Building Constraints, Geometric Invariants and Interpretability in Deep Learning: Applications in Computational Imaging and Vision

by

Suhas Anand Lohit

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2019 by the
Graduate Supervisory Committee:

Pavan Turaga, Chair
Andreas Spanias
Baoxin Li
Suren Jayasuriya

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

Over the last decade, deep neural networks also known as deep learning, combined with large databases and specialized hardware for computation, have made major strides in important areas such as computer vision, computational imaging and natural language processing. However, such frameworks currently suffer from some drawbacks. For example, it is generally not clear how the architectures are to be designed for different applications, or how the neural networks behave under different input perturbations and it is not easy to make the internal representations and parameters more interpretable. In this dissertation, I propose building constraints into feature maps, parameters and and design of algorithms involving neural networks for applications in low-level vision problems such as compressive imaging and multi-spectral image fusion, and high-level inference problems including activity and face recognition. Depending on the application, such constraints can be used to design architectures which are invariant/robust to certain nuisance factors, more efficient and, in some cases, more interpretable. Through extensive experiments on real-world datasets, I demonstrate these advantages of the proposed methods over conventional frameworks.

ACKNOWLEDGEMENTS

Prof. Pavan Turaga has been a wonderful advisor and has helped me immensely for the past half a decade. It is impossible to overstate how much I have learned from him. He has taught me the fundamental skills to become a good researcher, one of which is handling rejection with optimism. Equally importantly, he has treated me and all his other students with understanding and respect, has provided opportunities and encouragement, and has created a positive work environment. For this, I will always be grateful to him. It has truly been a joy working with him and in his group, the Geometric Media Lab (GML). This is a part of my life I will always cherish.

I thank Prof. Andreas Spanias, Prof. Baoxin Li and Prof. Suren Jayasuriya for serving on my committee and providing valuable comments which has greatly improved my dissertation. I would like to express my gratitude to Prof. Amit Ashok (UA), Prof. Aswin Sankaranarayanan (CMU), Prof. Matthew Buman (ASU), Prof. Rama Chellappa (UMD) and their students for important collaborative work which have contributed directly to my research and this dissertation. I have learned a great deal during my internships in the industry. I am very grateful to my supervisors there for giving me these invaluable opportunities which expanded my horizons beyond academic research – Dr. Farzin Aghdasi, Dr. Anil Ubale and Dr. Partha Sriram at Nvidia, Dr. Karan Sikka and Dr. Ajay Divakaran at SRI International, and Dr. Dehong Liu, Dr. Hassan Mansour and Dr. Petros Boufounos at Mitsubishi Electric Research Laboratories. I thank my graduate advisors, Toni Mengert (ECEE) and Kayla Elizondo (AME) for making my life a lot easier when it came to all the bureaucratic hoops I had to jump through.

Research reported in this dissertation was partly funded by ARO grant W911NF-17-1-0293, ONR Grant N00014-12-1-0124 sub-award Z868302, and NSF grant 1617999.

I also thank the School of ECEE for providing me with a TA position (2016-18), and to ASU for the University Graduate Fellowship (2015-16).

Two alumni of GML, whom I also consider close friends, Dr. Kuldeep Kulkarni and Dr. Qiao Wang were very helpful and kind to me, and my joint work with them forms an important part of this dissertation. I am especially grateful to them. I also thank all members of GML (past and present) including Dr. Rushil Anirudh, Dr. Vinay Venkataraman, Anirudh Som, Kowshik Thopalli, Rajhans Singh and Hongjun Choi for their support, lively conversations and friendship. I have realized that great people, as much as ideas, are a big reason for me to continue what I am doing.

In my junior year during my undergraduate degree, I got a research opportunity through the Indian Academy of Sciences to work with Prof. Malay K. Kundu for a summer at the Indian Statistical Institute in Kolkata, India. The time I spent there easily qualifies for being described as life-changing for me and I often look back to it fondly. I remember feelings of freedom when I worked there and of joy having found something I loved doing, which I felt could also be my career. I believe Prof. Kundu played an important role in my deciding to pursue a Ph.D., and I thank him.

Once every month for the past six years, I have promptly attended the philosophy reading group meetings organized by Prof. Steven Reynolds (ASU) and open to the public. I have loved every minute of it and learned so much more. I thank Prof. Reynolds for organizing it and all the regular attendees – Dale, Warren, Jean-Claude, Ken, Rod, Joe, Cheak – for making it so welcoming and interesting.

I must thank my Tempe friends – Aaron, Dheeraj, Jiachun, Manoj, Nikhilesh, Niveditha, Raksha, Vaishnavi and Varun – and my long-time friends – Ashwin, Darshan, Gauri, Likhith, Neil, Sanjay, Srinivas and Suneeth – for their support, kindness and all the amazing and meaningful conversations. And to the graduate student family at Matthews Center, thank you for making the last few years there such a

wonderful time – Albert, Alejandra, Brandon, Garrett, GML, Jennifer, John, Joshua, Joshua, Megan, Michael, Piyum, Sreenithy, Yanjun, Yifei and Yue.

I will forever be indebted to the people I love most – my parents, my sister and my grandparents – for their unwavering love and emotional support. This dissertation is dedicated to them. My father is one of the most hard working people I know and has given up a lot for us, and I am deeply grateful to him. My sister, mother and maternal grandmother have taught me that compassion and strength need not be mutually exclusive but should reinforce each other. My maternal grandfather however is the most special person to me. He has had a huge influence on me and I have always looked up to him. We discussed everything from classical music to string theory. He considered good education to be of paramount importance and, fittingly, was the person who taught me how to compute derivatives. He passed away during the first year of my research at ASU, and what was already a very difficult time for me, became the darkest part of my life. Life has since gotten better and today, I imagine him reading this document, smiling proudly.

If I am allowed to philosophize briefly, I have come to understand that even for the few things we actually control in our lives, life at any point is a multi-objective optimization problem with often conflicting objectives. Amongst several other things, I gave up on living close to my family which was very difficult. Some people I know have chosen/had to give up on their research aspirations because of such reasons. I feel very lucky as the people I care about the most have been understanding of this fact and always supported my decisions.

It is only hitting me now, as I am writing this, how much I will miss ASU and Tempe. I am grateful to everyone mentioned here for a multitude of reasons, and to you for taking the time to read these words. On Charlie Brown's face appears a Beautiful Smile.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

In this dissertation, I show that building constraints into layers, latent representations, and functions of deep neural networks for applications in computational imaging and vision leads to different kinds of advantages and improvements over conventional unconstrained frameworks. In the case of compressive imaging, I show how to build practical design constraints into the neural network models designed to solve the image reconstruction problem, leading to more efficient solutions. For the application of multi-spectral image fusion, I design hybrid data-driven and model-based solutions, where the neural network model is designed to perform a specific function of the algorithm rather than a fully black-box end-to-end mapping between inputs and desired outputs, which lends more interpretability to the algorithm. Similarly in action recognition and other time-series classification problems, I show that by adding special layers with constraints, we can add expressive power to deep network in an interpretable manner that allows for learning invariant representations to execution-rate and at the same time, also improved discriminative power. Taking the notion of invariants further, I finally discuss how we can impose geometric constraints at the output of neural networks leading to the general problem of non-linear regression on Riemannian manifolds using neural networks. This leads to architectures that can generate an illumination invariant from a single human face image which can result in illumination-robust face recognition. I now describe three application-areas where

I apply the above ideas. Each application will be discussed in separate chapters in more detail.

1.1 Applications in Imaging

In computational imaging and low-level vision, several important problems are ill-posed inverse problems. These include image denoising, super-resolution, deblurring etc. In this dissertation, we focus on two such applications – reconstruction of compressively sensed images, and fusion of multi-spectral images. Several other works now exist describing how deep learning can be employed for various other inverse problems. The reconstruction problem in compressive sensing can be viewed as the most general form of linear inverse problems and subsumes other applications such as super-resolution as special cases. First, in Chapter 2, I describe a purely data-driven neural network called ReconNet that approximately solves this problem for natural images and overcome several drawbacks of earlier iterative algorithms. Following the developments of the deep learning community such as generative adversarial networks (GANs), I propose several improvements over the base ReconNet architecture to significantly reduce the reconstruction error. In Chapter 3, I show how to translate two interesting practical design constraints into constraints on the measurement operators which can jointly with ReconNet – (1) rank constraints which model the trade-off between the number of measurements to sense, and the reconstruction quality and (2) subset-validity constraints such that subsets of the learned measurement operator are also valid measurement operator for the same reconstruction network, which allows for a single ReconNet to be used over a large range of measurement rates.

Since the development of purely data-driven approach for linear inverse problems, researchers have developed algorithms that combine earlier iterative methods with the data-driven methods with some success. This is mainly done through "unrolling"

the iterative algorithm and each stage is made up of neural network layers that can be trained. In Chapter 4, I show how these ideas are applicable to the problem of multi-spectral image fusion which is a form of super-resolution with side information. As a solution to the problem, based on earlier signal processing theory, I propose unrolled projected gradient descent, where the neural network layers perform the function of projection onto the manifold of high resolution multi-spectral images. Thus, we can build hybrid model-based and data-driven models for multi-spectral image fusion. Experimentally, we make significant gains in terms of both image fusion quality as well as computational speed-up.

1.2 Temporal Warping for Action Recognition

In Chapter 5, I use similar ideas of hybrid model and data-driven neural networks presented in Chapter 4, to perform end-to-end classification of time-series signals, in particular, human action recognition using skeletal sequences. An important nuisance factor in this case is the execution-rate of activities which can be modeled by warping of the time domain using order-preserving diffeomorphisms, which have geometric constraints. To this end, I build a differentiable module Temporal Transformer Network (TTN) that can generate such warping functions, satisfying the required constraints exactly, which are used to undo the effect of rate variations leading to better representations for rate-invariance, and the same time, improve discriminative power. This module can be easily integrated with classification architectures and trained end-to-end using standard backpropagation.

1.3 Regression on Non-Euclidean Spaces

An important idea presented in Chapter 5 is that of building certain kinds of invariances into neural network with the aid of differentiable layers that enforce geometric

constraints. In Chapter 6, I first pose the general question of using neural networks to perform regression on Riemannian manifolds, i.e mapping inputs to points on spaces which have specific geometric constraints and do not obey conventional vector space arithmetic. I focus on regression on the Grassmann manifold and I describe how this method can be used to train a deep network to map to illumination invariants for human face recognition, given a single face image under an unknown illumination condition. These illumination invariants are points on the Grassmann manifold. I propose two methods to solve this problem and demonstrate that incorporating designs that respects the underlying geometry in terms of satisfying constraints as well as meaningful loss functions, leads to significant improvements over conventional neural networks.

Finally, in Chapter 7, I summarize the contributions I have made and discuss directions for future research.

Chapter 2

CONVOLUTIONAL NEURAL NETS FOR RECONSTRUCTION OF COMPRESSIVELY SENSED IMAGES

In this chapter, I will demonstrate the usefulness of developing single, composite scores that are continuous and continuously updated for the purpose of building real-time feedback systems for movement training.

2.1 Introduction

Images and video data are now ubiquitous, and computer vision has grown tremendously with new applications being developed continuously in health-care, defence etc. Depending on the application, many constraints may arise when we build devices and algorithms to be deployed in the real world. In this chapter, we focus on two such constraints. Sensor costs can be prohibitively expensive in certain imaging modalities. For example, in short-wave infrared (SWIR) and medium-wave infrared (MWIR) imaging, the sensor cost can dominate the entire imaging system cost. Bandwidth and power constraints arise in mobile devices, surveillance applications, imagers in space probes etc. An effective way of designing algorithms, while satisfying these constraints to a large extent, is through compressive sensing.

Compressive Sensing (CS) is a signal acquisition paradigm that integrates sampling and compression into a single step performed by front-end hardware. CS theory tells us that one can acquire a small (relative to the ambient dimension) number of measurements which are random projections of a sparse signal and later reconstruct the entire signal perfectly by solving an inverse problem [26]. In the case of natural images, sparsity or compressibility of natural images in transform domains (such as

wavelets) is exploited for this purpose. This sub-Nyquist sampling feature of CS is particularly attractive in applications where sensing time (e.g. magnetic resonance imaging) or bandwidth (e.g. surveillance) is a constraint. One of the first camera architectures to be developed based on CS is the single pixel camera [120] and is commercially produced by the InView Corporation [1] . This camera, as the name suggests, consists of just one photodiode that operates at the required wavelengths and thus is suitable in applications like SWIR imaging where sensor cost is the main constraint. It is also worth mentioning that effort has been made in miniaturizing compressive imagers for possible use in mobile devices, cf. [132]. Here, the authors also show that such sensors can be more energy efficient than their traditional counterparts.

In order to employ such a camera in computer vision for image recognition, tracking etc., a natural pipeline emerges. Once the image is reconstructed from the low-dimensional CS measurements, existing computer vision algorithms can be used without modification. However, iterative reconstruction algorithms form a computational bottleneck in the pipeline. It may take as many as 5 minutes on a CPU (Table 2.2) to reconstruct a single image of size 256×256 using one of these algorithms. This is unacceptable in applications where inference needs to be done in real-time. These algorithms are also ineffective at low measurement rates below 0.1 for images, which is where the advantages of CS are most evident in terms of data reduction. In this chapter, we propose a new reconstruction algorithm that overcomes these drawbacks and is capable of yielding good quality images in real time. Inspired by the recent success of deep Convolutional Neural Networks (CNNs) in computer vision tasks such as super-resolution [39, 40], semantic segmentation [55], [113] etc., we design a novel architecture to map compressive measurements of an image block to the reconstructed image block. Once the architecture (and other hyper-parameters such as the learn-

[1] http://inviewcorp.com/technology/compressive-sensing/

ing rate schedule) is fixed, our approach is entirely data-driven which means that all parameters of the network are learned end-to-end based on training data.

2.1.1 Contributions

1. We describe a CS reconstruction algorithm called ReconNet, that is non-iterative and 3 orders of magnitude faster than conventional iterative approaches. ReconNet – **ReconNet (Euc)** trained using Euclidean loss was introduced in its earlier version [96]. Here, we propose **ReconNet (Euc + Adv)** trained using a combination of Euclidean and adversarial loss.

2. We carry out extensive experiments on a standard test dataset by simulating CS in software and show that our algorithm produces superior quality reconstruction in terms of PSNR at low measurement rates of 0.1 and below, as well as in the presence of noise. We also show that the ReconNet variant with adversarial loss results in sharper reconstructions and improved PSNRs at higher measurement rates than the vanilla ReconNet with just the Euclidean loss [96].

3. We demonstrate the robustness of our network to arbitrary sensor noise by showing high quality reconstructions from real CS measurements obtained using a scalable block compressive camera, although the network is trained using a synthetic set. This dataset will be released for public use.

4. The network complexity of ReconNet [96] is concentrated in the first fully connected layer which accounts for more than 80% of the parameters at higher measurement rates. We propose circulant layers as an alternative to this layer which greatly reduces the number of weights. We verify experimentally that even with a 95% reduction in the number of parameters in the first layer, the drop in PSNR is about 1-2 dB for a wide range of measurement rates.

5. Finally, we make the important observation that even reconstructions at very low measurement rates of about 0.01 retain sufficient semantic content that allow for effective high-level inference such as object tracking.

In Section 2.3.3, we modify the loss function to include adversarial loss which gives sharper reconstructions and higher PSNRs. In Section 2.7, we describe joint learning of the measurement matrix and the reconstruction algorithm and show supporting results. In Section 2.9, circulant layers are used to reduce the network complexity. In Section 2.8, additional results on reconstruction of real data are presented based on the new variants of ReconNet proposed.

2.2 Background and Related Work

We review relevant literature from compressive sensing, computer vision and deep learning here.

2.2.1 Compressive Sensing

As mentioned in Section 2.1, compressive sensing (CS) or compressive sampling is a relatively new paradigm in signal processing developed in the mid 2000s [26]. Here, we have a linear signal acquisition model (performed by hardware) as follows. For a signal $\mathbf{x} \in \mathbb{R}^n$, the measurement vector obtained via CS, henceforth referred to as compressive measurements, denoted by $\mathbf{y} \in \mathbb{R}^m$ is given by

$$\mathbf{y} = \Phi\mathbf{x}, \quad m << n, \tag{2.1}$$

where $\Phi \in \mathbb{R}^{m \times n}$ is called the measurement matrix. Recovering $\mathbf{x}$ from $\mathbf{y}$ is an inverse problem and not admit a unique solution in general. Researchers have shown theoretically that as long as $m = O(s \log(\frac{n}{s}))$, where s is the number of non-zeros in $\mathbf{x}$ when expressed in a transform domain Ψ and the entries of Φ are drawn

from a sub-Gaussian distribution such as a Gaussian, Bernoulli etc., it is possible to recover $\mathbf{x}$ from $\mathbf{y}$ perfectly [42], [25]. In this chapter, the data type of interest is natural images and it is worth mentioning that natural images, although not sparse, are "compressible" in the wavelet domain. The recovery/reconstruction problem has received a great amount of attention in the past decade and we briefly discuss the main algorithms and their drawbacks next.

Iterative algorithms for reconstruction

Several algorithms have been proposed to reconstruct images from CS measurements. The earliest algorithms leveraged the traditional CS theory described above [42, 25, 24] and solved the l_1-minimization in Eq. 2.2 with the assumption that the image is sparse in some transform-domain like wavelet, DCT, or gradient.

$$\min_{\mathbf{x}} \quad ||\mathbf{\Psi x}||_1 \qquad \text{s.t} \qquad ||\mathbf{y} - \mathbf{\Phi x}||_2 \leq \epsilon. \tag{2.2}$$

However, such sparsity-based algorithms did not work well, since images, though compressible, are not exactly sparse in the transform domain. This heralded an era of model-based CS recovery methods, wherein more complex image models that go beyond simple sparsity were proposed. Model-based CS recovery methods come in two flavors. In the first, the image model is enforced explicitly [45, 12, 91, 145], wherein in each iteration the image estimate is projected onto the solution set defined by the model. These models, often considered under the class of 'structured-sparsity' models, are capable of capturing the higher order dependencies between the wavelet coefficients. However, generally a computationally expensive optimization is solved to obtain the projection. In the second, the algorithms enforce the image model implicitly through a non-local regularization term in the objective function [136, 179, 41]. Recently, a new class of recovery methods called approximate message passing

(AMP) algorithms [43, 155, 121] have been proposed, wherein the image estimate is refined in each iteration using an off-the-shelf denoiser.

2.2.2 CNNs for Per-Pixel Prediction Tasks

There has been a great amount of exciting research in areas like semantic segmentation [55], [113], depth estimation [47], surface normal estimation [163] etc., where CNNs have outperformed all traditional methods. In such an application, an input image is mapped to a similar-sized output. Another related class of tasks which is of interest here is ill-posed inverse problems – problems where the output is of a higher dimension than that of the input. Example include automatic colorization [32], 3D reconstruction a single image [84] and super-resolution (SR) [39], [40] and image denoising [180]. For SR, the authors design a CNN, SRCNN, that takes an input image that is upsampled using bicubic interpolation and produces a super-resolved version of the original image of a lower resolution. The network architecture we design in this chapter is inspired by SRCNN. The reason for this is that the problem of CS reconstruction can be seen as a generalization of SR. However, although both CS recovery and SR can be cast as solving an inverse problem $\mathbf{y} = \Phi\mathbf{x}$, they are not considered under the same umbrella. The reason for this is discussed in more detail in Section 2.3.

To summarize, we make the observation that any neural network can be viewed as an algorithm that allows for efficient learning of a non-linear mapping from the input to the desired output. In our case, we apply this notion to learning the (necessarily non-linear) mapping from CS measurements to the image. This is also significant since 2D CNNs have until now been mainly shown to be useful for inputs which are images. CS measurements, however, are typically random projections of the scene and do not have the spatial correlational structure present in natural images. Thus,

they cannot be used directly as inputs to a 2D CNN. In Section 2.3, we describe the architecture that aims at resolving this apparent incompatibility.

Generative Adversarial Networks

In section 2.3.3, we discuss a modification of the loss function for ReconNet based on the recently popular Generative Adversarial Network (GAN) framework. It has been shown recently that for inverse problems such as image inpainting [133], super-resolution [102] and surface normal estimation [174], using a GAN framework yields sharper results, than by using just Eucliden loss. This is simply due to the averaging effect of Euclidean loss minimization. As described in the papers by Goodfellow et al [57] and Radford et al. [137], i.e., in the original formulation, a GAN learns to model the image distribution where the image is represented as a random variable R in an unsupervised fashion by learning a mapping from a small dimensional uniform random variable, z (which we can sample easily) to the image. In a GAN, two networks – a generator, $\mathcal{G}$ with parameters $\Theta_{\mathcal{G}}$ and a discriminator, $\mathcal{D}$, with parameters $\Theta_{\mathcal{D}}$ – are trained in an alternating fashion. $\mathcal{G}$ is a neural network responsible for generating an image for a given input. $\mathcal{D}$ is another neural network which learns to classify between "real" images and images output by the generator – "fake" images. During training, $\mathcal{D}$ tries to minimize this classification error by updating $\Theta_{\mathcal{D}}$. At the same time, $\mathcal{G}$ tries to maximize the loss of $\mathcal{D}$ by updating $\Theta_{\mathcal{G}}$, thereby trying to "fool" $\mathcal{D}$. The mathematical form of the optimization is given by

$$\min_{\Theta_{\mathcal{G}}} \max_{\Theta_{\mathcal{D}}} \quad \mathbb{E}_R[\log(\mathcal{D}(r))] + \mathbb{E}_Z[\log(1 - \mathcal{D}(\mathcal{G}(z)))] \tag{2.3}$$

Empirically, it has been shown that this optimization results in $\mathcal{D}$ being unable to classify better than chance and $\mathcal{G}$ learning to model the data distribution and generate "realistic" images.

2.2.3 Purely Data-Driven Approaches for CS Image and Video Reconstruction Using Deep Learning

There are several works that propose CS recovery algorithms based on deep learning. Many of these algorithms are based on designing the neural net architecture based on unrolling earlier iterative schemes that take into account signal priors [58, 15, 37, 80, 152, 30, 123]. However, in this work, we focus on methods that are purely data-driven and make no assumptions on the signal model. Ali et al. [126] first presented a stacked denoising auto-encoders (SDAs) based non-iterative approach for problem of CS reconstruction. In the preliminary version of this chapter [96], we proposed a convolutional architecture, which has fewer parameters, and is easily scalable to larger block-size at the sensing stage. One of the drawbacks of the approaches presented in both [126] and [96] is that the reconstructions are performed independently on each block. It results in the approaches not utilizing the strong dependencies that exist between the reconstructions of different blocks. In order to address this, Ali et al. [124] propose a network that can operate on the CS measurements of the entire image, while forcing the fully connected layer to be Φ^T. Ali et al. propose another method which learns to simultaneously compute non-linear measurements and the reconstruction layers using an autoencoder framework [125]. Yao et al. [173] modify the ReconNet architecture [96] by adding residual connections and present improved reconstruction performance. Dave et al. [37] show that by enforcing an image prior which captures long term spatial dependencies, one can recover better quality reconstructions than the iterative counterparts. Chakrabarti [27] proposes to learn the sensor multiplexing pattern in conjunction with the non-iterative reconstruction network. The success of the deep learning approaches in compressive recovery problem has not been limited to the image reconstruction problem. Researchers have shown

that they can be applied to the CS video recovery problems as well [73, 169]. Iliadis et al. [74] describe a novel encoder-decoder neural architecture in order to jointly learn a binary measurement matrix and the reconstruction for videos. For a more in-depth discussion of deep neural networks for inverse problems in imaging, please read the recent survey by Lucas et al. [114]

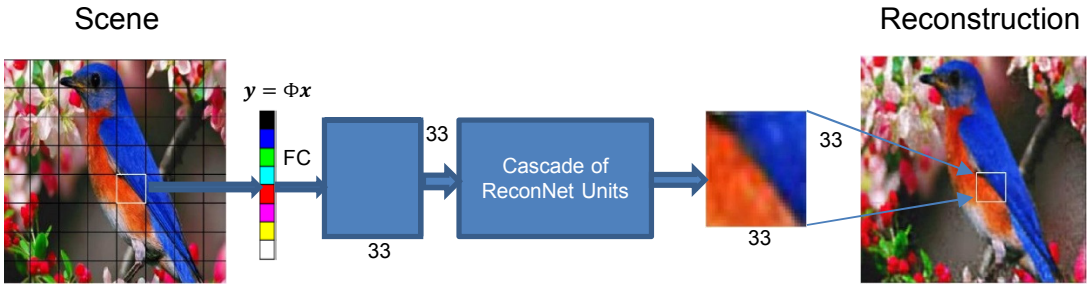


Figure 2.1: This shows the overview of the proposed non-iterative CS reconstruction algorithm: ReconNet. The architecture that we use for all the experiments operates using block CS. A scene is divided into blocks of size 33×33 and CS measurments of each block is passed through the ReconNet to obtain the reconstructed image patch. As a post-processing step, the image thus obtained is passed through BM3D denoiser to get rid of the blocky artifacts.

2.3 ReconNet

In this section, we describe in detail the network architecture and other implementation details. Figure 3.4 shows the overview of the proposed algorithm. Each image is divided into non-overlapping whose CS measurements are obtained separately. We need to reconstruct each image block from its compressive measurements. Then, the block reconstructions are arranged to form an image and passed through a denoiser to remove the blocky artifacts and produce the final reconstruction.

Although our network architecture was inspired by SRCNN [39], [40], the input in our case is a one-dimensional vector of CS measurements without any spatial structure, unlike an image in the case of SRCNN. Thus, in order to employ a CNN for reconstruction, we need to first resolve this incompatibility. One way to work around this problem is to seek inspiration from the SRCNN pipeline where an initial high resolution image is first obtained using bicubic interpolation and is used as the input to 3-layer CNN which produces the final super-resolved image. In our case, we could use an initial image estimate obtained using one of the many iterative approaches and then use the network to refine it to produce the final reconstruction. Although this is straightforward conceptually, the question of how many iterations of the algorithm to run to get the initial image estimate is hard to answer. While increasing the number of iterations improves the initial estimate, it also increases the run-time, thus moving away from the goal of fast implementation. On the other hand, too few iterations yield poor estimates. Therefore, we opt for a better and a more elegant solution – to use a fully connected layer at the beginning in order map the CS measurement vector to a two-dimensional array that may serve as an initial image estimate. However, all the parameters of the network are learned end-to-end. The presence of the fully connected layer, is also the main reason why we need to operate block-wise instead of trying to reconstruct the whole image directly. If we were to do the latter, the number of parameters in the fully connected layer would be too large to store the weights and would be easily prone to overfitting. We discuss alternatives to the fully connected layer later in Section 2.9.

2.3.1 ReconNet Unit Architecture

The input to the network is a vector of size $m \times 1$ denoted by $\Phi\mathbf{x}$, where Φ is the measurement matrix and $\mathbf{x}$ is the vectorized image block of size $n \times 1$ such that

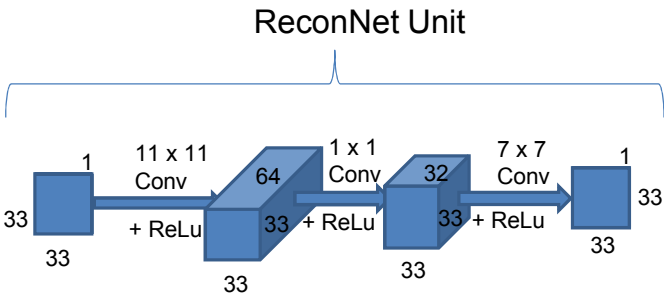


Figure 2.2: Each ReconNet Unit consists of 3 convolutional layers with ReLU non-linearity. Using appropriate zero-padding, the size of each feature map is always kept constant and equal to the block size.

$m << n$. In all the experiments, we set the block-size to be 33×33, ($n = 1089$) as this gives a good trade-off between reconstruction quality and network complexity.

The first layer is a fully connected layer that takes compressive measurements as input and outputs a feature map of size 33×33. This feature map is then input to a series of 'ReconNet units'. Each ReconNet unit consists of three convolutional layers as shown in Figure 2.2. ReLU non-linearity is employed. Using appropriate zero-padding, all feature maps produced by all convolutional layers are set to size 33×33, which is equal to the block size.

The first convolutional layer uses kernels of size 11×11 and generates 64 feature maps. The second convolutional layer uses kernels of size 1×1 and generates 32 feature maps. The third convolutional layer uses a 7×7 kernel and generates a single feature map. The output of the third layer of the last ReconNet unit is also the output of the network.

Once all the blocks of an image have been reconstructed, the entire image is input to a denoiser to reduce blocky artifacts that arise as a result of block-wise processing. We choose BM3D [36] as the denoiser as it is fast and yields good results.

2.3.2 *Training Data*

Ground Truth: We uniformly extract patches of size 33×33 from 91 natural images (these are the same images used for training in [39] and can be found on this website [2]) with a stride equal to 14 to form a set of 21760 patches. We retain only the luminance component of the extracted patches (during the test phase, in order to reconstruct RGB images, we replicate the network to recover the individual channels). These image blocks form the desired outputs or the ground truth of our training set. Experiments indicate that this training set is sufficient to obtain very competitive results compared to existing CS reconstruction algorithms.

Input data: To train our networks, we need CS measurements corresponding to each of the extracted image blocks. To this end, we simulate noiseless CS as follows. For a given measurement rate, we construct a measurement matrix, Φ by first generating a random Gaussian matrix of appropriate size, followed by orthonormalizing its rows. Then, we apply $\mathbf{y} = \Phi\mathbf{x}$ to obtain the set of CS measurements, where $\mathbf{x}$ is the vectorized version of the luminance component of an image block. Thus, an input-label pair in the training set can be represented as $(\Phi\mathbf{x}, \mathbf{x})$. We train networks for four different measurement rates (MR) $= 0.25, 0.10, 0.04$ and 0.01. Since, the total number of pixels per block is $n = 1089$, the number of measurements $n = 272, 109, 43$ and 10 respectively.

2.3.3 *Loss Function*

In this section, we describe the two variants of ReconNet based on the loss function used in training.

[2] mmlab.ie.cuhk.edu.hk/projects/SRCNN/SRCNN_train.zip

Euclidean Loss

The first variant of ReconNet [96] employs the Euclidean loss i.e., the average reconstruction error over all the training image blocks, given by

$$L(\Theta) = \frac{1}{B}\sum_{i=1}^{B} ||f(\mathbf{y_i}, \Theta) - \mathbf{x}_i||^2, \tag{2.4}$$

and is minimized by adjusting the parameters (weights and biases) in the network, Θ using mini-batch gradient descent with backpropagation. B is the total number of image blocks in one batch of the training set, x_i is the i^{th} patch and $f(\mathbf{y_i}, \Theta)$ is the network output for i^{th} patch. We set the batch size, $B = 128$ for all the networks. For each measurement rate, we train two networks, one with random Gaussian initialization for the fully connected layer, and one with a deterministic initialization, and choose the network which provides the lower loss on a validation set. For the network with deterministic initialization, the j^{th} weight connecting the i^{th} neuron of the fully connected layer is initialized to be equal to $\Phi^T_{i,j}$. In each case, weights of all convolutional layers are initialized using a random Gaussian with a fixed standard deviation. The learning rate is determined separately for each network using a linear search. Through experiments, we have found that two ReconNet units (6 convolutional layers in total) produce good performance. Adding further ReconNet units does not produce a significant boost in reconstruction quality and adds to network complexity. All networks are trained on an Nvidia Tesla K40 GPU using Caffe [79] for about a day (about 10^6 iterations) each even though the reconstruction errors are very close to the final value within few hours. For testing, we choose the best network by using a validation set. We refer to this network as **ReconNet (Euc)**, which uses Gaussian matrix for sensing and only the Euclidean loss function.

Euclidean + Adversarial Loss

Here, we describe the second variant of ReconNet by incorporating the GAN framework for CS reconstruction similar to [133]. See Section 2.2.2 for an overview of GANs and notation. In our case, **ReconNet acts as $\mathcal{G}$**. We build $\mathcal{D}$ that takes as input either the reconstructed block from ReconNet ("fake") or the desired block ("real") and outputs the probability of the input being a real image block. The loss function of $\mathcal{D}$ is the sum of two cross-entropy losses shown below:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{B}\sum_{i=1}^{B}(\mathcal{L}_{CE}(\mathcal{D}(\mathbf{x}_i), 1) + \mathcal{L}_{CE}(\mathcal{D}(\mathcal{G}(\mathbf{y}_i)), 0)). \tag{2.5}$$

The first loss term measures how well $\mathcal{D}$ is able to classify the real images while the second loss term measures its ability to classify the fake images generated by ReconNet, i.e, $\mathcal{G}$. Following the same notation as before, $\mathbf{y}_i$ denotes the i^{th} input training CS measurement vector and $\mathbf{x}_i$ denotes the ground truth 33×33 image block associated with it. $\mathcal{L}_{CE}()$ is the cross-entropy loss commonly used in binary classification, given by

$$\mathcal{L}_{CE}(\hat{\mathbf{c}}, c) = -c \log \hat{c} + (1 - c) \log (1 - \hat{c}) \tag{2.6}$$

The loss for $\mathcal{G}$ i.e., ReconNet is a linear combination of the Euclidean loss (from Equation 2.4) and the adversarial loss:

$$\mathcal{L}_{\mathcal{G}} = \frac{\lambda_{rec}}{B}\sum_{i=1}^{B}||\mathcal{G}(\mathbf{y_i}) - \mathbf{x}_i||^2 + \frac{\lambda_{adv}}{B}\sum_{i=1}^{B}\mathcal{L}_{CE}(\mathcal{D}(\mathcal{G}(\mathbf{y}_i)), 1) \tag{2.7}$$

The protocol for initializing and training the $\mathcal{G}$ portion is the same as in the case of Euclidean loss (see Section 2.3.3). However, we use just one ReconNet unit in this case as the reconstruction quality does not improve and also becomes harder to train due to the presence of $\mathcal{D}$ in addition to $\mathcal{G}$. Since $\mathcal{G}$ is fixed, the remaining

hyperparameters that need to be determined are the values of λ_{rec}, λ_{adv} and the structure of $\mathcal{D}$, which is another, much smaller, CNN. These hyperparameters were determined by measuring the reconstruction performance on the validation set for different settings. We use a $\mathcal{D}$ with the following architecture. It has 3 convolutional layers and each layer generates four feature maps of size 4×4 filters. At the end of the third convolutional layer, a fully connected layer maps the feature maps to a single probability value. Dropout with probability equal to 0.5 is used for this layer. λ_{rec} and λ_{adv} are set to 1 and 0.0001 respectively. Adam optimizer is used for learning [92]. The learning rates for $\mathcal{G}$ and $\mathcal{D}$ are set to 10^{-3} and 10^{-5} respectively and the momentum is set to 0.9. The training of these networks is done in an alternating fashion using TensorFlow. We update $\Theta_{\mathcal{G}}$ twice for every update of $\Theta_{\mathcal{D}}$ since this leads to faster convergence. Training is carried out for 10^5 iterations which means that $\Theta_{\mathcal{G}}$ are updated 2×10^5 times and $\Theta_{\mathcal{D}}$ are updated 10^5 times. We refer to this network as **ReconNet (Euc + Adv)**, which uses Gaussian matrix for sensing and the Euclidean + adversarial loss function.

