ABSTRACT

Title of dissertation: OPTIMIZATION ALGORITHMS USING PRIORS IN COMPUTER VISION

Sohil Atul Shah
Doctor of Philosophy, 2018

Dissertation directed by: Professor Tom Goldstein
Department of Electrical and Computer Engineering

Over the years, many computer vision models, some inspired by human behavior, have been developed for various applications. However, only handful of them are popular and widely used. Why? There are two major factors: 1) most of these models do not have any efficient numerical algorithm and hence they are computationally very expensive; 2) many models, being too generic, cannot capitalize on problem specific prior information and thus demand rigorous hyper-parameter tuning. In this dissertation, we design fast and efficient algorithms to leverage application specific priors to solve unsupervised and weakly-supervised problems. Specifically, we focus on developing algorithms to impose structured priors, model priors and label priors during the inference and/or learning of vision models.

In many application, it is known a priori that a signal is smooth and continuous in space. The first part of this work is focussed on improving unsupervised learning mechanisms by explicitly imposing these structured priors in an optimization framework using different regularization schemes. This led to the development of fast algorithms

for robust recovery of signals from compressed measurements, image denoising and data clustering. Moreover, by employing redescending robust penalty on the structured regularization terms and applying duality, we reduce our clustering formulation to an optimization of a single continuous objective. This enabled integration of clustering processes in an end-to-end feature learning pipeline.

In the second part of our work, we exploit inherent properties of established models to develop efficient solvers for SDP, GAN, and semantic segmentation. We consider models for several different problem classes. a) Certain non-convex models in computer vision (e.g., BQP) are popularly solved using convex SDPs after lifting to a high-dimensional space. However, this computationally expensive approach limits these methods to small matrices. A fast and approximate algorithm is developed that directly solves the original non-convex formulation using biconvex relaxations and known rank information. b) Widely popular adversarial networks are difficult to train as they suffer from instability issues. This is because optimizing adversarial networks corresponds to finding a saddle-point of a loss function. We propose a simple prediction method that enables faster training of various adversarial networks using larger learning rates without any instability problems. c) Semantic segmentation models must learn long-distance contextual information while retaining high spatial resolution at the output. Existing models achieves this at the cost of computationally expensive and memory exhaustive training/inference. We designed stacked u-nets model which can repeatedly process top-down and bottom-up features. Our smallest model exceeds Resnet-101 performance on PASCAL VOC 2012 by 4.5% IoU with $\sim 7\times$ fewer parameters.

Next, we address the problem of learning heterogeneous concepts from internet

videos using mined label tags. Given a large number of videos each with multiple concepts and labels, the idea is to teach machines to automatically learn these concepts by leveraging weak labels. We formulate this into a co-clustering problem and developed a novel bayesian non-parametric weakly supervised Indian buffet process model which additionally enforces the paired label prior between concepts.

In the final part of this work we consider an inverse approach: learning data priors from a given model. Specifically, we develop numerically efficient algorithm for estimating the log likelihood of data samples from GANs. The approximate log-likelihood function is used for outlier detection and data augmentation for training classifiers.

OPTIMIZATION ALGORITHMS USING PRIORS IN COMPUTER VISION

by

Sohil Atul Shah

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Dedication

To my beloved parents and beautiful sister Miloni

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Tom Goldstein for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past four years. There has never been an occasion when I've knocked on his door and he hasn't given me time. Without his remarkable theoretical insights and computational expertise this thesis wouldn't have been possible. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Professor Larry Davis who guided me during my initial years of PhD. He has always made himself available for help and advice. Without his continuous support on providing computational resources, this thesis would have been a distant dream.

I would also like to thank my external collaborator Dr. Vladlen Koltun. He provided me with an incredible internship opportunity following which he continued to advice and support me financially. It has been a pleasure to work with him and his team at Intel Labs which helped me gain exposure to plethora of research problems in computer vision and machine learning.

Thanks are due to Professor Rama Chellappa, Professor Behtash Babadi and Professor David Jacobs for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

My colleagues at the computer vision laboratory have enriched my graduate life in many ways and deserve a special mention. I am grateful to my co-author Abhay Yadav for his help on multiple research projects and his valuable insights. I have thoroughly enjoyed working with him and learned many important aspects of life. Pallabi Ghosh provided help by writing basic code on semantic segmentation and performing some of the experiments. My interaction with Bharat Singh, Yaming Wang, Soham De, Hao Li, Ang Li, Saurabh Sahu, Vlad Morariu and Xiyang Dai has been very fruitful.

I owe my deepest thanks to my family - my mother, father and sister who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. I would also like to humbly thank my closest childhood friends Austin Daniel, Avinash Thavva, Lloyd Gomez and Vikram Gopichetty who kept me motivated with their effective and continual communication all through my ups and downs of my PhD years.

My current and past housemates at my place of residence have been a crucial factor in me finishing smoothly. I'd like to express my gratitude to Ankit Gargava, Varun Manjunatha, Srinivasan Seshadhri, Sudharsan PB and Parth Agarwal for their friendship and support.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Lastly, thank you all and thank God!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A2D	Actor-Action Dataset
AC-GAN	Auxiliary Classifier - Generative Adversarial Networks
ACC	Clustering Accuracy
ADMM	Alternating Direction Method of Multipliers
AFLR	Adversarial Learned Fair Representations
AIM	Affine Invariant Metric
ALI	Adversarially Learned Inference
AM	Alternating Minimization
AMI	Adjusted Mutual Information
ANN	Adversarial Neural Networks
ASPP	Atrous Spatial Pyramid Pooling
AUC	Area Under the Curve
BCR	Bi-Convex Relaxation
Bi-GAN	Bidirectional Generative Adversarial Networks
BNP	Bayesian Non-Parametric
BQP	Binary-Valued Quadratic Problems
Co-LAMP	Convex Lattice Matching Pursuit
Co-SAMP	Compressive Sampling Matched Pursuit
CNN	Convolutional Neural Networks
CRF	Conditional Random Fields
CS	Compressive Sensing
DANNs	Domain Adversarial Neural Networks
DBM	Deep Boltzmann Machines
DC-GAN	Deep Convolutional Generative Adversarial Networks
DCC	Deep Continuous Clustering
D-VTV	Decorrelated Vectorized Total Variation
EM	Expectation-Maximization
FASTA	Fast Adaptive Shrinkage/Thresholding Algorithm
FBS	Forward-Backward Sampling
FC	Fully Connected
FCN	Fully Convolutional Networks
FPC	Fixed-Point Continuation
GAN	Generative Adversarial Networks
GMM	Gaussian Mixture Models

IBP	Indian Buffet Process
IoU	Intersection-over-Union
IRLS	Iterative Re-weighted Least Square
LAMP	Lattice Matching Pursuit
LC-KSVD	Label Consistent k-Singular Value Decomposition
LEM	Log Euclidean Metric
LMI	Linear Matrix Inequalities
m-kNN	Mutual k-Nearest Neighbor
MCMC	Markov Chain Monte Carlo
MIL	Multiple Instance Learning
MLE	Maximum Likelihood Estimation
MLP	Multi Layer Perceptron
MoG	Mixture of Gaussians
MRF	Markov Random Fields
NMI	Normalized Mutual Information
NVP	Non-Volume Preserving
PCA	Principal Component Analysis
PSD	Postive Semi-definite
PSNR	Peak Signal-to-Noise Ratio
RBM	Restricted Boltzmann Machines
RCC	Robust Continuous Clustering
RCC-DR	Robust Continuous Clustering with Dimensionality Reduction
ReLU	Rectified Linear Units
RIP	Restricted Isometry Property
RPCA	Robust Principal Components Analysis
SDAE	Stacked Denoised Auto-Encoders
SDP	Semi-definite Programming
SDR	Semi-definite Relaxation
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SUNets	Stacked U-Nets
SVM	Support Vector Machines

t-SNE	t-distributed Stochastic Neighbor Embedding
TV	Total Variation
W-GAN	Wasserstian GAN
WSC-SIIBP	Weakly Supervised, Constrained and Stacked Integrative Indian Buffet Process
YTC	YouTube Celebrities
YTF	YouTube Faces

Chapter 1: Introduction

Many computer vision and machine learning models were developed using prior knowledge derived from human experience. For example, humans are subconsciously wired to notice structure, motion, and patterns in images and videos. This suggests that the human brain carries some prior regarding continuity of objects across space and time. This so-called structured prior is widely used in popular vision and imaging applications, and is encoded in the form of conditional random fields (CRFs), total variation, and structured prediction models. However, many algorithms incorporate these prior at the expense of computation time, high memory consumption, and model approximation. This renders many models infeasible for large scale applications. For instance, if implemented naively, it is impractical to solve a fully connected CRF even on small images. It takes days for popular inference algorithms such as Markov Chain Monte Carlo (MCMC) sampling and graph cut methods to converge. However, an efficient implementation using mean-field approximation and low-pass gaussian filtering [1] can solve a CRF in seconds.

Strong priors enable us to solve difficult computer visions problems with relatively little data, but can we build computationally efficient algorithms for the resulting models? In this work, we develop new optimization algorithms that inject application-specific prior knowledge into selected computer vision problems. Specifically, we develop algorithms which impose structured priors, model priors, or label priors during the inference/learning of unsupervised and weakly supervised vision models.

