in *NIPS workshop: Feature Extraction*, Montreal, Canada, Dec. 2015, pp. 116–129.

[126] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning," *Preprint arXiv:1709.02893*, 2017.

[127] V. Papyan, Y. Romano, J. Sulam, and M. Elad, "Convolutional dictionary learning via local processing," *Preprint arXiv:1705.03239*, 2017.

[128] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "A block coordinate variable metric forward-backward algorithm," *J. Global Optim.*, vol. 66, no. 3, pp. 457–485, Nov. 2016.

[129] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Math. Program.*, vol. 146, no. 1-2, pp. 459–494, Aug. 2014.

[130] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods," *Mathematical Programming*, vol. 137, no. 1, pp. 91–129, Feb. 2013.

[131] J. J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bull. Soc. Math. France*, vol. 93, pp. 273–299, 1965.

[132] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems*

in Science and Engineering. New York: Springer-Verlag, 2010, pp. 185–212.

[133] J. Bolte, P. L. Combettes, and J. C. Pesquet, "Alternating proximal algorithm for blind image recovery," in *Proc. of ICIP*, Hong Kong, China, Sep. 2010, pp. 1673–1676.

[134] E. C. A. Benfenati and J.-C. Pesquet, "A proximal approach for a class of matrix optimization problems," Tech. Rep., 2017, http://arxiv.org/abs/1801.07452,.

[135] H. H. Bauschke and P. L. Combettes, "A Dykstra-like algorithm for two monotone operators," *Pac. J. Optim.*, vol. 4, no. 3, pp. 383–391, 2007.

[136] P. N. Bellhumer, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 7, pp. 711–720, 1997.

[137] M. Shekhar, S. Patel and C. R., "Analysis sparse coding models for image-based classification," in *Proc. of ICIP*, Paris, France, Oct. 2014, pp. 5207–5211.

[138] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[139] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[140] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE transactions on image processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[141] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2216–2223.

[142] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2015, pp. 1823–1831. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.212

[143] N. Das, D. Mandal, and S. Biswas, "Simultaneous semi-coupled dictionary learning for matching rgbd data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 175–183.

[144] S. P. Mudunuri and S. Biswas, "A coupled discriminative dictionary and transformation learning approach with applications to cross

domain matching," *Pattern Recognition Letters*, vol. 71, pp. 38–44, 2016.

[145] R. Mehrotra, D. Chu, S. A. Haider, and I. A. Kakadiaris, "Towards learning coupled representations for cross-lingual information retrieval."

[146] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, ser. CoNLL '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 247–256. [Online]. Available: http://dl.acm.org/citation.cfm?id=2018936.2018965

[147] D. Mimno, H. M. Wallach, J. Naradowsky, D. A. Smith, and A. McCallum, "Polylingual topic models," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 880–889.