2.4 Synthetic Experiments

In this section, we conduct extensive experiments on simulated CS data, and compare the performance of ReconNet with state of the art CS image recovery algorithms, both in terms of reconstruction quality and time complexity.

Baselines We compare both variants of our algorithm described in Section 2.3 with three iterative CS image reconstruction algorithms, TVAL3 [103], NLR-CS [41] and D-AMP [121]. We use the code made available by the respective authors on their websites. Parameters for these algorithms, including the number of iterations, are set to the default values. Since the reconstruction is performed block-wise, blocky arti-

facts arise. We use BM3D [36] denoiser to reduce these artifacts since it gives a good trade-off between time complexity and reconstruction quality. The code for NLR-CS provided on author's website is implemented only for random Fourier sampling. The algorithm first computes an initial estimate using a DCT or wavelet based CS recovery algorithm, and then solves an optimization problem to get the final estimate. Hence, obtaining a good estimate is critical to the success of the algorithm. However, using the code provided on the author's website, we failed to initialize the reconstruction for random Gaussian measurement matrix. Similar observation was reported by [121]. Following the procedure outlined in [121], the initial image estimate for NLR-CS is obtained by running D-AMP (with BM3D denoiser) for 8 iterations. Once the initial estimate is obtained, we use the default parameters and obtain the final NLR-CS reconstruction.

We compare with [126] which presents an SDA based non-iterative approach to recover from block-wise CS measurements. Using our own implementation of SDA, we show that ReconNet outperforms SDA.

We also create another deep network baseline based on the ReconNet architecture. Once we compute the compressive measurements $\mathbf{y}$ using the camera, we can project the measurements back into the pixel domain by pre-multiplying with Φ^T to form a pseudo-image i.e., $\hat{\mathbf{x}} = \Phi^T\mathbf{y}$. We then use the pseudo-image as an input to the ReconNet units directly without the first fully-connected layer. We note this architecture was suggested by Ali et al. for reconstruction of CS measurements [125] and by Lohit et al. [109] in the case of direct inference from compressive measurements. This architecture has far fewer learnable parameters and all the weights are in the convolutional layers, as the first layer is not learned.

For fair comparison, we denoise the image estimates recovered by baselines as well. The only parameter to be input to the BM3D algorithm is the estimate of the

standard Gaussian noise, σ. To estimate σ, we first compute the estimates of the standard Gaussian noise for each block in the intermediate reconstruction given by $\sigma_i = \sqrt{\frac{||y_i - \Phi x_i||^2}{m}}$, and then take the median of these estimates.

2.4.1 Reconstruction of Simulated CS Data

For our simulated experiments, we use a standard test set of 11 grayscale images, compiled from two sources [3] [4]. We conduct both noiseless and noisy block-CS image reconstruction experiments at four different measurement rates 0.25, 0.1, 0.04 and 0.01. We train two sets of networks – The first set of networks is ReconNet Variant 1 trained with just Euclidean loss. The second set is ReconNet variant 2 trained with Euclidean + adversarial loss.

Reconstruction from noiseless CS measurements

For a given test image, to simulate noiseless block-wise CS, we first divide the image into non-overlapping blocks of size 33×33, and then compute CS measurements for each block using Equation 2.1. For each measurement rate, the sensing matrix used is the same random Gaussian measurement matrix as was used to generate the training data for the network corresponding to this measurement rate in Section 2.3.2. The PSNR values in dB for both reconstructions before passing through the denoiser (indicated by w/o BM3D) as well as final denoised versions (indicated by w/ BM3D) for all the measurement rates are presented in Table 2.1. It is clear from the PSNR values that both variants of our algorithm outperforms traditional reconstruction algorithms at low measurement rates of $0.1, 0.04$ and 0.01. Also, the degradation in performance with lower measurement rates is more graceful. Further, in Figure 2.3, we show the

[3] https://web.archive.org/web/20160403234531/http://dsp.rice.edu/software/DAMP-toolbox

[4] http://see.xidian.edu.cn/faculty/wsdong/NLR_Exps.htm

final reconstructions of parrot and house images for various algorithms at measurement rate of 0.1 compared to ReconNet (Euc). From the reconstructed images, one can notice that our algorithm, as well as SDA, are able to retain the finer features of the images while other algorithms fail to do so. NLR-CS and DAMP provide poor quality reconstruction. Even though TVAL3 yields PSNR values comparable to our algorithm, it introduces undesirable artifacts in the reconstructions.

For visual comparison between the baseline of $\Phi^T\Phi\mathbf{x} + 2$ ReconNet units, ReconNet (Euc) and ReconNet (Euc + Adv), see 1^{st} and 2^{nd} columns of Figure 2.8. We observe that at higher measurement rates of 0.25 and 0.10, there is improvement in reconstruction of the test set with ReconNet (Euc + Adv) over ReconNet (Euc) both in terms of PSNR ($\sim$1 dB increase) and visual quality. The reconstructed blocks are sharper than those obtained in the case of Euclidean loss. At lower measurement rates of 0.04 and 0.01, PSNR values decrease for ReconNet (Euc + Adv) when compared to ReconNet (Euc). However, we can observe that more detail is preserved and the reconstructed images tend to be sharper when adversarial loss is used, in all cases.

Performance in the presence of noise

We demonstrate that our algorithm is robust to Gaussian noise by performing reconstruction from noisy CS measurements. We use ReconNet (Euc) for all the experiments here and we expect the same trends to follow for other variants as well. We perform this experiment at three measurement rates - $0.25, 0.10$ and 0.04. We emphasize that we **do not** train separate networks for different noise levels but use the same networks as used in the noiseless case. In order to simulate the noisy CS process, we add standard random Gaussian noise of increasing standard deviation to the noiseless CS measurements (from the previous section) of each block. In each measurement rate, we test the algorithms at three levels of noise corresponding to $\sigma = 10, 20, 30$

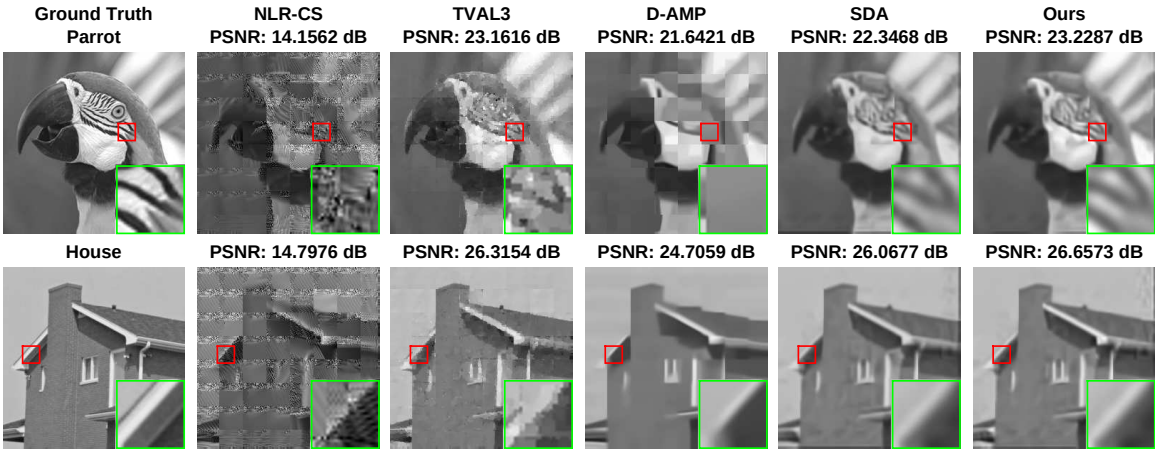


Figure 2.3: Comparison of reconstruction performance of various algorithms with ReconNet (Euc) in terms of PSNR (in dB) and visual quality at MR = 0.1 and no noise for Parrot and House images. Our algorithm outperforms all the iterative algorithms. SDA also yields competetive results. The zoomed in portions show that finer structures are better retained in our case.

(3.9%, 7.8% and 11.7% of the dynamic range (0-255) respectively), where σ is the standard deviation of the Gaussian noise distribution. The reconstructions obtained from the algorithms are denoised using BM3D. The mean PSNR for various noise levels for different algorithms at different measurement rates are shown in Figure 2.4. It can be observed that our algorithm beats all other algorithms at high noise levels. This shows that the method proposed in this chapter is robust to all levels of noise.

2.4.2 Time Complexity

In addition to competitive reconstruction quality, for our algorithm without the BM3D denoiser, the computation is real-time and is about **3** orders of magnitude faster than traditional reconstruction algorithms. **However, it is important to note that the codes for the traditional reconstruction algorithms are mainly in Matlab, and hence, it may be amenable to further optimization (for ex-**

ample, by using faster routines in C++). We compare various algorithms in terms of the time taken to produce the reconstructions of a 256×256 image from noiseless CS measurements at various measurement rates. For traditional CS algorithms, we use an Intel Xeon E5-1650 CPU to run the implementations provided by the respective authors. For ReconNet, we report computational time for both the CPU implementation of Caffe on Intel Xeon E5-1650 as well as the GPU implementation on an inexpensive mid-range Nvidia GTX 980 GPU. Note that, for our algorithm, we use a network with two ReconNet units. The average time taken for the all algorithms of interest are given in table 2.2. Depending on the measurement rate, the time taken for block-wise reconstruction of a 256×256 on the GPU for our algorithm is about 145 to 390 times faster than TVAL3, 1400 to 2700 times faster than D-AMP, and 14782 to 15660 times faster than NLR-CS. In the case of CPU implementation, the speed-ups are 5.6 to 15 times faster, 52.9 to 105.2 times faster and 569 to 600 times faster compared to TVAL3, D-AMP and NLR-CS respectively. We believe that the speedup achieved by our algorithm is not solely because of the utilization of the GPU. It is because unlike traditional CS algorithms, our algorithm being CNN based relies on much simpler convolution operations, for which very fast implementations exist. More importantly, the non-iterative nature of our algorithm makes it amenable to parallelization. SDA, also a deep-learning based non-iterative algorithm shows significant speedups over traditional algorithms at all MRs.

2.4.3 Comparison with LDAMP

In addition to purely data-driven (like ReconNet) and purely model-based iterative algorithms (like D-AMP), there has been recent work on a new class of algorithms that combines the two approaches. This is achieved by unrolling the iterative algorithms. Although the reconstruction process is not entirely replaced by the neural network,

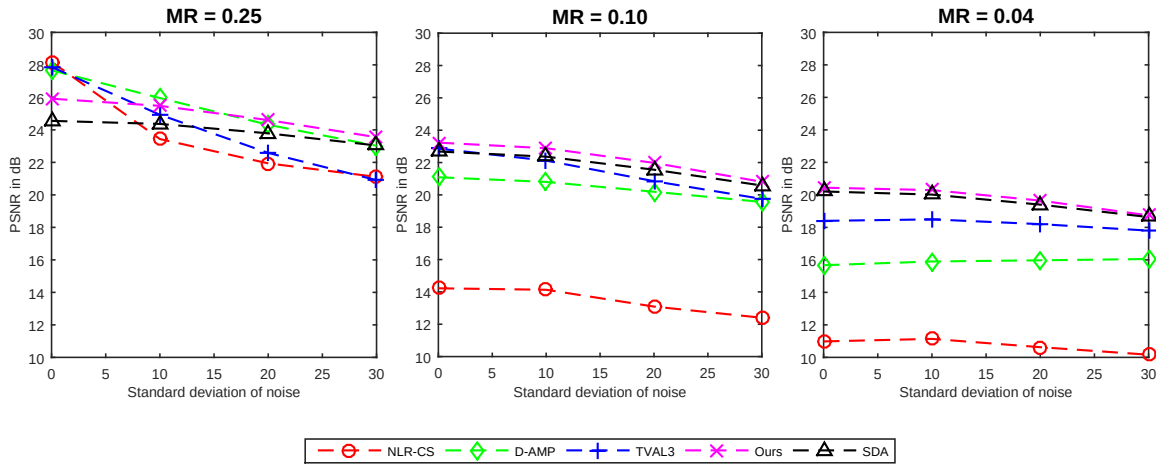


Figure 2.4: The figure shows the degradation of performance of various algorithms as the measurement rate and noise level in the CS measurements are increased. Our algorithm exhibits a more graceful trend compared to the iterative apporaches and outperforms them at low MRs and high noise levels.

the main operation of each iteration (denoising, in the case of D-AMP) is performed by a trained neural network. Here, we compare ReconNet, which is a purely data-driven algorithm, with Learned D-AMP (LDAMP) [123], which is the state of the art in the class of algorithms obtained by unrolling the iterative algorithms. In LDAMP, the iterations of D-AMP [121] are unrolled and using a trained neural network for the denoiser in each iteration. We will refer to each unrolled D-AMP iteration as a D-AMP unit. Each D-AMP unit contains a denoiser which is built using a CNN with residual connections with 16 layers of 3×3 filters and 64 feature maps in each layer. The authors in [123] used 10 D-AMP units and thus, 160 layers of convolutions. Based on the codes provided by the authors of LDAMP, we trained models for MRs $= 0.01$, 0.04, 0.10 and 0.25.

We observe the following important differences in architecture. First, the 16-layer CNN in **each** denoiser contains a little more than 5×10^5 parameters. Thus, using

10 D-AMP units would result in 5 million parameters. Also, as shown in Table 2.3, the with identical hardware settings, the computation time for reconstruction is between 36 times (for MR = 0.01) and 97 times (for MR = 0.25) that of ReconNet. **Thus, unlike ReconNet, LDAMP cannot be run in real-time with currently available hardware.** This is expected, given that number of convolutional layers in LDAMP is 160 to that of ReconNet which has 6. Figure 2.5 shows the comparison between LDAMP trained with different number of D-AMP units and Reconnet. For low MRs 0.01 and 0.04, both methods yield very similar results. With 10 D-AMP units, LDAMP is better than ReconNet by only 0.21 dB and 0.06 dB at MR = 0.04 and 0.01 respectively, in spite of the fact that ReconNet is about 36 and 45 times faster, and contains about 72 and 150 times fewer learnable parameters at MR = 0.04 and 0.01 respectively. For higher MRs = 0.10 and 0.25, LDAMP is clearly better with 10 D-AMP units by 6.02 dB and 2.37 dB respectively. However, the improved reconstruction comes at the expense of huge computation cost, both in terms of memory and time. Interestingly, ReconNet yields a performance comparable to LDAMP with 5 D-AMP units at MR = 0.10 and 3 D-AMP units at MR = 0.25, with far fewer parameters and is tens of times faster. The trends for LDAMP are similar to that of iterative algorithms in that, while it is excellent at high MRs (for 10 D-AMP units), its performance falls much faster than ReconNet as MR is reduced.

Another important difference is that for LDAMP, the CS measurements are obtained at the image-level and not the block-level. Therefore, the issue of blocky artifacts appearing in the reconstructed image in the case of ReconNet, is not present in LDAMP. We believe this to be an important reason for why LDAMP yields better reconstructions, although at the expense of a lot more parameters and computation time. We partly alleviate this issue for ReconNet using a denoiser post-reconstruction. Perhaps more importantly, in the case of LDAMP, for an image size of 256×256, and

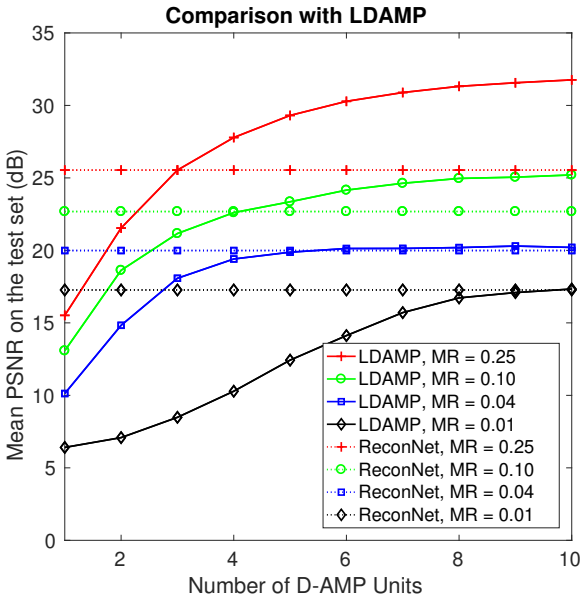


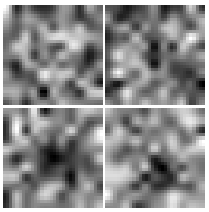
Figure 2.5: Comparison of reconstruction performance of LDAMP versus ReconNet at different MRs and different number of D-AMP units. At MRs = 0.25 and 0.10, although LDAMP with many D-AMP units is significantly better than ReconNet, this comes at a large expense of network parameters and computation, e.g. at MR = 0.10, ReconNet is 35 times lighter and 60 times faster than LDAMP with 10 D-AMP units. Interestingly, the increased computation is not beneficial for LDAMP at MR = 0.4 and 0.01 where both ReconNet and LDAMP (10 D-AMP units) perform nearly at the same level in terms of PSNR. Note that BM3D is not used for ReconNet.

MR = 0.25, the measurement matrix Φ is of size $65536 \times 16378 \approx 1$ billion parameters. This means that, unlike ReconNet, it is not clear if it is possible to learn the Φ jointly with the reconstruction network, which we later show yields about 3 dB gain in mean PSNR (Table 2.5).

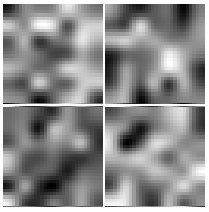
2.5 Analysis of Trained Models

In the section, we provide some insight to the inner workings of ReconNet by visualizing intermediate outputs as well as the learned filters. The output feature map after the fully connected layer, the first ReconNet unit and the second ReconNet unit (also the final output) are shown in Figure 2.7 for the Cameraman image at MRs = 0.25 and 0.1. As it can be seen from the images, the network is able to create a very coarse approximation of the desired output image at the end of the fully connected layer. The output at the end of the first ReconNet unit contains more high frequency content of the final output. This is remarkable because the convolutional layers, assume that spatial correlational structure is a property of the input on which they are acting. Although the CS measurements have no such property, the convolutional layers, through backpropagation, force the fully-connected layer to produce an "image-like" output. It is to be noted that except designing the architecture, this was not explicitly enforced and is simply a result of the optimization.

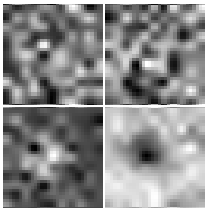
Figure 2.6 shows, both in spatial and frequency domain, the filters of different convolution layers of the trained network at MR = 0.25. We make the observation that they are very different from the filters learned in the case of inference tasks like image recognition, where, for example, first layer filters may look like edge filters, and filters in intermediate layers take on object-like appearances. This difference is not unexpected, since the current task is that of image reconstruction, where one needs to re-introduce low as well as high-frequency information through the layers, and one does not need to learn invariances to deformations. While the filters shown do not reveal significant structure in the spatial domain, their speckle-field like structure in the frequency domain indicates that the filters enhance frequencies across a wide spectral range in non-trivial ways.



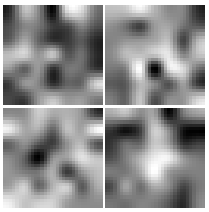
(a) conv 1 filters in spatial domain



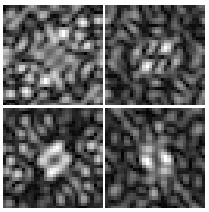
(b) conv 3 filters in spatial domain



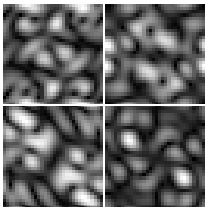
(c) conv 4 filters in spatial domain



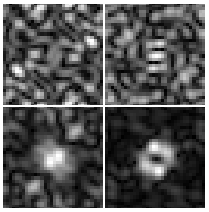
(d) conv 6 filters in spatial domain



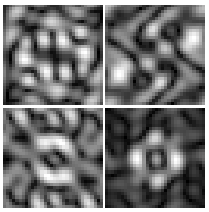
(e) 11 x 11 conv 1 filters in frequency domain



(f) 7 x 7 conv 3 filters in frequency domain



(g) 11 x 11 conv 4 filters in frequency domain



(h) 7 x 7 conv 6 filters in frequency domain

Figure 2.6: Visualizing some (smoothed) trained filters in the convolutional layers in the spatial and frequency domains for MR = 0.25. The speckle-field like structure in the frequency domain indicates that the filters enhance frequencies across a wide spectral range in non-trivial ways.

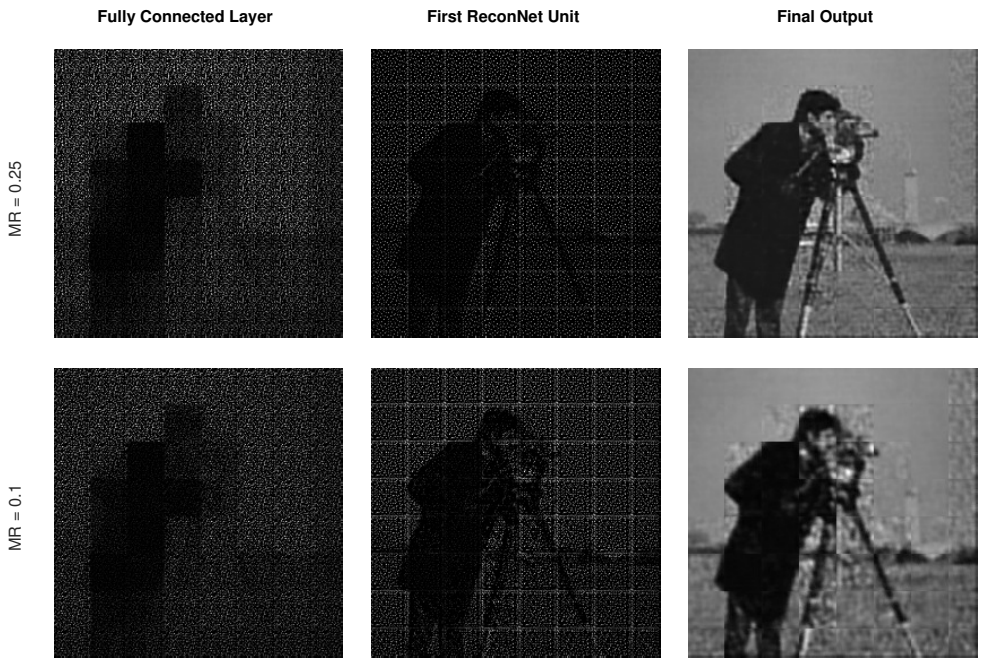


Figure 2.7: The figure shows the intermediate outputs for the Cameraman image. Each row corresponds to a different MR. The first column shows the output obtained at the end of the fully connected layer. The second column shows the output after the first ReconNet unit. The third column shows the output of the network.

2.6 Efficient Training Strategy for New Measurement Matrix

In Section 2.3.2, a new network was trained from scratch for each MR. However, it may not be practical to train a entirely new network just to operate a slightly different MR or with a different Φ at the same MR. In this section, we show that for a new Φ of a desired measurement rate, one **does not** need to train the network from scratch, and that it may be sufficient to follow a suboptimal, yet effective and computationally light training strategy outlined below, ideally suited to practical scenarios.

We adapt the convolutional layers (C1-C6) of a pre-trained network for the same or slightly higher MR, henceforth referred to as the *base network*, and train **only**

the first fully connected (FC) layer with random initialization for 1000 iterations (or equivalent time of around 2 seconds on a Titan X GPU), while keeping C1-C6 **fixed**. The network used here is ReconNet (Euc) [96] and we expect similar trends for other variants. The mean PSNR (without BM3D) for the test set at various MRs, the time taken to train models and the MR of the base network are given in Table 2.4. From the table, it is clear that the overhead in computation for new Φ is trivial, while the mean PSNR values are comparable to the ones presented in Table 2.1. One can obtain better quality reconstructions at the cost of more training time if C1-C6 layers are also fine-tuned along with FC layer.

2.7 Learning the Measurement Matrix

Until now we have considered CS reconstruction where the measurements are acquired with a predefined sensing matrix – a random Gaussian matrix. However, with a small addition to the the ReconNet framework, we show that it is possible to jointly, in a single network, learn the measurement matrix (Φ) as well as the reconstruction algorithm. The earlier framework describes a network that map the input CS measurements to the output image block. Here, we attach an additional fully connected layer in the front that maps an input image of size 33×33 to a vector of dimension m. Thus the input-desired output pair in the training set is $(\mathbf{x}, \mathbf{x})$. This can be seen as a variation of the autoencoder, with the constraint in the architecture that the "encoder" part of the network must be a single linear layer. This constraint arises because of the nature of the single pixel camera which can only capture linear projections of the scene. After training, the weights of the first fully connected layer correspond to the (locally) optimal measurement matrix, and the all the following layers form the reconstruction network.

Figure 2.8: The figures show reconstruction results for the "Parrot" image at two measurement rates of 0.1 and 0.04 from measurements obtained using different variants of ReconNet. We can observe that learning the measurement matrix as well as using adversarial loss while training produce superior quality reconstruction (both independently and together) at both measurement rates when compared to the basic version of ReconNet. MM refers to the measurement matrix.

As before, we train two sets of networks: **ReconNet (Euc, learn Φ)** – jointly learning Φ and reconstruction algorithm using only Euclidean loss and **ReconNet (Euc + Adv, learn Φ)** – jointly learning Φ and reconstruction algorithm using Euclidean + adversarial loss. The training set consists of 21760 image patches from the same set of 91 images. Since we are learning Φ as well as the reconstruction network, each image patch in the training set forms both the input and the desired output image patch. Table 2.5 shows the mean PSNR obtained on the test set using variants of ReconNet with the learned Φ compared to ReconNet with the random Gaussian Φ. We observe a significant gain in terms of PSNR at the lower measurement rates – 2.83 dB, 3.15 dB and 2.17 dB at MR = 0.10, 0.04 and 0.01 respectively without

using adversarial loss. With adversarial loss, the gains are 3.25 dB, 3.33 dB and 2.4 dB at MR = 0.10, 0.04 and 0.01 respectively. Figure 2.8 illustrates the differences in visual quality obtained for the Parrot and House images at two different measurement rates of 0.1 and 0.04 for all four variants of ReconNet. Clearly, more detail is preserved in the case of learned Φ and using adversarial loss further sharpens reconstructions.

2.8 Reconstruction of Real Data From Compressive Imager

The previous section demonstrated the superiority of our algorithm over traditional algorithms for simulated CS measurements. Here, we show that our networks trained on simulated data can be readily applied to the real world, by reconstructing images from CS measurements obtained from our block SPC. We compare our reconstruction results with other algorithms.

2.8.1 Scalable Optical Compressive Imager Testbed

Here we employ a compressive imaging system implementation [87],[86], which is scalable with respect to field of view and/or resolution and avoids limitations inherent in a single-pixel implementation [120]. Scalability is achieved via a block wise measurement approach. The compressive imaging system is implemented via two imaging arms and a discrete mirror device (DMD) as shown in Figure 2.9. The DMD is an array of electronically controllable bi-stable mirrors of 10.8μm pitch, which modulates the incoming light intensity field with 8 bits gray-scale transmission patterns. Since the sensing matrix contains both positive and negative values, two sets of measurements are obtained – one for the positive entries and the other for the negative entries – the difference is used as the measurements from the camera. The first arm of the system images the object or scene onto the DMD surface, mapping in to an area of about 262 $\times$ 262 micro-mirror element (or about 2.85mm). The second im-

ages the DMD plane onto a detector array, which is 1/3" 640×480 CCD with a pixel pitch/size of 7.4μm operating at 12-bit quantization. Given the object, the DMD and the sensor planes that optical conjugates, the block of modulated patterns are each mapped to a small number of contiguous detectors whose outputs are digitally combined to return a single measurement per block. Thus the blocks and their mapping to group of detectors essentially behave like parallel Single Pixel Cameras (SPC). In this architecture, the modulation patterns on the DMD are generated by unfolding each row of the projection matrix Φ, which are temporally scanned to acquire all the measurements, in parallel for all blocks.

It is important to highlight that one of the underlying challenges of implementing such a compressive imaging hardware is to ensure the correct calibration of the system, i.e. to minimize the deviation from the actual physical system measurement model to the idealized (and usually simplified) one. With this testbed, we have partly automated this arduous calibration process. We employ uniform white object and display a series of known transmission patterns on the DMD to localize and identify the pixels of the sensor associated with a particular block. We refer to [87] and [86] for more details about this calibration process. This calibration process thus dynamically discovers the distorted mapping between the DMD plane and the sensor plane and also measures the bias and scaling non-uniformities across the blocks. Finally, the target images are shown on a display facing the imaging arm and the system, which are pre-corrected for the gamma correction applied by the display panel.

2.8.2 Reconstruction Experiments

We use the set up described above to obtain CS measurements for each of the blocks (of size 33×33) in the scene. Operating at MR's of 0.1 and 0.04, we implement the 8-bit quantized versions of two kinds of measurement matrices:

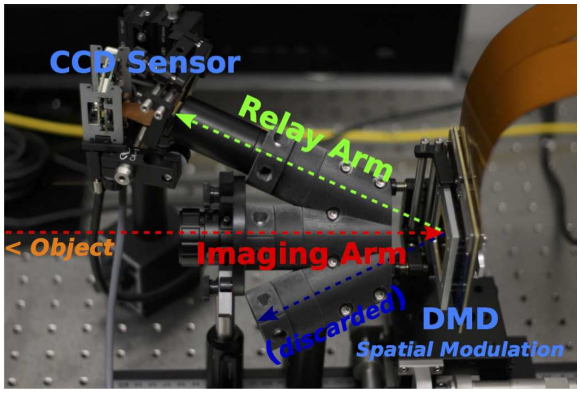


Figure 2.9: Compressive imager testbed layout with the imaging arm in the center, and 2 DMD imaging arms on the side.

1. Orthogonalized random Gaussian matrices used to train networks in Section 2.3.2 and Section 2.3.3

2. Learned measurement matrices, from Section 2.7 and Section 2.3.3. In this case, the measurement matrices are implemented by the camera hardware by programming the DMD and the outputs are the CS measurements that are fed into the $\mathbf{2^{nd}}$ of the trained networks directly.

Note that the CS measurements are input to the corresponding networks **trained on the simulated CS measurements**; no further training is done on the real data. Using these measurements we test four variants of ReconNet – two kinds of measurement matrices (Gaussian or learned) and two kinds of loss functions(Euclidean or Euclidean + Adversarial Loss). Figures 2.10a and 2.10b show the reconstruction results MR = 0.10 and 0.04 respectively. The first and second columns show the reconstructions obtained using D-AMP and TVAL3 which are iterative algorithms. The next five columns show the results obtained using the variants of ReconNet. It can be observed that our algorithm (columns 3,4 and 5) yields reconstructions that preserve more detail compared to the iterative approaches, thus demonstrating that our algorithm is robust to unseen sensor noise. For ReconNet, learning the mea-

surement matrix improves results significantly. Using adversarial loss in addition to Euclidean loss yields sharper results as in the case of simulated CS data. Also, the degradation in reconstruction quality when measurement rate is reduced is less in the case of ReconNet than the iterative algorithms.

2.9 Reducing Memory Footprint With Circulant Layers

A drawback of the architecture presented in Section 2.3.1 is the large size of the first fully connected (FC) layer that maps the CS measurements to a 2D array. As a numerical example, consider ReconNet operating at an MR $= 0.1$ with a block size of 33×33. Then, the FC layer contains $109 * 1089 = 118701$ weights. By comparison, the rest of the layers are all convolutional and contain a total of 22720 parameters. In this section, we discuss ways to reduce the complexity of this layer, which would be useful in systems with storage constraints, such as mobile platforms.

In inference applications using deep learning such as image recognition, CNN architectures usually employ one or two fully connected layers at the end to map the convolutional feature maps to probability distributions over the class labels. Depending on the size of the feature maps, the number of classes etc., these FC layers tend to be large (relative to the rest of the network). Recent research has shown that we can reduce the complexity of these layers from $O(d^2)$ to $O(d)$ without any loss in performance. One particular paper is that of Cheng et al [31] which replaces the fully connected layer – represented by a weight matrix without any constraints on the weights – with a circulant layer where the weight matrix is constrained to be circulant matrix. They proceed to show that in spite of a large reduction in the number of parameters, the performance of the network largely remains the same and in some cases, even performs better! They also discuss how to efficiently compute the output

of such a layer using FFTs. In this chapter, we propose this layer as an alternative for the first fully connected layer.

A circulant matrix $C \in \mathbb{R}^{d \times d}$ is completely defined by a vector $\mathbf{c} = (c_0, c_1, ..., c_{d-1})$ as follows:

$$\mathbf{C} = circ(\mathbf{c}) = \begin{bmatrix} c_0 & c_{d-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & \dots & c_3 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{d-1} & c_{d-2} & \dots & c_1 & c_0 \end{bmatrix}. \tag{2.8}$$

It can be shown that for an input $\mathbf{x} \in \mathbb{R}^d$, the output $\mathbf{y} \in \mathbb{R}^d$ of a circulant layer can be computed efficiently using

$$\mathbf{y} = \mathbf{C}\mathbf{x} = \mathbf{c} \circledast \mathbf{x} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{c}) \circ \mathcal{F}(\mathbf{x})), \tag{2.9}$$

where $\circledast$ represents circular convolution and $\circ$ is the element-wise multiplication operator. $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent Fourier and inverse Fourier transforms respectively. We have implemented this layer in TensorFlow which computes the gradients using automatic differentiation.

In our case, the input vector $\mathbf{x} \in \mathbb{R}^M$ has a dimension less than that of the output of the first layer which is a vector with dimension equal to the number of pixels (N) in the block. Thus, in order to use the circulant layer instead of an FC layer, we will append $N - M$ zeros to each input $\mathbf{x}$ and hence, $\mathbf{C} \in \mathbb{R}^{N \times N}$ and $\mathbf{c} \in \mathbb{R}^{\mathbb{N}}$. Therefore, the number of weights in the first layer of ReconNet can be reduced from MN to N by employing a circulant layer instead of an FC layer. For MR = 0.10, this corresponds to a 99.1% reduction in parameters for the first layer.

However, for higher measurement rates, this leads to significant under-fitting since the number of trainable parameters becomes small. We observed empirically that we can increase the reconstruction quality by using **multiple** circulant layers as the first

layer instead of just one. At the output of the first layer, we have multiple feature maps from the circulant layers which are combined into a single tensor. Thus, the convolutional layer that follows this layer must be modified. If the number of circulant layers is γ, then each filter in the following convolutional layer are of size $11 \times 11 \times \gamma$. This is only a modest increase in parameters for this layer compared to the $11 \times 11 \times 1$ filters which we would need in the case of an FC layer or a single circulant layer.

We evaluate this by training networks at four measurement rates using ReconNet (Euc) as the network architecture. We increased the number of circulant layers from 1 to a value γ such that the reduction in the parameters of the first layer is no less than 95% when compared to using a fully connected layer at the same measurement rate. The training and testing sets are same as in the previous sections. Table 2.6 shows the mean PSNR obtained for the test set using ReconNet using circulant layer instead of an FC layer. We observe that the reduction in PSNR is within 2 dB at most measurement rates even with 95% reduction in parameters of the first layer. Although this may be a significant decrease in performance in terms of PSNR, perceptually, the quality is remains competitive, as shown in Figure 2.11. The main difference is that the reconstructions with circulant layers have more artifacts near the edges of blocks. The reconstructions can be further improved by adding more circulant layer as shown in Table 2.6. Thus, the number of circulant layers serves as an interesting parameter with which we can choose the point of trade-off between memory requirements and reconstruction performance. In Section 2.10, we further show that circulant layers can be used for memory-efficient CS object-tracking.