In many signal recovery applications, one is recovering a signal (image) which should be smooth / continuous in space. One good example is the reconstruction of sparse signals from compressed measurements. However, we note that most of the current compressed sensing models do not explicitly impose smoothness priors. We propose the use of spatial smoothness priors that assist in speeding up and improving the quality of high definition image and video recovery. In the past, some models have already exploited the priors using non-convex regularization in the form of wavelet trees [2], MRFs [3] etc, which guarantees global convergence only for specific cases. In chapter 2, we develop a new mechanism for signal recovery that regularizes pixel neighborhoods using block sparse ($l_{1,2}$) regularizers. This regularizer is convex and can explicitly impose the above mentioned continuity criteria in many models involving sparse variables. This improves recovery from compressed measurements, and also boosts results on image denoising, robust PCA (RPCA), and structured dictionary learning applications [4].

Motivated by the successful application of spatial priors for signal recovery, in chapter 3 we propose a new optimization formulation for data clustering. The set of high-dimensional data samples is known to span a low-dimensional smooth manifold [5]. Similar to the imaging application, we argue that imposing a suitable local neighborhood structure prior can be beneficial in finding the underlying data clusters. This led to the development of a new clustering formulation, robust continuous clustering (RCC) [6], which is regularized using the local connectivity structure tapped from an underlying manifold space in an unsupervised manner. The advantage of our formulation over existing clustering formulations is that it optimizes a smooth continuous objective using efficient linear solvers. The presented algorithm is fast, easy to use and scales efficiently to high dimensions and large datasets. Moreover, it does not rely on prior knowledge of the number of ground-truth clusters. Experiments demonstrate that our method achieves high accuracy across datasets from multiple domains, outperforming the best prior algorithm by a factor of 3 in average rank.

The use of global continuous formulation eliminates the discrete reconfigurations of the objective that characterize prior clustering algorithms. This allows clustering to be integrated as a module in an end-to-end feature learning pipelines. We demonstrate this further in chapter 4 by extending the RCC algorithm to perform joint clustering and non-linear dimensionality reduction by efficiently optimizing a global continuous objective [7]. The integrated formulation can be easily optimized using the standard gradient descent solvers. The data is embedded into a lower-dimensional space by a deep autoencoder. The autoencoder is optimized as part of the clustering process while the resulting low-dimensional projection produces clustered data. Experiments on multiple datasets demonstrates that the presented algorithm outperforms state-of-the-art clustering schemes, including RCC and recent methods that use deep networks.

In the second part of this dissertation, we independently analyze three different well-established models and exploit their inherent properties to develop efficient solvers and algorithms. Specifically, we present new solvers for semi-definite programming (SDP) and generative adversarial networks (GAN) and a new deep network for semantic segmentation tasks.

In chapter 5, we present a fast approximate SDP solver which is based on the biconvex relaxation procedure [8]. Many computer vision problems are being modeled in the form of binary-valued quadratic problem (BQP). BQPs are non-convex formulation and are popularly solved through semi-definite relaxation to continuous space. This is made possible by lifting BQPs into the high-dimensional space and remodeling them as SDPs. This reformulation is easily solved using convex SDP solvers. However, this is achieved at the expense of high computational power (polynomial time) and memory consumption ($\Theta(N^2)$, for N unknowns). It is known that SDP matrices $\mathbf{X} \in \mathbb{R}^{N \times N}$ can be decomposed into $\mathbf{X} = \mathbf{Y}\mathbf{Y}^T$, where $\mathbf{Y} \in \mathbb{R}^{N \times k}$ (k is the rank information; $k = 1$ for BQPs). For the case where the rank information is known, one may indirectly try to solve for the optimal SDP matrix $\mathbf{X}^*$ by solving the non-linear optimization problem modeled in the search space of $\mathbf{Y}$. This can indeed give significant speedup for low rank SDPs and reduces memory consumption to $\Theta(N(k + 1))$. In practice, the optimal SDPs are extremely low-rank i.e., $k << N$ and hence defining a general rank-constrained model will lead to a smaller search space for many problems. In the past, [9, 10] have suggested different approaches to solve low rank SDP problems involving a linear objective and very few linear matrix equalities or inequalities (LMI). However, in order to cater to the needs of many vision applications, such as segmentation and metric learning, it is necessary to solve a rank-constrained *general* SDP problem involving many linear equalities and LMI. We propose a biconvex optimization algorithm that can solve general SDPs with *any* number of linear equality and inequality constraints. The alternate sub-problem which emerges out of the proposed formulation are convex, and they are readily solved using proximal operators and least squares solvers. Consequently, the proposed solver runs $4 - 35\times$ faster than the previous state-of-the-art approach while achieving similar or better performance. Our algorithm, if properly initialized, is guaranteed to converge closer to the global minima. Thus, we will also discuss a newly developed initialization technique which is based on the generalized eigenvalue approach for matrices. Note that, apart from the RCC formulation, the biconvex relaxation is another instance of my work wherein the discrete assignment problem is transformed into a continuous objective.

In chapter 6, we present a simple modification to stochastic gradient descent that can stabilize any adversarial neural networks (ANNs) [11]. ANNs solve many important problems in data science, but are notoriously difficult to train. These difficulties come from the fact that optimal weights for adversarial nets correspond to saddle points, and not minimizers, of the loss function. The alternating stochastic gradient methods typically used for such problems do not reliably converge to saddle points, and when convergence does happen it is often highly sensitive to learning rates. Inspired by the well-known "predictor-corrector" methods that have been successfully applied for saddle-point optimization in the non-stochastic setting, we propose a new prediction method for solving saddle-point problems in stochastic settings. We show, both in theory and practice, that the proposed method reliably converges to saddle points, and is stable with a wider range of training parameters than a non-prediction method. This makes adversarial networks less likely to "collapse," and enables faster training with larger learning rates. Moreover, our method adds only a single prediction step to any stochastic solver with an increase of just 10% in training time. We demonstrate the efficacy of prediction methods on three different categories of adversarial networks: GANs, domain adaptation and learning fair classifiers. Furthermore, the qualitative and quantitative comparison to Unrolled-GAN [12], Stacked-GAN [13] and AC-GAN [14] (on ImageNet) suggests that our algorithm stabilizes GAN training across learning rates, momentum, and image resolutions while achieving comparable or better inception scores.

In chapter 7, we present a novel deep neural network model for semantic image segmentation [15]. Many imaging tasks require global information about all pixels in an image. Conventional bottom-up classification networks globalize information by decreasing resolution; features are pooled and down-sampled into a single output. But for semantic segmentation and object detection tasks, a network must provide higher-resolution pixel-level outputs. To globalize information while preserving resolution, many researchers propose the inclusion of sophisticated auxiliary blocks such as dense CRF and spatial pyramid pooling, but these come at the cost of a considerable increase in network size and computational cost. We argue that in all contemporaneous work, the continued use of conventional pre-trained classification networks as a backbone model is fundamentally incorrect. This is due to the fact that classification networks were specifically designed to work well on single object classification and not for pixel-level localization tasks. Thus, when extended to object detection and segmentation tasks, it is not clear whether the complete potential of these networks has been properly tapped. In this chapter, we propose a stacked u-nets (SUNets) model, which explicitly characterizes the primary requirement of the localization task. SUNets iteratively performs multiple top-down and bottom-up processes which optimally exchange contextual information and combine features from different resolution scales. SUNets leverage the information globalization power of u-nets in a deeper network architecture that is capable of handling the complexity of natural images. SUNets perform extremely well on image classification and semantic segmentation tasks using a very small number of parameters. Indeed, our smallest model exceeds the performance of ResNet-101 on the PASCAL VOC 2012 semantic segmentation task by 4.5% mIoU, while having $\sim 7\times$ fewer parameters. These results and other analysis suggest that the use of conventional classification models hinders their performance on pixel localization tasks.

In the third part of this dissertation (chapter 8), we present a weakly-supervised framework for learning heterogeneous concepts from large scale internet videos using co-occurrence *label* priors [16]. The idea is, given a large number of videos each with multiple label tags, similar to ones we find on YouTube, can we build algorithms to teach machines to identify and localize (both spatial and temporal) visual concepts (such as objects, actions, persons, etc.)? Philosophically speaking, this problem is inspired by the way babies learns. For example, consider a scenario in which a human is shown many videos of an unknown celebrity. Each of these videos contain various concepts and some of them have nothing to do with this particular celebrity. The videos are also provided with multiple label tags which may be associated with these concepts. Some of these tags are indeed related to this celebrity - possibly describing its action in the corresponding videos. Given this scenario, there is a good chance that after watching many videos, a human subject will be able to recognize and localize the celebrity in question along with various other concepts. Ambiguity during the learning process is bound to occur due to the presence of multiple label tags per video. Thus, to automate this process, one requires a model that can learn to map each and every label tag to different concepts in each video such that the similar tags coherently share the visual feature representation across the video database. This being a difficult problem, a lot of research was focused on solving a related but simplified problem on images using various context priors and deep learning techniques [17, 18].

We developed a novel model that utilizes multiple paired labels of type (dog, walking), (cat, running) etc and jointly learns heterogeneous concepts from large scale video databases. The commonly used phrases on the internet describe subjects, predicates, objects, and actions, and hence the paired labels can be readily extracted. By imposing these types of weak but paired label priors into the learning algorithm, we believe that the localization of one concept, for *e.g.*cat, in a video will in turn help to easily localize and learn the visual appearance of its paired concept, for *e.g.*running. This process, if inculcated in the model learning, will successfully learn faster as well as better visual appearance models. We propose to solve the above problem by remodeling a non-parametric and unsupervised Bayesian model named the Indian Buffet Process (IBP). Our weakly-supervised stacked IBP model performs co-clustering while using the paired label priors. We also develop posterior inference for the proposed formulation using mean-field variational approximation. The model parameters are learned during the inference process. Comparative evaluations on the Casablanca and the A2D datasets show that the proposed approach significantly outperforms other state-of-the-art techniques: 24% relative improvement for pairwise concept classification in the Casablanca dataset and 9% relative improvement for localization in the A2D dataset as compared to the most competitive baseline.