2.10 Real-time High Level Vision Using Compressive Imagers

It is now clear that our CS reconstruction algorithm is non-iterative, real-time and capable of producing good quality reconstruction results, over a broad range

of measurement rates. In this section, we demonstrate that despite the expected degradation in PSNR as the measurement rate is decreased to an extremely low value of 0.01 (10 measurements for a 33×33 block), our algorithm still yields reconstructions where rich semantic content is still retained. As stated earlier, in many resource-constrained inference applications the goal is to acquire the least amount of data required to perform effective high-level image understanding.

To demonstrate how CS imaging can applied in such scenarios, we present an example proof of concept real-time high level vision application - object tracking. To this end, we simulate frame-wise video compressive imaging at measurement rates of 0.01 and 0.10 by obtaining block CS measurements of each frame on 15 publicly available videos [168] (BlurBody, BlurCar1, BlurCar2, BlurCar4, BlurFace, BlurOwl, Car2, CarDark, Dancer, Dancer2, Dudek, FaceOcc1, FaceOcc2, FleetFace, Girl2) used to benchmark tracking algorithms. Then, we perform object tracking on-the-fly as we recover the frames of the video using all the variants of ReconNet without the denoiser. For object tracking we use a state of the art algorithm based on kernelized correlation filters [66]. We call this pipeline, ReconNet+KCF. For comparison, we conduct tracking on original videos as well. We use the default values of the tracking algorithm in all cases. Figure 2.12 shows the average precision curve over the 15 videos, in which each datapoint indicates the mean percentage of frames that are tracked correctly for a given location error threshold. Using a location error threshold of 20 pixels, the average precision over 15 videos for variants of ReconNet+KCF at MR = 0.01 is between 68.14% and 77.46%. At MR = 0.10, we obtain impressive tracking performance between 79.49% and 84.89 % for different variants. By comparison, tracking on the original videos yields an average precision value of 84.9%. Learning the measurement matrix gives a significant boost of about 8 and 5 percentage points at MR = 0.01 and 0.10 respectively.

The effect of loss function is more nuanced. Euclidean + Adversarial loss seems to decrease tracking performance at MR = 0.10 with any measurement matrix and at MR = 0.01 with a Gaussian measurement matrix by about 3 percentage points over a large range of location error thresholds when compared to just Euclidean loss. However, we observe the opposite in the case of MR = 0.01 using a learned measurement matrix. Here, Euclidean + Adversarial loss outperforms Euclidean loss by about 3 percentage points. ReconNet + KCF operates at around 10 Frames per Second (FPS) for a video with frame size of 480×720 to as high as 56 FPS for a frame size of 240×320.

We also conducted tracking experiments using circulant layers (Section 2.9) instead of the first FC layer in the network for the case of Gaussian Φ and Euclidean loss. At both MR = 0.01 and 0.10, the circulant layers provide nearly the same tracking performance as their FC variants, especially at lower location error thresholds. This clearly demonstrates the effectiveness of circulant layers for high-level inference.

2.11 Conclusion

In this chapter we have described ReconNet – a non-iterative algorithm for CS image reconstruction based on CNNs. The advantages of this algorithm are two-fold – it can be easily implemented while making it 3 orders of magnitude faster than traditional iterative algorithms essentially making reconstruction real-time and it provides excellent reconstruction quality retaining rich semantic information over a large range of measurement rates. We have also discussed novel ways to improve the basic version of our algorithm. We have proposed learning the measurement matrix jointly with the reconstruction network as well as training with adversarial loss based on recently popular GANs. In both cases, we have shown significant improvements in reconstruction quality over a range of measurement rates. Using the ReconNet +

KCF pipeline, efficient real-time tracking is possible using CS measurements even at a very low measurement rate of 0.01. This also means that other high-level inference applications such as image recognition can be performed using a similar framework i.e., ReconNet + Recognition from CS measurements. We hope that this work will generate more interest in building practical real-world devices and applications for compressive imaging.

Image Name	Algorithm	MR = 0.25		MR = 0.10		MR = 0.04		MR = 0.01	
		w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D
Monarch	TVAL3 [103]	**27.77**	**27.77**	21.16	21.16	16.73	16.73	11.09	11.11
	NLR-CS [41]	25.91	26.06	14.59	14.67	11.62	11.97	6.38	6.71
	D-AMP [121]	26.39	26.55	19.00	19.00	14.57	14.57	6.20	6.20
	SDA [126]	23.54	23.32	20.95	21.04	18.09	18.19	15.31	15.38
	$\Phi^T\Phi\mathbf{x}$ + 2 ReconNet Units	22.36	22.86	20.66	21.15	17.75	17.96	15.31	15.41
	ReconNet (Euc) [96]	24.31	25.06	21.10	21.51	**18.19**	**18.32**	**15.39**	**15.49**
	ReconNet (Euc + Adv)	25.83	25.16	**21.74**	**21.94**	17.81	18.05	13.99	14.14
Cameraman	TVAL3	**25.69**	**25.70**	**21.91**	**21.92**	18.30	18.33	11.97	12.00
	NLR-CS	24.88	24.96	14.18	14.22	11.04	11.43	5.98	6.31
	D-AMP	24.41	24.54	20.35	20.35	15.11	15.11	5.64	5.64
	SDA	22.77	22.64	21.15	21.30	**19.32**	19.55	17.06	17.19
	$\Phi^T\Phi\mathbf{x}$ + 2 ReconNet Units	21.89	22.27	20.71	21.15	18.71	19.08	16.92	17.05
	ReconNet (Euc) [96]	23.15	23.59	21.28	21.66	19.26	**19.72**	**17.11**	**17.49**
	ReconNet (Euc + Adv)	25.11	25.20	21.94	22.18	19.58	19.95	17.09	17.37
Peppers	TVAL3	29.62	29.65	**22.64**	22.65	18.21	18.22	11.35	11.36
	NLR-CS	28.89	29.25	14.93	14.99	11.39	11.80	5.77	6.10
	D-AMP	**29.84**	**28.58**	21.39	21.37	16.13	16.46	5.79	5.85
	SDA	24.30	24.22	22.09	22.34	**19.63**	19.89	**16.93**	**17.02**
	$\Phi^T\Phi\mathbf{x}$ + 2 ReconNet Units	23.08	23.41	21.72	22.10	19.02	19.44	16.74	16.90
	ReconNet (Euc) [96]	24.77	25.16	22.15	**22.67**	19.56	**20.00**	16.82	16.96
	ReconNet (Euc + Adv)	27.90	27.90	23.68	24.09	19.84	20.29	16.93	17.16
Mean PSNR	TVAL3	27.84	27.87	22.84	22.86	18.39	18.40	11.31	11.34
	NLR-CS	28.05	**28.19**	14.19	14.22	10.58	10.98	5.30	5.62
	D-AMP	**28.17**	27.67	21.14	21.09	15.49	15.67	5.19	5.23
	SDA	24.72	24.55	22.43	22.68	19.96	20.21	**17.29**	17.40
	$\Phi^T\Phi\mathbf{x}$ + 2 ReconNet Units	23.73	24.32	22.28	22.86	19.55	19.96	17.24	17.38
	ReconNet (Euc) [96]	25.54	25.92	22.68	23.23	**19.99**	**20.44**	17.27	**17.55**
	ReconNet (Euc + Adv)	27.11	26.90	**23.22**	**23.48**	19.65	20.00	16.66	16.90

Table 2.1: PSNR values in dB for three test images as well as the mean PSNR values for the entire test set using different algorithms at different MRs with a Gaussian Φ. At low measurement rates of 0.1, 0.04 and 0.01, both variants of our algorithm yields superior quality reconstructions than the traditional iterative reconstruction algorithms. It is evident that the reconstructions are very stable for our algorithm.

Algorithm	MR=0.25	MR=0.10	MR=0.04	MR=0.01
TVAL3 (CPU)	2.943	3.223	3.467	7.790
NLR-CS (CPU)	314.852	305.703	300.666	314.176
D-AMP (CPU)	27.764	31.849	34.207	54.643
ReconNet (CPU)	0.5249	0.5258	0.5284	0.5193
ReconNet (GPU)	0.0213	0.0195	0.0192	0.0244
SDA (GPU)	0.0042	0.0029	0.0025	0.0045

Table 2.2: Time complexity (in seconds) of various algorithms (without BM3D) for reconstructing a single 256×256 image on both CPU (Xeon E5-1650) and GPU (GTX 980) Taking only about 0.02 seconds at any given MR, ReconNet can recover images from CS measurements in real-time.

MR	No. of D-AMP units			ReconNet
	1	5	10	
0.25	0.5582	0.7865	1.1584	0.0119
0.10	0.2803	0.4518	0.7290	0.0124
0.04	0.1762	0.3011	0.5357	0.0119
0.01	0.0930	0.0882	0.4521	0.0120

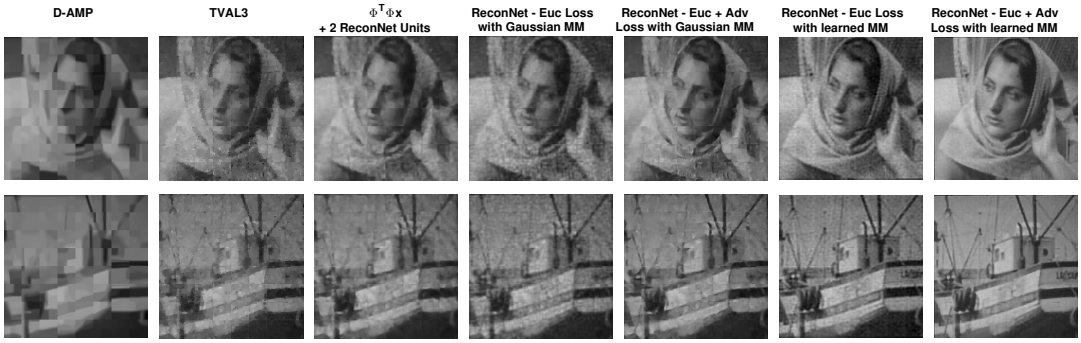
Table 2.3: Time in seconds to reconstruct a 256×256 image on a Titan X GPU. Clearly ReconNet is much faster than LDAMP at all MRs. Note that BM3D is not used for ReconNet.

New Φ MR	**0.1**	**0.08**	**0.04**	**0.01**
Base network MR	0.25	0.1	0.1	0.25
Mean PSNR (dB)	21.73	20.99	19.66	16.60
Training Time (seconds)	2	2	2	2

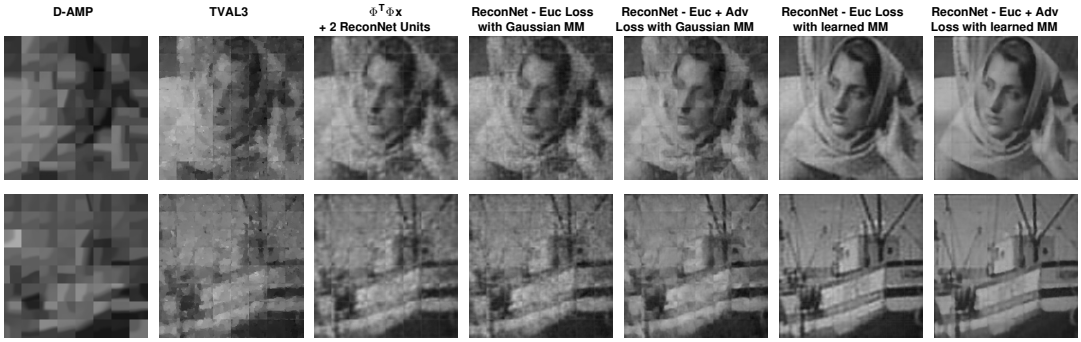
Table 2.4: Networks for a new Φ can be obtained by training only the FC layer of the base network at minimal computational overhead, while maintaining comparable PSNRs.

Loss function and measurement matrix type	MR = 0.25		MR = 0.10		MR = 0.04		MR = 0.01	
	w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D	w/o BM3D	w/ BM3D
Euclidean with Gaussian Φ [96]	25.54	25.92	22.68	23.23	19.99	20.44	17.27	17.55
Euclidean + adversarial with Gaussian Φ	27.11	26.90	23.22	23.48	19.65	20.00	16.66	16.90
Euclidean with learned Φ	26.59	26.44	25.51	25.73	23.14	23.51	19.44	19.74
Euclidean + adversarial with learned Φ	30.53	29.42	26.47	25.94	22.98	23.00	19.06	19.31

Table 2.5: This table shows the mean reconstruction PSNR on the test set for different variations of ReconNet i.e., with different loss functions and measurement matrices (Φ). We see that the PSNR improves significantly at all measurement rates when a the measurement matrix is changed from a Gaussian matrix to a jointly learned one (Section 2.7). We also observe that at higher measurement rates of 0.25 and 0.10, using adding adversarial loss to Euclidean loss (2.7) while training improves PSNR by about 1 dB in the case of a Gaussian Φ and about 3 dB when Φ is learned.



(a) MR = 0.10



(b) MR = 0.04

Figure 2.10: The figure shows reconstruction results for 2 images whose measurements are collected using our block SPC. The results are for two measurement rates (a) 0.10 and (b) 0.04. The iterative methods in first and second columns use a Gaussian Φ. The next four columns shows the reconstructions obtained using different variants of ReconNet based on the loss function used in training and whether or not Φ was learned. "MM" and Φ both stand for measurement matrix, "Euc Loss" stands for Euclidean loss and "Adv Loss" stands for adversarial loss. Clearly, all variants of ReconNet with a Gaussian Φ (columns 3,4,5) perform better than both TVAL3 and D-AMP. Both learning the measurement matrix and using adversarial loss make reconstructions sharper and less noisy.

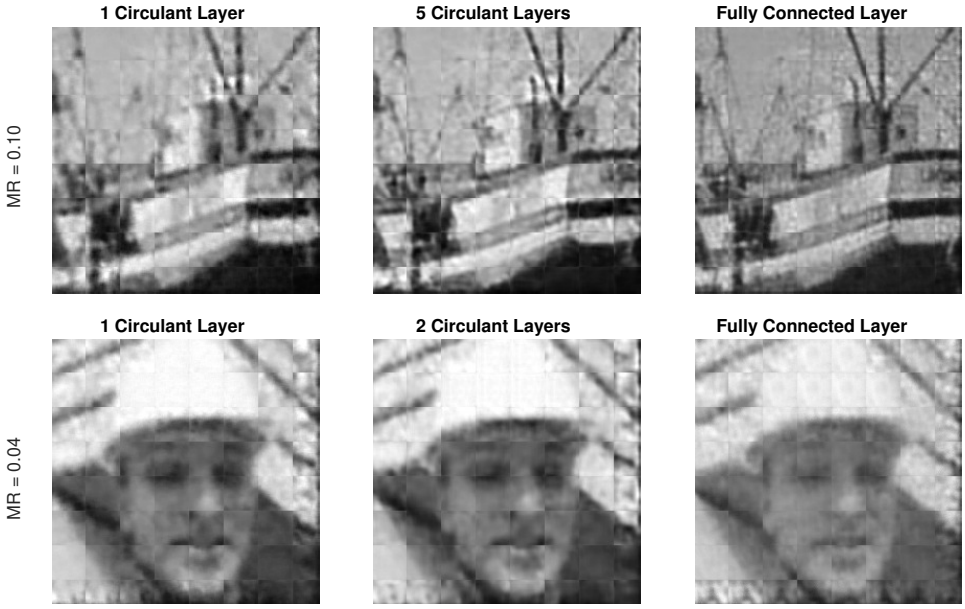
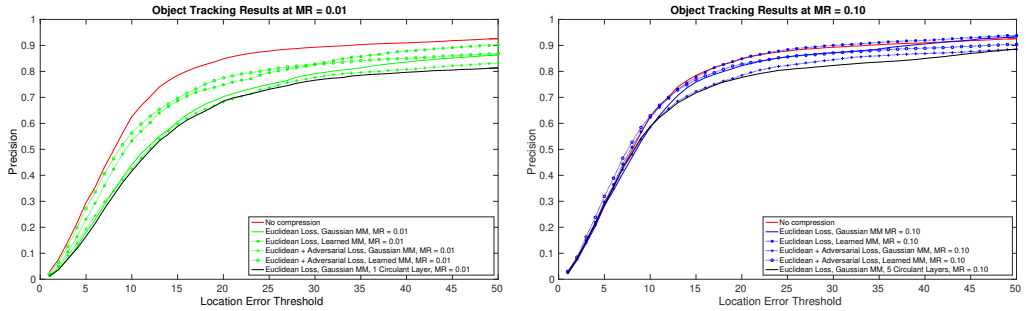


Figure 2.11: The figures show reconstruction results for the "Boats" and "Foreman" image for circulant layers compared to fully connected layer. The images demonstrate that circulant layers produce good quality images and at the same time, provide huge savings in terms of storage.



(a) MR = 0.01 (b) MR = 0.10

Figure 2.12: The figure shows the variation of average precision with location error threshold for ReconNet+KCF and original videos. Clearly, semantic content required for object tracking is retained even in reconstructions at MR = 0.01

MR	No. of Circulant Layers	% Reduction in Parameters in the First Layer	Mean PSNR using circulant layers		Mean PSNR using an FC layer	
			without BM3D	with BM3D	without BM3D	with BM3D
0.25	1	99.63	20.92	21.31	25.54	25.92
	13	95.22	23.52	23.89		
0.10	1	99.08	20.3	20.71	22.68	23.23
	5	95.41	21.24	21.65		
	15	86.24	22.07	22.57		
0.04	1	97.67	18.83	19.18	19.99	20.44
	2	95.34	19.11	19.48		
	5	88.37	19.29	19.69		
0.01	1	90	16.51	16.77	17.27	17.55

Table 2.6: Comparison of mean PSNR (in dB) of reconstruction of the test set using a one or more circulant layers instead of a fully connected layer as the first layer of ReconNet. We see that the reduction in PSNR using circulant layers is within 2 dB even with 95% reduction in parameters in the first layer. (The entries in the last two columns are from Table 2.1)

Chapter 3

IMPOSING DESIGN CONSTRAINTS ON LEARNED MEASUREMENT OPERATORS FOR RECONNET

3.1 Introduction

Advances in computational imaging have led to cameras tailored to application domains. Indeed, different constraints arise depending on the task at hand which necessitate design and exploration of novel camera architectures. For example, mobile devices have energy constraints, imaging in short-wave infrared (SWIR) domain in high resolution is very expensive with conventional sensors, and magnetic resonance imaging is very slow when sampled at the Nyquist rate. We can overcome these issues using spatial multiplexing cameras, where, instead of recording pixel intensities, projections of the scene onto a chosen basis subset are computed by the camera.

As mentioned in Chapter 2, learning-based methods using neural networks have recently made significant progress in producing excellent image reconstruction in real-time. These methods are non-iterative in nature and require a simple feed-forward operation through the trained network. These networks are trained using Gaussian measurements as in [126, 96], and further improved by jointly learning the measurement operator as in [126, 107, 4].

If high-level inference is the eventual goal of image acquisition, then it can be performed directly in the compressed domain, bypassing expensive image reconstruction. Theoretical and experimental evidence for this idea has been provided by Calderbank et al. [22] and Davenport et al. [116] for linear classifiers. Deep learning can make the process of direct inference on compressive measurements much more effective as

shown by Lohit et al. [108] for Gaussian measurements, and improved by learning the measurement matrix jointly with the image recognition network as shown by Adler et al. [6]. Below, we mention two interesting design constraints on the learned measurement operator that we believe can lead to more efficient compressive imaging systems.

In all the deep learning based works mentioned above,

(1) the measurement rate is a user-defined quantity and at the time of training, it may not be clear what it should be. In this chapter we propose a modified optimization problem which encodes the trade-off between the number of measurements and the network performance. This trade-off is expressed in terms of a parameter λ that may represent the cost per measurement in terms of sensing time, bandwidth, energy etc. We provide a solution to the problem of finding the optimal number of measurements by minimizing the rank of the measurement operator Φ. To this end, we propose two different regularizers on the values of Φ – the nuclear norm and the $l_{2,1}$ norm. These regularizers can be readily integrated into the backpropagation framework. We demonstrate the effectiveness of the proposed framework for two different applications – image reconstruction and object recognition.

(2) for a given trained network, the MR is defined prior to training, and thus, cannot be used at different MRs at test time. **In this chapter, we extend the ideas of deep learning based data-driven CS reconstruction and inference to applications where it is necessary and useful to allow for the MR to vary at test time, keeping the network fixed. We call such a system – Rate-Independent CS**, and it refers to the combination of the measurement operator, usually implemented using a single pixel camera (SPC) [116], and the reconstruction/inference network that follows it. To this end, we propose learning "super-operators" which satisfy what we called subset-validity constraints. This is

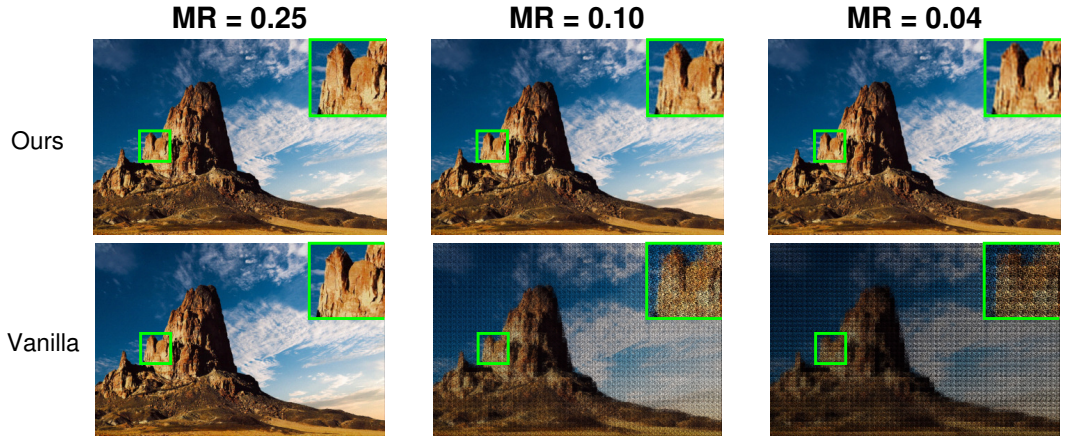


Figure 3.1: Sample visualizations comparing our algorithm Rate-Independent CS which is stable over a range measurement rates $[0.04 - 0.25]$, with vanilla training algorithm that is trained for a single MR=0.25 (272 measurements) and tested over the chosen MR range. Clearly, our algorithm produces better quality images over the entire range of MRs.

illustrated in Figure 3.6. As applications of Rate-Independent CS, we envision power and storage-constrained mobile systems, where MR is a function of available energy, memory, or time-varying bandwidth constraints or even content-based dynamically varying MR. This also means that only a single network needs to be stored in the system and no access to training data is necessary. Hence, our approach is memory efficient, compared to earlier purely data-driven deep-learning-for-CS methods. For the task of image reconstruction, compared to earlier approaches like [107], we get huge improvements in PSNR of up to 15 dB, for the case wherein a network trained at a MR $= 0.25$ is used to reconstruct measurements acquired at MR $= 0.04$. Some examples are shown in Figure 3.1.

3.2 Related Work

Image reconstruction: An important problem in CS is to recover the original signal $\mathbf{x} \in \mathbb{R}^n$ from the compressive measurements $\mathbf{y} \in \mathbb{R}^m$. The earlier CS methods posed this problem as a constrained optimization problem using prior models on the set of natural images as the constraint set, such as sparsity in a transform domain [128, 21, 103] and more complicated models such as in [12, 41, 122]. These algorithms are iterative in nature, computationally expensive and do not perform well at low measurement rates (< 0.1) [96]. Over the past few years, instead of hand-crafted priors, deep learning-based approaches have been shown to be superior to traditional methods. Deep learning methods can further be divided into two categories: (1) deep nets are used as a step of *unrolled* iterative algorithms [58, 82, 15, 123, 28, 178], (2) purely data-driven i.e., deep nets map the $\mathbf{y}$ directly to $\mathbf{x}$ [126, 97, 173]. We focus on the latter approach in this chapter as it is currently not clear how the unrolled methods can be used to learn the measurement matrix jointly. Mousavi et al.[126] proposed the first non-iterative deep learning networks for CS reconstruction. This model is based on stacked denoising auto-encoders. Kulkarni et al. [97] propose a CNN based model called ReconNet. Lohit et al.[109] extend ReconNet where the measurement matrix is jointly learned with the reconstruction network. Similar ideas have also been applied for video compressive sensing [76, 75].

Image recognition: Calderbank et al. [22] have shown theoretically that it is possible to do the inference task like object recognition directly from CS measurement without reconstructing the image. Davenport et al. [116] use the CS version of match filtering called as 'smashed filter' for image recognition. Lohit et al. [108] propose deep learning for object classification in the compressed domain. This model uses a

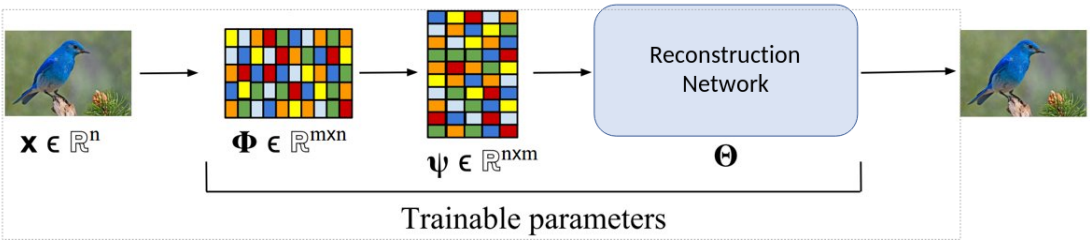


Figure 3.2: The figure shows the general framework for jointly learning the measurement matrix and the reconstruction network. The measurement matrix is denoted by $\Phi \in \mathbb{R}^{m \times n}$ and forms the first fully connected (FC) layer in the network. The output of this layer are the compressive measurements of the scene. The second layer $\Psi \in \mathbb{R}^{n \times m}$ maps the compressive measurements back into the 2D space.

fully-connected layer to project the CS measurement to pixel space and the rest of model is a typical CNN based classification network, further improved by Adler et al. [4] by jointly learning the measurement matrix.

3.2.1 Contributions

1. We propose addition of rank regularizers to compressive reconstruction and compressive inference network that can take into account the cost of sensing and model the trade-off between reconstruction performance and number of measurements.

2. We propose a novel optimization problem in the context of compressive imaging to learn a single measurement operator and reconstruction/inference algorithm jointly. This is done such that the MR is reduced by simply using a smaller subsets of the rows of the measurement matrix (satisfying subset-validity constraints), and the reconstruction/inference algorithm remains fixed. Based on insights from existing methods, we design a three-stage training algorithm to solve this problem.

3. Using widely-employed network architectures for purely data-driven CS reconstruction, we demonstrate experimentally that reconstruction networks trained with the

proposed algorithms allow for learning reduced-rank and rate-independent measurement operators that yield excellent reconstruction performance.

4. As a proof of concept, we describe object tracking where MR is varied dynamically depending on image content or a pre-determined adaptation scheme. We use a single network to reconstruct the frames at different MRs and show that the object tracking performance remains competitive.

5. Finally, we show that the algorithms are general enough that they allow for learning reduced-rank and rate-independent inference networks for tasks like image recognition rather than reconstruction. Experiments on MNIST and CIFAR-10 show the wide applicability of our algorithm.

3.3 Learning a Low-Rank Φ: LowRankCS

We will first describe the framework for learning task-specific measurement operators. We note that while the reconstruction/inference network architecture will depend on the task and other design choices, the general framework remains the same as shown in Figure 3.6. Once the network is trained, the first FC layer, denoted by Φ, acts as the measurement operator that is implemented in the compressive imager, whose outputs are the compressive measurements. The second FC layer, Ψ, maps the compressive measurements into the original dimension n and is reshaped into a 2D array to be compatible with the reconstruction/inference network architecture. The remaining parameters in the network are denoted by Θ. That is, the training problem can be described as

$$\min_{\Phi,\Psi,\Theta} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{d}_i), \tag{P1}$$

where $\mathbf{d}_i$ is the desired output. For image reconstruction $\mathbf{d}_i = \mathbf{x}_i$ and $L(.)$ may be chosen as the Euclidean loss. In the case of the image recognition, $\mathbf{d}_i = \mathbf{p}_i$, where $\mathbf{p}_i$ is a one-hot vector with the position of the one indicating the ground-truth label and the $L(.)$ is the cross-entropy loss.

The main advantage of CS is that one can achieve a high performance even with a small number of measurements. We want to have as few measurements as possible while maintaining a high level of performance. However, in the set-up described in the above paragraph, the number of measurements is a user-defined quantity, determined by the number of rows in Φ, m and may not be known a priori. Instead, in this chapter, we formulate the following alternative optimization problem which can express the trade-off between the number of measurements and the performance of the reconstruction/inference algorithm. We refer to this framework as **LowRankCS**. We assume that there is a cost per measurement that is given by λ and that the number of measurements cannot exceed m. We can write the optimization problem as

$$\min_{\Phi,\Psi,\Theta} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{d}_i) + \lambda \, \text{rank}(\Phi). \tag{P2}$$

The constraint on the maximum number of measurements is satisfied by setting the first fully connected layer (which implements the measurement operator Φ) to be of size $m \times n$. In the initial framework without rank minimization, the rows of Φ, with m rows, are implemented by the camera. Our goal is not just to reduce the rank of Φ, but the number of measurements that need to be acquired. Thus, we cannot directly use the Φ from Problem P2. We use the singular value decomposition (SVD) of Φ instead: $\Phi = U\Sigma V^T$. Say, the rank of Φ returned by Problem P2 is some $k < m$. Then, we use the top k columns of U, top k singular values of Σ and top k rows of V^T, denoted by U_k, Σ_k and V_k^T respectively, to obtain $\Phi = \Phi_k = U_k \Sigma_k V_k^T$. Now, instead of the rows of Φ being implemented by the camera, the k rows of V_k^T are implemented

by the camera. Thus, the new signal acquisition process is given by $\mathbf{z} = V_k^T \mathbf{x}$ and then $\mathbf{y}$ is computed using $\mathbf{y} = \Phi\mathbf{x} = \Phi_k\mathbf{x} = U_k\Sigma_k\mathbf{z}$.

The rank, being a discrete quantity cannot be directly minimized using gradient-based optimization and rank minimization is known to be NP-hard [138]. Therefore, P2 cannot be optimized directly and we turn to regularizers on the values of Φ that may help us at least solve an approximate version of P2. Then, P2 is transformed to

$$\min_{\Phi,\Psi,\Theta} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{d}_i) + \lambda \, \text{Reg}(\Phi). \tag{P3}$$

Next, we describe two different regularization functions used and their sub-gradients, used for backpropagation.

Nuclear norm [49] : There are several works in the literature that employ the nuclear norm for rank minimization problems and have some recovery guarantees in certain settings e.g. rank minimization with linear equality constraints [138] and matrix completion problems [20, 23]. A related paper by Hegde et al. [64] proposes nuclear norm minimization for dimensionality reduction and metric learning, and can be used to design task-specific measurement matrices. However, the reconstruction/inference algorithm is designed independently unlike the end-to-end framework presented here. Nuclear norm regularization has also been applied in the deep learning setting for multi-task learning [172]. We employ the nuclear norm as a regularization function in P3. The nuclear norm of Φ, denoted $||\Phi||_*$, is given by $||\Phi||_* = \sum_{i=1}^{m} \sigma_i = trace(\sqrt{\Phi^T\Phi})$. We still need to provide the gradient function for the backpropagation algorithm. However, the nuclear norm is not differentiable everywhere. As it is a convex function, we can employ a sub-gradient such as the following [165]: $\partial||\Phi||_* = UV^T$.

$\ell_{2,1}$ **norm** : The $\ell_{2,1}$ regularizer has been used in applications to promote structured sparsity such as multi-task feature selection [131, 105, 129]. For our case, the $\ell_{2,1}$ regularizer is given by $||\Phi||_{2,1} = \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{n} \Phi_{ij}^2}$. This regularizer encourages rows of Φ to be zero. Clearly, all zero rows of Φ can be removed, thus reducing the rank of the measurement operator. This function is not differentiable when the row of the matrix is zero. Hence, we use its sub-gradient with respect to the $(i,j)^{th}$ element of Φ, given by $(\partial||\Phi||_{2,1})_{ij} = \frac{\Phi_{ij}}{||\Phi_{.,j}||_2}$, if $\Phi_{.,j} \neq \mathbf{0}$ and 0 otherwise.

3.3.1 Experimental Results on LowRankCS

In this section, we describe the experimental results for the LowRankCS framework, where we learn the optimal number of measurements through rank minimization. For each of the proposed regularizers, we train networks for varying values of λ and conduct the experiments at two different measurement rates MR = 0.25 and 0.10 and use the test set to demonstrate that the Φ thus obtained is indeed low-rank and that its rank depends on the value of λ and the regularizer. Once the training is complete, we determine the approximate rank of the learned measurement matrix as follows. We compute the performance of the algorithm on the test set for increasingly low rank approximations of Φ, using SVD, given by $\{\Phi_k = U_k \Sigma_k V_k^T, k = 1, 2, \ldots, m\}$, with $\Phi_m = \Phi$. Rank of Φ is then determined to be the value of k for which the performance on the test set suddenly drops.

Experiments on image reconstruction

Here, we describe the n/w, the training and testing protocols used for the image reconstruction problem: given $\mathbf{y}$, return $\mathbf{x}$, where $\mathbf{y} = \Phi\mathbf{x}$. It is important to note that Φ is also learned (see Figure 3.6). The n/w architecture we employ in this chapter is the extended ReconNet architecture described in [107] for jointly learning

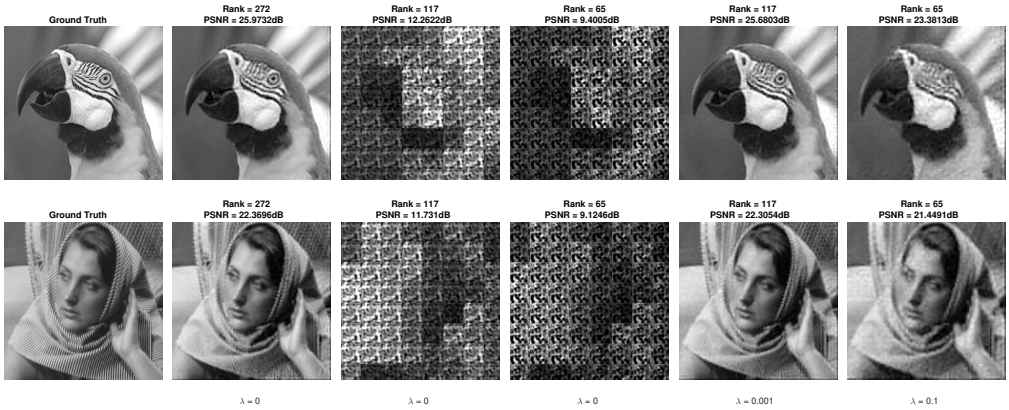


Figure 3.3: The figures show reconstructions for two test images using no regularization ($\lambda = 0$) and with nuclear norm regularizer with different values of λ. Note that, without regularization, low-rank approximation of Φ results in poor performance.

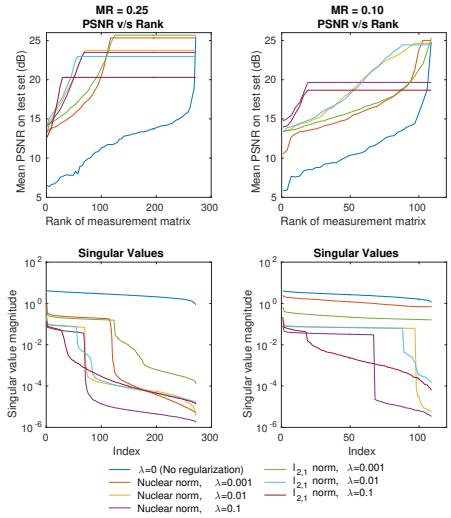


Figure 3.4: The plots show LowRankCS results for image reconstruction application. The top row shows the mean test PSNR for different low-rank approximations of Φ_k's. The bottom row shows the singular values of the various Φ's obtained. We see that the rank of Φ is lower with regularization and with higher value of λ. Best viewed in color.

λ	MR = 0.25 (m=272)		MR = 0.10 (m=109)	
	Rank	PSNR	Rank	PSNR
0	272	25.50	109	24.82
10^{-3}	117	25.33	103	25.01
10^{-2}	69	23.61	97	24.63
10^{-1}	65	23.00	19	19.64

Table 3.1: Results for LowRankCS for the image reconstruction problem showing that we can design low rank measurement operators using the nuclear norm regularizer. *Rank* denotes the rank of the learned Φ and *PSNR* denotes the mean reconstruction PSNR (dB) over the test set. Note that the sensing and reconstruction are performed block-wise with a block size of $33 \times 33, n = 1089$.

the measurement operator and the reconstruction algorithm, a series of convolutional layers. Note that the n/w is for block-wise reconstruction which means that the sensing and reconstruction process is for non-overlapping blocks of the image, not the entire image. The first layer is an FC layer of size $m \times n$ which serves as the measurement operator Φ. The second layer, Ψ, is also an FC layer of size $n \times m$ which converts the output of the first layer into the same dimension as the original signal and is then reshaped to form a 2D array of the same dimensions as the image. This is then followed by two ReconNet units described in [96], consisting of six convolutional layers with ReLU activation. The loss function consists of two terms as shown in P3. The first term for this case is simply the Euclidean loss between the desired output $\mathbf{x}$ (the original image itself) and the estimated reconstruction $\hat{\mathbf{x}}$. We perform experiments with both regularizers described in Section 3.3 with different values of $\lambda = 0, 10^{-1}, 10^{-2}, 10^{-3}$.

We train all networks for 5×10^5 iterations with the Adam optimizer [92] with an initial learning rate of 10^{-4}. The training set and testing are identical to the ones used in [96]. The training set consists of 91 grayscale natural images. About 24000 image blocks of size 33×33 are constructed from these images and serve as both the inputs as well as the desired output for the network. Thus $n = 1089(33 \times 33)$. For the test set, set of 11 standard test images – Barbara, Boats, Cameraman, Fingerprint, Flintstones, House, Lena, Foreman, Monarch, Parrots and Peppers – is employed.

The results are shown in Figure 3.4 and Table 3.1. Compared to the case with $\lambda = 0$, PSNR curves remain flat for a large range before falling. This indicates that the learned Φ is indeed approximately low-rank. This can also be inferred from the plots of the singular values (in log scale) which show significant decay compared to the case without any regularization. These remarks are true for both the nuclear norm and $\ell_{2,1}$-norm regularizers. Both regularizers yield similar results in terms of test PSNR. Note that the PSNRs obtained with regularization for m equal to the ranks determined by the algorithm are approximately equal to the PSNRs obtained without regularization for the same values of m. But our algorithm can determine the optimal rank automatically, given the cost λ. If we do not know the desired value of λ, but only the desired PSNR, we can envision a alternating optimization strategy for updating the value of λ until the desired PSNR is achieved. Interestingly, we see that for small values of $\lambda = 10^{-3}$, even though the $rank(\Phi) \approx m$, the mean PSNR values on the test set are higher indicating that they are also effective at reducing overfitting. Figure 3.3 shows the reconstructions for two test images demonstrating that nuclear norm regularization is effective for designing low rank Φ.

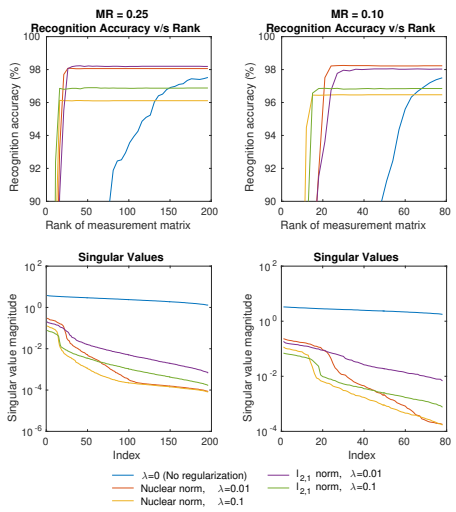


Figure 3.5: The plots show LowRankCS results for the image recognition problem on the MNIST database. The top row shows the test accuracy for different low-rank approximations of Φ as the measurement matrix. Bottom row shows the singular values of the various Φ_k's obtained. We clearly see that the rank of Φ is lower with regularization and with higher value of the cost parameter, λ. Best viewed in color.

Experiments on image recognition

Here, we learn low-rank measurement operators for MNIST digit recognition to illustrate that LowRankCS is also applicable to inference algorithms. The network is a modified version of the LeNet-5 architecture [101] with 3 convolutional layers, two FC layers and a softmax layer. We add two FC layers at the input of the network, of which the first serves as Φ and the second FC layer is the matrix Ψ. All networks are trained for 5×10^4 iterations using the Adam optimizer [92] with an initial learning rate of 10^{-4}. The dataset consists of 50000 training and 10000 testing grayscale images of hand-written digits of size 28×28. Thus $n = 784(28 \times 28)$.

The results are shown in Figure 3.5. As desired, both regularizers return Φ of low rank and the rank decreases with an increase in λ, while maintaining high per-

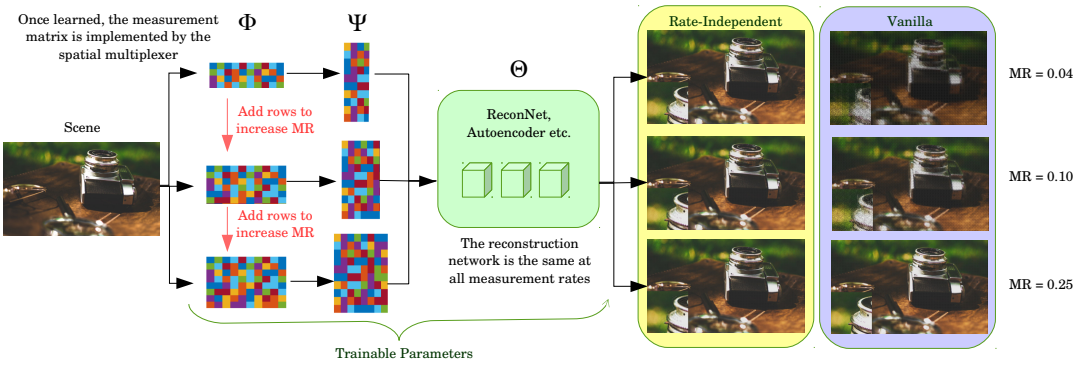


Figure 3.6: The figure shows Rate-Independent CS for image reconstruction. The measurement matrix, $\Phi \in \mathbb{R}^{m \times n}$ forms the first fully-connected layer in the network. After training, Φ is implemented in the SPC, and the output of this layer are the compressive measurements of the scene. The second layer $\Psi \in \mathbb{R}^{n \times m}$ maps the compressive measurements back into the 2D space. The rest of the reconstruction network (ReconNet, autoencoder, DR^2-Net [173] etc.) is represented by Θ. For image recognition, we use an inference network instead.

formance. Note that the accuracies obtained with regularization for m equal to the ranks determined by the algorithm are approximately equal to the accuracies obtained without regularization for the same values of m. But our algorithm can determine the rank automatically, given the cost λ. We can also see that the singular values decay much faster than in the case with $\lambda = 0$ (no regularization). For small values of λ, regularization leads to better performance in many cases, which is an added bonus.

3.3.2 Super-operators and Rate-Independent CS

This section describes the main contribution of the chapter. As mentioned earlier, it is of practical value if we can train the combination of the measurement matrix Φ and a single reconstruction/inference network such that the system can operate at when some rows of Φ are dropped, without having to modify rest of the network. To

this end, we propose a new training algorithm that makes the system performance ***stable*** with respect to a chosen range of measurement rates (MR). By this we mean the following: For a given range of MRs, we want the performance of the algorithm to be highest at the upper limit of the range and decrease slowly as the MR is decreased, such that the performance at any particular MR in the range is approximately equal to that of a system that is trained for a single MR. For convenience, we call such a system Rate-Independent CS, as the system is required to work at all rates. This is an interesting design constraint that has not been considered in the literature of either compressive sensing or deep learning.

We write the problem under consideration formally as follows by defining what we call subset-validity constraints. Denoting the measurement operator, the back projection operator and the reconstruction/inference function by Φ, Ψ and $f(\Theta, \cdot)$, we want to solve the joint optimization problem

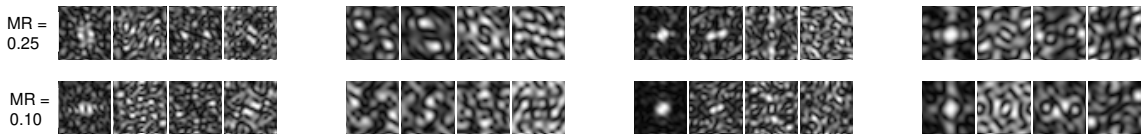
$$\begin{aligned} \min_{\Phi_r, \Psi_r, \Theta} & \frac{1}{2}||\mathbf{x} - f(\Theta, \Psi_r \Phi_r \mathbf{x})||_2^2 \\ \text{s.t.} \quad & \Phi_{r+1} = [\Phi_r, \Phi(r+1, :)] \\ & \Psi_{r+1} = [\Psi_r, \Psi(:, r+1)], \forall r = k, \ldots, m. \end{aligned} \tag{3.1}$$

$f(\Theta, \cdot)$ is chosen to be a neural network with a predetermined architecture and Θ are its learnable parameters. In the case of reconstruction, $f(\Theta, \cdot)$ is itself trained to approximate another optimization problem which is to solve the underdetermined linear system

$$f(\Theta, \Psi_r \Phi_r \mathbf{x}) \approx \arg\min_{\mathbf{x} \in \mathcal{C}} ||\mathbf{y} - \Phi \mathbf{x}||, \tag{3.2}$$

where $\mathcal{C}$ is the set of real images. In the case of inference such as image recognition, $f(\Theta, \cdot)$ is trained to perform the required task. That is, we want to design

a **single** measurement matrix $\Phi \in \mathbb{R}^{m \times n}$ such that each contiguous subset of the rows of Φ denoted by $\Phi_r = \Phi(1:r,:)$ is a valid measurement matrix for the **same** reconstruction/inference network, $f(\cdot)$. These are the subset-validity constraints and we refer to a Φ that satisfies these constraints as a super-operator. This overcomes a major drawback of previous purely data-driven algorithms and makes the system more efficient as only a single Φ needs to be stored. We will now describe a simple algorithm, inspired by key insights, to solve the above optimization.



(a) conv 1 filters (b) conv 3 filters (c) conv 4 filters (d) conv 6 filters

Figure 3.7: Visualizing filters of ReconNet [107] in the frequency domain for MR = 0.25 and 0.10. We clearly see several similarities between the filters at the two MRs. This means that the filters can be used across MRs.

Motivation for three-stage training process

We will first describe the motivation through which the algorithm can be derived. We use the extended ReconNet architecture [107], for insight into designing Rate-Independent CS. We first trained ReconNet at two different MRs = 0.25 and 0.1. Then, we visualized the filters in the convolutional layers. Although the filters are not exactly the same, one can immediately observe remarkable similarities in the filter structures. When observed in the frequency domain, both sets of filters have a "speckle-field" structure and more interestingly, even more similarities emerge. Figure 3.7 shows some pairs of filters, from every layer, at the two MRs that are very similar to each other. This may be because of the fact that output generated at the end of

the second FC layer are image-like and have spatial correlated structure (observed by 107) at both MRs, just differing in quality. Thus, similar convolutional operations are required to generate the final high-quality output. This leads to our idea that convolutional filters from one MR can be reused over the MR range of $[k, m]$. We later show that this is also true for autoencoders, not just CNNs.

Once the measurements are obtained from the spatial multiplexer (the first layers, Φ, in the network), the second layer maps it back into a 2D array to obtain a pseudo-image. In our experiments, the second layer Ψ is constrained to be $\Phi^\dagger$, the pseudoinverse of Φ given by $\Phi^T(\Phi\Phi^T)^{-1}$. This reduces the total number of parameters in the network by a large amount, especially for ReconNet-like architecture with convolutional layers, thus reducing the chance of overfitting. Now, we describe the three-stage training algorithm required to ensure that the network obeys subset-validity constraints of Rate-Independent CS. As the convolutional filters are fixed for the entire range of MRs, the only difference in network architecture are the first (Φ) and second (Ψ) FC layers. We will denote the parameters in the system, except Φ and Ψ, by Θ.

Stage 1: In the first stage, we train the convolutional layers (or the later FC layers in the case of autoencoders), for which we train the entire network at the base measurement rate of MR $= \frac{m}{n}$. That is, Φ is set to be of size $m \times n$. This ensures that the convolutional layers are most suited for the upper limit of the MR range, thus leading to the highest performance of the network at MR $= \frac{m}{n}$, as required. We call this the base network.

Stage 2: In the second stage, we freeze all the parameters Θ. We set the size of 1^{st} FC layers to be of $k \times n$ (thus, size of 2^{nd} FC layer is $n \times k$), and optimize over only these parameters. This ensures that the network performs well at the MR $= \frac{k}{n}$.

Stage 3: In the third stage, we add a single row at a time to the 1st FC layer and optimize over only the newly added variables. All the other variables, the remaining rows of Φ, and Θ, are held constant. **Thus, the output of the three-stage training algorithm is a super-operator Φ such that any subset of its rows $\Phi(1:r,:), k \leq r \leq m$, is a valid measurement matrix for the reconstruction/inference network defined by $\Psi(:, 1:r)$ and Θ.** The proposed training algorithm is summarized in Algorithm 1. The general framework for jointly learning Φ and the reconstruction/inference network is illustrated in Figure 3.6.

Comparison with a random Gaussian Φ

Earlier works in literature such as 126 and 107 have convincingly demonstrated that for the same network architecture, learning Φ jointly with the reconstruction algorithm results in substantially improved PSNRs ($\sim$3dB) over random Φ. In our problem, we have made similar observations: that using a random Φ performs worse than learning the Φ, especially when one starts dropping rows. We have not included this in the chapter, as it is now commonly known.

Comparison with unrolled iterative algorithms

There have been several algorithms such as LDAMP [123] and ISTA-Net [178] that combine learning based methods with earlier iterative approaches. These algorithms have provided excellent results in the case of a predefined measurement matrix such as a Gaussian matrix for which the network can be readily used at different measurement rates. However, it is not clear how to employ these networks to learn the measurement matrix jointly with the reconstruction. Our experiment with ISTA-Net for this purpose failed: the learning was erratic and did not converge. As such, we compare the proposed method with purely data-driven networks. It is also worth

Algorithm 1 Training Algorithm for Rate-Independent CS.

Input: $k \longleftarrow$ Min. #rows of Φ, $m \longleftarrow$ Max. #rows of Φ and other hyperparameters

Output: Φ_{new}, $\Psi_{new} = \Phi_{new}^{\dagger}$ and Θ, the remaining parameters in the reconstruction/inference network

Initialize network with size of 1st FC layer = size(Φ) = $m \times n$

Stage 1

for iter = 1 **to** max_iters_1 **do**

Optimize over Φ and Θ

end for

Stage 2

$\Phi_k \longleftarrow \Phi(1:k,:); \Psi_k = \Phi_k^{\dagger}$ Replace Φ with Φ_k in the network

for iter = 1 **to** max_iters_2 **do**

Optimize over Φ_k, holding Θ constant

end for

Stage 3

$\Phi_{new} \longleftarrow \Phi_k; \Psi_{new} = \Phi_{new}^{\dagger}$

for $r = k+1$ **to** m **do**

for iter = 1 **to** iters_per_row **do**

$\Phi_{new} \longleftarrow [\Phi_{new}; \Phi(r,:)]; \Psi_{new} = \Phi_{new}^{\dagger}$

Optimize over $\Phi(r,:)$, holding $\Phi(1:r-1,:)$ and Θ constant

end for

end for

noting that, at lower measurement rates for a given performance level, networks like LDAMP and ISTA-Net are several times slower and bigger than the purely data-driven counterparts.

3.3.3 Experimental Results

In this section, we describe experimental results for the Rate-Independent CS framework (Section 3.3.2, Algorithm 1), where we learn Φ and the reconstruction/inference network that can be operated over a range of MRs. We compare our results with the **"vanilla" framework** [107, 5], where the system is trained for a single MR. We carry out experiments for both image reconstruction as well image recognition. As mentioned, we need to input the values of the operating MR range $[k, m]$ to the algorithm, which are the minimum and maximum values of MR for which the system can operate. We demonstrate that our algorithm is effective as follows. Given a trained network (Φ, Ψ, Θ), we start with $\Phi(1, :)$ as the measurement matrix and $\Psi(:, 1)$ as the second FC layer. We observe the performance of the system on the test set. Then, we add one row at a time to the measurement matrix, and one column at a time to the second FC layer and measure the change in performance, compared to the vanilla algorithm which is trained for a single MR.

Rate-Independent CS for image reconstruction

In this section, we will describe the network, the training and testing protocols used for the image reconstruction problem: given $\mathbf{y}$, return $\mathbf{x}$, where $\mathbf{y} = \Phi\mathbf{x}$. We reiterate that Φ is also learned (see Figure 3.6), and forms the first layer of the network while training. We experiment on two network architectures – **the extended ReconNet architecture [107] which uses 6 convolutional layers, and a 3-layer autoencoder [126]** – for jointly learning the Φ and the reconstruction algorithm.

Note that the networks are designed for block-wise reconstruction which means that the sensing and reconstruction process is for non-overlapping blocks of the image rather than the entire image. The loss function is simply the Euclidean loss between the desired output $\mathbf{x}$ (the original image itself) and the estimated reconstruction $\hat{\mathbf{x}}$.

Datasets: The training and test sets are identical to those employed by 96. The training set contains 91 natural images that can be downloaded from this website [1] . The training set for the network is obtained by constructing image blocks of size 33×33. The test set consists of 11 standard images employed in image processing literature obtained from these two links given here [2] [3] .

Results: Figure 3.8 shows the results in terms of the mean PSNR obtained on the test set for different combinations of $[k, m]$. Figure 3.9 shows the reconstructions for a subset of test images comparing the vanilla framework with Rate-Independent CS using ReconNet.

We can clearly observe the desired behavior of Rate-Independent CS. For all cases tested, the performance only decreases gradually for Rate-Independent CS, as the number of measurements is decreased, within the operating range. This is not true in the case of the vanilla algorithm, where the performance falls much more quickly when the test MR strays from the training MR. It also appears that the performance at a measurement rate depends on the value of k. For example, a lower $k = 0.04n$ leads to a lower mean PSNR at MR $= 0.10$. This can be explained by observing that Stage 3 of the algorithm builds on the Φ learned in Stages 1 and 2 and is expected to be sub-optimal compared to the vanilla training algorithm for the specific

[1] mmlab.ie.cuhk.edu.hk/projects/SRCNN/SRCNN_train.zip

[2] https://web.archive.org/web/20160403234531/http://dsp.rice.edu/software/DAMP-toolbox

[3] http://see.xidian.edu.cn/faculty/wsdong/NLR_Exps.htm

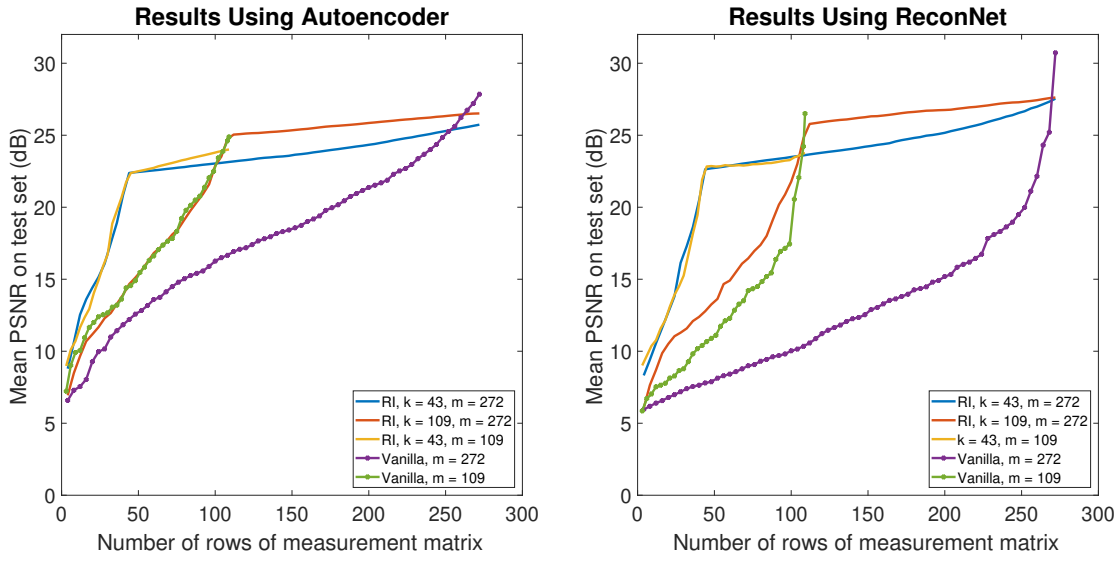


Figure 3.8: Rate-Independent CS results for image reconstruction for the Rate-Independent CS framework compared to the vanilla training algorithm trained for a single measurement rate. We test on two architectures – autoencoder [126] and ReconNet [107]. Clearly, Rate-Independent CS is stable over the entire chosen operating range while the performance of the vanilla framework drops considerably in the same range.

MR=0.10. Naturally, the performance at MR $= \frac{m}{n}$ depends on $m - k$, but only to a small extent, as required. This can be observed easily from the plots. Using ReconNet as the underlying architecture, we note that a rate-independent network for $MR = [0.10, 0.25]$ performs on average 9 db and up to 15.2 dB better than a vanilla network trained for $MR = 0.25$, when tested over all MRs. Similar results are observed for all other cases.

How rate-independence modifies Φ: Here, we compare the Φ learned in the Rate-Independent framework, with the vanilla algorithm in Figure 3.10. We can observe that the rows of rate-independent Φ look like a sampling of rows of Φ trained

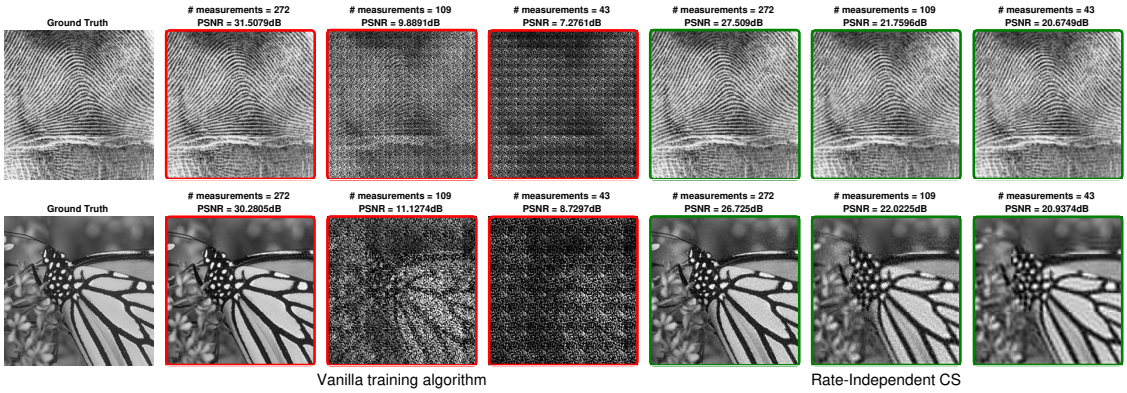
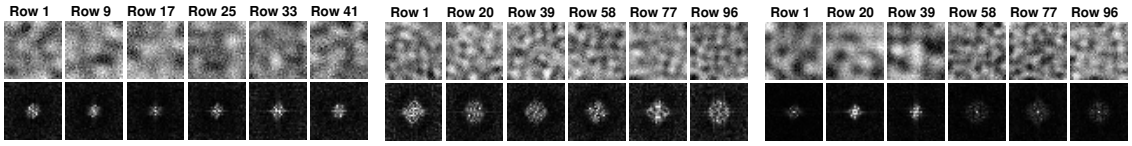


Figure 3.9: The figures show examples of reconstruction for 2 test set images using the vanilla training algorithm [107] trained for MR = 0.25, compared to Rate-Independent CS, proposed in this chapter ($[k, m] = [0.04n, 0.25n]$) We observe that Rate-Independent CS performs significantly better than the vanilla approach over a range of MRs. ReconNet[107] is used as the reconstruction network (Θ).

at different MRs. For instance, rows $1, 20, 39$ in column (c) are visually similar to the images shown in column (a), whereas rows $58, 77, 96$ in column (c) are similar to the ones in column (b). These observations suggest that rate-independent Φ shares characteristics of vanilla Φ's across a range of MRs.



(a) Vanilla Φ, MR = 0.04 (b) Vanilla Φ, MR = 0.10 (c) Rate-Independent Φ

Figure 3.10: The figures show the visualization of the rows of the learned Φ in the spatial (top) and Fourier (bottom) domains for the ReconNet architecture. The earlier rows of Φ in the case of Rate-Independent CS for the MR range [0.04,0.10] resemble the rows of Φ obtained using the vanilla training algorithm at MR = 0.04, while the later rows look similar to the rows of the vanilla Φ for MR = 0.10.

Rate-Independent CS for Object Tracking

We use object tracking in order to provide a proof of concept for Rate-Independent CS. In our framework, the frames of the video for object tracking are reconstructed using the measurements acquired by the spatial multiplexer that is learned in conjunction with the reconstruction algorithm, and the goal is to track the main object in the scene. We choose ReconNet [107] as the underlying network architecture. We use Kernelized Correlation Filter (KCF) [66], an off-the-shelf tracker, for our experiments. We use HoG features for the images and Gaussian kernel for the tracker. We use a well-known dataset of 15 publicly available videos [168]. We consider three different simple MR adaptation schemes in order to illustrate how Rate-Independent CS can be used for sample-efficient object tracking:

(a) Linearly decreasing MR: For a given video, we use an initial MR= m, and steadily decrease such that the final frame of the video is acquired at MR= k. We compare the performance of Rate-Independent CS with that of the vanilla framework, for the case where the MR is decreased linearly with the frame number. In this experiment, we choose the operating MR range to be $[k, m] = [0.04, 0.25]$, and the vanilla network is trained at MR= 0.25. This adaptation scheme could be utilized for an energy/memory constrained application. The initial MR is set to $m = 0.25$.

(b) Content-based MR adaptation using Euclidean loss: Here, we compute the normalized Euclidean loss between successive frames of the video. If the difference is smaller α, we reduce the number of measurements for the next frame by a fixed amount ΔMR. If the difference is larger than β, we increase the number of measurements for the next frame by ΔMR. This scheme can be viewed as a simple form of motion-based MR adaptation. For the experiment, we choose the operat-

ing MR range to be $[0.04, 0.25]$, and the vanilla network is trained at MR= 0.25, $\alpha = 0.15, \beta = 0.3, \Delta MR = 3$.

(c) Content-based MR adaptation using maximum correlation: For each frame of the video, the tracking algorithm outputs the maximum response value of cross-correlation $\in [0, 1]$ between the templates and the frame, which indicates the confidence of the tracker, and is used to localize the object. We utilize this value to determine the MR for sensing the next frame. If the max. correlation is smaller than γ, we increase the number of measurements for the next frame by a fixed amount ΔMR. If the max. correlation is larger than γ, we decrease the number of measurements for the next frame by ΔMR. In this experiment, we choose the operating MR range to be $[0.04, 0.10]$, and the vanilla network is trained at MR= 0.25, $\gamma = 0.3, \Delta MR = 3$.

In each of the above cases, we compare the performance of the reconstructions obtained using Rate-Independent CS to the vanilla training algorithm, using ReconNet as the reconstruction architecture. The performance is measured in terms of the average precision of the localizations, for a pixel error threshold of 20 pixels. In each case, we also calculate the average MR over the entire database, that the heuristics lead to.

Results: The results are shown in Table 3.2 and visualizations are provided in Figure 3.11. Clearly, as expected, Rate-Independent ReconNet is superior in terms of the tracking performance by a huge margin, because the vanilla network yields poor reconstructions at MRs for which it is not trained specifically. For comparison, the tracking performance with full images (oracle, i.e no compression) is about 80% for a 20 pixel error threshold. Also, for the heuristics and the thresholds chosen, the average MR for the Rate-Independent case is much lower, and at the same time our approach maintains high tracking performance. We also observe that the MR determination based on maximum correlation is superior to the one based on Euclidean loss between

Method	MR adaptation approach					
	Linear Decrease		Euclidean Loss		Detector Confidence	
	AP	Avg. MR	AP	Avg. MR	AP	Avg. MR
Vanilla	54.29 %	0.1444	46.08 %	0.1113	63.31 %	0.0749
Rate-Independent CS	79.56 %	0.1444	70.82 %	0.1106	77.47 %	0.0462

Table 3.2: Object tracking performance comparison of the Rate-Independent framework with the single-MR vanilla framework for dynamically varying MR. Results clearly show that Rate-Independent CS is superior in terms of average precision (AP) as well as the average number of measurements made.

successive frames. It is possible to have more sophisticated approaches to determining the best MR for each frame, and optimize the values of α, β, γ and ΔMR, but such an elaborate study is beyond the scope of this chapter.

Rate-Independent CS for Image Recognition

In this section, we extend the ideas presented for the image reconstruction problem to a very different task of image recognition, in order to demonstrate the wide applicability of the proposed method. We would like to perform image/object recognition directly on the multiplexed measurements obtained from the camera, bypassing reconstruction. Also note that the measurement operator learned in this case is optimized for the task of image recognition. We use two widely used datasets – MNIST and CIFAR-10 – for this purpose. For MNIST, we use a modified version of the LeNet-5 architecture[101] with 3 convolutional layers, two FC layers and a softmax layer. We add two FC layers at the input of the network, of which the first serves as Φ and the second FC layer is the matrix Ψ. For CIFAR-10, we modify the 32 layer ResNet

Figure 3.11: Visual results on two videos for the object tracking, using maximum correlation of the detector to dynamically vary MR. For each video, the top row shows the frames acquired with no compression (conventional imaging, referred to as the oracle). The second and third rows display the reconstructions using Rate-Independent ReconNet (trained for MR $= [0.04, 0.10]$ and vanilla ReconNet (trained for MR $= 0.10$) respectively. Blue, green and red boxes show the object locations for ground-truth, Rate-Independent ReconNet and vanilla ReconNet respectively. Unlike the rate-independent framework, as MR is varied, the reconstructions are very poor in the vanilla case, leading to poor tracking performance.

model for CIFAR [63] by adding two fully connected layers at the input of network similar to MNIST model.

Results: Figure 3.12 shows comparison between Rate-Independent CS and vanilla training algorithm for different variations of $[k, m]$. We observe that the performance of Rate-Independent CS decreases slowly as the number of measurements decreases within the operating range for both MNIST and CIFAR-10. This is contrary to the performance of vanilla training algorithm where the performance falls more steeply as we move away from training MR. For the case of CIFAR-10, we note that a rate-

independent network for MR $= [0.10, 0.25]$ performs on average 8.49% points and up to 29.12 % points better than a vanilla network trained for MR $= 0.25$, when tested over all MRs.

3.4 Conclusion

In this chapter, we considered how we can encode interesting design constraints into the learning of measurement operators jointly with the reconstruction/inference network. We show how to determine the best number of measurements for compressive sensing jointly by solving a rank minimization problem. We propose regularization functions that encourage sparse singular values, which can be readily integrated

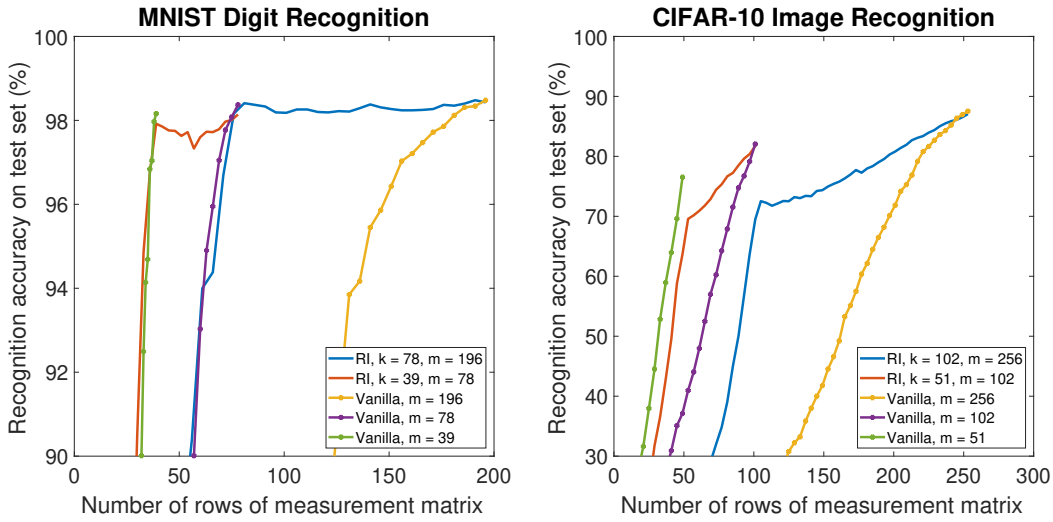


Figure 3.12: Rate-Independent CS results for image recognition for MNIST handwritten digit recognition ($n = 784$) and CIFAR-10 object recognition ($n = 1024$) – compared to the vanilla training algorithm trained for a single MR. Clearly, Rate-Independent CS is stable over the chosen operating range compared to the vanilla framework.

into deep learning frameworks. We demonstrate the effectiveness of the proposed methods on both image reconstruction and recognition. We also design a novel training algorithm for compressive imaging that enables training a single network that can be operated over a range of measurement rates by ensuring that predetermined subsets of the measurement operator are also valid measurement operators. This overcomes a major drawback of previous data-driven algorithms and makes the system more efficient. Our rate-independent framework performs significantly better than previous algorithms that work only for a single measurement rate. We demonstrate this on two important problems – image reconstruction and image recognition. Furthermore, through object tracking, we have shown how the rate-independent framework can be utilized in systems with time-varying constraints on the measurement rate. We hope that the algorithm and results presented in this chapter will enable researchers to adopt spatial multiplexing/ compressive imaging more easily in real-world scenarios.

Chapter 4

UNROLLED PROJECTED GRADIENT DESCENT FOR MULTI-SPECTRAL IMAGE FUSION

The research in this chapter was carried out when I worked as an intern in Mitsubishi Electric Research Laboratories, Cambridge, MA.