In the final chapter 9, we consider the completely reverse approach of learning a data prior from a given trained model. Generative adversarial networks are trained to generate data that is indistinguishable from the input training samples. Such adversarially trained networks are known to generate high quality samples without the need for explicitly specifying the likelihood function. Among the generative models, GANs are widely popular due to their simple training and sampling procedure. In contrast, models such as RBMs [19] and DBMs [20], which are trained by explicitly maximizing the data likelihood, suffer from complex training and sampling procedure due to the use of mean field inference and MCMC. The variational autoencoder, which simultaneously learns a generative model and approximate inference, produces blurry samples and is limited in its application to low-dimensional deep representations. Ideally, a good generative model is one that generates perceptually high quality samples while employing simpler training, sampling, and evaluation. In this chapter, we develop a approximate procedure to estimate the log-likelihood of the generated and test data samples using GANs. We investigate this by building the Jacobian of the generator transformation function. We demonstrate that this approximate function can be used for outlier detection and data augmentation for training a classifier.

Part I

Structured Prior

Chapter 2: Block Sparsity

2.1 Introduction

A large number of existing models used in sparse signal processing and machine learning rely on ℓ_1-norm regularization in order to recover sparse signals or to identify sparse features for classification tasks. Sparse ℓ_1-norm regularization is also prominently used in the image-processing and computer vision domain, where it is used for segmentation, tracking, and background subtraction tasks. In computer vision and image processing, we are often interested in regions that are not only sparse, but also *spatially smooth*, i.e., regions with contiguous support structure. In such situations, it is desirable to have regularizers that promote the selection of large, contiguous regions rather than merely sparse (and potentially isolated) pixels. In contrast, simple ℓ_1-norm regularization adopts an unstructured approach that induces sparsity wherein each variable is treated independently, disregarding correlation among neighboring variables.

For imaging applications, ℓ_1-norm regularization may result in regions with spurious active (or isolated) pixels or non-smooth boundaries in the support set. This issue is addressed by the image-segmentation literature, where spatially correlated priors (such as total variation or normalized cuts) are used to enforce smooth support boundaries [21–24]. An important hallmark of existing image-segmentation methods is that they are able to enforce spatially contiguous support. However, the concept of correlated support has yet to be ported to more complex reconstruction tasks, including (but not limited to) robust PCA, dictionary learning and compressive background subtraction. The development of such structured sparsity models has been an active research topic [2,3,25–28], with new models and applications still emerging [29,30].

In this chapter, we develop a class of convex priors based on overlapping block/group sparsity, which are able to enforce sparsity of the support set and promote *spatial smoothness*. Our work is inspired by the ℓ_1/ℓ_2-norm spatial coherence priors used in [26], as well as group sparsity priors used in statistics (e.g., group lasso) [31,32]. Specifically, we propose new regularizers for imaging and computer vision applications and develop computationally efficient global minimization algorithms that are suitable for overlapping pixel-cliques. Finally, we propose the use of our regularizers within greedy pursuit methods for compressive reconstruction. The code is available at `https://github.com/shahsohil/CoLaMP`.

2.2 Related Work

Existing work on spatially-smooth support-set regularization can be divided into two main categories: (i) non-convex models that rely on graphs and trees, and (ii) convex models that rely on group-sparsity inducing norms. Cevher *et al*. [3] promote sparsity using Markov random fields (MRFs) in combination with compressive-sensing signal recovery, which is referred to as lattice matching pursuit (LaMP). LaMP recovers structured sparse signals using fewer noisy measurements than methods that ignore spatially correlated

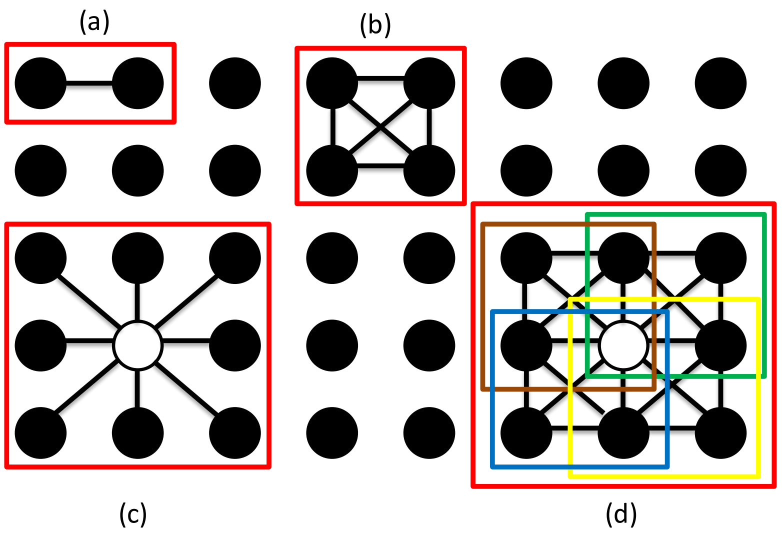


Figure 2.1: Illustration of cliques and overlapping cliques.

support sets. Baraniuk *et al*. [2] prove theoretical guarantees on robust recovery of structured sparse signals using a non-convex algorithm; their approach has been validated using wavelet-tree-based hierarchical group structure, as well as signals with non-overlapping blocks in the support set. Huang *et al*. [25] developed a theory of greedy approximation methods for general non-convex structured sparse models. All these methods, however, are limited in that they are either non-convex, computationally expensive, or do not allow for overlapping (or not aligned) group structure. Jenatton *et al*. [26] showed the possibility of coming up with a problem-specific optimal group-sparsity-inducing norm using prior knowledge of the underlying structure. While they consider a convex relaxation of the structured sparsity problem, it remains unclear how their proposed active-set algorithm for least squares regression can be generalized to a broader range of applications.

2.3 Model and Algorithms

Consider the measurement model $\mathbf{y} = \mathbf{\Phi}\mathbf{x}_0 + \mathbf{z}_0$, where $\mathbf{y} \in \mathbb{R}^M$ is the observed signal, $\mathbf{x}_0 \in \mathbb{R}^N$ is the original sparse signal we wish to recover, $\mathbf{z}_0 \in \mathbb{R}^M$ is a non-sparse component of the signal (comprising both the background image and potential noise), $\mathbf{\Phi} \in \mathbb{R}^{M\times N}$ is the linear operator that models the signal acquisition process. Based on this model, we study signal recovery by solving convex optimization problems of the following general form:

$$\{\hat{\mathbf{z}}, \hat{\mathbf{x}}\} = \underset{\mathbf{z}\in\mathbb{R}^M, \mathbf{x}\in\mathbb{R}^N}{\arg\min}\ D(\mathbf{x}, \mathbf{z} \,|\, \mathbf{y}, \mathbf{\Phi}) + J(\mathbf{x}). \tag{2.1}$$

Here, $D : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ is a convex data-consistency term, and $J : \mathbb{R}^M \to \mathbb{R}$ is a regularizer that enforces both sparsity and support smoothness on the vector $\mathbf{x}$. The proposed regularizer is a hybrid ℓ_1/ℓ_2-norm penalty of the form

$$J(\mathbf{x}) = \textstyle\sum_{c\in\mathcal{C}} \|\mathbf{x}_c\|_2, \tag{2.2}$$

where $\mathcal{C}$ is a set of cliques over the graph $\mathcal{G}$ defined over the pixels of $\mathbf{x}$. This regularizer (2.2) is a natural generalization of the *group (or block) sparsity* model that has been explored in the literature for a variety of purposes including statistics and radar [26, 28, 29, 31]. We focus on the case where the collection of sparse cliques consist of regularly-spaced groups of adjacent pixels. For example, consider two types of cliques shown in Figure 2.1(a) and 2.1(b). Notice the (a) 2-clique and (b) 4-clique wherein all nodes are connected to each other. These cliques can be translated over the entire image graph to generate various overlapping clique geometries as shown in (c) and (d), respectively. In (c), eight overlapping cliques, each of size two, overlap at a central point. In the image processing literature this is referred to as an 8-connected neighborhood [33]. In contrast, Figure 2.1(d) uses a higher-order connectivity model, which is obtained using four rectangular cliques of size four (each shown in a different color). Overlapping group-sparsity models of the form depicted in Figure 2.1(d) effectively enforce spatial coherence of the recovered support. When such an overlapping group-sparsity model is used, all pixels in a clique tend to be either zero or non-zero at the same time (see, e.g., [27]). Since each pixel shares multiple overlapping cliques with its neighbors, this regularizer suppresses "rogue" (or isolated) pixels from entering the support without their neighbors and hence, promotes smooth (contiguous) support boundaries.

2.3.1 Applications

The proposed regularizer (2.2) can be used as a building block for various applications in computer vision, image processing, and compressive sensing. We will focus on the following four imaging applications:

1) Compressive sensing signal recovery: Consider a signal $\mathbf{x} \in \mathbb{R}^N$ that is K-sparse, i.e., only KN entries of $\mathbf{x}$ are non-zero. In the CS literature, the signal is acquired via $M < N$ linear projections $\mathbf{y} = \mathbf{\Phi x}$. The K-sparse signal $\mathbf{x}$ can then be recovered if, for example, the matrix $\mathbf{\Phi}$ satisfies the *2K-RIP* or similar conditions [2, 34]. The underlying recovery problem is usually formulated as follows:

$$\mathbf{x}^\star = \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min} \, \|\mathbf{y} - \mathbf{\Phi x}\|_2^2 \quad \text{subject to } \|\mathbf{x}\|_0 = K. \tag{2.3}$$

When the sparse signals are images, simple sparse recovery may not exploit the entire image structure; this is particularly true for background-subtracted surveillance video.