4.1 Introduction

Multi-spectral (MS) imaging, widely used in remote sensing related areas, allows sensing of images across a wider range of wavelengths compared to conventional RGB imagers. The bands of interest in multi-spectral imaging include RGB, near infrared (NIR) and short-wave IR (SWIR). Advantages of MS imaging lie in several aspects such as (a) better discrimination of objects with different material properties which may otherwise be very similar in the RGB bands and (b) more information gathering capability in the presence of harsh atmospheric conditions such as haze and fog, as the IR waves can travel more easily through these media, compared to visible light.

Multi-spectral sensing presents an interesting challenge. It is necessary in many applications to have both high spatial and spectral resolutions. However, there is a fundamental trade-off between the bandwidth of the sensor and the spatial resolution it can have. High spatial resolution typically can be achieved by panchromatic image covering the visible bands. This leads to the problem of multi-spectral image fusion. Given a set of low resolution images obtained at different wavelengths as well as a high resolution panchromatic image which does not have spectral information, we would like to fuse these two modes of information in order to produce a set of images which has both high spectral and high spatial resolutions.

MS image fusion is basically an ill-posed problem. To solve this problem, various methods have been proposed, either model-based [177, 176, 7, 104] or data-driven methods [100, 114, 119, 166]. Following many recent studies on model-based deep learning [152, 28, 59], we formulate a combination of model-based and data-driven solution based on deep learning in order to solve the multi-spectral image fusion problem. We unroll the iterations of the Projected Gradient Descent (PGD) algorithm, where the projection step is replaced with a convolutional neural network (CNN). Compared to other existing purely data-driven techniques, our work is based on well studied signal processing frameworks and thus, more interpretable and provides superior performance compared to the various baselines considered.

4.1.1 Contributions

1. We unroll the iterations of PGD using a learned CNN as the projection operator onto the set of high resolution multi-spectral images. This provides a signal processing-based perspective to the solution and makes the deep learning solution more interpretable.

2. In order to overcome the additional challenge of the unknown forward operator, $\mathbf{A}$, we consider two possible solutions. In the first case, we set $\mathbf{A} = \mathbf{I}$, the identity operator. With this, the fusion problem becomes a denoising problem. We observe that this reduces to existing deep learning solutions.

3. In the second case, we learn both the projection operator as well as the forward operator $\mathbf{A}$ by parameterizing it based on the desirable properties. We show improved results using this framework compared to baselines.

4.2 Prior Art

In this section, we briefly review relevant algorithms for MS fusion as well as other inverse problems. For thorough surveys, the readers are referred to [8, 112, 183].

Model-based iterative methods: In order to solve ill-posed problems, there is a rich literature on simple prior models of the desired signal. In the case of the multi-spectral fusion, priors include sparsity in the gradient domain – total-variation regularization, low-rank models [177, 176], over-complete dictionary learning with regularizers on the coefficients [7, 104]. These methods are simple to design and have theoretical guarantees. However, in terms of recovery performance as well as time complexity during testing, these methods fare poorly compared to purely data-driven methods described next.

Purely data-driven approaches: In recent years, the resurgence of deep learning [100] has led to feed-forward non-iterative approaches for several solving inverse problems in low-level vision including compressive sensing, single-image super-resolution, deblurring [114] and multi-spectral fusion [119, 166]. These methods are model-agnostic and simply learn a mapping from the measurements to the desired signal in a purely data-driven fashion. Compared to the model-based iterative methods, these methods generally yield superior results, and are also computationally faster owing to their non-iterative nature (a feed-forward operation at test time) as well as the ease of implementation on Graphics Processing Units (GPUs).

Model-based deep learning: Although purely data-driven approaches using deep learning perform very well compared to model-based shallow methods, they are less interpretable than the latter. In order to bridge the gap between understanding

and performance, many methods recently combine iterative methods with the deep learning. This can be achieved in several ways. Sun *et al.* first proposed the ADMM-Net for MRI [152]. Here, the iterations of ADMM are unrolled and the projection operator as well as the shrinkage functions are learned from data. Chang *et al.* proposed the OneNet [28] for inverse problems like super-resolution and restoration. It unrolls the ADMM algorithm such that projection operator is a deep learning method. More recently, Gupta *et al.* [59] propose a similar approach in the case of Projected Gradient Descent (PGD) and also provide theoretical guarantees for convergence. In this chapter, we combine PGD with deep learning for the problem of MS image fusion. We unroll the iterations of PGD such that the projection operator is computed using a trained convolutional neural network (CNN) and all the parameters are learned end-to-end using a training dataset. This problem is different from other inverse problems in two aspects – (a) we are given the pan-chromatic image which acts as important side information, and (b) the forward operator $\mathbf{A}$ is usually unknown. We describe the proposed algorithm next.

4.3 Unrolling PGD Using a CNN as the Projection Operator

We first consider a general problem where we have measurements $\mathbf{y} \in \mathbb{R}^m$ of unknowns $\mathbf{x} \in \mathbb{R}^n$ via a linear forward operator $\mathbf{A} \in \mathbb{R}^{m \times n}$, *i.e.*

$$\mathbf{y} = \mathbf{A}\mathbf{x}. \tag{4.1}$$

In real applications, we typically have $m < n$, leading to an underdetermined linear system with infinite solutions in general. In order to have a unique solution, we solve a constrained optimization problem as following

$$\mathbf{x}^* = \text{argmin}\, \frac{1}{2}||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 \;\; \text{s.t. } \mathbf{x} \in \mathcal{C}, \tag{4.2}$$

where $\mathcal{C}$ is the constraint set. In our case, $\mathcal{C}$ is the set of feasible images. Generally, the set $\mathcal{C}$ is chosen based on domain knowledge, *e.g.*, the set of images with small ℓ_1

norm of the wavelet coefficients. A popular approach to solving the above problem in image processing is by employing the Projected Gradient Descent (PGD). It consists of two alternating steps:

$$\mathbf{w}^{k+1} = \mathbf{x}^k + \alpha \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k), \tag{4.3}$$

$$\mathbf{x}^{k+1} = \Pi_{\mathcal{C}}(\mathbf{w}^{k+1}). \tag{4.4}$$

The first step in the above optimization is gradient descent, which is guaranteed to reduce the value of the cost function given a suitable value of the learning rate α. However, the output of the gradient descent step is not guaranteed to be a feasible point. The second step is to map the intermediate output from gradient descent to the closest point in the set of feasible solutions through the projection operator $\Pi_{\mathcal{C}}$.

The MS image fusion problem can be formulated as a linear inverse problem. Let I_P, I_L, and I_H represent the vectorized versions the panchromatic image, low resolution, and high resolution multi-spectral images, respectively. Hence $\mathbf{y} = (I_P; I_L)$ and $\mathbf{x} = (I_P; I_H)$. The forward operator $\mathbf{A}$ models the mapping from high resolution to the low resolution multi-spectral images.

However, there are several challenges to solving MS image fusion. First, in the case of multi-spectral aerial images considered in this chapter (as well as natural images in general), it is not possible to provide a precise mathematical definition of a feasible set and it is also not clear what a good approximate to the constraint set is. The goodness of approximate constraint set may also depend on the properties of $\mathbf{A}$. Second, although we know that the forward operator $\mathbf{A}$ can be represented as a combination of blurring and downsampling, the exact coefficients of the blur kernel are unknown.

In existing methods, some of the widely used hand-crafted priors include the sparsity priors in wavelet and gradient domains. In the case of dictionary learning, an

over-complete sparsifying basis is also learned from the data and sparsity priors are used on the coefficients. However, these techniques fall short in terms of providing high quality solutions and have given way to purely data-driven non-linear methods using deep learning, as explained in Section 4.2. A main drawback of deep learning methods they are not easily interpretable and it is unclear what functions they perform in terms of signal processing.

Following similar works for compressive imaging, MRI, and super-resolution, etc., we propose the following framework. We ***unroll*** the iterations of PGD (Eqs. (4.3) and (4.4)), and use a trained CNN in each step in place of the projection operator. We choose *apriori* the number of iterations to unroll and train the entire pipeline end-to-end. This is also illustrated in Fig. 4.1. Therefore, although the core architecture of the CNN is hand-crafted, at a higher level, the algorithm is based on well-studied methods.

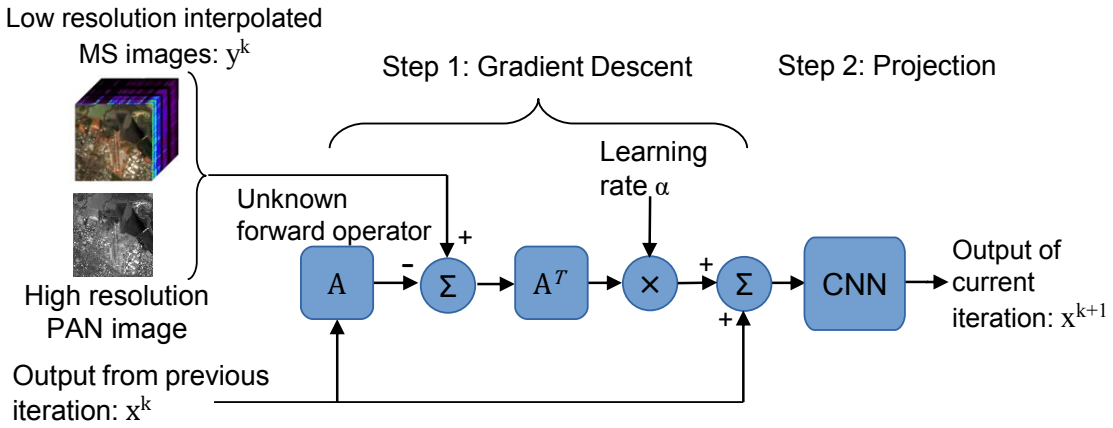


Figure 4.1: A single stage of unrolled PGD framework we propose in this chapter. The gradient descent step is carried out as usual and a CNN is used as the projection operator onto the set of high resolution multi-spectral images. Note that both forward operator $\mathbf{A}$ and the layers in the CNN are learned end-to-end jointly.

4.3.1 *Using the Denoising Formulation i.e.,* $\mathbf{A} = \mathbf{I}$

We can solve the optimization problem by using the bicubic interpolations of the low resolution images as the measurements $\mathbf{y}$ and by simply setting $\mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Now, the original problem of fusion reduces to the problem of denoising. We note that under this new formulation, the iterations of PGD reduce to simply applying the projection operator directly on the measurements $\mathbf{y}$ in a single step. In our case, where the CNN is performing the operation of the projection operator in the unrolled PGD, only a single stage of CNN layers that maps $\mathbf{y}$ to $\mathbf{x}$ suffices, i.e., the number of stages = 1. Here, we make the interesting observation that this formulation, depending on the network architecture, reduces to the framework presented in [119] and [166], but arrived at using a signal processing methodology.

4.3.2 *Using the General Formulation i.e.,* $\mathbf{A}$ *is Learned*

As a natural extension to the above, we investigate the possibility of *jointly* learning the forward operator $\mathbf{A}$, the learning rate α and the fusion algorithm. After bicubic interpolation, the low resolution images are of the same size, albeit blurred versions of the high resolution images. This suggests modeling $\mathbf{A}$ as a blurring operator. We assume that the same blurring filter, which we represent as a square convolutional kernel $\mathbf{K_A}$ of size $S \times S$, operates on the entire image as well as on different spectral channels. We structure the kernel to be of the form

$$\mathbf{K_A} = \mathbf{K_B} + \mathbf{K_I}, \tag{4.5}$$

$$\text{s.t.} \quad \sum_{i=1}^{S}\sum_{j=1}^{S} \mathbf{K_A}(i,j) = 1 \text{ and } \mathbf{K_A}(i,j) \geq 0, \forall i,j \in \{1,\ldots,S\}$$

where the coefficients of $\mathbf{K_B}$ are learned and $\mathbf{K_I}$ is the identity filter. This encourages the central coefficient of the filter to be higher than the other elements and the

constraints on $\mathbf{K_A}$ ensure that it is a valid blurring filter, and the corresponding operator $\mathbf{A}$ is a valid blurring operator. In our expts., we have chosen the size of $\mathbf{K_A}$ to be 9×9.

4.4 Experiments

Image Name	**Bicubic**	**Shrinkage Fields [141]**	**DeepCASD [167]**	**Unrolled PGD**					
				$\mathbf{A} = \mathbf{I}$ (reduces to [166])			$\mathbf{A}$ is learned		
				Number of Layers			Number of Iterations		
				4	12	20	1	3	5
Moffett	32.24	34.21	34.53	37.44	38.29	37.46	37.59	38.52	38.17
	0.4788	0.6981	0.7185	0.9710	0.9768	0.9729	0.9706	0.9778	0.9776
Cambria Fire	35.32	37.51	37.62	37.83	38.91	38.71	37.99	38.91	39.33
	0.5887	0.7941	0.7987	0.9734	0.9734	0.9696	0.9775	0.9765	0.9771
Cuprite	32.44	34.33	34.52	36.88	37.56	36.82	37.95	38.56	39.02
	0.5060	0.7437	0.7616	0.9750	0.9842	0.9823	0.9794	0.9837	0.9840
Los Angeles	27.96	30.39	30.50	36.27	37.38	37.28	36.42	37.79	37.77
	0.4888	0.7628	0.7761	0.9702	0.9755	0.9760	0.9712	0.9777	0.9790
Mean	31.99	34.11	34.29	37.11	38.03	37.57	37.49	38.45	38.57
	0.5156	0.7497	0.7637	0.9760	0.9775	0.9752	0.9746	0.9789	0.9794

Table 4.1: The table shows the experimental results of multi-spectral image fusion in terms of PSNR in dB (the top number in each cell) and SSIM (the bottom number in each cell) on the test set. Clearly, the results using unrolled PGD are superior to all the baselines considered. Also observe that the results improve further when $\mathbf{A}$ is learned. Note that when $\mathbf{A} = \mathbf{I}$, PGD reduces to a single projection operator as in [166]. Then number of layers refers to the number of layers in the projection CNN. In the case where $\mathbf{A}$ is learned, the "number of iterations" refers to the number of steps of PGD we unroll. The CNN in each projection operation contains 4 layers of convolutions + ReLU.

Training and testing datasets: For all our experiments, we use a dataset of 138 high resolution aerial MS images with 16 channels and a panchromatic image of size 256×256 pixels. These images are synthesized using the AVIRIS hyper-spectral image database [127]: each multi-spectral image is generated by a weighted linear combination of a band of hyper-spectral images. For training, we also need access to the low resolution images. The low resolution multi-spectral images are produced by first low-pass filtering (anti-aliasing) and then downsampling by a factor of 2 (we focus on $2\times$ super-resolution, however the method can be extended to other factors).

The test set consists of four low resolution multi-spectral images with the same 16 channels – Moffett, Cambria Fire, Cuprite and Los Angeles – of size 512×512 and a panchromatic image, also of size 512×512 [167]. As before, we form 256×256 images of the test set which serve as the input to the algorithm. Thus, at test time, the goal is to fuse the lower resolution 256×256 multi-spectral images with the 512×512 panchromatic image in order to provide a high resolution multi-spectral output of resolution 512×512.

Network architectures: As described in Section 4.3, the projection operator (Equation 4.4) is learned from training data, and we choose to implement it using a CNN. Based on the work of Wei *et al.* [166], the architecture of this network is simple with **4 layers** of 2D convolutions + Rectified Linear Units (ReLUs) with a residual connection connecting the bicubic interpolated low resolution multi-spectral images ($\mathbf{y}$) to the output of the penultimate layer of the CNN. We set the the filter size in all layers to be 9×9 and we use 32 feature maps for layers 1 and 2. Layer 3 produces 17 feature maps in order to be compatible with the number of channels of the input, and the output produces the desired 16 multi-spectral channels.

Training and testing protocol: The low resolution 128×128 multi-spectral training images are first upsampled using bicubic interpolation to match the resolution of the panchromatic image i.e., 256×256. Using the 138 pairs of low-res and high-res training images and a stride of 11, we first create a dataset of about 60832 patches (of size $32 \times 32 \times 17$) of the interpolated low-res multi-spectral and panchromatic images which form the input to the fusion algorithm, and $32 \times 32 \times 16$ high-res multi-spectral images which form the desired output. About 1800 of these patches are used as the validation set in order to select the hyperparameters. For the case with $\mathbf{A} = \mathbf{I}$, the PGD algorithm reduces to simply applying the projection operator (the CNN, in our case) **once** on the low-res input $\mathbf{y}$, and thus, the algorithm essentially reduces to the one described by Wei *et al.* [166]. For this case, we train 3 networks with varying depths of $4, 12$ and 20 layers. When $\mathbf{A}$ is learned, we also need to choose the number of iterations, n_{iter} of PGD to unroll. We conduct experiments with $n_{iter} = 1, 3, 5$. The networks are trained using Adam optimizer [93] for 2×10^5 iterations with a batch size of 32. We observed empirically that the validation error converges for the chosen number of iterations and takes a few hours to train on an Nvidia TitanX. We use the mean squared error over the batch between the desired high resolution patches and the output of the algorithm as the loss function. All the networks are trained using Tensorflow [2]. During test time, the images are split into non-overlapping patches and the fed through the trained networks. We use both Peak Signal-to-Noise Ratio (PSNR) as well as Structural Similarity Index (SSIM) [164] to measure the performance of the algorithms. The measures are computed channel-wise and averaged over the 16 multi-spectral channels.

Baselines: We provide experimental comparisons of the proposed approach with three baseline algorithms: (1) Bicubic interpolation, where the output of the algo-

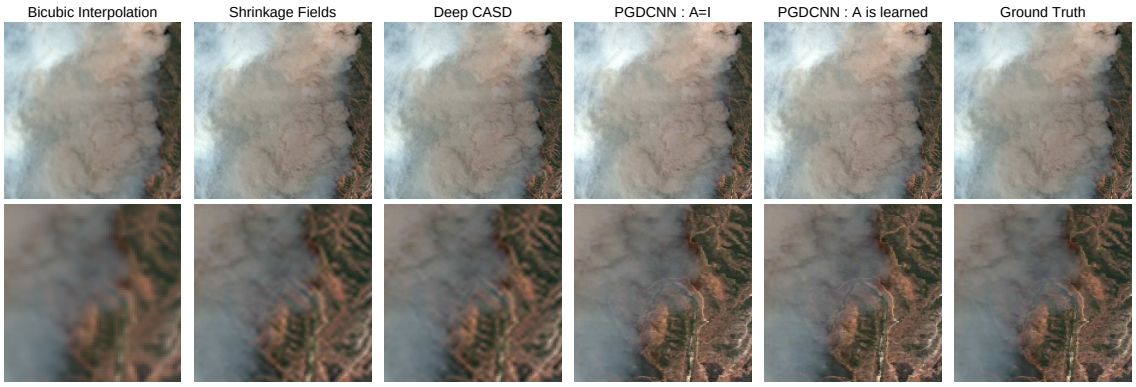


Figure 4.2: Visual comparison of results for the "Cambria Fire" image (top row) and the zoomed in portions (bottom row). It is clear that the unrolled PGD (PGDCNN) provides much sharper spatial resolution and preserves spectral information compared to the baselines.

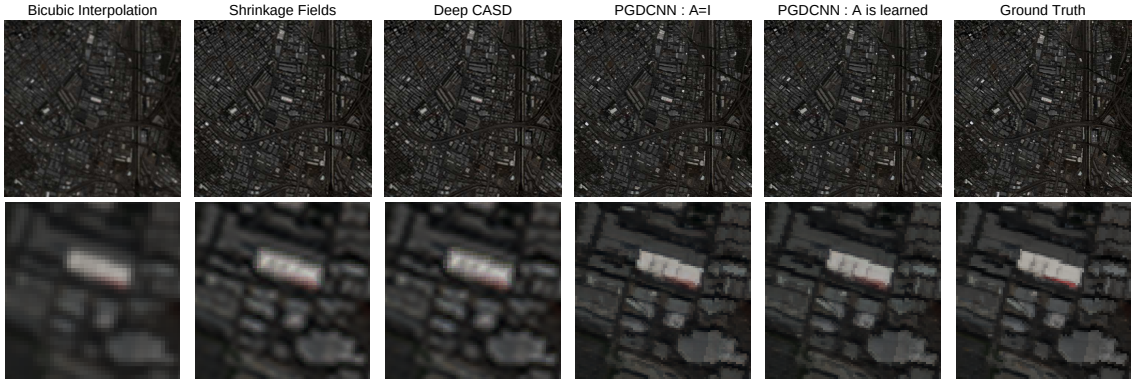


Figure 4.3: Visual comparison of results for the "Los Angeles" image (top row) and the zoomed in portions (bottom row). It is clear that the unrolled PGD (PGDCNN) provides much sharper spatial resolution and preserves spectral information compared to the baselines.

rithm is the channel-wise upsampling of the lower resolution images using bicubic interpolation, (2) Shrinkage field networks by Schmidt and Roth [141] which is a trainable architecture, but is applied to each channel independently and (3) Deep Coupled Analysis and Synthesis Dictionary (CASD), a recent work by Wen *et al.*[167]

uses channel-wise outputs from the Shrinkage Field Network and a CASD framework in order to exploit the inter-channel relationships in order to improve fusion results.

Results: The results of multi-spectral fusion on the test images using various algorithms are shown in Table 4.1. We use two performance metrics in order to measure the performance of the algorithms: PSNR and SSIM. From the table, we clearly observe that the results using unrolled PGD when the forward operator $\mathbf{A}$ is learned are superior to bicubic interpolation, shrinkage fields and DeepCASD by 3-6 dB, and about 0.6 dB better than unrolled PGD with $\mathbf{A} = \mathbf{I}$. Figs. 4.2 and 4.3 show visual results on two test images illustrating that the proposed algorithm provides much sharper resolution and preserves spectral information compared to the baselines. We also note that, unrolled PGD-based multi-spectral fusion can be performed extremely fast. On an Nvidia TITANX with batch processing of all the patches at once, it takes 0.09, 0.16 and 0.22 seconds for $n_{iter} = 1, 3, 5$ respectively.

4.5 Conclusion

In this chapter, we developed a data-driven based approach for multi-spectral fusion that was inspired by commonly used projected gradient descent (PGD). We unroll PGD for a predefined number of iterations such that the projection operation is performed by a convolutional neural network (unrolled PGD). As the forward operator $\mathbf{A}$ is not known in our case, we develop two variants of unrolled PGD. First, we use a denoising formulation with $\mathbf{A} = \mathbf{I}$, the identity operator. We make the observation that this reduces to existing black-box deep learning methods. Second, as a generalization of this approach, we model $\mathbf{A}$ as a blur kernel and learn its coefficients jointly with the CNN. Our experiments show that the learning-the-projection operation outperforms several baselines considered, and improves the results further with a learned operator $\mathbf{A}$.

Chapter 5

TEMPORAL TRANSFORMERS: JOINT LEARNING OF INVARIANT AND DISCRIMINATIVE TIME WARPS

5.1 Introduction

Guaranteed invariances of machine learning algorithms to nuisance parameters is an important design consideration in critical applications. Classically, invariances can only be guaranteed under a model-based approach. Learned representations however have not been able to guarantee invariances, except by empirical tests [56]. Learning invariant representations that build on analytical models of phenomena may hold the cue to bridge this gap, and can also help lend explainability to the model.

However, deep learning presents an especially hard challenge for learning explainable invariants, primarily due to incompatibility between the mathematical approaches that underlie invariant design, and the architectures prevalent in deep learning. There have been recent attempts at leveraging model-based and data-driven approaches to learning invariant representations across spatial transforms [88, 83, 170], illumination [156, 110], and view-point [106]. On the other hand, learning invariant/robust representations to temporal rate-variations has received significantly lesser attention. If tackled well, many applications of human activity modeling will benefit, including more robust recognition algorithms for human-robot interaction, richer synthesis of human motion for computer-generated imagery, and health applications.

Hybrid model- and data-based approach: In this chapter, our chosen application is activity classification from RGBD devices, where skeleton data may be available. Activities such as walking can be performed at different rates by different

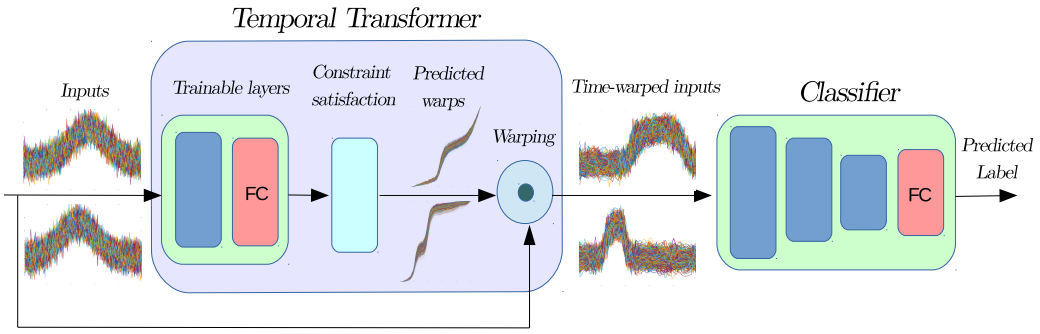


Figure 5.1: The Temporal Transformer Network (TTN) is a trainable module that is easily added at the beginning of a time-series classifier. Its function is to warp the input sequences so as to maximize the classification performance, as shown in the figure for two classes of waveforms which become more discriminative after passing through the TTN. The sub-modules of the TTN are explained in Section 5.4.

subjects owing to physiological, and biomechanical factors [33, 162]. We would like to design representations that provide robust classification against such nuisance factors. To do this, we adopt a model-based approach, and constrain certain layers in deep-models using the model. The model for temporal variability is adopted from past work in elastic temporal alignment which considers temporal variability as a result of a temporal diffeomorphism acting on a given time-series [150]. The space of such diffeomorphisms has a group structure, and can be converted to simpler geometric constraints by exploiting contemporary square-root forms derived from the diffeomorphic maps [147].

Compatibility with deep architectures: We design a novel module, which we refer to as a Temporal Transformer Network (TTN). The hallmark of this module is that it can be easily integrated into existing time-series classifiers such as Temporal Convolution Networks (TCNs) [90] and Long Short-Term Memory (LSTM) networks [67]. TTN is explainable in the sense that it is designed so as to interact with the

classification network in a predefined, predictable and visualizable manner. TTN is a trainable network added at the beginning of the classifier and operates on the input sequence by performing selective temporal warping of the input sequences. As such, it has the ability to factor out rate variations, if present in the data, as well as increase the inter-class separation by learning to align sequences in dissimilar classes away from each other.

Application impact: Recognition of human activities from sensors such as motion capture (mocap) and depth cameras like the Microsoft Kinect and Intel RealSense has been gathering a great deal of interest in the recent past. The cost of these sensors is ever-reducing and the increasing effectiveness of pose estimation algorithms [157] makes 3D skeletons an important sensing modality for action recognition. As the problem of action recognition presents a large amount of variability both inter-class as well as intra-class we choose it as the focus of this chapter.

Contributions

- We propose the Temporal Transformer Network (TTN), which performs joint representation learning as well as class-aware discriminative alignment for time-series classification including action trajectories.
- We design the TTN to generate highly expressive non-parametric, order-preserving diffeomorphisms, which have favorable theoretical properties.
- We exploit the non-uniqueness of the optimal alignment (between equivalence classes) to develop discriminative warps for improved classification.
- The proposed TTN can be easily integrated into existing time-series classification architectures, with just a single line of code for the warping module.

We validate our contributions by demonstrating improved performance on small and large databases real datasets, as well as synthetic datasets, for action recognition from 3D pose obtained from two different modalities – Kinect and mocap. The combined architecture of the TTN and the classifier consistently yields improved classification performance compared to several baseline classifiers.

5.2 Related Work

Deep learning of invariant representations: One of the main inspirations for this work is the paper by Jaderberg et al. [77] on Spatial Transformer Networks (STNs) where a smaller network first predicts a geometric transform of the input grid parameterized by affine transforms or thin plate splines. The transformation is then applied to the input before feeding it to the classification network. A recent work is that of Skafte-Detlefsen et al. [144] who improve the performance of spatial transformers by replacing affine transforms and thin plate splines with a richer class of parameterized diffeomorphic transforms called continuous piecewise-affine transforms, but at the expense of complex implementation and considerably longer training times. Both these works are aimed at building invariances to spatial geometric transforms of images. Capsule networks by Sabour et al.[139] expand the expressive capacity of CNNs by allowing them to learn explicit spatial relationships. An interesting recent work by Tallec and Ollivier [154] show that LSTM networks have the capability to learn to warp input sequences. Our experiments show that by integrating LSTMs with the module designed in this chapter, the performance can be further increased, as the proposed framework can also lead to more discriminative representations.

In this chapter, we design a module to predict warping functions in the temporal domain which when applied to the input sequences lead to higher classification performance. This requires the predicted warping functions, γ's to satisfy the order-

preserving property. Moreover, in our case, the predicted warping functions are non-parameterized and thus, can span the entire space of rate-modifying transforms. The warping is also elastic, as opposed to rigid deformations which is the case with STNs. We note that these are important design requirements in the case of temporally varying signals, that are different from transforms in the 2D spatial domain.

Alignment of time-series data: The most commonly used method to align time-series data is perhaps Dynamic Time Warping (DTW) [14, 140]. DTW tries to minimize the $\mathbb{L}^2$ distance between two time series after a time warping is applied to one of them, and is agnostic of class information. To address some of DTW's shortcomings, new methods have been proposed recently, including the elastic functional data and shape analyses [149, 150] which defined proper metrics that are invariant to time warping, and soft-DTW which is a differentiable loss function that may be integrated into neural networks [35].

One of the major differences between our proposed approach and the aforementioned optimization-based time warping methods is that our approach does discriminative warping based on class information and does not need signal templates. This is further discussed in Section 5.4.

3D action recognition using deep learning: As sensing systems like the Microsoft Kinect, Intel RealSense and motion capture are getting more effective at acquiring depth and human pose estimation with even up to millimeter precision, research and commercial interest in employing 3D pose data for action recognition has understandably increased. Recent experiments suggest that for small datasets, recognition accuracies are better with 3D pose information compared to video frames [53]. That simple landmark-based or skeleton-based action recognition can be effective is supported by evidence from works in psychology which show that humans are excellent at recognizing actions only from a few points on the human body [81].

Recurrent neural architectures, especially Long Short-Term Memory (LSTM) networks have been used to perform 3D action recognition e.g. [44, 142]. Song et al. propose including layers for spatial and temporal attention (STA-LSTM) [146] which greatly improves the recognition performance. For majority of the experiments in this chapter, we will use the temporal convolution network (TCN) with residual connections [99] as they are effective, simple to build and faster to train compared to LSTM based networks. Additionally, Kim and Reiter have shown excellent results on using TCNs for 3D action recognition [90]. This network outperforms STA+LSTM [146] for 3D action recognition. They further show that TCNs can learn both spatial and temporal attention without the need for special attention layers. Also, the network filter activations are interpretable by design because of the residual connections. We also note that the TCN architecture presented in [90] incorporates pooling mechanisms inside the network.

We note that more recently, newer architectures have proposed modifications to baseline architectures by using graph convolutions to better take into account the spatial structure of the joints in the human body [171]. However, it has a much higher computational load. Other representations include image-based ones [85, 106], and fusing skeleton information with velocity information [29] etc. Our contributions in this chapter are orthogonal to these works, and the main focus of the chapter is to design a specialized module for learning rate-robust discriminative representations. As such, for our experiments, we choose two effective widely-used simple-to-implement architectures as our baselines – TCNs and LSTMs – and demonstrate improvements in recognition performance over these frameworks.

5.3 Diffeomorphic Models for Rate Variation

A continuous time-series can be represented as a single-parameter curve, which we denote by $\alpha(.)$, where $t \in [0,1]$ is the parameter. In our case, t is time and we assume that each $\alpha(t) \in \mathbb{R}^N$. Figure 5.2 shows an illustration of this representation. Another curve β is a *resampling* of α if $\beta = \alpha \circ \gamma$, where $\circ$ is a function composition, and γ is the *resampling function*. We focus on the set of γ's which form the group of order-preserving diffeomorphisms Γ. In physical signals such as human actions, two actions α_1, α_2 differing only by a change of rate of execution obey the equation $\alpha_1 = \alpha_2 \circ \gamma$, for some $\gamma \in \Gamma$. Given a 1-differentiable function γ defined on the domain $[0,T]$, for γ to be an element of Γ, γ needs to satisfy the following conditions:

$$\gamma(0) = 0, \gamma(1) = 1, \text{ and } \gamma(t_1) < \gamma(t_2), \text{ if } t_1 < t_2, \tag{5.1}$$

The above conditions fix the boundary conditions, and imply that any $\gamma \in \Gamma$ is a monotonically increasing function. This property is also called order-preserving which is important to the current discussion of action recognition as actions are critically dependent on sequencing/ordering of poses/frames. It is easy to show that

- $\forall \gamma_1, \gamma_2 \in \Gamma, \gamma_1 \circ \gamma_2 \in \Gamma$,
- $\gamma_{Id} \in \Gamma$,
- $\forall \gamma \in \Gamma, \exists \gamma^{-1} \in \Gamma$ s.t. $\gamma \circ \gamma^{-1} = \gamma_{Id}$, where $\gamma_{Id}(t) = t$, the identity warping function.

These properties imply that the set of γ's form a group Γ, where the group action is function composition. We denote by $\dot{\gamma}$, the first derivative of $\gamma \in \Gamma$, or

$$\gamma(t) = \int_0^t \dot{\gamma}(t)dt, \int_0^1 \dot{\gamma}(t)dt = \gamma(1) - \gamma(0) = 1 \tag{5.2}$$

Further, due to the monotonically increasing property of γ, we have $\dot{\gamma} > 0$. This in conjunction with (5.2) implies that $\dot{\gamma}$ has the properties of a probability distribution function (positive, and integrates to 1), and the corresponding γ is thus equivalent to a cumulative distribution function.

As we work with digitized signals from sensors such as Kinect and mocap, we represent a discrete time series by $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$. In this chapter, we will work with time series in $\mathbb{R}^N$ i.e., each $\mathbf{x}_t \in \mathbb{R}^N$ is called a frame of the sequence X. More clearly, we have $\alpha(t) = \mathbf{x}_t, t \in \{1, 2, \ldots, T\}$. The warping function in the case of a discrete time signal is a discretized version of $\gamma \in \Gamma$, which we represent using γ with a slight abuse of notation. The derivative $\dot{\gamma}$ can be computed by first order numerical differencing. Thus, (5.2) now becomes

$$\gamma(t) = \sum_{i=0}^{t} \dot{\gamma}(i) \quad \text{and} \quad \frac{1}{T}\sum_{i=0}^{T} \dot{\gamma}(t) = 1. \tag{5.3}$$

Two sequences α and β are said to be *equivalent* if there exists a $\gamma \in \Gamma$ such that $\alpha = \beta \circ \gamma$, and the set $\{\alpha \circ \gamma\}, \forall \gamma \in \Gamma$ is called the equivalence class of α under rate variations and is denoted by $[\alpha]$. In classical elastic alignment, given two signals, a metric between sequences is defined as the minimal distance between their equivalence classes. However, this approach can be used to develop class-specific templates, and phase-amplitude separation [117] that reduces intra-class variance, but does not promote inter-class separation. Once the equivalence classes are defined, metrics are designed to compute distances between equivalence classes and develop methods to compute statistical measures such as mean and variance, which can be used to compute optimal alignments [150].

5.4 Temporal Transformers for Learning Discriminative Warping Functions

The main idea presented in this chapter is to use a specialized module, which we call a *Temporal Transformer Network* (TTN) for neural network-based classification which, given an input test sequence X, generates a warping function γ used to warp the input sequence by computing $X \circ \gamma$ and feed it to the classification network. It is important to note that the warping is carried out using linear interpolation. This makes it possible to train both the TTN and the classifier jointly end-to-end as the entire pipeline is (sub-) differentiable. Another notable aspect of this framework is that the warping functions are predicted without a "class-template". Even though this sounds paradoxical, we will soon show that this allows our framework to jointly learn features as well as achieve discriminative warps. This capability is makes our framework more powerful than template-based matching techniques like Dynamic Time Warping (DTW) and variants [14, 140].

Key Insight: Given two input sequences X_1 and X_2 such that they differ only by a warping transform, the trained TTN would ideally predict γ_1 and γ_2, corresponding to X_1 and X_2 respectively such that $X_1 \circ \gamma_1 = X_2 \circ \gamma_2$. However, we note that γ_1 and γ_2 are not unique because $X_1 \circ \gamma_1 \circ \gamma = X_2 \circ \gamma_2 \circ \gamma, \forall \gamma \in \Gamma$. We believe this to be an important and often overlooked fact, and is formally referred to as invariance to group-action. The non-uniqueness of the warping functions is not really exploited in the temporal alignment literature. The goal in past work in alignment has been to learn a class-specific template using an EM-style optimization method, which is not unique due to invariance to group-action. Non-uniqueness here presents an opportunity that can be exploited to develop discriminative warps for classification problems.

The non-uniqueness of the warping functions can be advantageous as it expands the expressive capacity for classification. Minimizing intra-class variations – rate

variations in our case – is only one part of the problem. For classification, we would also like to maximize inter-class variations. For example, if we have four sequences X_1, X_2, X_3 and X_4 such that X_1, X_2 belong to Class A, and X_3, X_4 belong to class B, the TTN has the capacity to predict $\gamma_1, \gamma_2, \gamma_3$ and γ_4 such that

- $d(X_1 \circ \gamma_1, X_2 \circ \gamma_2) < d(X_1, X_2)$
- $d(X_3 \circ \gamma_3, X_4 \circ \gamma_4) < d(X_3, X_4)$
- $d(X_i \circ \gamma_i, X_j \circ \gamma_j) > d(X_i, X_j), i = 1, 2$ and $j = 3, 4$,

where $d(.)$ is the Euclidean distance between sequences. However, we do not explicitly train the networks to achieve the above. Both the TTN and the classifier are trained so as to maximize classification performance by minimizing the cross-entropy loss between the predicted and true distribution over the class labels given the input sequence. The TTN can be divided into three sub-modules:

Trainable layers: As shown in Figure 5.1, the input to the TTN trainable layers is an input sequence. The input is then passed through a few layers of convolutions and fully-connected layers. The network outputs a vector of length T, such that the first element is set to be zero. T is the length/number of frames in the input sequence. Let us denote this vector by $\mathbf{v} \in \mathbb{R}^T$.

Constraint satisfaction layers: The output $\mathbf{v}$ is unconstrained, and hence, we need to convert it into a valid warping function that satisfies Equations (5.1).

We divide the vector $\mathbf{v}$ by its norm, to get a unit-vector, followed by squaring each of its entries. This will have the effect of converting the vector into a point on the probability simplex. Thus, we use the following mappings:

$$\dot{\gamma} = \frac{\mathbf{v} \odot \mathbf{v}}{\|\mathbf{v}\|^2}, \quad \text{and} \quad \gamma(t) = T \cdot \sum_{i=1}^{t} \dot{\gamma}(i), \tag{5.4}$$

where, $\odot$ is the Hadamard product (element-wise multiplication). This is treated as the network's estimate of the derivative of the warping function, denoted by $\dot{\gamma}$. We compute the cumulative sum and multiply it by the length of the input sequence, T, in order to form the warping function γ as shown in Equation (5.4).

Differentiable temporal resampling: The warping function γ is then applied to the input sequence using linear interpolation. We assume that the sampling rate of the signal is high enough in relation to the speed of the activity, (in practice, 20 frames/sec is plenty for most common action recognition applications) that simple linear interpolation of the frames is sufficient to get intermediate skeletons to look realistic. The warping is done using the equation $Y(t_t) = X(t_s) = X(\gamma(t_t))$, where X and Y are the input and output sequences respectively, and t_s and t_t are the source and target indices respectively. The frames of output sequence Y are to be defined at regular intervals $t_t = 1, 2, \ldots, T$. As the source indices corresponding to these times may not be integers, we use linear interpolation to find the values of $X(\gamma(t_t))$. This operation is sub-differentiable, as in the case of STN. Thus, we can write the expressions of the required gradients as follows (these expressions are adapted from Jaderberg et al. [77]). If X^j is the input sequence of the j^{th} joint, Y^j is the warped sequence output by the TTN module and $i \in \{1, 2, \ldots, T\}$ is the time index, we have:

$$\frac{\partial Y_i^j}{\partial X_\tau^j} = \sum_{\tau=1}^{T} \max(0, 1 - |t_i^s - \tau|) \tag{5.5}$$

$$\frac{\partial Y_i^j}{\partial t_i^s} = \sum_{\tau=1}^{T} X^j \cdot \begin{cases} 0, & \text{if } |t_i^s - \tau| \geq 1 \\ 1, & \text{if } \tau \geq t_i^s \\ -1, & \text{if } \tau < t_i^s \end{cases} \tag{5.6}$$

5.5 Experimental Results

All networks in this chapter are trained and tested using Tensorflow [2]. Due to space constraints, some training and testing details and results are provided in the supplement.

5.5.1 Synthetic datasets

(1) Demonstrating discriminative properties of TTN: We consider a two-class classification problem where the two classes are one-dimensional time series signals of length 100. Let us denote each sequence in the dataset by $X \in \mathbb{R}^{100}$. All the signals are Gaussian functions with varying amplitude. Signals in class 1 are centered at $t = 0.45$ while signals in class 2 are centered at $t = 0.55$. Further, we corrupt the function with additive Gaussian noise ($\mathcal{N}(0, 0.2)$). Samples of these functions are shown in Figure 5.3. We generate 8000 training and 2000 test sequences evenly balanced between classes 1 and 2. We use a simple classifier with a one-layer fully connected layer. The TTN is a 2-layer network with 1 convolutional layer producing 1 feature map with a filter of size 8, and 1 fully-connected layer. We train the networks for 10^3 iterations using Adam optimizer with an initial learning rate of 10^{-4} for the classifier. The weights of the TTN are updated at one-tenth the learning rate of the classifier. Figure 5.3 shows the test signals, corresponding outputs of the TTN, as well as the TTN-generated warping functions for every test input. It is clear from the figures that the TTNs predict class-specific warping functions in order to separate the peaks in the signals which makes them more discriminative. Note that this behavior arises automatically by minimizing the cross-entropy loss. In order to visualize the TTN outputs better, we perform post-processing by warping the TTN outputs with

	Vanilla	TTN
Unwarped	100.00 ± 0.00 %	**100.00 ± 0.00** %
Warped	96.31 ± 0.021 %	**99.03 ± 0.15** %

Table 5.1: Recognition Results (%) for Synthetic Dataset 2. Addition of TTN Clearly Outperforms the Baseline.

γ_μ^{-1}, where $\gamma_\mu = \sum_{i=1}^{N} \gamma_i$, where N is the size of the test set. This experiment clearly shows that TTNs are effective at increasing inter-class variations, as desired.

(2) Demonstrating rate-invariance of TTN: In this case, we construct a dataset such that rate variations in the signals are the major nuisance parameter. In this scenario, intuitively, minimizing classification error should lead to the following: different signals belonging to the same class, but differing (approximately) only by a γ should come closer to each other after passing through a trained TTN module. In class 1, we have signals which are a single Gaussian function with random warping applied and additive Gaussian noise added to them. Signals in class 2 are similar except that they a are mixture of two Gaussian functions. As before, we generated 8000 training sequences and 2000 test sequences evenly balanced between classes 1 and 2. These are shown before (Column 1) and after random warping (Column 2) in Figure 5.4. The TTN, classifier and the training and testing protocol are the same as in dataset (1) above. From column 3 in Figure 5.4, it is clear the TTN leads to reduction in intra-class rate variations. Table 5.1 shows the classification accuracies obtained with and without the TTN module (averaged over 10 runs). When no warping is present in the input data, both variants yield perfect accuracy. When warping is introduced in the dataset, the performance of the vanilla model (i.e., without TTN) drops significantly. With the addition of the TTN module, most of the lost performance can be recovered.

5.5.2 *ICL First-Person Hand Action dataset*

In this section, we conduct experiments on a recently released real-world dataset of hand actions [53]. The dataset contains 3D hand pose sequences with 21 joint locations per frame of 45 daily hand action categories interacting with 26 objects, such as "pour juice", "put tea bag" and "read paper". These sequences are performed by 6 subjects and are acquired using an accurate mocap system. For our experiments, we use the training/test splits suggested by the authors of the dataset [53], with subjects 1,3,4 used for training and the rest for testing. The training set contains 600 sequences and the test set contains 575 sequences. As the sequences are of varying lengths, we uniformly sample the sequences such that all sequences contain 50 samples. If the sequences are shorter than 50 samples, we use zero-padding. As there are 21 joints per frame, each input sequence is of dimension 50×63 ($21 \times 3 = 63$). We normalize the sequences such that the wrist position of the first frame is at the origin. We conduct our experiments with two different types of classifiers widely used for action recognition: (1) Temporal Convolutional Network (TCN) and (2) 2-layer LSTM, showing that the proposed TTN framework can yield better results for both the classifier architectures.

The TTN module consists of 3 FC layers with *tanh* nonlinearity and hidden states of dimensions 16 and 16. The final FC layer produces a vector of length 50 (equal to the input sequence length), with the first element set to zero (see Section 5.4).

The TCN architecture contains 1 temporal convolutional layer with 16, 32 or 64 feature maps, and 1 FC layer. We refer to these networks as TCN-16, TCN-32 and TCN-64 respectively. We run the algorithm 5 times and report the mean and standard deviation obtained in Table 5.2.

The LSTM architecture is similar to the one proposed in [53] containing two layers of LSTMs with a state dimension of 1024 and a dropout probability of 0.2. We use momentum optimizer with a momentum of 0.9 for training.

The results obtained are shown in Table 5.2. In addition to our experiments, we have reported results given in [53] for other important algorithms used for 3D pose-based action recognition including JOULE-pose [69], Moving Pose [175], Hierarchical Recurrent Neural Networks (HBRNN) [44], Transition Forests (TF) [52], and Lie Groups [159] and the Gram Matrix method [182], which also uses Dynamic Time Warping for sequence alignment. Among the baseline neural networks, TCN-32 led to the best results for this dataset, and addition of more layers did not yield better performance. We observe that addition of the TTN consistently improves performance over the baseline networks by at least 1% point (TCN-32) and up to 3.8 % points (TCN-16). These results also suggest that the TTN can recover some lost performance, when one reduces the number of parameters in the architecture. In the case of the LSTM classifier, we observe an improvement of 2.25% points using TTN + LSTM over just the LSTM.

Introducing distortions in the data: As datasets are usually collected in lab settings, there are relatively "clean" and do not contain many rate variations. Here, we introduce artificial rate variations in the data in order to better illustrate the usefulness of the TTN module. Here, we set the sequence length to 100 such that the original sequences of length 50 range from $t = 25$ to 75, and the rest of the values are set to zero. Now, we apply random "affine warps" to the training and test data. By an affine warp, we mean a warping function of the form $\gamma(t) = at + b$, $t = 25$ to 75, which is a linear time-scaling with an offset. We use $a \in [0.75, 1.25]$ and $b \in 0, 1, \ldots, 49$.

We observe that the induced distortion leads to a huge drop in performance of TCN-32 from 81.74% to 70.43 %. With the TTN, the performance drop is much

lower – from 82.75 to 78.26% and TTN+TCN-32 performs about 8% points higher than TCN-32. Furthermore, from Figure 5.5, which shows the inputs, generated warping functions and the TTN outputs, it can be readily observed that the TTN performs class-aware alignment of the sequences which then makes the classification problem much easier. This experiment shows that addition of the TTN enhances the interpretability of the network, and also delivers superior performance when there are larger rate variations are present in the data.

5.5.3 NTU RGB-D dataset

In this section, we conduct experiments on a large-scale dataset of human actions called the NTU RGB-D dataset [142] which contains about 56000 sequences of 3D skeleton positions acquired by a Microsoft Kinect. 25 joint locations are provided for each skeleton. The dataset contains actions belonging to 60 human activities performed by 45 subjects, with some actions containing two actors. The data are acquired using a Microsoft Kinect. We sample 50 frames per sequence uniformly. We conduct two sets of experiments for this dataset – Cross Subject (CS) and Cross View (CV) – as per the protocol suggested by the authors in [142] using the same training and testing splits.

We construct a TTN module with 2 temporal convolutional layers and 3 FC layers with ReLU non-linearity. We use a filter size of 8 and 16 output feature maps in each conv layer. The FC layers produce hidden representations of sizes 16, 16 and 50 respectively. The TTN module is trained with a learning rate that is one-tenth of the learning rate for the parameters of the network.

We use the Temporal Convolution Network (TCN) described in [90]. The network consists of 10 convolutional layers with batch normalization and ReLU non-linearity.

Method	Accuracy (%)
Moving Pose [175]	56.34
JOULE-pose [69]	74.60
HBRNN [44]	77.40
TF [52]	80.69
Lie Group [159]	82.69
Gram Matrix [182]	***85.39***
2-layer LSTM	76.17
2-layer LSTM + TTN	**78.43**
TCN-16	76.28 ± 0.29
TCN-16 + TTN	**80.14 ± 0.33**
TCN-64	79.10 ± 0.76
TCN-64 + TTN	**81.32 ± 0.36**
TCN-32	81.74 ± 0.27
TCN-32 + TTN	**82.75 ± 0.31**
TCN-32 (affine warp)	70.43
TCN-32 + TTN (affine warp)	**78.26**

Table 5.2: Action Recognition Results on the ICL Hand Action Dataset Showing that LSTM+TTN and TCN+TTN Frameworks Consistently Outperform LSTM and TCN Baselines.

Method	CS (%)	CV (%)
Lie Groups [159]	50.08	52.76
FTP Dynamic Skeletons [69]	60.23	65.22
HBRNN [44]	59.07	63.97
2-layer LSTM [142]	60.69	67.29
2-layer part-LSTM [142]	62.93	70.27
STA-LSTM [146]	73.40	81.20
VA-LSTM [181]	79.40	87.60
STA-GCN [171]	***81.50***	***88.30***
TCN [90]	76.54	83.98
TCN + TTN	**77.55**	**84.25**

Table 5.3: Action Recognition Results on the NTU RGB-D Dataset Showing that TCN+TTN Frameworks Outperforms the TCN.

While training, the TTN parameters are updated at one-tenth the learning rate of the TCN.

The results obtained for this dataset are shown in Table 5.3. For cross-subject experiments, we observe from the table that the addition of the TTN module results in a performance improvement of about 1 percentage point over the baseline TCN. We also found that by using 2 parallel TTNs and concatenating the TTN outputs results in further improvement with a final performance of 77.80%. The addition of the TTN module leads to lesser improvement in the case of cross-view experiment. This can be explained by the fact there are likely fewer rate variations in the case of cross-view protocol compared to cross-subject.

Method	w/o TTN (%)	w/ TTN (%)
TCN (4 layers)	70.72	**71.63**
TCN (7 layers)	75.06	**75.30**
TCN (10 layers)	76.54	**77.55**

Table 5.4: Ablation results on the TCN for the NTU database. Cross-subject action recognition results show that the TTN+TCN consistently performs better than TCN for different sizes of TCN.

We study the effect of the number of layers in the classifier network on the performance. The TCN architecture consists of 3 blocks of conv layers. We remove 1 block at a time and compare the cross-subject classification rate. The results are shown in Table 5.4. We see that the addition of the TTN produces better results in all cases.

5.6 Discussion and Future Work

In this work, we have proposed the Temporal Transformer Network (TTN) which can be readily integrated into classification pipelines. TTN has the ability to generate rate-invariant as well as discriminative warping functions for general time-series classification. We have shown improved classification results using different types of classifiers – TCNs and LSTMs – on challenging 3D action recognition datasets acquired using different modalities – Kinect and mocap. We have demonstrated the rate-invariant and discriminative properties of the TTN.

In the future, we would like to apply the ideas presented in this chapter to video action recognition. However, it is not immediately clear how to perform temporal warping for videos as interpolation of video frames or their features may not correspond to the true frame interpolation. One possible solution is jointly train the image-level features and the action classification pipeline jointly along with the TTN

module. Temporal transformers can also be applied in general time-series classification which includes recognition from wearables, speech, etc. Unsupervised pattern discovery with inbuilt warp-invariant metrics will be another interesting direction for further research.

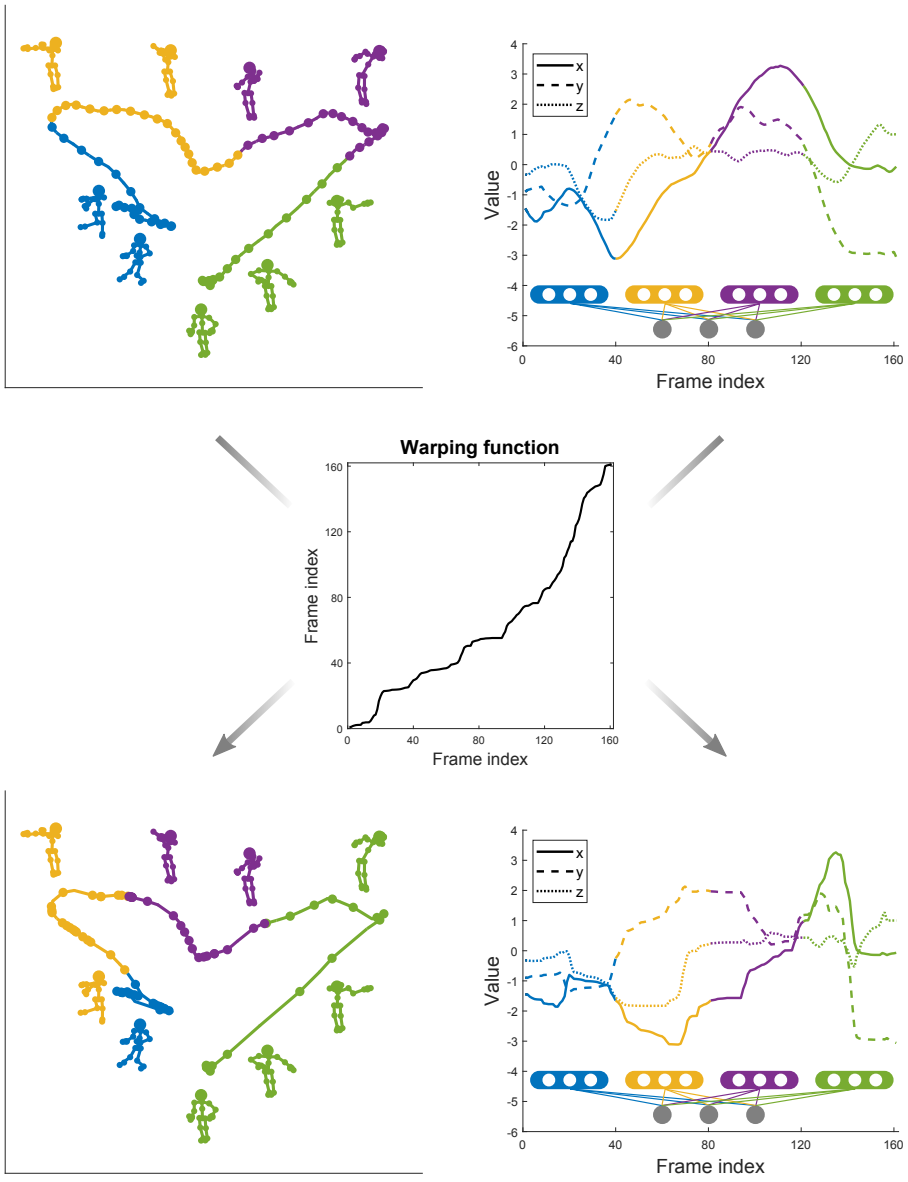


Figure 5.2: Top-left, Bottom-left: the trajectories and sampling points of the same action ("wearing jacket") before and after a time warping (Center). The trajectories are visualized in $\mathbb{R}^3$ by using the sums of x, y, z coordinates of all joints. Notice how the time series of x, y, z, which are the inputs of a neural network (Top-right and Bottom-right), are quite different despite the action being the same. Here the action is arbitrarily divided into 4 segments, shown in different colors, to highlight the rate variation.

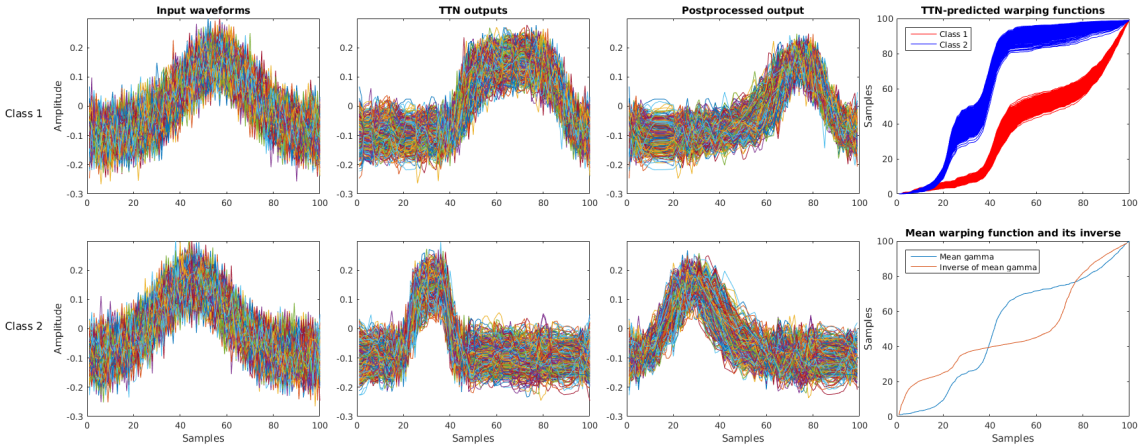


Figure 5.3: Results on synthetic dataset 1. Rows 1 and 2 show waveforms corresponding to classes 1 and 2 respectively. Columns 1 and 2 show the test inputs and the TTN outputs respectively. It is clear by comparing these columns that the TTN outputs are much better discriminated after warping. The TTN-predicted warping functions also show that the TTN performs class-dependent warping. Column 3 is a better visualization of column 2 after some post-processing (see text).

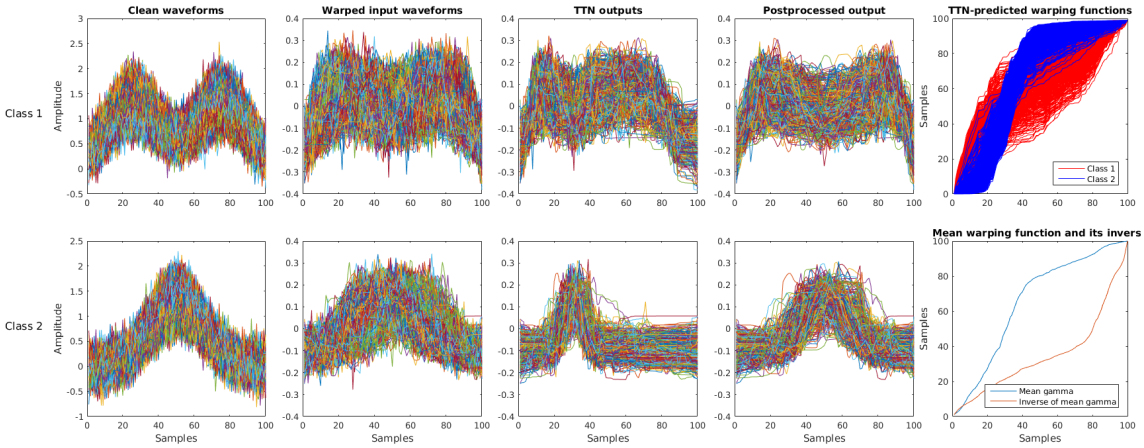


Figure 5.4: Results on synthetic dataset 2. Rows 1 and 2 show waveforms corresponding to classes 1 and 2 respectively. Columns 1, 2 and 3 show the clean waveforms, test inputs(after random warping) and the TTN outputs respectively. It is clear by comparing these columns that the TTN outputs are much more closely clustered especially for class 2, showing that the TTN outputs are robust to rate-variations. Column 4 is a better visualization of column 3 after some post-processing (see text).

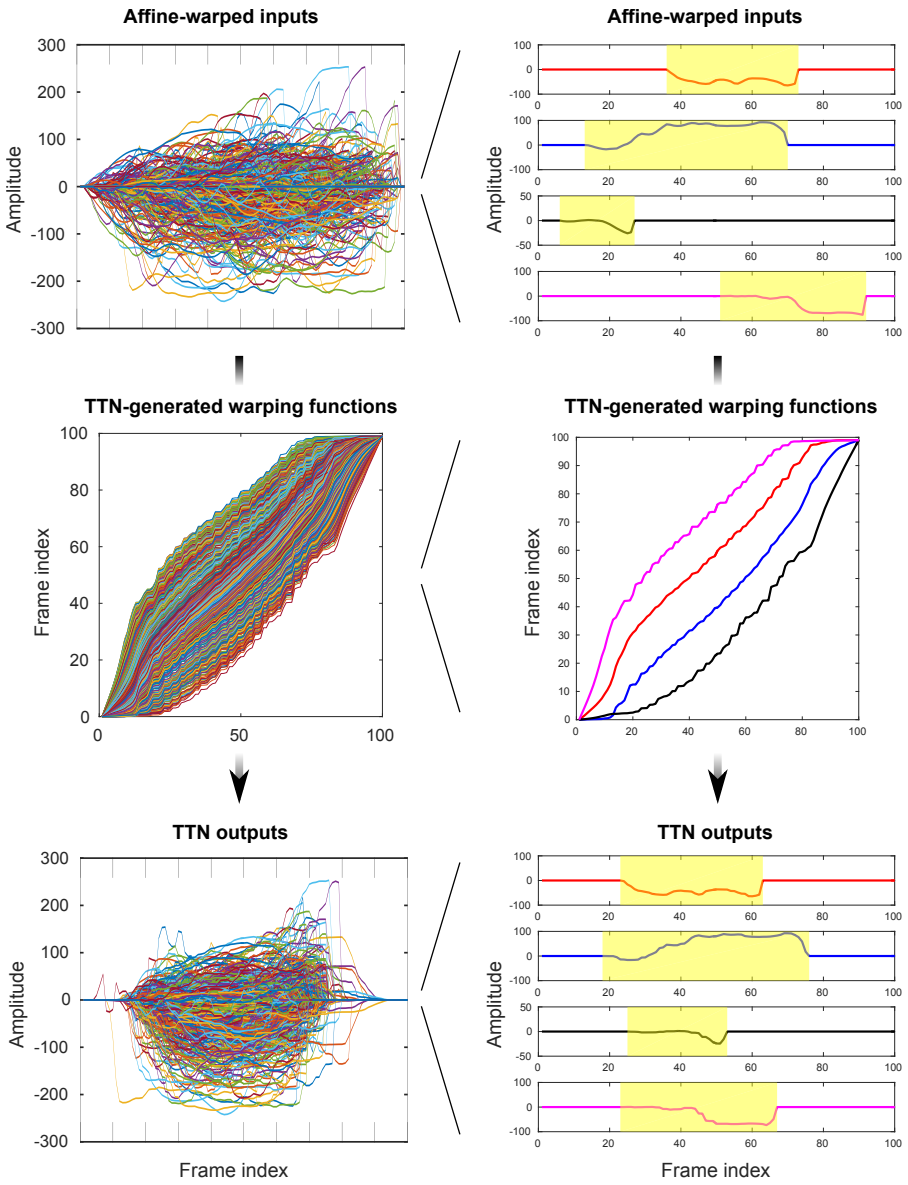


Figure 5.5: Visualizations of results of TCN-32 + TTN on ICL action dataset with induced rate variations. In the left column are shown waveforms corresponding to joint 1 of all test sequences. In the right column, 4 of those sequences are shown for clarity. We see clearly that the generated warping functions undo the affine-warp distortion in the test data, and the TTN outputs are nearly perfectly aligned leading to much better classification results.

Chapter 6

DEEP NON-LINEAR REGRESSION ON RIEMANNIAN MANIFOLDS

6.1 Introduction

Many applications in computer vision employ data that are naturally represented on manifolds [115]. Shapes that are invariant to affine transforms [13] and linear dynamical systems [158] can be represented as points on the Grassmannian. In diffusion tensor imaging, each "pixel" of the "image" is a symmetric positive definite (SPD) matrix and the space of SPD matrices forms a manifold [135]. Lie groups like $SO(3)$ and $SE(3)$ are used to represent human skeletons [159, 160]. Predicting probability density functions is another area of interest, applicable to multi-class classification and bag-of-words models [98], and saliency prediction [78].

Several years of research has presented us with various tools for statistics, and thereby machine learning approaches to be deployed when the objects of interest have manifold-valued domains (c.f. [151]). In deep learning, it is usually the case that data samples are viewed as elements of vector spaces. Any additional structure that the data may possess is left to be learned through the training examples. However, recently, there has been interest in employing deep learning techniques for non-Euclidean inputs as well: [17] including graph-structured data [18, 65, 130] and 3D shapes viewed as Riemannian manifolds [118]. Also, deep networks that preserve the input geometry at each layer have been studied for inference problems, e.g., for symmetric positive definite matrices [70], Lie groups [71] and points on the Stiefel manifold [72]. Another recent work considers weight matrices which are constrained to be orthogonal, i.e., points of the Stiefel manifold, and propose a generalized version

of backpropagation [62]. These works do not consider output variables with geometric constraints.

In contrast to the above, instead of enforcing geometry at the inputs, our goal is to design a general framework to extend neural network architectures where output variables (or deeper feature maps) lie on manifolds of known geometry, typically due to certain invariance requirements. We do not assume the inputs themselves have known geometric structure and employ standard back-propagation for training. Equivalently, one may consider this approach as trying to estimate a mapping from an input $\mathbf{x} \in \mathbb{R}^N$ to a manifold-valued point $\mathbf{m} \in \mathcal{M}$ i.e., $f : \mathbb{R}^N \rightarrow \mathcal{M}$, using a neural network, where $\mathbf{m}$ is the desired output.

That is, this chapter provides a framework for regression that is applicable to predicting manifold-valued data and at the same time is able to leverage the power of neural nets for feature learning, using standard backpropagation for unconstrained optimization. In this chapter, we focus on two manifolds that are of wide interest in computer vision – the hypersphere and the Grassmannian. We describe the applications next.

Face $\rightarrow$ Illumination Subspace as regression on the Grassmannian: The *illumination subspace* of a human face is a popular example from computer vision where for a particular subject, the set of all face images of that subject under all illumination conditions can be shown to lie close to a low dimensional subspace [60]. These illumination subspaces are represented as points on the Grassmannian (or Stiefel, depending on application) manifold. Several applications such as robust face recognition have been proposed using this approach. In this work, in order to demonstrate how deep networks can be employed to map to Grassmannian-valued data, we consider the problem of estimating the illumination subspace from a single input image of a sub-

ject under unknown illumination. We refer to this application as Face→Illumination Subspace (F2IS).

Multi-class classification as regression on the unit hypersphere: Classification problems in deep learning use the softmax layer to map arbitrary vectors to the space of probability distributions. However, more formally, probability distributions can be easily mapped to the unit hypersphere, under a square-root parametrization [147] inspired by the Fisher-Rao metric used in information geometry. Thus, multi-class classfication can be posed as regression to a hypersphere. Indeed, there has been work recently that consider *spherical-loss functions* which use the Euclidean loss on unit-norm output vectors of a network [161, 38]. In this work, we propose loss-functions for the classification problem based on the geometry of the sphere.

Main contributions: In this chapter, we address the training of neural networks using standard backpropagation to output elements that lie on Riemannian manifolds. To this end, we propose two frameworks in this chapter:

(1) We discuss how to map to simpler manifolds like the hypersphere directly using a combination of geodesic loss functions as well as differentiable constraint satisfaction layers such as the normalization layer in the case of the hypersphere.

(2) We also propose a more general framework that is applicable to Riemannian manifolds that may not have closed-form expressions for the geodesic distance or when the constraints are hard to encode as a layer in the neural network. In this framework, the network maps to the tangent space of the manifold and then the exponential map is employed to find the desired point on the manifold.

We carry out experiments for the applications described above in order to evaluate the proposed frameworks and show that geometry-aware frameworks result in improved performance compared to baselines that do not take output geometry into account.

6.2 Related Work

We will now point to some related work that also examine the problem of predicting outputs with geometric structure using neural networks. Byravan and Fox [19] and Clark et al. [34] design deep networks to output $SE(3)$ transformations. The set of transformations $SE(3)$ is a group which also possesses manifold structure, i.e., a Lie group. It is not straightforward to predict elements on $SE(3)$ since it involves predicting a matrix constrained to be orthogonal. Instead, the authors map to the Lie algebra $se(3)$ which is a linear space. We note that the Lie algebra is nothing but the tangent space of $SE(3)$ at the identity transformation and can be considered a particular case of the general formulation presented in this chapter. Huang et al. [71] use the logarithm map to map *feature maps* on $SE(3)$ to $se(3)$ before using regular layers for action recognition. However, the logarithm map is implemented within the network, since for $SE(3)$, this function is simple and differentiable. In contrast, in this work, we require the network *output* to be manifold-valued and thus do not impose any geometry requirements at the input or for the feature maps. This also means that a suitable loss function needs to be defined, taking into account, the structure of the manifold of interest.

In a more traditional learning setting, there has been work using *geodesic regression*, a generalization of linear regression, on Riemannian manifolds [51, 50, 143, 68, 89], where a geodesic curve is computed such that the average distance (on the manifold) from the data points to the curve is minimized. This involves computing gradients on the manifold. Recent work has also included non-linear regression on

Riemannian manifolds [11, 10]. Here, the non-linearity is provided by a pre-defined kernel function and the mapping algorithm solves an optimization problem iteratively. Our work is a non-iterative deep-learning based approach to the problem described in Banerjee et al. [11] as regression with the independent variable in $\mathbb{R}^N$ and the dependent variable lying on a manifold $\mathcal{M}$. That is, unlike these works, the mapping $f : \mathbb{R}^N \to \mathcal{M}$ in our case is a hierarchical non-linear function, learned directly from data without any hand-crafted feature extraction, and the required mapping is achieved by a simple feed forward pass through the trained network.

All neural nets in the chapter are trained and tested using Tensorflow [1], making use of its automatic differentiation capability.

6.3 Two Approaches for Deep Manifold-Aware Prediction

We propose two ways of predicting manifold-valued data using neural networks with standard backpropagation. See Figure 6.1 for visual illustration and important notation.