Background-subtracted frames are generally more sparse than frames containing background information, and can thus be reconstructed from far fewer measurements M.

We propose to extend the problem in (2.3) by adding a regularizer of the form (2.2) to promote correlation in the support set of the foreground objects. The optimization problem defined in (2.3) is non-convex and is commonly solved using greedy algorithms [3, 35, 36]. We will show that the use of our prior (2.2) leads to faster signal recovery with a small number of measurements compared to existing methods.

2) Total-variation denoising: Total variation (TV) denoising restores a noisy image $\mathbf{y}$ (vectorized) by finding an image that lies close to $\mathbf{y}$ in an ℓ_2-norm sense, while simultaneously having small total variation; This is accomplished by solving,

$$\mathbf{x}^{\star} = \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min}\, \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 + \lambda\|\nabla_d \mathbf{x}\|_1, \tag{2.4}$$

where $\nabla_d : \mathbb{R}^N \to \mathbb{R}^{2N}$ is a discrete gradient operator that acts on an N-pixel image, and produces a stacked horizontal and vertical gradient vector containing all first-order differences between adjacent pixels. TV-based image processing assumes that images have a piecewise constant representation, i.e., the gradient is sparse and locally contiguous [37, 38]. Numerous generalizations of TV exist, including the recently proposed vectorial TV for color images [39, 40]. Such regularizers are of the form of (2.4) merely by changing the definition of the discrete gradient operator.

We propose to extend total variation by penalizing the gradient of cliques in order to enforce a greater degree of spatial coherence. In particular, we consider

$$\mathbf{x}^{\star} = \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min}\, \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 + J(\nabla_d \mathbf{x}), \tag{2.5}$$

where $J(\cdot)$ denotes the regularizer (2.2). Furthermore, we explore formulations where the discrete gradient operator is given by the decorrelated color TV operator described in [40] and the structured sparsity prior is on 3-D blocks.

3) Robust PCA (RPCA): Suppose $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_L]$ is a matrix of L measurement vectors, and $\mathbf{Y}$ is the sum of a low rank matrix $\mathbf{Z}$ and a sparse matrix $\mathbf{X}$. For this case, Candès *et al.*show that exact recovery of these components is possible using the following formulation [41]:

$$\begin{aligned} \{\hat{\mathbf{Z}}, \hat{\mathbf{X}}\} = \underset{\mathbf{Z},\mathbf{X}\in\mathbb{R}^{N\times L}}{\arg\min} \; \|\mathbf{Z}\|_* + \lambda\|\mathbf{X}\|_1 \\ \text{subject to } \; \mathbf{Y} = \mathbf{Z} + \mathbf{X}. \end{aligned} \tag{2.6}$$

The nuclear-norm in (2.6) promotes a low rank solution for $\mathbf{Z}$; the ℓ_1-norm penalty promotes sparsity in $\mathbf{X}$. A well-known application of RPCA is background subtraction in videos with a stationary background. For such datasets, the shared background in the frames $\{y_i\}$ can be represented using a low-rank subspace. The moving foreground objects often have sparse support, and thus are absorbed into the sparse term $\mathbf{X}$.

We propose to replace the ℓ_1-norm regularization prior on $\mathbf{X}$ in (2.6) with the proposed regularizer in (2.2); this enables us to promote spatial smoothness in the support set of the foreground objects.

4) Structured Dictionary Learning: Similar to RPCA we consider another rank-minimization application, where additionally the dictionary model is also being learned. In [42], the authors demonstrated that given the labels it is possible to learn label consistent dictionaries which can improve the discriminative capability of inferred latent coefficient

$\mathbf{Z}$. This formulation is given by,

$$\{\hat{\mathbf{Z}}, \hat{\mathbf{X}}, \hat{\mathbf{D}}\} = \underset{\mathbf{Z}\in\mathbb{R}^{M\times L}, \mathbf{X}\in\mathbb{R}^{N\times L}, \mathbf{D}\in\mathbb{R}^{N\times M}}{\arg\min} \|\mathbf{Z}\|_* + \lambda\|\mathbf{X}\|_1 + \beta\|\mathbf{Z}\|_1 + \alpha\|\mathbf{Z}-\mathbf{Q}\|_F^2$$
$$\text{subject to } \mathbf{Y} = \mathbf{D}\mathbf{Z} + \mathbf{X}. \tag{2.7}$$

where $\mathbf{Y}$ denotes column-stacked feature input, $\mathbf{Z}$'s are the latent codes and $\mathbf{X}$ denotes some sparse noise. The class prior is given by the matrix $\mathbf{Q}$. The final penalty term propagate class structure into the learning process, hence helping dictionary atoms to evolve and arrange according to the structure in $\mathbf{Q}$. We propose to replace the penultimate and the final penalty term on $\mathbf{Z}$ using the block sparse regularizer in (2.2). Note that, in this model the block sparsity is promoted on the set of non-overlapping blocks.

2.3.2 Optimization Algorithms

We now develop efficient numerical methods for solving problems involving the regularizer (2.2). A common approach to enforce group sparsity in the statistics literature is consensus ADMM [43, 44], which we will briefly discuss in Section 2.3.2.1. For image processing and vision applications, where the datasets as well as the cliques tend to be large, the high memory requirements of ADMM render this approach unattractive. As a consequence, we propose an alternative method that uses fast convolution algorithms to perform gradient descent that exhibits low memory requirements and requires low computational complexity. In particular, our approach is capable of handling large-scale problems, such as those in video applications, which are out of the scope of memory-hungry ADMM algorithms.

We note that numerical methods for overlapping group sparsity have been studied in the context of statistical regression [28, 44, 45], but for different purposes. Yuan *et al.* [45] solves the regression variable selection problem using an accelerated gradient descent approach, whereas Deng [43] and Boyd [44] use consensus ADMM, which does not scale to high-dimensional problems. Compared to these methods, our approach provides significant speedups (see Section 2.4).

2.3.2.1 Proximal Minimization and ADMM

The simplest instance of the problem (2.1) is the proximal operator for the penalty term J in (2.2), defined as follows:

$$\mathrm{prox}_J(\mathbf{v}, \lambda) = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{v}\|^2 + \lambda J(\mathbf{x}). \tag{2.8}$$

Proximal minimization is a key sub-step in a large number of numerical methods. For example, the ADMM for TV minimization [37, 38] requires the computation of the proximal operator of the ℓ_1-norm. For such methods, the regularizer (2.2) is easily incorporated into the numerical procedure by replacing this proximal minimization with (2.8).

In the simplest case where the cliques in $\mathcal{C}$ are small and no other regularizers are needed, the proximal minimization (2.8) can be computed using ADMM [38, 44]. Similar approaches have been used for other applications of overlapping group sparsity [43]. It is key to realize that the regularization term in (2.8) can be reformulated as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{x} - \mathbf{v}\|_2^2 + \lambda \sum_{i=1}^{s} \sum_{c\in\mathcal{C}_i} \|\mathbf{x}_c\|_2. \tag{2.9}$$

Here, $\mathcal{C}_1, \ldots, \mathcal{C}_s$ are clique subsets for which the cliques in $\mathcal{C}_i$ are *disjoint*. For example, consider the case where the set of cliques contains all 2×2 image patches as shown in

Figure 2.1(d). For such a scenario, we need four subsets of disjoint cliques to represent every possible patch. The reformulated problem for the example graph will be of the form (8.10) with $s = 4$. In general, if cliques are formed by translating an $l \times l$ patch, l^2 subsets of cliques are required so that every subset contains only disjoint cliques.

To apply ADMM to this problem, we need to introduce s auxiliary variables $\mathbf{z}^1, \dots, \mathbf{z}^s$ each representing a copy of the original pixel values. The resulting problem is

$$\begin{aligned} \{\hat{\mathbf{x}}, \hat{\mathbf{z}}^i \, \forall i\} &= \underset{\mathbf{x}, \{\mathbf{z}^i\}_{i=1}^s}{\arg\min} \|\mathbf{x} - \mathbf{v}\|_2^2 + \lambda \sum_{i=1}^{s} \sum_{c \in \mathcal{C}_i} \|\mathbf{z}_c^i\|_2 \\ &\text{subject to } \; \mathbf{z}^i = \mathbf{x}, \;\; \forall i. \end{aligned} \tag{2.10}$$

This is an example of a consensus optimization problem, which can be solved using ADMM (see [43] for more details). An important property of this ADMM reformulation is that each vector $\mathbf{z}^i$ can be updated in closed form—an immediate result of the disjoint clique decomposition.

2.3.2.2 Forward-Backward Splitting (FBS) with Fast Fourier Transforms

The above discussed ADMM approach has several drawbacks. First, it is difficult to incorporate more regularizers (in addition to the support regularizer J) without the introduction of an excessive amount of additional auxiliary variables. Furthermore, the method becomes inefficient and memory intensive for large clique sizes and large data-sets (as it is the case for multiple images). For instance in RPCA, if the cliques are generated by $l \times l$ patches, l^2 variables $\{\mathbf{z}^i\}$ are required, each having the same dimensionality as original image data-set NL. Additionally, the dual variables for each equality constraints in (2.10) will require another $l^2 NL$ storage entries. As a consequence, for large values of

l, the memory requirements of ADMM become prohibitive.