Mapping to the manifold via geodesic-loss functions: In this case, the network directly maps input vectors to elements on the manifold $\mathcal{M}$ and is required to learn the manifold constraints from the data. If we represent the neural network as a mapping `NN`, we have `NN`: $\mathbb{R}^N \to \mathcal{M}$. Firstly, unlike simple manifolds like the sphere, manifolds in general do not have a differentiable closed-form expression, that are also efficiently computable, for the geodesic distance function that can be used as a loss function for the neural network. Although one can still resort to using a differentiable loss function such as the Euclidean distance, this approach is not mathematically correct and does not yield the right estimate for distance on the manifold. Secondly, the network output has to satisfy the manifold constraints. In the case of

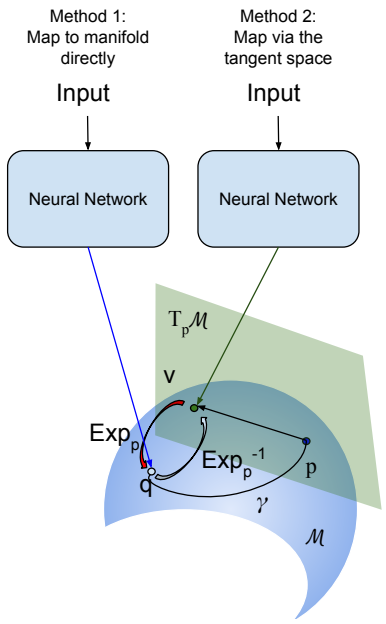


Figure 6.1: This figure illustrates the two approaches presented in this chapter for training neural networks to predict manifold-valued data. It also explains some basic concepts from differential geometry visually. $\mathcal{M}$ is a manifold, $T_P\mathcal{M}$ is the tangent space at $p \in \mathcal{M}$. p is called the pole of the tangent space. The curve connected p and $q \in \mathcal{M}$ is the geodesic curve γ. v is a point on $T_p\mathcal{M}$ such that the exponential map $\exp_p(v) = \gamma(1)$ and the logarithm map $\exp_p^{-1}(q) = v$.

the sphere, it is simple to enforce the unit-norm constraint at the output layer of the neural network using a differentiable normalization layer. It is however less clear how to map to more complicated manifolds such as the Stiefel and Grassmann manifolds where the points are usually represented by tall-thin orthonormal matrices. That is, in addition to the unit-norm constraints, orthogonality constraints between all pairs of columns in the matrix need to be enforced. The Grassmann manifold, presents a more difficult challenge, since each point in this space is an equivalence class of points on the Stiefel manifold that are orthogonal transforms of each other. As we will see

later, the data representation that respects this equivalence (projection matrix) does not admit a feasible way for a neural network to map to this manifold directly.

Mapping via the tangent space – toward a general framework: This is a more general formulation that is applicable to all the manifolds of interest. Here, the network first maps to a vector on the tangent space constructed at a suitable pole $p \in \mathcal{M}$, which forms the intermediate output. Once the network outputs the required tangent, the exponential map ($\exp_p$) is employed to find the corresponding point on the manifold. Mathematically, we decompose the desired function $f : \mathbb{R}^N \to \mathcal{M}$ as $f = \exp_p \circ \texttt{NN}$ and $\texttt{NN}$: $\mathbb{R}^N \to T_p\mathcal{M}$. Intuitively, since the tangent space is a vector space that encodes geometric constraints implicitly, it is attractive here, as neural networks have been shown to be effective for estimating vector-valued data. We note that an assumption is implicit in this framework: all the data points of interest on the manifold are much closer to p than the cut-locus of the manifold and in this case, the distance on the tangent space serves as a good approximation to the geodesic distance. This is the same assumption that goes into currently successful approaches for statistical computing methods on manifolds [135]. In practice, we find this assumption is respected in our applications as well.

6.4 Deep Regression on the Grassmannian for F2IS

Face → Illumination Subspace: We will now describe an ill-posed inverse problem from computer vision that serves as our canonical application to illustrate prediction on the Grassmann manifold using a neural network. It is well known that the set of images of a human face in frontal pose under all illuminations lies close to a low-dimensional subspace, known as the illumination subspace [60, 48]. If we compute the eigenvectors of this set of images for different subjects using PCA, we observe

that the top 5 principal components (PCs) capture nearly 90% of the variance. More importantly, for this chapter, an obvious pattern can be observed between the subject under consideration and the PCs of the illumination subspace. Firstly, the identity of the subject can be easily determined from the PCs. Secondly, as noted by Hallinan [60], we observe that the illumination patterns of top 5 principal components are the same across subjects only up to certain permutations and sign flips. [1] Using the terminology in [60], we can interpret the visualizations of the top 5 PC's as a face under the following respective illuminations: *frontal lighting, side lighting, lighting from above/below, extreme side lighting and lighting from a corner*. The 1^{st} and 2^{nd} PC's have eigenvalues in a similar range and sometimes exchange places depending on the subject. The 3^{rd} PC, corresponding to eigenvalue is always at the same place. The 4^{th} and 5^{th} PCs have eigenvalues in a similar range and can interchange places for a few subjects.

The illumination subspace refers to the linear *span* of these eigenvectors, and is a point on the Grassmannian. **When we represent the subspace by its projection matrix representation, the representation becomes invariant to both sign flips and permutations** (in fact, invariant to the full set of right orthogonal transforms).

In this chapter, as an example of predicting points on Grassmann manifold, we define the following ill-posed inverse problem: given a human face in frontal pose under an unknown illumination, output the corresponding illumination subspace. We will refer to this problem as the "Face $\rightarrow$ Illumination Subspace" problem or F2IS. In our expts., we consider the illumination subspace to be of dimension $d = 3, 4$,or 5.

[1] It is clear that for an eigenvector $\mathbf{e}$, $-\mathbf{e}$ is also an eigenvector.

Geometry of the Grassmannian: The Grassmann manifold, denoted by $\mathcal{G}_{n,d}$, is a matrix manifold and is the set of d-dimensional subspaces in $\mathbb{R}^n$. To represent a point on $\mathcal{G}_{n,d}$, we can use an orthonormal matrix, $\mathbf{U} \in \mathbb{R}^{n\times d}$ ($\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$), to represent the equivalence class of points in $\mathbb{R}^{n\times d}$, such that, two points are equivalent if their columns span the same d-dimensional subspace. That is, $\mathcal{G}_{n,d} = \{[\mathbf{U}]\}$, where $[\mathbf{U}] = \{\mathbf{UQ}|\mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{Q} \text{ is orthogonal}\}$. In order to uniquely represent the equivalence class $[\mathbf{U}] \in \mathcal{G}_{n,d}$, we use its projection matrix representation $\mathbf{P} = \mathbf{U}\mathbf{U}^T \in \mathbb{R}^{n\times n}$, where $\mathbf{U}$ is some point in the equivalence class. $\mathbf{U}\mathbf{U}^T$ contains $\frac{n(n+1)}{2}$ unique entries as it is a symmetric matrix. Clearly, for any other point in the same equivalence class $\mathbf{UQ}$, its projection matrix representation is $(\mathbf{UQ})(\mathbf{UQ})^T = \mathbf{U}\mathbf{U}^T$, as required. **Thus, the space of all rank d projection matrices of size $n \times n$, $\mathcal{P}_n$ is diffeomorphic to $\mathcal{G}_{n,d}$.** The identity element of $\mathcal{P}_n$ is given by $\mathbf{I}_{\mathcal{P}_n} = diag(\mathbf{I}_d, \mathbf{0}_{n-d})$, where $\mathbf{0}_{n-d}$ is the matrix of zeros of size $(n-d) \times (n-d)$. In order to find the exponential and logarithm maps for $\mathcal{G}_{n,d}$, we will view $\mathcal{G}_{n,d}$ as a quotient space of the orthogonal group, $\mathcal{G}_{n,d} = \mathcal{O}_n/(\mathcal{O}_{n-d} \times \mathcal{O}_d)$. The Riemannian metric in this case is the standard inner product [46] and thus, the distance function induced on the tangent space is the Euclidean distance function. Using this formulation, given any point $\mathbf{P} = \mathbf{U}\mathbf{U}^T \in \mathcal{P}_n$, a geodesic of $\mathcal{P}_n$ at $\mathbf{I}_{\mathcal{P}_n}$ passing through $\mathbf{P}$ at $t = 0$, is a particular geodesic $\alpha(t)$ of $\mathcal{O}(n)$ completely specified by a skew-symmetric $\mathbf{X} \in \mathbb{R}^{n\times n}$: $\alpha(t) = \exp_m(t\mathbf{X})I_P \exp_m(-t\mathbf{X})$, where $\exp_m(.)$ is the matrix exponential and $\mathbf{P} = \alpha(1)$, such that $\mathbf{X}$ belongs to the set M given by $M = \left\{ \begin{bmatrix} \mathbf{0}_d & \mathbf{A} \\ -\mathbf{A}^T & \mathbf{0}_{n-d} \end{bmatrix} \mid \quad \mathbf{A} \in \mathbb{R}^{d\times(n-d)} \right\}$. $\mathbf{X}$ serves as the tangent vector to $\mathcal{G}_{n,d}$ at the identity and is completely determined by $\mathbf{A}$. The geodesic between two points $\mathbf{P}_1, \mathbf{P}_2 \in \mathcal{P}_n$, is computed by rotating $\mathbf{P}_1$ and $\mathbf{P}_2$ to $\mathbf{I}_{\mathcal{P}_n}$ and some $\mathbf{P} \in \mathcal{P}_n$ respectively. The exponential map, takes as inputs, the pole and the tangent vector and returns the subspace $span(\mathbf{U})$, represented by some point $\mathbf{UQ}$. Refer Srivastava

and Klassen [148] and Taheri et al. [153] for algorithms to compute exponential and log maps for the Grassmannian.

Synthetic dataset for F2IS: We use the Basel Face Model dataset [134] in order to generate 250 random 3D models of human faces $\{S_i\}, i = 1 \dots 250$ (200 for training and 50 for testing chosen randomly). [2] We then generate a set of 64 faces for each subject where each face is obtained by varying the direction of the point source illumination for the frontal pose i.e., $S_i = \{\mathbf{F}_i^j\}, j = 1 \dots 64$. The directions of illumination are the same as the ones used in the Extended Yale Face Database B [54]. Each face image is converted to grayscale and resized to 28×28. Once we have the 250 sets of faces under the 64 illumination conditions, we calculate the illumination subspace for each subject as follows. For each subject, we first subtract the mean face image of that subject under all illumination conditions and then calculate the principal components (PCs). For a subject i and an illumination condition j, we will denote the input face image by $\mathbf{F}_i^j$ and the desired d top PCs by $\mathbf{E}_i^1, \mathbf{E}_i^2, \dots, \mathbf{E}_i^d$ (note that the PCs do not depend on the input illumination condition). It is clear that every $\mathbf{E}_i^k, k = 1, 2, \dots, d$ is of size 28×28 and $\langle \mathbf{E}_i^k, \mathbf{E}_i^l \rangle = 1$, if $k = l$ and 0 otherwise.

If we lexicographically order each $\mathbf{E}_i^k$ to form a vector $vec(\mathbf{E}_i^k)$ of size 784×1 and for each subject, arrange the $\mathbf{E}_i^k$'s to form a matrix $\mathbf{U}_i = [vec(\mathbf{E}_i^1)\ vec(\mathbf{E}_i^2) \dots vec(\mathbf{E}_i^d)]$, then the orthonormality constraint can be rewritten as $\mathbf{U}_i^T \mathbf{U}_i = \mathbf{I}_d$, where $\mathbf{I}_d$ is the identity matrix of size $d \times d$. As we argued earlier, due to the nature of the problem, $\mathbf{U}_i$ should be represented as a point on the Grassmann $\mathcal{G}_{784,d}$ using the projection matrix representation. With this notation, the desired mapping is $f : \mathbb{R}^{28 \times 28} \to \mathcal{G}_{784,d}$ such that $f(\mathbf{F}_i^j) = \mathbf{U}_i \mathbf{Q} \in [\mathbf{U}_i]$, the required equivalence class or equivalently, $\mathbf{U}_i \mathbf{U}_i^T$.

[2] We use a synthetic dataset because we were unable to find any large publicly available real database that would enable training of neural networks without overfitting.

For the inputs $\mathbf{F}_i^j$ during training and testing, we do not use all the illumination directions ($j's$). We only use illumination directions that light at least half of the face. This is because for extreme illumination directions, most of the image is unlit and does not contain information about the identity of the subject, which is an important factor for determining the output subspaces. We select the same 33 illumination directions for all subjects to form the inputs for the network. We randomly split the dataset into 200 subjects for training and 50 subjects for testing. Therefore there are $33 \times 200 = 6600$ and $33 \times 50 = 1650$ different input-output pairs for training and testing respectively. The 33 illumination directions used for creating inputs for both the training and test sets are a subset of the illumination directions used in the Extended Yale Face Database B [54].

6.4.1 Proposed frameworks for solving F2IS

We propose two frameworks which employ networks with nearly the same architecture: The network consists of 3 `conv` layers and two `fc` layers. ReLU non-linearity is employed. Each `conv` layer produces 16 feature maps. All the filters in the `conv` layers are of size 11×11. The first `fc` layer produces a vector of size 512. Size of the second `fc` layer depends on the framework. Both networks are trained using the Adam optimizer [92] using a learning rate of 10^{-3} for 50000 iterations with a mini-batch size of 30. Euclidean loss between the desired output and ground truth is employed in both cases. We show that the choice of representation of the desired output is crucial in this application. We carry out three sets of experiments using subspace dimension $d = 3, 4$ and 5.

Baseline: The first framework is a baseline that attempts to directly map to the desired PCs represented as a matrix $\mathbf{U}_i$, given $\mathbf{F}_i^j$, i.e., $\texttt{NN}(\mathbf{F}_i^j) = \mathbf{U}_i$. We use the

Euclidean loss function between the ground-truth $\mathbf{U}_i$ and the network output $\hat{\mathbf{U}}_i$ for training: $L_b = ||\mathbf{U}_i - \hat{\mathbf{U}}_i||_F^2$ That is, instead of regressing to the desired subspace, the network attempts to map to its basis vectors (PCs). However, the mapping from $\mathbf{F}_i^j$ to $\mathbf{U}_i$ is consistent across subjects **only up to certain permutations and sign flips in the PCs**. Hence, without correcting these inconsistencies ad hoc, the problem is rendered too complicated, since during the training phase, the network receives conflicting ground-truth vectors depending on the subject. Thus, this framework performs poorly as expected. It is important to note that mapping to the correct representation $\mathbf{U}\mathbf{U}^T$ (which respects Grassmann geometry and is invariant to these inconsistencies) directly is not feasible because the size of $\mathbf{U}\mathbf{U}^T$ is too large ($\frac{784\times785}{2}$) and has rank constraints. This necessitates mapping via the tangent space which is discussed next.

Mapping via the Grassmann tangent space – GrassmannNet-TS: The second framework represents the output subspaces as points on the Grassmann manifold and first maps to the Grassmann tangent space and then computes the required subspace using the Grassmann exponential map. This circumvents the problem of very large dimensionality encountered in the first approach since the tangent vector has a much smaller intrinsic dimensionality. This representation has a one-to-one mapping with the projection matrix representation and thus, is naturally invariant to the permutations of the PCs and all combinations of sign flips present in the data. And the mapping we intend to learn becomes feasible in a data-driven framework. Mathematically, $\texttt{NN}$: $\mathbb{R}^{784\times d} \rightarrow T_p\mathcal{G}_{784,d}$.

As shown in Section 6.4, a tangent at some pole p is given by the matrix $\mathbf{X}$, which in turn is completely specified by the matrix $\mathbf{A} \in \mathbb{R}^{(784-d)\times d}$, a much smaller matrix. Therefore, we design a network to map an input face $\mathbf{F}_i^j$ to the desired matrix $\mathbf{A_i}$.

Input	Ground-truth PCs	Output of baseline n/w	Output of GrassmannNet-TS
			
		$D_G = 1.6694$	$D_G = 0.7006$
			
		$D_G = 1.2998$	$D_G = 0.7238$
			
		$D_G = 0.7797$	$D_G = 0.5966$
			
		$D_G = 1.5355$	$D_G = 0.6170$
			
		$D_G = 1.6760$	$D_G = 0.4420$
			
		$D_G = 1.6703$	$D_G = 0.4939$

Table 6.1: Test results for two input images using $d = 5$. We can clearly observe that the GrassmannNet-TS (with pole $\mathbf{U}_{Fr}^{d}$) framework performs much better than the baseline that attempts to regress directly to the PCs. The numbers below the output images indicate the subspace distance from the ground truth (lower the better). Note that the outputs need not be exactly the same as the groundtruth PCs since the quantity of interest is the subspace spanned by the groundtruth PCs.

Input	Ground-truth PCs	Output of baseline n/w	Output of GrassmannNet-TS
			
		$D_G = 1.9400$	$D_G = 0.5489$
			
		$D_G = 1.2735$	$D_G = 0.6245$
			
		$D_G = 0.6750$	$D_G = 1.1580$
			
		$D_G = 1.2047$	$D_G = 0.6674$
			
		$D_G = 0.6225$	$D_G = 0.3653$
			
		$D_G = 1.5900$	$D_G = 0.9339$

Table 6.2: Test results for two input images using $d = 4$. As in the case of $d = 5$, GrassmannNet-TS (with pole $\mathbf{U}_{Fr}^{d}$) framework performs much better than the baseline that attempts to regress the PCs directly.

Input	Ground truth PC's	Output of baseline n/w	Output of GrassmannNet-TS
			
		$D_G = 0.5766$	$D_G = 0.4854$
			
		$D_G = 0.7095$	$D_G = 0.4787$
			
		$D_G = 0.4429$	$D_G = 0.4009$
			
		$D_G = 1.0904$	$D_G = 0.3042$
			
		$D_G = 0.9012$	$D_G = 0.3118$
			
		$D_G = 0.9244$	$D_G = 0.3294$

Table 6.3: Test results for six input images using $d = 3$. From the figures, we can clearly observe that the GrassmannNet-TS (with the Fréchet mean of the training set as the pole) framework performs much better than the baseline that attempts to regress the PCs directly. The numbers below the output images indicate the subspace distance from the ground truth (lower the better).

.

An training pair can be represented as $(\mathbf{F}_i^j, \mathbf{A}_i)$ and the let the output of the network be the vectorized version of a matrix $\hat{\mathbf{A}}_i \in \mathbb{R}^{(784-d)\times d}$. The $\mathbf{A}_i$'s are computed using the Grassmann logarithm map in [148]. We also note that $\mathbf{A}$ does not possess any additional structure to be enforced and thus a neural network can be easily trained to map to this space using just the Euclidean loss between $\mathbf{A}_i$ and the network output $\hat{\mathbf{A}}_i$: $L_G = ||\mathbf{A}_i - \hat{\mathbf{A}}_i||_F^2$.

Pole of tangent space: This is a design choice and we conduct experiments with two different poles:

(1) We compute the the illumination subspace of the entire training set. We will denote these PCs by $\mathbf{E}_{Tr}^k, k = 1, 2, \ldots, d$ and the corresponding matrix representation by $\mathbf{U}_{Tr}^d$, which forms the pole of the Grassmann tangent space.

(2) It is common practice to use the Fréchet (also known as geometric or Karcher) mean as the pole of the tangent space. We compute the Fréchet mean of the ground-truth subspaces of the training set using the iterative algorithm given by Turaga et al. [158]. We denote this pole as $\mathbf{U}_{Fr}^d$.

During the testing phase, using the output $\hat{\mathbf{A}}$ matrix, we employ the exponential map for a given pole to find the corresponding point on the Grassmann manifold. This framework of first mapping to the Grassmann tangent space using a network and then to the corresponding subspace using the Grassmann exponential map is referred to as **GrassmannNet-TS**.

6.4.2 Experimental Results for F2IS

For the frameworks described in Sections 6.4.1, we describe the results on the test set of F2IS here. We compute the distance between predicted and ground-truth subspace as the measure to quantify the efficacy of the proposed frameworks. Various

Subspace Dim d	Baseline	GrassmannNet-TS	
		Pole = $\mathbf{U}^d_{Tr}$	Pole = $\mathbf{U}^d_{Fr}$
3	0.6613	0.3991	0.3953
4	1.0997	0.5489	0.5913
5	1.4558	0.8694	0.6174

Table 6.4: Mean geodesic distance between predictions and ground-truth on the test set using the proposed frameworks. GrassmannNet-TS expectedly provides excellent results compared to the baseline framework for all subspace dimensions. Note that the max $D_G(.)$ possible for $d = 3, 4$ and 5 are $2.72, 3.14$ and 3.51 respectively

measures exist that quantify this notion based on principal angles between subspaces [61]. We use the Grassmann geodesic distance. For two subspaces represented by $\mathbf{U}_1$ and $\mathbf{U}_2 \in \mathcal{G}_{n,d}$, the geodesic distance is given by $D_G(\mathbf{U}_1, \mathbf{U}_2) = \left(\sum_{i=1}^{d} \theta_i^2 \right)^{1/2}$, where θ_i's are the principal angles obtained by the SVD of $\mathbf{U}_1^T \mathbf{U}_2 = \mathbf{W}(\cos \mathbf{\Theta}) \mathbf{V}^{\mathbf{T}}$, where $\cos \Theta = diag(\cos \theta_1, \ldots, \cos \theta_d)$. We use the implementation in [94] for computing the principal angles. We report the arithmetic mean of this distance measure for the entire test set. Note that the maximum value of $D_G(.)$ is $\frac{\pi\sqrt{d}}{2}$.

The results based on the mean subspace distance on the test set for the proposed frameworks for different values of the subspace dimension d are presented in Table 6.4. The baseline, as expected, performs poorly. This is because during the training phase, the network received conflicting ground-truth information because of the permutation and sign flips inherently present in the data. On the other hand, GrassmannNet-TS yields excellent performance as it is invariant to these transformations by design. We reiterate that this is possible only in the case of regression directly on the Grassmann tangent space since mapping to the Grassmann manifold is infeasible because of the

very large number of variables required to represent the projection matrix. The outputs of the baseline as well as GrassmannNet-TS using the Fréchet mean as the pole are also presented visually in Tables 6.1 and 6.2 for two test images for $d = 4, 5$ respectively, and show similar trends. The choice of the pole does not seem to affect the results significantly except in the case of $d = 5$ where the Fréchet mean performs better.

6.5 Deep Regression on the Unit Hypersphere for Multi-Class Classification

Reformulating classification as mapping to the unit hypersphere: For multi-class classification problems, deep networks usually output a probability distribution, where one uses the mode of the distribution to predict the class label. What ensures that the output elements form a probability distribution is the "softmax layer". However, by using a square-root parametrization – replacing each element in the distribution by its square-root – we can map a probability distribution to a point on the non-negative orthant of a unit hypersphere $\mathcal{S}^C$, where C is now the number of classes. The square-root parameterization reduces the complicated Riemannian metric on the space of probability density functions, the Fisher-Rao metric, to the simpler Euclidean inner product on the tangent space of the unit hypersphere with closed form expressions for the geodesic distance, exponential and logarithm maps [147]. In this work, equipped with the knowledge of differential geometry of the sphere, we propose different loss functions for the tackling the classification problem. We consider two main variants – learning a network to map to the sphere directly or map to its tangent space. We note that the constraint for a point to be on a sphere or to be a probability distribution is simple and can be easily satisfied by using an appropriate normalization (dividing by its 2-norm or using softmax). However, mapping to the

tangent space of the sphere provides a novel perspective to the same problem and is more general since as we showed earlier, it is necessary for the Grassmannian.

Consider a classification problem with C classes. For a given input vector $\mathbf{x}$, let the ground-truth probability distribution over the class labels be $\mathbf{c}_{pd}$. The corresponding point on $\mathcal{S}^C$ is given by $\mathbf{c}_S$, such that $\mathbf{c}_S(i) = \sqrt{\mathbf{c}_{pd}(i)}, i = 1 \dots C$. The pole $\mathbf{u}_S$ for constructing the tangent space $T_{\mathbf{u}_S}\mathcal{S}^C$ is chosen to be the point on $\mathcal{S}^C$ corresponding to the uniform distribution $\mathbf{u}_{pd}, \mathbf{u}_{pd}(i) = \frac{1}{C}, i = 1 \dots C$. Let ξ be the desired point on $T_{\mathbf{u}_S}\mathcal{S}^C$ for the input $\mathbf{x}$ and is given by output of the log map $\xi = \exp^{-1}_{\mathbf{u}_S}(\mathbf{c}_S)$. **Let the output of the last fully connected layer be denoted by $\hat{\mathbf{o}}$.**

Geometry of the unit hypersphere [3]: The the n-dimensional unit sphere denoted as $\mathcal{S}^n$ and is defined as $\mathcal{S}^n = \{(x_1, x_2, \dots, x_{n+1}) \in \mathbb{R}^{n+1} | \quad \sum_{i=1}^{n+1} x_i^2 = 1\}$. Given any two points $\mathbf{x}, \mathbf{y} \in \mathcal{S}^n$, the geodesic distance between $\mathbf{x}$ and $\mathbf{y}$ is calculated using $d(\mathbf{x}, \mathbf{y}) = \cos^{-1}\langle \mathbf{x}, \mathbf{y} \rangle$. For a given point $\mathbf{x} \in \mathcal{S}^n$, the tangent space of $\mathcal{S}^n$ at $\mathbf{x}$ is given by $T_{\mathbf{x}}\mathcal{S}^n = \{\xi \in \mathbb{R}^n | \quad \mathbf{x}^T\xi = 0\}$. Since the Riemannian metric (the inner product on the tangent space) is the usual Euclidean inner product, the distance function on the tangent space induced by this inner product is the Euclidean distance. The exponential map $\exp : T_{\mathbf{x}}\mathcal{S}^n \rightarrow \mathcal{S}^n$ is computed using the following formula: $\exp_{\mathbf{x}} \xi = \cos(||\xi||)\mathbf{x} + \sin(||\xi||)\frac{\xi}{||\xi||}$, where $\xi \in T_{\mathbf{x}}\mathcal{S}^n$. For $\mathbf{x}, \mathbf{y} \in \mathcal{S}^n$, the inverse exponential map $\exp^{-1} : \mathcal{S}^n \rightarrow T_{\mathbf{x}}\mathcal{S}^n$ is given by $\exp^{-1}_{\mathbf{x}}(\mathbf{y}) = \frac{d(\mathbf{x},\mathbf{y})}{||P_{\mathbf{x}}(\mathbf{y}-\mathbf{x})||} P_{\mathbf{x}}(\mathbf{y} - \mathbf{x})$. $P_{\mathbf{x}}(\mathbf{v})$ is the projection of a vector $\mathbf{v} \in \mathbb{R}^n$ onto $T_{\mathbf{x}}\mathcal{S}^n$, given by $P_{\mathbf{x}}(\mathbf{v}) = (\mathbf{I}_n - \mathbf{x}\mathbf{x}^T)\mathbf{v}$, $\mathbf{I}_n$ is the $n \times n$ identity matrix.

6.5.1 Mapping to the Hypersphere Directly: SNet-M

In this case, the network directly outputs points on the sphere and a training pair is represented as $(\mathbf{x}, \mathbf{c}_S)$. We call this framework Snet-M. At test time, the

Framework	Desired Output of Network Lies on	Loss Function	Test Accuracy on MNIST (%)	Test Accuracy on CIFAR-10 (%)
Baseline		Cross Entropy	99.224 (0.0306)	78.685 (0.3493)
SNet-M	$\mathcal{S}^C$	$L_{S_{euc}}$	99.263 (0.0479)	79.738 (0.4009)
		$L_{S_{geo}}$	99.293 (0.0343)	**80.024** (0.5131)
SNet-TS	$T_{\mathbf{u_S}}\mathcal{S}^C$	$L_{T_{euc}}$	99.293 (0.0691)	77.548 (0.5620)
		$L_{T_{orth}}$	99.279 (0.0448)	77.708 (0.3517)
		$L_{T_{proj}}$	**99.332** (0.0600)	76.047 (1.6225)

Table 6.5: Avg test accuracy (std. dev.) over 10 runs using different loss functions on S^C and $T_{\mathbf{u}_S}S^C$, compared to the cross entropy loss.

network outputs a point on the sphere and the corresponding probability distribution is computed by squaring the elements of the output. We propose the following loss functions on the sphere. While training, the loss is averaged over the entire batch. In this case, we also employ a normalizing layer as the last layer of the network which guarantees that $\hat{\mathbf{o}}$ lies on $\mathcal{S}^C$.

(1) **Euclidean loss on $\mathcal{S}^C$** : This simply measures the Euclidean distance between two points on a sphere and does not take into account the non-linear nature of the manifold: $L_{S_{euc}} = ||\mathbf{c}_S - \frac{\hat{\mathbf{o}}}{||\hat{\mathbf{o}}||_2}||_2^2$.

(2) **Geodesic loss on $\mathcal{S}^C$** : The "true" distance between the two points on the sphere is given by $\theta = \cos^{-1}\langle \mathbf{c}_S, \frac{\hat{\mathbf{o}}}{||\hat{\mathbf{o}}||_2}\rangle$. Since minimizing this function directly leads to numerical difficulties, we instead minimize its surrogate, $L_{S_{geo}} = 1 - \cos\theta$.

6.5.2 Mapping to the Hypersphere via its Tangent Space: SNet-TS

Here, given an input, the algorithm first produces an intermediate output on $T_{\mathbf{u_S}}\mathcal{S}^C$ and then the exponential map is used to compute the desired point on $\mathcal{S}^C$. The corresponding probability distribution is computed by simply squaring each element of the vector. We refer to this framework as SNet-TS. A training example, then, is of the form $(\mathbf{x}, \xi)$, where ξ is the desired tangent vector. We propose the following loss functions on $T_{\mathbf{u_S}}\mathcal{S}^C$.

(1) **Euclidean loss on** $T_{\mathbf{u}_S}\mathcal{S}^C$: Measures the Euclidean distance between two points on the tangent space of the sphere : $L_{T_{euc}} = ||\xi - \hat{\mathbf{o}}||_2^2$. The output, $\hat{\mathbf{o}}$, is however not guaranteed to lie on $T_{\mathbf{u_S}}\mathcal{S}^C$ since a point on $T_{\mathbf{u_S}}S^C$ needs to satisfy the constraint $\mathbf{x}^T\xi = 0$. Therefore, we first project $\hat{\mathbf{o}}$ to the $T_{\mathbf{u_S}}S^C$ and then use the exponential map.

(2) **Euclidean + Orthogonal loss on** $T_{\mathbf{u}_S}\mathcal{S}^C$: In order to improve the "tangentness" of the output vector, we add the inner product loss that encourages the orthogonality of the output vector relative to the pole, which is the tangent space constraint: $L_{T_{orth}} = ||\xi - \hat{\mathbf{o}}||_2^2 + \lambda(\hat{\mathbf{o}}^T\mathbf{u}_S)^2$.

(3) **Projection loss on** $T_{\mathbf{u}_S}\mathcal{S}^C$: Since a closed form expression exists to project an arbitrary vector onto $T_{\mathbf{u}_S}\mathcal{S}^C$, we implement the projection layer as the last layer that guarantees that the output of the projection layer lies on $T_{\mathbf{u}_S}\mathcal{S}^C$. We compute the Euclidean loss between the projected vector and the desired tangent: $L_{T_{proj}} = ||\mathbf{c}_S - P_{\mathbf{u}_S}\hat{\mathbf{o}}||_2^2$.

6.5.3 Experiments on Image Classification

Image classification problem is a widely studied problem in computer vision and will serve as an example to demonstrate training a network to map to points on a unit hypersphere and its tangent space. We now describe the experiments conducted using MNIST and CIFAR-10 datasets. We train 6 networks with different loss functions. The first network is a baseline using the softmax layer to output a probability distribution directly and employs the well-known cross-entropy loss. The next two networks use the SNet-M framework and $L_{S_{euc}}$ and $L_{S_{geo}}$ as the loss functions. The desired outputs in this case lie on $\mathcal{S}^C$ and the network employs a normalizing layer at the end in order force the output vector to lie on the $\mathcal{S}^C$. The ground-truth output vectors are obtained by using the square-root parametrization. The final 3 networks employ the SNet-TS framework and $L_{T_{euc}}$, $L_{T_{orth}}$ and $L_{T_{proj}}$ as the loss functions. The desired output vector, in this case, should lie on $T_{\mathbf{u}_S}\mathcal{S}^C$. The required logarithm and exponential maps are computed using the Manifold Optimization toolbox [16]. We note that the purpose of the experiments is to show that for some chosen network architecture, the proposed loss functions that are inspired by the geometry of the hypersphere, perform comparably with the cross-entropy loss function.

MNIST: The MNIST dataset [101] consists a total of 60000 images of hand-written digits (0-9). Each image is of size 28×28 and is in grayscale. The task is to classify each image into one of the 10 classes (0-9). The dataset is split into training and testing sets with 50000 and 10000 images respectively. We use the LeNet-5 architecture as the neural network [101]. The network consists of 2 convolutional and max-pooling (`conv`) layers followed by 2 fully-connected (`fc`) layers. ReLU non-linearity is employed. The filters are of size 5×5. The first and second `conv` layers produce 32 and 64 feature maps respectively. The first and second fc layers output 1024 and 10

elements respectively. The networks are trained for 50000 iterations with a batch size of 100 using Adam optimizer [92] with learning rate of 10^{-3}.

CIFAR-10: The CIFAR-10 dataset [95] consists a total of 60000 RGB natural images. Each image is of size 32×32. The task is to classify each image into one of the 10 classes (Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck). The dataset is split into training and testing sets with 50000 and 10000 images respectively. The network consists of 2 `conv` layers with max-pooling and local response normalization followed by 2 `fc` layers. ReLU non-linearity is employed. Each input image is mean subtracted and divided by its standard deviation. Data augmentation using 10 24×24 random crops per input image is employed to reduce overfitting. At test time, the central 24×24 region is used as the input to the network, after performing the same normalization as the training inputs. The networks are trained for 500000 iterations with a batch size of 100 using Adam optimizer with learning rate of 10^{-3}.

For each dataset, we fix the network architecture and train the 6 versions of the network with different loss function as described above. We use $\lambda = 1$ for $L_{T_{orth}}$. The image recognition accuracies obtained on the test set (averaged over 10 runs) are shown in Table 6.5.

The results indeed show that some of the proposed loss functions tend to perform better than cross entropy. For both datasets and especially CIFAR-10, SNet-M yields better performance than cross entropy and within this framework, geodesic loss performs better compared to Euclidean loss. SNet-TS shows improvements in accuracy in the case of MNIST, albeit with higher variance in the accuracy.

6.6 Conclusion

In this chapter, we have studied the problem of learning invariant representations, which are at the heart of many computer vision problems, where invariance to physical factors such as illumination, pose, etc often lead to representations with non-Euclidean geometric properties. We have shown how deep learning architectures can be effectively extended to such non-linear target domains, exploiting the knowledge of data geometry. Through two specific examples – predicting illumination invariant representations which lie on the Grassmannian, and multi-class classification by mapping to a scale-invariant unit hypersphere representation – we have demonstrated how the power of deep networks can be leveraged and enhanced by making informed choices about the loss function while also enforcing the required output geometric constraints exactly. Extensions to other geometrically constrained representations, such as symmetric positive-definite matrices are evident. On the theoretical side, extending the current framework to applications where data points may have wider spread from their centroid, and to non-differentiable manifolds which arise in vision remain interesting avenues for the future.

Chapter 7

DISCUSSION AND FUTURE WORK

In this dissertation, I have presented several contributions in computer vision and computational imaging which employ deep neural networks. I have demonstrated how we can build invariant/robust representations, efficient and more interpretable architectures through enforcing different types of constraints on latent representations, parameters and design of neural network-based algorithms. These modifications come with significant improvements in performance such as image reconstruction quality in the case of computational imaging applications, and recognition accuracies in the case of higher-level computer vision applications considered in this disseration.

Specifically, I described how to use deep learning for two important computational imaging applications – (1) compressive sensing where we want to solve reconstruct an image from a small set of linear measurements of the scene. I first show how we can use a convolutional neural network, ReconNet, to significantly improve over traditional optimization-based methods which use hand-crafted priors or simple linear-models (such as dictionary learning), and further enhance reconstruction quality by jointly learning the measurement matrix along with the reconstruction network. Then, I described how we can make simple modifications to the loss function and training algorithm in order to incorporate interesting constraints on the measurement operator that makes the algorithm more efficient. (2) multi-spectral image fusion where given a high resolution panchromatic image a set of low resolution multispectal images, we want to fuse the two modes of information together to create a set of high resolution multispectral images. In this case, I show how we can design a more interpretable neu-

ral network-based architecture inspired by earlier signal processing literature, which leads to significant improvements in final multi-spectral fusion quality.

I have also carefully demonstrated how the addition of special layers and constrained representations can encode required invariances for high-level inference applications in computer vision. In particular, I have designed Temporal Transformers for human skeletal action recognition which is designed to generate easily interpretable rate-invariant and discriminative warping functions in a template-free manner such that, when applied to the input signal aids in improving classification performance. Finally, I showed how to use a learning framework to create illumination invariants from a single images. Unlike the traditional approach which requires access to images under all illumination conditions and a subspace fitting through optimization, I pose the problem as a non-linear regression on the Grassmannian. I design manifold-aware loss functions which respect the geometry of the underlying space and produces significantly better illumination subspaces from a single image, compared to conventional Euclidean frameworks.

This dissertation also opens up many interesting avenues for future research which I discuss below.

7.1 Aligning Trajectories on Manifolds Using Neural Networks

In Chapter 5, I presented Temporal Transformer Networks which are trained to perform template-free alignment of time-series signals in order to maximize classification accuracy. Instead, we can build a Siamese architecture which takes in two time series signals and generates a warping function at the output. This network can be trained in a completely unsupervised fashion by employing the warping layer designed in Chapter 5 and by using an alignment loss function between signal 1 and the warped version of signal 2. Once trained, time series alignment of similar test sequences can

be performed by a simple feed-forward pass through the network, rather than solving a computationally more expensive dynamic programming algorithm, which is the conventional method.

This idea can be readily extended from trajectories in $\mathbb{R}^N$ to trajectories on manifolds. This is directly applicable to human action recognition from skeletal data in the following way. Using 3D joint locations, the sequences can be seen as trajectories on $\mathbb{R}^{J\times 3}$, where J is the number of joints. If we use relative pairs of SE(3) transformation matrices between pairs of joints, such as in the paper by Raviteja et al. [159], as the representation of the skeleton, then the sequence becomes a trajectory on the product manifold of $SE(3) \times SE(3) \times \cdots \times SE(3)$. Once we have these trajectories, we can use the framework presented by Anirudh et al. [9] for action recognition. This involves, as the first step converting the sequences to the Square-Root Velocity Framework (SRVF) representation for Euclidean trajectories, and Transport SRVF (TSRVF) representation for non-Euclidean trajectories. Then, the sequences are aligned with the "mean" sequence of the training dataset, followed by feature extraction and classification.

The alignment is performed using an algorithm based on dynamic programming. Instead, we can envision training a neural network to perform the alignment to the mean sequence using simple feed-forward operation. Due to the fact that the eventual goal is action recognition, it is likely that the alignment need not be perfect and can be approximate, while maintaining the same classification performance.

7.2 Extensions to Deep Non-Linear Regression onto Riemannian Manifolds

In Chapter 6, non-linear regression on the Grassmannian and the unit-hypersphere were presented. On similar lines, I plan to extend the ideas to the manifold of symmetric positive definite matrices (SPD). This manifold is of significance in the

medical imaging community in the context of Diffusion Tensor Imaging (DTI), where the acquired "image" is a 2D field of 3×3 SPD matrices [135]. Operations such as denoising and super-resolution DT images can be posed as regression on the product space of SPD matrices.

In Chapter 6, for regression onto complicated manifolds like the Grassmann and Stiefel, we resort to first regressing the tangent vector and then mapping it to the corresponding point on the manifold. This formulation, although effective in the cases presented, assumes that the data are concentrated on a small part of the manifold such that the tangent space approximation is valid. In cases where this assumption may not hold, I propose investigating the possibility of regressing directly on the manifold by employing matrix back-propagation and retraction map (which can be implemented as a differentiable layer) instead of the standard back-propagation and the exponential map.

7.3 Enforcing Multiple Invariances Simultaneously in Neural Network Frameworks: Deep Learning in Shape Spaces

In chapters 4 and 5, I designed methods to encode certain geometric invariances into deep neural networks for create rate-invariant representaions for human skeletal actions and illumination-invariant representations for faces. However, in most applications, we would like to build multiple such invariances into deep architectures. Two relevant and interesting examples are that of shape clustering and classification [150], which are important problems in many applications in life sciences. We start of by first representing shapes are closed curves in $\mathbb{R}^2$. Shapes however are invariant to the following transformations: translation, scale and rotation. Once the curve is discretized for processing on a computer, we need an extra invariance to the sampled points on the curve. Formally, if we denote the space of curves by $\mathcal{C}$, then the

space of shapes $\mathcal{S}$ is the quotient space given by $\mathcal{C}/(\Gamma \times SO(2))$, where Γ is the set of resampling functions and $SO(2)$ is the 2D rotation group. In order to achieve both sets of invariances, we can jointly employ a spatial transformer layer [77] for learning invariances to spatial transformations and another warping layer for resampling invariance which is very similar to the Temporal Transformer [111], thus building two sets of invariances into the same neural network. As these warping layers are differentiable, they can be incorporated readily at the front-end of classification and clustering networks.

REFERENCES

[1] Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems", Software available from tensorflow.org (2015).

[2] Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning.", (????).

[3] Absil, P.-A., R. Mahony and R. Sepulchre, *Optimization algorithms on matrix manifolds* (Princeton University Press, 2009).

[4] Adler, A., D. Boublil, M. Elad and M. Zibulevsky, "A deep learning approach to block-based compressed sensing of images", arXiv preprint arXiv:1606.01519 (2016).

[5] Adler, A., D. Boublil and M. Zibulevsky, "Block-based compressed sensing of images via deep learning", in "Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on", pp. 1–6 (IEEE, 2017).

[6] Adler, A., M. Elad and M. Zibulevsky, "Compressed learning: A deep neural network approach", Signal Processing with Adaptive Sparse Structured Representations (2017).

[7] Aharon, M., M. Elad, A. Bruckstein *et al.*, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation", IEEE Transactions on signal processing **54**, 11, 4311 (2006).

[8] Amro, I., J. Mateos, M. Vega, R. Molina and A. K. Katsaggelos, "A survey of classical methods and new trends in pansharpening of multispectral images", EURASIP Journal on Advances in Signal Processing **2011**, 1, 79 (2011).

[9] Anirudh, R., P. Turaga, J. Su and A. Srivastava, "Elastic functional coding of riemannian trajectories", IEEE transactions on pattern analysis and machine intelligence **39**, 5, 922–936 (2017).

[10] Banerjee, M., R. Chakraborty, E. Ofori, M. S. Okun, D. E. Viallancourt and B. C. Vemuri, "A nonlinear regression technique for manifold valued data with applications to medical image analysis", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", (2016).

[11] Banerjee, M., R. Chakraborty, E. Ofori, D. Vaillancourt and B. C. Vemuri, "Nonlinear regression on riemannian manifolds and its applications to neuroimage analysis", in "International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 719–727 (Springer, 2015).

[12] Baraniuk, R. G., V. Cevher, M. F. Duarte and C. Hegde, "Model-based compressive sensing", IEEE Trans. Inf. Theory **56**, 4, 1982–2001 (2010).

[13] Begelfor, E. and M. Werman, "Affine invariance revisited", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", vol. 2, pp. 2087–2094 (IEEE, 2006).

[14] Bellman, R. and R. Kalaba, "On adaptive control processes", IRE Transactions on Automatic Control **4**, 2, 1–9 (1958).

[15] Borgerding, M. and P. Schniter, "Onsager-corrected deep networks for sparse linear inverse problems", arXiv preprint arXiv:1612.01183 (2016).

[16] Boumal, N., B. Mishra, P.-A. Absil, R. Sepulchre *et al.*, "Manopt, a matlab toolbox for optimization on manifolds.", Journal of Machine Learning Research **15**, 1, 1455–1459 (2014).

[17] Bronstein, M. M., J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data", IEEE Signal Processing Magazine (2017).

[18] Bruna, J., W. Zaremba, A. Szlam and Y. LeCun, "Spectral networks and locally connected networks on graphs", International Conference on Learning Representations (2014).

[19] Byravan, A. and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks", IEEE International Conference on Robotics and Automation (2016).

[20] Cai, J.-F., E. J. Candès and Z. Shen, "A singular value thresholding algorithm for matrix completion", SIAM Journal on Optimization **20**, 4, 1956–1982 (2010).

[21] Cai, T. and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise", IEEE Transactions on Information Theory **57**, 7, 4680–4688 (2011).

[22] Calderbank, R., S. Jafarpour and R. Schapire, "Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain", (2009).

[23] Candès, E. J. and B. Recht, "Exact matrix completion via convex optimization", Foundations of Computational mathematics **9**, 6, 717 (2009).

[24] Candès, E. J., J. Romberg and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", IEEE Trans. Inf. Theory **52**, 2, 489–509 (2006).

[25] Candes, E. J. and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?", IEEE Trans. Inf. Theory **52**, 12, 5406–5425 (2006).

[26] Candes, E. J. and M. B. Wakin, "An introduction to compressive sampling", IEEE Signal Processing Magazine pp. 21 – 30 (2008).

[27] Chakrabarti, A., "Learning sensor multiplexing design through back-propagation", in "Adv. Neural Inf. Proc. Sys.", (2016).

[28] Chang, J., C.-L. Li, B. Póczos, B. Kumar and A. C. Sankaranarayanan, "One network to solve them all—solving linear inverse problems using deep projection models", International Conference on Computer Vision (2017).

[29] Chao Li, D. X. S. P., Qiaoyong Zhong, "Skeleton-based action recognition with convolutional neural networks", in "ICMEW", (2017).

[30] Chen, Y. and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration", IEEE transactions on pattern analysis and machine intelligence **39**, 6, 1256–1272 (2017).

[31] Cheng, Y., F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections", in "IEEE Intl. Conf. Comp. Vision.", pp. 2857–2865 (2015).

[32] Cheng, Z., Q. Yang and B. Sheng, "Deep colorization", in "IEEE Intl. Conf. Comp. Vision.", pp. 415–423 (2015).

[33] Choi, H., Q. Wang, M. Toledo, P. Turaga, M. Buman and A. Srivastava, "Temporal alignment improves feature quality: an experiment on activity recognition with accelerometer data", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops", pp. 349–357 (2018).

[34] Clark, R., S. Wang, H. Wen, A. Markham and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem", AAAI Conference on Artificial Intelligence (2017).

[35] Cuturi, M. and M. Blondel, "Soft-dtw: a differentiable loss function for time-series", in "International Conference on Machine Learning", pp. 894–903 (2017).

[36] Dabov, K., A. Foi, V. Katkovnik and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering", IEEE Trans. Image Process. **16**, 8, 2080–2095 (2007).

[37] Dave, A., A. K. Vadathya and K. Mitra, "Compressive image recovery using recurrent generative model", arXiv preprint arXiv:1612.04229 (2016).

[38] de Brébisson, A. and P. Vincent, "An exploration of softmax alternatives belonging to the spherical loss family", International Conference on Learning Representations (2016).

[39] Dong, C., C. C. Loy, K. He and X. Tang, "Learning a deep convolutional network for image super-resolution", in "Euro. Conf. Comp. Vision", pp. 184–199 (Springer, 2014).

[40] Dong, C., C. C. Loy, K. He and X. Tang, "Image super-resolution using deep convolutional networks", IEEE Trans. Pattern Anal. Mach. Intell. **38**, 2, 295–307 (2016).

[41] Dong, W., G. Shi, X. Li, Y. Ma and F. Huang, "Compressive sensing via nonlocal low-rank regularization", IEEE Trans. Image Process. **23**, 8, 3618–3632 (2014).

[42] Donoho, D. L., "Compressed sensing", IEEE Trans. Inf. Theory **52**, 4, 1289–1306 (2006).

[43] Donoho, D. L., A. Maleki and A. Montanari, "Message-passing algorithms for compressed sensing", Proceedings of the National Academy of Sciences **106**, 45, 18914–18919 (2009).

[44] Du, Y., W. Wang and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1110–1118 (2015).

[45] Duarte, M. F., M. B. Wakin and R. G. Baraniuk, "Wavelet-domain compressive signal reconstruction using a hidden markov tree model", in "Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on", pp. 5137–5140 (IEEE, 2008).

[46] Edelman, A., T. A. Arias and S. T. Smith, "The geometry of algorithms with orthogonality constraints", SIAM journal on Matrix Analysis and Applications **20**, 2, 303–353 (1998).

[47] Eigen, D., C. Puhrsch and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network", in "Adv. Neural Inf. Proc. Sys.", pp. 2366–2374 (2014).

[48] Epstein, R., P. W. Hallinan and A. L. Yuille, "5±2 eigenimages suffice: an empirical investigation of low-dimensional lighting models", in "Proceedings of the Workshop on Physics-Based Modeling in Computer Vision", p. 108 (IEEE, 1995).

[49] Fazel, M., *Matrix rank minimization with applications*, Ph.D. thesis, Stanford (2002).

[50] Fletcher, P. T., "Geodesic regression and the theory of least squares on Riemannian manifolds", International Journal of Computer Vision **105**, 2, 171–185 (2013).

[51] Fletcher, T., "Geodesic regression on Riemannian manifolds", in "Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability", pp. 75–86 (2011).

[52] Garcia-Hernando, G. and T.-K. Kim, "Transition forests: Learning discriminative temporal transitions for action recognition and detection", in "2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 407–415 (IEEE, 2017).

[53] Garcia-Hernando, G., S. Yuan, S. Baek and T.-K. Kim, "First-person hand action benchmark with RGB-D videos and 3D hand pose annotations", in "Proceedings of Computer Vision and Pattern Recognition (CVPR)", (2018).

[54] Georghiades, A. S., P. N. Belhumeur and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose", IEEE Transactions on Pattern Analysis and Machine Intelligence **23**, 6, 643–660 (2001).

[55] Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in "IEEE Conf. Comp. Vision and Pattern Recog", pp. 580–587 (IEEE, 2014).

[56] Goodfellow, I., H. Lee, Q. V. Le, A. Saxe and A. Y. Ng, "Measuring invariances in deep networks", in "Advances in neural information processing systems", pp. 646–654 (2009).

[57] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks", in "Adv. Neural Inf. Proc. Sys.", pp. 2672–2680 (2014).

[58] Gregor, K. and Y. LeCun, "Learning fast approximations of sparse coding", in "Proceedings of the 27th International Conference on Machine Learning (ICML-10)", pp. 399–406 (2010).

[59] Gupta, H., K. H. Jin, H. Q. Nguyen, M. T. McCann and M. Unser, "CNN-based projected gradient descent for consistent CT image reconstruction", IEEE transactions on medical imaging **37**, 6, 1440–1453 (2018).

[60] Hallinan, P. W., "A low-dimensional representation of human faces for arbitrary lighting conditions", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", (1994).

[61] Hamm, J. and D. D. Lee, "Grassmann discriminant analysis: a unifying view on subspace-based learning", in "Proceedings of the International Conference on Machine Learning", pp. 376–383 (ACM, 2008).

[62] Harandi, M. and B. Fernando, "Generalized backpropagation,\'{E} tude de cas: Orthogonality", arXiv preprint arXiv:1611.05927 (2016).

[63] He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 770–778 (2016).

[64] Hegde, C., A. C. Sankaranarayanan, W. Yin and R. G. Baraniuk, "Numax: A convex approach for learning near-isometric linear embeddings", IEEE Transactions on Signal Processing **63**, 22, 6109–6121 (2015).

[65] Henaff, M., J. Bruna and Y. LeCun, "Deep convolutional networks on graph-structured data", arXiv preprint arXiv:1506.05163 (2015).

[66] Henriques, J. F., R. Caseiro, P. Martins and J. Batista, "High-speed tracking with kernelized correlation filters", IEEE Trans. Pattern Anal. Mach. Intell. **37**, 3, 583–596 (2015).

[67] Hochreiter, S. and J. Schmidhuber, "Long short-term memory", Neural computation **9**, 8, 1735–1780 (1997).

[68] Hong, Y., R. Kwitt, N. Singh, B. Davis, N. Vasconcelos and M. Niethammer, "Geodesic regression on the Grassmannian", in "European Conference on Computer Vision", pp. 632–646 (Springer, 2014).

[69] Hu, J.-F., W.-S. Zheng, J. Lai and J. Zhang, "Jointly learning heterogeneous features for RGB-D activity recognition", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 5344–5352 (2015).

[70] Huang, Z. and L. Van Gool, "A Riemannian network for SPD matrix learning", AAAI Conference on Artificial Intelligence (2017).

[71] Huang, Z., C. Wan, T. Probst and L. Van Gool, "Deep learning on Lie groups for skeleton-based action recognition", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016).

[72] Huang, Z., J. Wu and L. Van Gool, "Building deep networks on Grassmann manifolds", arXiv preprint arXiv:1611.05742 (2016).

[73] Iliadis, M., L. Spinoulas and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing", arXiv preprint arXiv:1603.04930 (2016).

[74] Iliadis, M., L. Spinoulas and A. K. Katsaggelos, "Deepbinarymask: Learning a binary mask for video compressive sensing", in "Preprint", (2016).

[75] Iliadis, M., L. Spinoulas and A. K. Katsaggelos, "Deepbinarymask: Learning a binary mask for video compressive sensing", arXiv preprint arXiv:1607.03343 (2016).

[76] Iliadis, M., L. Spinoulas and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing", Digital Signal Processing **72**, 9–18 (2018).

[77] Jaderberg, M., K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks", in "Advances in neural information processing systems", pp. 2017–2025 (2015).

[78] Jetley, S., N. Murray and E. Vig, "End-to-end saliency mapping via probability distribution prediction", in "2016 IEEE Conference on Computer Vision and Pattern Recognition", pp. 5753–5761 (2016).

[79] Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", arXiv preprint arXiv:1408.5093 (2014).

[80] Jin, K. H., M. T. McCann, E. Froustey and M. Unser, "Deep convolutional neural network for inverse problems in imaging", IEEE Transactions on Image Processing (2017).

[81] Johansson, G., "Visual perception of biological motion and a model for its analysis", Perception & psychophysics **14**, 2, 201–211 (1973).

[82] Kamilov, U. S. and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms", IEEE Signal Processing Letters **23**, 5, 747–751 (2016).

[83] Kanazawa, A., A. Sharma and D. Jacobs, "Locally scale-invariant convolutional neural networks", arXiv preprint arXiv:1412.5104 (2014).

[84] Kar, A., S. Tulsiani, J. Carreira and J. Malik, "Category-specific object reconstruction from a single image", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1966–1974 (2015).

[85] Ke, Q., M. Bennamoun, S. An, F. Sohel and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition", in "Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on", pp. 4570–4579 (IEEE, 2017).

[86] Kerviche, R., N. Zhu and A. Ashok, "Information optimal scalable compressive imager demonstrator", in "IEEE Conf. Image Process.", (2014).

[87] Kerviche, R., N. Zhu and A. Ashok, "Information-optimal scalable compressive imaging system", in "Classical Optics 2014", (Optical Society of America, 2014).

[88] Khasanova, R. and P. Frossard, "Graph-based isometry invariant representation learning", arXiv preprint arXiv:1703.00356 (2017).

[89] Kim, H. J., N. Adluru, M. D. Collins, M. K. Chung, B. B. Bendlin, S. C. Johnson, R. J. Davidson and V. Singh, "Multivariate general linear models (mglm) on Riemannian manifolds with applications to statistical analysis of diffusion weighted images", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2705–2712 (2014).

[90] Kim, T. S. and A. Reiter, "Interpretable 3d human action analysis with temporal convolutional networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops", (2017).

[91] Kim, Y., M. S. Nadar and A. Bilgin, "Compressed sensing using a Gaussian scale mixtures model in wavelet domain", in "IEEE Conf. Image Process.", pp. 3365–3368 (IEEE, 2010).

[92] Kingma, D. and J. Ba, "Adam: A method for stochastic optimization", Internal Conference on Learning Representations (2015).

[93] Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980 (2014).

[94] Knyazev, A. V. and M. E. Argentati, "Principal angles between subspaces in an a-based scalar product: algorithms and perturbation estimates", SIAM Journal on Scientific Computing **23**, 6, 2008–2040 (2002).

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

[96] Kulkarni, K., S. Lohit, P. Turaga, R. Kerviche and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements", in "IEEE Conf. Comp. Vision and Pattern Recog", (2016).

[97] Kulkarni, K. and P. Turaga, "Reconstruction-free action inference from compressive imagers", IEEE Trans. Pattern Anal. Mach. Intell. **PP**, 99 (2015).

[98] Lafferty, J. D. and G. Lebanon, "Diffusion kernels on statistical manifolds", Journal of Machine Learning Research **6**, 129–163 (2005).

[99] Lea, C., R. Vidal, A. Reiter and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation", in "European Conference on Computer Vision", pp. 47–54 (Springer, 2016).

[100] LeCun, Y., Y. Bengio and G. Hinton, "Deep learning", nature **521**, 7553, 436 (2015).

[101] LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE **86**, 11, 2278–2324 (1998).

[102] Ledig, C., L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network", arXiv preprint arXiv:1609.04802 (2016).

[103] Li, C., W. Yin, H. Jiang and Y. Zhang, "An efficient augmented lagrangian method with applications to total variation minimization", Computational Optimization and Applications **56**, 3, 507–530 (2013).

[104] Liu, D. and P. T. Boufounos, "Dictionary learning based pan-sharpening", in "Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on", pp. 2397–2400 (IEEE, 2012).

[105] Liu, J., S. Ji and J. Ye, "Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization", in "Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence", pp. 339–348 (AUAI Press, 2009).

[106] Liu, M., H. Liu and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition", Pattern Recognition **68**, 346–362 (2017).

[107] Lohit, S., K. Kulkarni, R. Kerviche, P. Turaga and A. Ashok, "Convolutional neural networks for non-iterative reconstruction of compressively sensed images", IEEE Transactions on Computational Imaging (2018).

[108] Lohit, S., K. Kulkarni and P. Turaga, "Direct inference on compressive measurements using convolutional neural networks", in "IEEE International Conference on Image Processing", pp. 1913–1917 (IEEE, 2016).

[109] Lohit, S., K. Kulkarni, P. Turaga, J. Wang and A. Sankaranarayanan, "Reconstruction-free inference on compressive measurements", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops", pp. 16–24 (2015).

[110] Lohit, S. and P. Turaga, "Learning invariant riemannian geometric representations using deep nets", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 1329–1338 (2017).

[111] Lohit, S., Q. Wang and P. Turaga, "Temporal transformer networks: Joint learning of invariant and discriminative time warping", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 12426–12435 (2019).

[112] Loncan, L., L. B. Almeida, J. M. Bioucas-Dias, X. Briottet, J. Chanussot, N. Dobigeon, S. Fabre, W. Liao, G. A. Licciardi, M. Simoes *et al.*, "Hyperspectral pansharpening: A review", arXiv preprint arXiv:1504.04531 (2015).

[113] Long, J., E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation", in "IEEE Conf. Comp. Vision and Pattern Recog", (2015).

[114] Lucas, A., M. Iliadis, R. Molina and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods", IEEE Signal Processing Magazine **35**, 1, 20–36 (2018).

[115] Lui, Y. M., "Advances in matrix manifolds for computer vision", Image and Vision Computing **30**, 6, 380–388 (2012).

[116] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly and R. G. Baraniuk, "The smashed filter for compressive classification and target recognition", Computat. Imag. V **6498**, 142–153 (2007).

[117] Marron, J. S., J. O. Ramsay, L. M. Sangalli and A. Srivastava, "Functional data analysis of amplitude and phase variation", Statistical Science **30**, 4, 468–484 (2015).

[118] Masci, J., D. Boscaini, M. Bronstein and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds", in "Proceedings of the IEEE International Conference on Computer Vision workshops", pp. 37–45 (2015).

[119] Masi, G., D. Cozzolino, L. Verdoliva and G. Scarpa, "Pansharpening by convolutional neural networks", Remote Sensing **8**, 7, 594 (2016).

[120] M.B. Wakin, J.N. Laska, M.F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K.F. Kelly and R.G. Baraniuk, "An architecture for compressive imaging", in "IEEE Conf. Image Process.", (2006).

[121] Metzler, C. A., A. Maleki and R. G. Baraniuk, "From denoising to compressed sensing", IEEE Trans. Inf. Theory (2016).

[122] Metzler, C. A., A. Maleki and R. G. Baraniuk, "From denoising to compressed sensing", IEEE Transactions on Information Theory **62**, 9, 5117–5144 (2016).

[123] Metzler, C. A., A. Mousavi and R. G. Baraniuk, "Learned d-amp: A principled cnn-based compressive image recovery algorithm", Advances in Neural Information Processing Systems (2017).

[124] Mousavi, A. and R. G. Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks", International Conference on Acoustics, Speech and Signal Processing (2017).

[125] Mousavi, A., G. Dasarathy and R. G. Baraniuk, "Deepcodec: Adaptive sensing and recovery via deep convolutional neural networks", arXiv preprint arXiv:1707.03386 (2017).

[126] Mousavi, A., A. B. Patel and R. G. Baraniuk, "A deep learning approach to structured signal recovery", in "2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)", pp. 1336–1343 (IEEE, 2015).

[127] NASA, "NASA AVIRIS data repository", (2019).

[128] Needell, D. and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples", Applied and Computational Harmonic Analysis **26**, 3, 301–321 (2009).

[129] Nie, F., H. Huang, X. Cai and C. H. Ding, "Efficient and robust feature selection via joint $\ell_{2,1}$-norm minimization", in "Advances in Neural Information Processing Systems", pp. 1813–1821 (2010).

[130] Niepert, M., M. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs", in "Proceedings of the International Conference on Machine Learning", (2016).

[131] Obozinski, G., B. Taskar and M. Jordan, "Multi-task feature selection", Technical Report, University of California, Berkeley (2006).

[132] Oike, Y. and A. E. Gamal, "CMOS image sensor with per-column $\sigma\delta$ adc and programmable compressed sensing", IEEE Journal of Solid-State Circuits **48**, 1, 318–328 (2013).

[133] Pathak, D., P. Krähenbühl, J. Donahue, T. Darrell and A. Efros, "Context encoders: Feature learning by inpainting", in "IEEE Conf. Comp. Vision and Pattern Recog", (2016).

[134] Paysan, P., R. Knothe, B. Amberg, S. Romdhani and T. Vetter, "A 3d face model for pose and illumination invariant face recognition", in "Sixth IEEE International Conference on Advanced video and signal based surveillance", pp. 296–301 (IEEE, 2009).

[135] Pennec, X., P. Fillard and N. Ayache, "A Riemannian framework for tensor computing", International Journal of Computer Vision **66**, 1, 41–66 (2006).

[136] Peyré, G., S. Bougleux and L. Cohen, "Non-local regularization of inverse problems", in "Euro. Conf. Comp. Vision", pp. 57–68 (Springer, 2008).

[137] Radford, A., L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", arXiv preprint arXiv:1511.06434 (2015).

[138] Recht, B., M. Fazel and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization", SIAM review **52**, 3, 471–501 (2010).

[139] Sabour, S., N. Frosst and G. E. Hinton, "Dynamic routing between capsules", in "Advances in Neural Information Processing Systems", pp. 3856–3866 (2017).

[140] Sakoe, H. and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", IEEE Transactions on Acoustics, Speech, and Signal Processing **26**, 1, 43–49 (1978).

[141] Schmidt, U. and S. Roth, "Shrinkage fields for effective image restoration", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2774–2781 (2014).

[142] Shahroudy, A., J. Liu, T.-T. Ng and G. Wang, "NTU RGB+D: A large scale dataset for 3d human activity analysis", in "The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2016).

[143] Shi, X., M. Styner, J. Lieberman, J. G. Ibrahim, W. Lin and H. Zhu, "Intrinsic regression models for manifold-valued data", in "International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 192–199 (Springer, 2009).

[144] Skafte Detlefsen, N., O. Freifeld and S. Hauberg, "Deep diffeomorphic transformer networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4403–4412 (2018).

[145] Som, S. and P. Schniter, "Compressive imaging using approximate message passing and a markov-tree prior", Signal Processing, IEEE Transactions on **60**, 7, 3439–3448 (2012).

[146] Song, S., C. Lan, J. Xing, W. Zeng and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data.", in "Proc. of AAAI Conference on Artificial Intelligence", (2017).

[147] Srivastava, A., I. Jermyn and S. Joshi, "Riemannian analysis of probability density functions with applications in vision", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1–8 (IEEE, 2007).

[148] Srivastava, A. and E. Klassen, "Bayesian and geometric subspace tracking", Advances in Applied Probability **36**, 01, 43–56 (2004).

[149] Srivastava, A., E. Klassen, S. H. Joshi and I. H. Jermyn, "Shape analysis of elastic curves in euclidean spaces", IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 7, 1415–1428 (2011).

[150] Srivastava, A. and E. P. Klassen, *Functional and shape data analysis* (Springer, 2016).

[151] Srivastava, A. and P. Turaga, *Riemannian computing in computer vision* (Springer International Publishing, 2015).

[152] Sun, J., H. Li, Z. Xu *et al.*, "Deep admm-net for compressive sensing mri", in "Advances in Neural Information Processing Systems", pp. 10–18 (2016).

[153] Taheri, S., P. Turaga and R. Chellappa, "Towards view-invariant expression analysis using analytic shape manifolds", in "IEEE International Conference on Automatic Face & Gesture Recognition and Workshops", pp. 306–313 (IEEE, 2011).

[154] Tallec, C. and Y. Ollivier, "Can recurrent neural networks warp time?", International Conference on Learning Representations (2018).

[155] Tan, J., Y. Ma and D. Baron, "Compressive imaging via approximate message passing with image denoising", IEEE Trans. Signal Process. **63**, 8, 2085–2092 (2015).

[156] Tang, Y., R. Salakhutdinov and G. Hinton, "Deep lambertian networks", in "Proceedings of the 29th International Coference on International Conference on Machine Learning", pp. 1419–1426 (Omnipress, 2012).

[157] Toshev, A. and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1653–1660 (2014).

[158] Turaga, P., A. Veeraraghavan, A. Srivastava and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 11, 2273–2286 (2011).

[159] Vemulapalli, R., F. Arrate and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a Lie group", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 588–595 (2014).

[160] Vemulapalli, R. and R. Chellapa, "Rolling rotations for recognizing human actions from 3D skeletal data", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4471–4479 (2016).

[161] Vincent, P., A. de Brébisson and X. Bouthillier, "Efficient exact gradient update for training deep networks with very large sparse targets", in "Advances in Neural Information Processing Systems", pp. 1108–1116 (2015).

[162] Wang, Q., S. Lohit, M. J. Toledo, M. P. Buman and P. Turaga, "A statistical estimation framework for energy expenditure of physical activities from a wrist-worn accelerometer", in "2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)", pp. 2631–2635 (IEEE, 2016).

[163] Wang, X., D. F. Fouhey and A. Gupta, "Designing deep networks for surface normal estimation", in "IEEE Conf. Comp. Vision and Pattern Recog", (2015).

[164] Wang, Z., A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity", IEEE transactions on image processing **13**, 4, 600–612 (2004).

[165] Watson, G. A., "Characterization of the subdifferential of some matrix norms", Linear Algebra and its Applications **170**, 33–45 (1992).

[166] Wei, Y., Q. Yuan, H. Shen and L. Zhang, "Boosting the accuracy of multispectral image pansharpening by learning a deep residual network", IEEE Geoscience Remote Sensing Letters (2017).

[167] Wen, B., U. Kamilov, D. Liu, H. Mansour and P. T. Boufounos, "DeepCASD: An end-to-end approach for multi-spectural image super-resolution", Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on (2018).

[168] Wu, Y., J. Lim and M. Yang, "Object tracking benchmark", IEEE Trans. Pattern Anal. Mach. Intell. **37**, 9, 1834–1848 (2015).

[169] Xu, K. and F. Ren, "CSvideonet: A recurrent convolutional neural network for compressive sensing video reconstruction", arXiv preprint arXiv:1612.05203 (2016).

[170] Xu, Y., T. Xiao, J. Zhang, K. Yang and Z. Zhang, "Scale-invariant convolutional neural networks", arXiv preprint arXiv:1411.6369 (2014).

[171] Yan, S., Y. Xiong and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition", in "Proc. of AAAI Conference on Artificial Intelligence", (2018).

[172] Yang, Y. and T. M. Hospedales, "Trace norm regularised deep multi-task learning", arXiv preprint arXiv:1606.04038 (2016).

[173] Yao, H., F. Dai, D. Zhang, Y. Ma, S. Zhang and Y. Zhang, "Dr-net: Deep residual reconstruction network for image compressive sensing", arXiv preprint arXiv:1702.05743 (2017).

[174] Yoon, Y., G. Choe, N. Kim, J.-Y. Lee and I. S. Kweon, "Fine-scale surface normal estimation using a single nir image", Euro. Conf. Comp. Vision (2016).

[175] Zanfir, M., M. Leordeanu and C. Sminchisescu, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection", in "Proceedings of the IEEE international conference on computer vision", pp. 2752–2759 (2013).

[176] Zha, Z., X. Zhang, Q. Wang, L. Tang and X. Liu, "Analysis of the group sparsity based on the rank minimization methods", arXiv preprint arXiv:1709.03979 (2017).

[177] Zhang, H., W. He, L. Zhang, H. Shen and Q. Yuan, "Hyperspectral image restoration using low-rank matrix recovery", IEEE Transactions on Geoscience and Remote Sensing **52**, 8, 4729–4743 (2014).

[178] Zhang, J. and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1828–1837 (2018).

[179] Zhang, J., S. Liu, R. Xiong, S. Ma and D. Zhao, "Improved total variation based image compressive sensing recovery by nonlocal regularization", in "Circuits and Systems (ISCAS), 2013 IEEE International Symposium on", pp. 2836–2839 (IEEE, 2013).

[180] Zhang, K., W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising", IEEE Transactions on Image Processing (2017).

[181] Zhang, P., C. Lan, J. Xing, W. Zeng, J. Xue and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data", in "2017 IEEE International Conference on Computer Vision (ICCV)", pp. 2136–2145 (IEEE, 2017).

[182] Zhang, X., Y. Wang, M. Gou, M. Sznaier and O. Camps, "Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4498–4507 (2016).

[183] Zhu, X. X., D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu and F. Fraundorfer, "Deep learning in remote sensing: a comprehensive review and list of resources", IEEE Geoscience and Remote Sensing Magazine **5**, 4, 8–36 (2017).