We propose a new forward-backward splitting algorithm that exploits fast convolution operators and prevents the excessive memory overhead of ADMM-based methods. To this end, we propose to "smoothen" the objective via *hyperbolic regularization* of the ℓ_2-norm as

$$\|\mathbf{x}_c\|_2 \approx \|\mathbf{x}_c\|_{2,\epsilon} = \sqrt{x_1^2 + \cdots + x_n^2 + \epsilon^2} \tag{2.11}$$

for some small $\epsilon > 0$. For the sake of clarity, we describe the forward-backward splitting approach in the specific case of robust PCA. Note, however, that other regularizers are possible with only minor modifications.

Using the proposed support prior (8.7), we write

$$\{\hat{\mathbf{Z}},\hat{\mathbf{X}}\} = \arg\min_{\mathbf{Z},\mathbf{X}} \|\mathbf{Z}\|_* + \lambda J_\epsilon(\mathbf{X}) + \tfrac{\mu}{2}\|\mathbf{Y}-\mathbf{Z}-\mathbf{X}\|_F^2 \tag{2.12}$$

where

$$J_\epsilon(\mathbf{X}) = \textstyle\sum_{t=1}^{L} \sum_{c\in\mathcal{C}} \|\mathbf{X}_{t,c}\|_{2,\epsilon} \tag{2.13}$$

is the smoothed support regularizer, and $\mathbf{X}_{t,c}$ refers to the clique c drawn from column t of $\mathbf{X}$. We note that this formulation differs from that in Liu *et al.* [46], where the structured sparsity is induced across columns of $\mathbf{X}$ rather than blocks, and is solved using conventional ADMM.

The forward-backward splitting (or proximal gradient) method is a general framework for minimizing objective functions with two terms [47]. For the problem (2.12), the method alternates between gradient descent steps that only act on the smooth terms in (2.12), and a backward/proximal step that only acts on the nuclear norm term. The gradient of the (smoothed) proximal regularizer in (2.12) is given column-wise (i.e., image-wise) by

$$\nabla J_\epsilon(\mathbf{X}_t) = \textstyle\sum_{c\in C} \mathbf{X}_{t,c}\|\mathbf{X}_{t,c}\|_{2,\epsilon}^{-1}. \tag{2.14}$$

The gradient formula (2.14) requires the computation of the sum (2.13) for every clique c, and then, a summation over the reciprocals of these sums; this is potentially expensive if done in a naïve way. Fortunately, every block sum can be computed simultaneously by squaring all of the entries in $\mathbf{X}$, and then convolving the result with a block filter. The result of this convolution contains the value of $\|\mathbf{X}_{t,c}\|_{2,\epsilon}^2$ for all cliques c. Each entry in the result is then raised to the $-1/2$ power, and convolved again with a block filter to compute the entries in the gradient (2.14). Both of these two convolution operations can be computed quickly using fast Fourier transforms (FFTs), so that the computational complexity becomes independent of clique size.

Algorithm 1 shows the pseudocode for solving (2.12). In Steps 1 and 2, the values of $\mathbf{X}$ and $\mathbf{Z}$ are updated using gradient descent on (2.12), ignoring the nuclear norm regularizer. Step 3 accounts for the nuclear-norm term using its proximal mapping, which is given by

$$\operatorname{prox}_*(\mathbf{Q},\delta) = \mathbf{U}(\operatorname{sign}(\mathbf{S}) \circ \max\{|\mathbf{S}| - \delta, 0\})\mathbf{V}^T,$$

where $\mathbf{Q} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is a singular value decomposition of $\mathbf{Q}$, $|\mathbf{S}|$ denotes element-wise absolute value, and $\circ$ denotes element-wise multiplication.

The forward-backward splitting (FBS) procedure in Algorithm 1 is known to converge for sufficiently small stepsizes α [48]. Practical implementations of FBS 1 include adaptive stepsize selection [49], backtracking line search, or acceleration [48]. We use the

Algorithm 1 Forward-backward proximal minimization

Input: $\mathbf{Y}, \mu > 0, \lambda, \mathcal{C}_i, \alpha > 0$
Initialize: $\mathbf{X}^{(0)} = \mathbf{0}$, $\mathbf{Z}^{(0)} = \mathbf{0}$
Output: $\mathbf{X}^{(n)}, \mathbf{Z}^{(n)}$

```
1: while not converged do
2:     Step 1: Forward gradient descent on $X$,
3:     $\mathbf{X}_k^{(n)} = \mathbf{X}_k^{(n-1)} - \alpha\lambda\nabla J_\epsilon(\mathbf{X})$
              $+\alpha\mu(\mathbf{Y}_k - \mathbf{Z}_k^{(n-1)} - \mathbf{X}_k^{(n-1)})$
4:     Step 2: Forward gradient descent on $\mathbf{Z}$,
5:       $\mathbf{Z}_k^{(n)} = \mathbf{Z}_k^{(n-1)} + \alpha\mu(\mathbf{Y}_k - \mathbf{Z}_k^{(n-1)} - \mathbf{X}_k^{(n-1)})$
6:     Step 3: Backward gradient descent on $\mathbf{Z}$,
7:         $\mathbf{Z}^{(n)} = \mathrm{prox}_*(\mathbf{Z}^{(n)}, \alpha)$
8: end while
```

FASTA solver from [47], which combines such acceleration techniques.

We note that FBS 1 only requires a total of $4NL$ storage entries for $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and gradient $\nabla J_\epsilon(\mathbf{X})$. However, in order to solve RPCA formulation using ADMM we require $2l^2NL$ storage entries for auxiliary variables (as discussed before) and $4NL$ storage entries for the variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and dual variable of $\mathbf{Y} = \mathbf{X} + \mathbf{Z}$, leading to total of $(2l^2 + 4)NL$ storage entries. Since the memory usage and runtime of FBS is independent of the clique size, the advantage of FBS over ADMM is much greater for larger cliques.

2.3.2.3 Matching Pursuit Algorithm

For compressive-sensing problems involving large random matrices, *matching pursuit* algorithms (such as CoSaMP [36]) are an important class of sparse recovery methods. When signals have structured support, model-based matching pursuit routines have been proposed that require non-convex minimizations over Markov random fields [3]. In this section, we propose a model-based matching pursuit algorithm that achieves structured compressive signal recovery using *convex* sub-steps for which global minimizers are efficiently computable.

The proposed method, Convex Lattice Matching Pursuit (CoLaMP), is a greedy algorithm that attempts to solve

$$\begin{gathered}\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\boldsymbol{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda J(\mathbf{x}) \\ \text{subject to } \|\mathbf{x}\|_0 \leq K.\end{gathered} \tag{2.15}$$

The complete method is listed in Algorithm 2. In Step 1, CoLaMP proceeds like other matching pursuit algorithms; the unknown signal is estimated by multiplying the residual by the adjoint of the measurement operator. In Step 2, this estimate is refined by solving a support regularized problem of the form (2.8). We solve this problem either via ADMM or the FBS method in Algorithm 1). In Step 3, a least-squares (LS) problem is solved to identify the signal that best matches the observed data, assuming the correct support was identified in Step 2. This LS problem is solved by a conjugate gradient method. Finally, in Step 4, the residual (the discrepancy between $\boldsymbol{\Phi}\mathbf{x}$ and the data vector $\mathbf{y}$) is calculated. The algorithm is terminated if the residual becomes sufficiently small or a maximum number of iterations is reached.

CoLaMP has several desirable properties. First, the support set regularization (Step 2) helps to prevent signal support from growing quickly, and thus minimizes the cost of the least-squares problem in Step 3. Secondly, the use of a convex prior guarantees that a global minimum is obtained for every subproblem in Step 2, regardless of the considered clique structure. This is in stark contrast to other model-based recovery algorithms, such as LaMP[1], and model-based CoSaMP [2], which requires the solution to non-convex optimization problems to enforce structured support set models.

[1] It is possible to restrict LaMP to planar Ising models, in which case a global optimum is computable [3].

Algorithm 2 CoLaMP - Convex Lattice Matching Pursuit

```
Input: $\mathbf{y}, \mathbf{\Phi}, K, \lambda, \epsilon$
Initialize: $\mathbf{x}^{(0)} = \mathbf{0}, \mathbf{s}^{(0)} = \mathbf{0}, \mathbf{r}^{(0)} = \mathbf{y}$
Output: $\mathbf{x}^{(n)}$
 1: while $n \leq$ max_iterations and $\|\mathbf{r}^{(n)}\|_2 > \epsilon$ do
 2:     Step 1: Form temporary target signal
 3:         $\mathbf{v}^{(n)} \leftarrow \Phi^T \mathbf{r}^{(n-1)} + \mathbf{x}^{(n-1)}$
 4:     Step 2: Refine signal support using convex prior
 5:         $\mathbf{x}_r^{(n)} = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{v}^{(n)}\|_2^2 + \lambda J(x),$
 6:         $\mathbf{s} \leftarrow \mathrm{supp}(\mathbf{x}_r^{(n)})$
 7:     Step 3: Estimate target signal
 8:         Solve $\mathbf{\Phi}_s^T \mathbf{\Phi}_s \mathbf{x}_s = \mathbf{\Phi}_s^T \mathbf{y}$, with $\mathbf{\Phi}_s = \mathbf{\Phi}(:, \mathbf{s})$
 9:         Set all but largest $K$ entries in $\mathbf{x}_s$ to zero,
10:         $\mathbf{x}^{(n)}(s) = \mathbf{x}_s(s)$
11:     Step 4: Calculate data residual
12:         $\mathbf{r}^{(n)} \leftarrow \mathbf{y} - \mathbf{\Phi}\mathbf{x}^{(n)}$
13:         $n \leftarrow n + 1$
14: end while
```

2.4 Experiments

We now apply the proposed regularizer to a range of datasets to demonstrate its efficacy for various applications. Unless stated otherwise, we showcase our algorithms using overlapping cliques of size 2×2 as shown in Figure 2.1(b). Note that the numerical algorithms need not be restricted to those discussed above as different schemes (such as primal-dual decomposition) are needed for different situations.

2.4.1 Compressive Image Recovery

We first consider the recovery of background-subtracted images from compressive measurements. We use the "walking2" surveillance video data [50] with frames of dimension 288×384. Test data is generated by choosing two frames from a video sequence and computing the pixel-wise difference between their intensities. We compare the output of

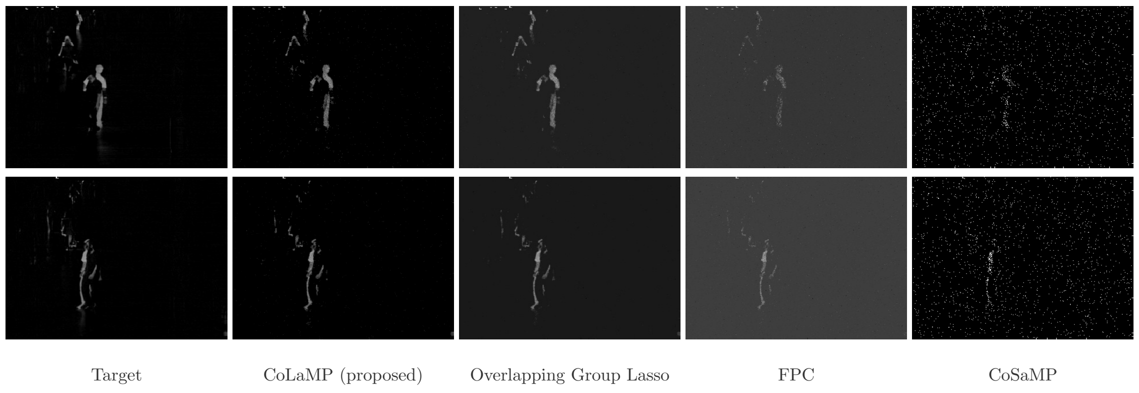


Figure 2.2: Compressed sensing recovery results for background subtracted images using $M = 3K$.

our proposed CoLaMP algorithm to that of other state-of-the-art recovery algorithms, such as overlapping group lasso [31], fixed-point continuation (FPC) [51] and CoSaMP [36]. Note that CoSaMP defines the support set using the $2K$ largest components of the error signal. The group lasso algorithm is equivalent to minimizing the objective in (2.15) using variational method. Unlike the CoLaMP algorithm, this method does not consider prescribed signal sparsity K. An example recovery using $M = 3K$ measurements is shown in Figure 2.2. The sparsity level K is chosen such that the recovered images account for 97% of the compressive signal energy. The average K across datasets is 2800 and we fix $\lambda = 2$. Note that the spatially clustered pixels are recovered almost perfectly. Further, we randomly generated 50 such test images from the above dataset and compared the performance of the CoLaMP, group lasso, and FPC algorithms under varying numbers of measurements from $1K$ to $5K$. The performance is measured in terms of the magnitude of reconstruction error normalized by the original image magnitude. Results are shown in Figure 2.4 (left). We clearly see that the proposed smooth sparsity prior significantly improves the reconstruction quality over FPC. Furthermore, our algorithm is $7\times$ faster

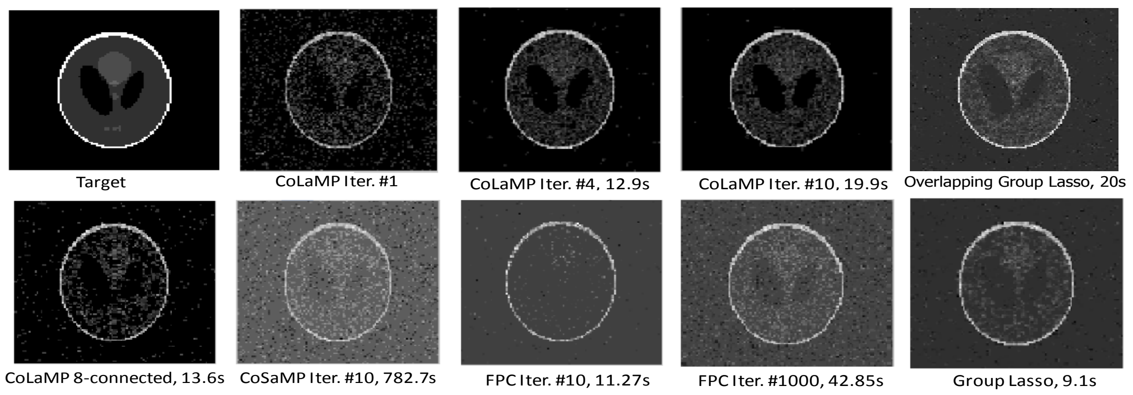


Figure 2.3: Robust recovery results for the phantom image from a noisy compressed signal.

than the group lasso algorithm. For $M/K = 3$, the average runtime is $215s$ for CoLaMP and $1510s$ for the group lasso algorithm.

2.4.2 Robust Signal Recovery

We next showcase the suitability of CoLaMP for signal recovery from noisy compressive measurements. We consider a 100×100 Shepp—Logan phantom image with a support size of $K = 2636$. A Gaussian random measurement matrix was used to sample $M = 2K$ measurements, and the measurements were corrupted with additive white Gaussian noise. The signal-to-noise ratio of the resulting measurements is 10 dB. Figure 2.3 shows the original and recovered images for various recovery algorithms. We also show the output from the first few iterates of the CoLaMP algorithm. The support of the target signal is almost exactly recovered within four iterations of CoSaMP and stabilizes by the end of 10 iterations. Figure 2.3 also shows the recovery times of various algorithms running on the same laptop computer. CoLaMP is approximately $40\times$ faster than the CoSaMP algorithm and it is at least $2\times$ faster than FPC.

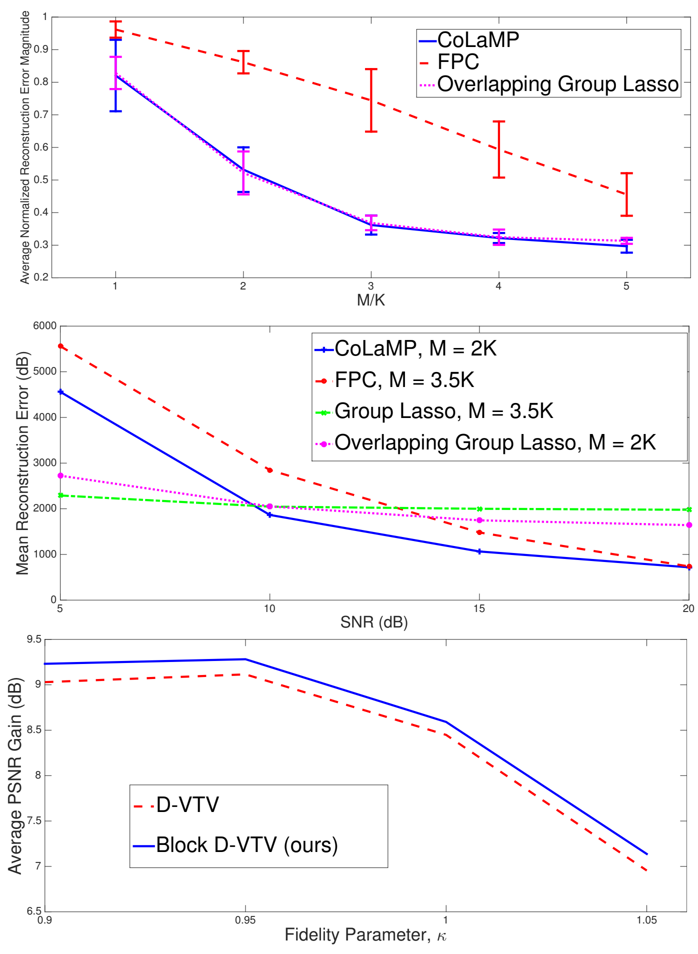


Figure 2.4: Quantitative Comparison: (left) Recovery performance of compressed sensing on background subtracted images; (center) Robust compressed sensing recovery error at various SNR; (right) Average denoising gain in PSNR (dB) for various values of κ

To enable a fair comparison, we also show the output obtained with CoLaMP using the 8-connected pixel clique in Figure 2.1(c), as well as the output of the group lasso algorithm [32], where each clique is of size 2×2. All these algorithms and our proposed method are implemented using ADMM. Not surprisingly, while all these algorithms beat CoLaMP in terms of runtime, their recovered signals do not match CoLaMP in terms of perceived closeness to target signal as shown in Figure 2.3. The CoLaMP results are regularized by $\lambda_0 = 16$. We then used an increasing value of $\lambda_n = 1.02^n \lambda_0$ where n is the iteration number. In practice, we obtain better results if λ increases over time as it will heavily penalize sparse, blocky noise. For all other algorithms, we used the implementations provided by the authors.

For detailed quantitative comparisons, we repeat the above experiment using 100 Gaussian random measurement matrices and record the average reconstruction error with SNR varying from $5\,\text{dB}$ to $20\,\text{dB}$. For each algorithm, M is fixed to the minimal measurement number required to give close to perfect recovery in the presence of noise. For CoLaMP and overlapping group lasso, we set $M = 2K$, whereas for FPC and non-overlapping group lasso we set $M = 3.5K$. Figure 2.4 (center) illustrates that CoSaMP outperforms FPC at all SNRs even with $1.5K$ fewer measurements. Group lasso performs best at low SNR while its performance flattens out starting at $10\,\text{dB}$.

2.4.3 Color Image Denoising

We now consider a variant of the denoising problem (2.5) where the image gradient is defined over color images using the decorrelated vectorized TV (D-VTV) proposed in [40]

$$\begin{aligned} \hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^{3N}}{\arg\min} \sum_{c\in\mathcal{C}} \lambda\|\nabla_d \mathbf{x}_c^{\ell}\|_2 + \|\nabla_d \mathbf{x}_c^{ch}\|_2 \\ \text{subject to} \quad \|\mathbf{x}-\mathbf{y}\|_2 \leq \kappa m. \end{aligned} \tag{2.16}$$

Here, $\nabla_d \mathbf{x}^{\ell} \in \mathbb{R}^{2N}$ and $\nabla_d \mathbf{x}^{ch} \in \mathbb{R}^{4N}$ represent the stacked gradients of luminance and chrominance channels of the input color image, the constant m depends on the noise level, and κ is a fidelity parameter. To solve this problem numerically, we use the primal-dual algorithm described in [40], but we replace the shrinkage operator with the proximal operator (2.8) to adapt our clique-based regularizer.

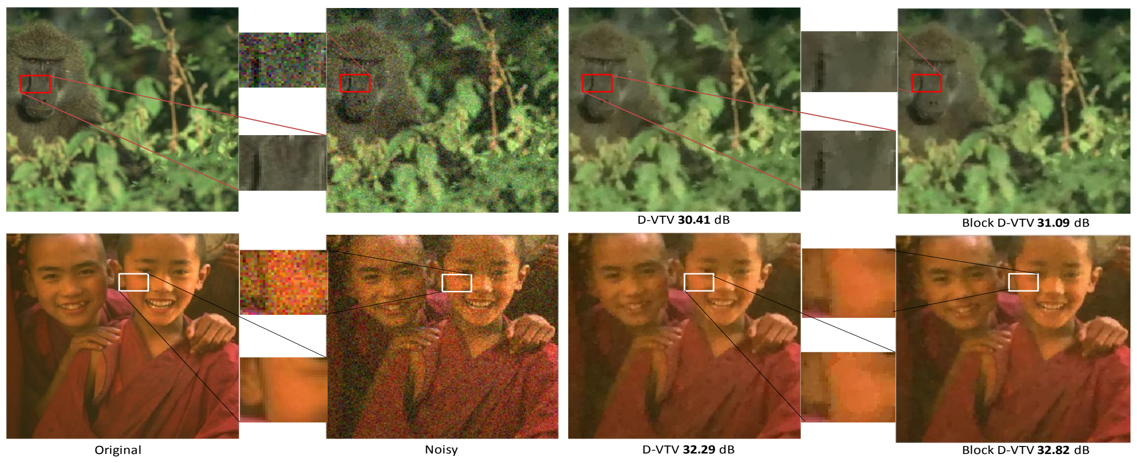
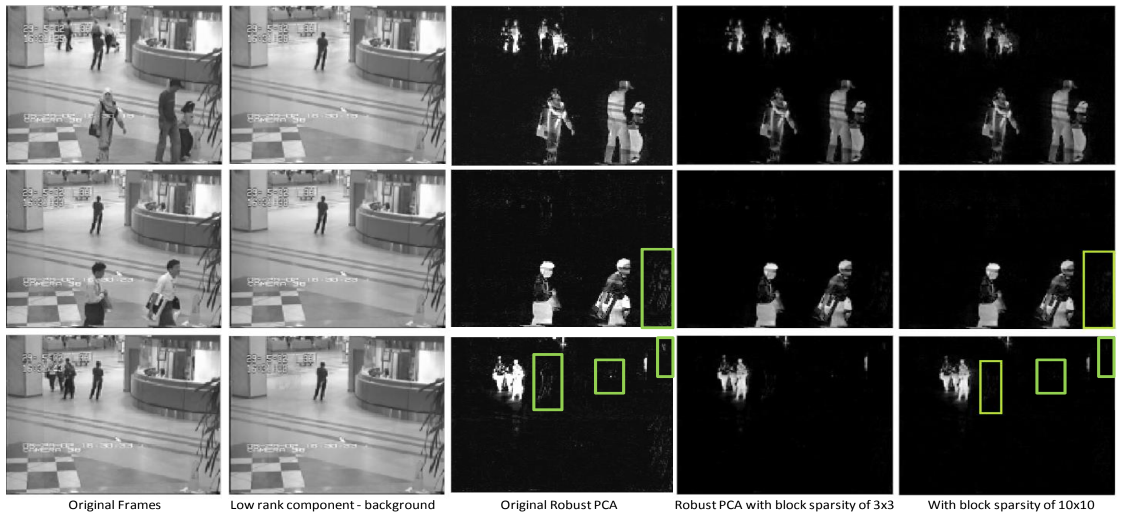


Figure 2.5: Restoration of noisy images using Block D-VTV and existing D-VTV (best viewed in color).

Following a protocol similar to D-VTV [40], we conduct experiments using 300 images from the Berkeley Segmentation Database [52]. Noisy images with average PSNR 20 dB are obtained by adding white Gaussian noise. The resulting denoised output of our method (Block D-VTV) is compared to D-VTV in Figure 2.5. The zoomed-in version reveals that our method exhibits less uneven color artifacts and less pronounced staircasing artifacts than the D-VTV results. A quantitative comparison measured using average PSNR gain (in dB) is drawn in Figure 2.4 (right) for various values of κ. Our method outperforms D-VTV by 0.25 dB. Also note that our method, Block D-VTV, obtains relatively better PSNR gain than the state-of-the-art D-VTV method at smaller values of κ. This observed gain is significant because smaller κ values lead to a tighter fidelity constraint and thus a smaller solution space around the noisy input. In such situations, Block D-VTV helps to improve image quality by leveraging input from neighboring pixels.



Original Frames
Low rank component - background
Original Robust PCA
Robust PCA with block sparsity of 3x3
With block sparsity of 10x10

Figure 2.6: Sparse-and-low-rank decomposition using original robust PCA and proposed approach.

2.4.4 Video Decomposition

We consider the robust PCA (RPCA) problem for structured sparsity of size 10×10 as formulated in (2.12) and using Algorithm 2. We consider the same airport surveillance video data [53] as in [41] with frames of dimension 144×176. For a clique formed from $l \times l$ patches, we observed that $\lambda = 1/(l\sqrt{n_1})$ works best for our experiments as opposed to $\lambda = 1/\sqrt{n_1}$ used in [41]. This is because each element of the matrix $\mathbf{X}$ is shared by l^2 sparsity inducing terms. The resulting low rank components (background) and foreground components of three such example video frames are shown in Figure 2.6. For all the approaches, the low rank components are nearly identical. We observe that the rank of the low-rank component remains the same. As highlighted with the green box, the noisy sparse edges appearing in the original RPCA disappear from the foreground component using our proposed method. We also display the foreground component obtained using smaller overlapping cliques of size 3×3, but solved using ADMM as opposed to forward-backward splitting (Algorithm 1). We found that for clique size of 10×10 the ADMM method becomes intractable because it requires approximately 50$\times$ more memory than the proposed forward-backward splitting method with fast convolutions (i.e., $204NL$ vs. $4NL$).

2.4.5 Structured Dictionary Learning

We finally consider learning structured dictionary on the Extended YaleB Face database. This database contains 2,414 frontal face cropped images of 38 people taken under various lighting conditions. The original images of sizes 192×168 is downsampled by eight for our experiments. Following the protocol in [42] we randomly select 32 images from each class for training and learn 20 dictionary atoms per class. Hence the non-overlapping structured sparsity on $\mathbf{Z}$ is of size 20×32. The above procedure is repeated five times and the average recognition accuracy on test samples is reported in Table 2.1. During inference, each test sample is classified based on the maximum group energy of the inferred latent coefficients $\max_{g\in\mathcal{G}} \|\mathbf{z}_g\|_2$. The upper half represents all the

Method	Accuracy
Ours	**97**.1
Label consistent low rank [42]	90
Low Rank with structural coherence [54]	89.5
Discriminative Low Rank** [55]	**98.2**
Low Rank and Sparse face representation [56]	84.5
Locally constrained linear coding [57]	76
LC-KSVD** [58]	96.7

Table 2.1: Recognition accuracy on Extended YaleB database using various dictionary learning approach.

algorithm involving low-rank decomposition of matrix $\mathbf{Z}$ whereas ** indicates the result using all the training samples as dictionary atoms.

2.5 Conclusions

We have proposed a novel structured support regularizer for convex sparse recovery. Our regularizer can be applied to a variety of problems, including sparse-and-low-rank decomposition and denoising. For compressive signal recovery using large unstructured matrices, our convex regularizer can be used to improve the recovery quality of existing matching-pursuit algorithms. Compared to existing algorithms for this task, our proposed approach enjoys the capability of fast signal reconstruction from fewer measurements while exhibiting superior robustness against spurious artifacts and noise. For color image denoising, the restored images reveal more homogeneous color effects. For robust PCA, we achieve improved foreground-background separation with far fewer artifacts. For dictionary learning, our proposed approach promotes diversity among the dictionary atoms with an improved classification accuracy. We envision many more applications that could benefit of the proposed regularizer, including deblurring, inpainting and multitask classification.

Chapter 3: Robust Continuous Clustering

3.1 Introduction

The success with the use of spatial prior for signal recovery motivated us to explore further and study the utilization of local structure prior used in relevant vision problems. The literature survey led us to draw connection with the recent algorithm developed for segmentation and clustering [59–62]. Indeed, most of these algorithm requires similarity graph as input prior. In a widely used spectral clustering [59] algorithm the similarity graph is utilized to construct a graph Laplacian matrix. This is followed by eigen-decomposition and bipartition of graph using second smallest eigen vector. In [60–62], the authors explore the concept of cluster formation based on continuous MRF objective. Each of these work independently proposes different algorithm for different penalty on the pairwise term. The pairwise terms are once again constructed based on the similarity graph input prior. However, we noted that all of these clustering algorithms are non-scalable, very sensitive to the hyper-parameters and outliers. Moreover, most of these algorithms is designed to operate using the number of ground truth clusters which is practically not feasible to obtain for real world data.

In this chapter, we present a clustering algorithm that is fast, easy to use, and effective in high dimensions. The algorithm optimizes a clear continuous objective using standard numerical methods that scale to massive datasets. The number of clusters need not be known in advance. The hyper-parameters can be set automatically and need not be tuned for different datasets. As we will see later, unlike the spectral and other traditional algorithms, the advantage of optimizing over continuous objective is differentiability and hence it can be easily back-propagated.

The operation of the algorithm can be understood by contrasting it with other popular clustering techniques. In center-based algorithms such as k-means [63, 64], a small set of putative cluster centers is initialized from the data and then iteratively refined. In affinity propagation [65], data points communicate over a graph structure to elect a subset of the points as representatives. In the presented algorithm, each data point has a dedicated representative, initially located at the data point. Over the course of the algorithm, the representatives move and coalesce into easily separable clusters.

Our formulation is indeed based on the convex relaxations for clustering [60–62]. However, our objective is deliberately not convex. We use redescending robust estimators that allow even heavily mixed clusters to be untangled by optimizing a single continuous objective. Despite the nonconvexity of the objective, the optimization can still be performed using standard linear least-squares solvers, which are highly efficient and scalable. Since the algorithm expresses clustering as optimization of a continuous objective based on robust estimation, we call it Robust Continuous Clustering (RCC).

One of the characteristics of the presented formulation is that clustering is reduced to optimization of a continuous objective. This enables the integration of clustering in end-to-end feature learning pipelines. We demonstrate this by extending RCC to perform joint clustering and dimensionality reduction. The extended algorithm, called RCC-DR, learns an embedding of the data into a low-dimensional space in which it is clustered. Embedding and clustering are performed jointly, by an algorithm that optimizes a clear global objective.

We evaluate RCC and RCC-DR on a large number of datasets from a variety of domains. These include image datasets, document datasets, a dataset of sensor readings from the Space Shuttle, and a dataset of protein expression levels in mice. Experiments demonstrate that our method significantly outperforms prior state-of-the-art techniques. RCC-DR is particularly robust across datasets from different domains, outperforming the best prior algorithm by a factor of 3 in average rank. The code is available at `https://bitbucket.org/sohilas/robust-continuous-clustering/src`.

3.2 Model and Algorithms

3.2.1 Formulation

We consider the problem of clustering a set of n data points. The input is denoted by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, where $\mathbf{x}_i \in \mathbb{R}^D$. Our approach operates on a set of representatives $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$, where $\mathbf{u}_i \in \mathbb{R}^D$. The representatives $\mathbf{U}$ are initialized at the corresponding data points $\mathbf{X}$. The optimization operates on the representation $\mathbf{U}$, which coalesces to reveal the cluster structure latent in the data. Thus the number of clusters need not be known in advance.

The Robust Continuous Clustering (RCC) objective has the following form:

$$\mathbf{C}(\mathbf{U}) = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{u}_i\|_2^2 + \frac{\lambda}{2} \sum_{(p,q)\in\mathcal{E}} w_{p,q}\, \rho\big(\|\mathbf{u}_p - \mathbf{u}_q\|_2\big). \quad (3.1)$$

Here $\mathcal{E}$ is the set of edges in a graph connecting the data points. The graph is constructed automatically from the data. We use mutual k-nearest neighbors (m-kNN) connectivity [66], which is more robust than commonly used kNN graphs. The weights $w_{p,q}$ balance the contribution of each data point to the pairwise terms and λ balances the strength of different objective terms.

The function $\rho(\cdot)$ is a penalty on the regularization terms. The use of an appropriate robust penalty function ρ is central to our method. Since we would like representatives $\mathbf{u}_i$ of observations from the same latent cluster to collapse into a single point, a natural penalty would be the ℓ_0 norm ($\rho(y) = [y \neq 0]$, where $[\cdot]$ is the Iverson bracket). However, this transforms the objective into an intractable combinatorial optimization problem. At another extreme, recent work has explored the use of convex penalties, such as the ℓ_1 and ℓ_2 norms [60, 61]. This has the advantage of turning objective (3.1) into a convex optimization problem. However, convex functions—even the ℓ_1 norm—have limited robustness to spurious edges in the connectivity structure $\mathcal{E}$, because the influence of a spurious pairwise term does not diminish as representatives move apart during the optimization. Given noisy real-world data, heavy contamination of the connectivity structure by connections across different underlying clusters is inevitable. Our method uses robust estimators to automatically prune spurious intercluster connections while maintaining veridical intracluster correspondences, all within a single continuous objective.

The second term in objective (3.1) is related to the mean shift objective [67]. The RCC objective differs in that it includes an additional data term, uses a sparse (as opposed to a fully-connected) connectivity structure, and is based on robust estimation.

Our approach is based on the duality between robust estimation and line processes [68]. We introduce an auxiliary variable $l_{p,q}$ for each connection $(p,q) \in \mathcal{E}$ and optimize a joint objective over the representatives $\mathbf{U}$ and the line process $\mathbb{L} = \{l_{p,q}\}$:

$$\mathbf{C}(\mathbf{U},\mathbb{L}) = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{u}_i\|_2^2 + \frac{\lambda}{2}\sum_{(p,q)\in\mathcal{E}} w_{p,q}\left(l_{p,q}\|\mathbf{u}_p - \mathbf{u}_q\|_2^2 + \Psi(l_{p,q})\right). \quad (3.2)$$

Here $\Psi(l_{p,q})$ is a penalty on ignoring a connection (p,q): $\Psi(l_{p,q})$ tends to zero when the connection is active ($l_{p,q} \to 1$) and to one when the connection is disabled ($l_{p,q} \to 0$). A broad variety of robust estimators $\rho(\cdot)$ have corresponding penalty functions $\Psi(\cdot)$ such that objectives (3.1) and (3.2) are equivalent with respect to $\mathbf{U}$: optimizing either of the two objectives yields the same set of representatives $\mathbf{U}$. This formulation is related to iteratively reweighted least squares (IRLS) [69], but is more flexible due to the explicit variables $\mathbb{L}$ and the ability to define additional terms over these variables.

Objective (3.2) can be optimized by any gradient-based method. However, its form enables efficient and scalable optimization by iterative solution of linear least-squares systems. This yields a general approach that can accommodate many robust nonconvex functions ρ, reduces clustering to the application of highly optimized off-the-shelf linear system solvers, and easily scales to datasets with hundreds of thousands of points in tens of thousands of dimensions. In comparison, recent work has considered a specific family of concave penalties and derived a computationally intensive majorization-minimization scheme for optimizing the objective in this special case [62]. Our work provides a highly efficient general solution.

While the presented approach can accommodate many estimators in the same computationally efficient framework, our exposition and experiments will use a form of the well-known Geman-McClure estimator [70]:

$$\rho(y) = \frac{\mu y^2}{\mu + y^2}, \tag{3.3}$$

where μ is a scale parameter. The corresponding penalty function that makes objectives (3.1) and (3.2) equivalent with respect to $\mathbf{U}$ is

$$\Psi(l_{p,q}) = \mu(\sqrt{l_{p,q}} - 1)^2. \tag{3.4}$$

3.2.2 Optimization

Objective (3.2) is biconvex on $(\mathbf{U}, \mathbb{L})$. When variables $\mathbf{U}$ are fixed, the individual pairwise terms decouple and the optimal value of each $l_{p,q}$ can be computed independently in closed form. When variables $\mathbb{L}$ are fixed, objective (3.2) turns into a linear least-squares problem. We exploit this special structure and optimize the objective by alternatingly updating the variable sets $\mathbf{U}$ and $\mathbb{L}$. As a block coordinate descent algorithm, this alternating minimization scheme provably converges.

When $\mathbf{U}$ are fixed, the optimal value of each $l_{p,q}$ is given by

$$l_{p,q} = \left(\frac{\mu}{\mu + \|\mathbf{u}_p - \mathbf{u}_q\|_2^2}\right)^2. \tag{3.5}$$

This can be verified by substituting (3.5) into (3.2), which yields objective (3.1) with respect to $\mathbf{U}$.

When $\mathbb{L}$ are fixed, we can rewrite (3.2) in matrix form and obtain a simplified expression for solving $\mathbf{U}$:

$$\arg\min \frac{1}{2}\|\mathbf{X} - \mathbf{U}\|_F^2 + \frac{\lambda}{2} \sum_{(p,q)\in\mathcal{E}} w_{p,q} l_{p,q} \|\mathbf{U}(\mathbf{e}_p - \mathbf{e}_q)\|_2^2, \tag{3.6}$$

where $\mathbf{e}_i$ is an indicator vector with the i^{th} element set to 1. This is a linear least-squares problem that can be efficiently solved using fast and scalable solvers. The linear least-squares formulation is given by

$$\mathbf{U}\mathbf{M} = \mathbf{X}, \text{ where } \quad \mathbf{M} = \mathbf{I} + \lambda \sum_{p,q\in\mathcal{E}} w_{p,q} l_{p,q} (\mathbf{e}_p - \mathbf{e}_q)(\mathbf{e}_p - \mathbf{e}_q)^\top. \tag{3.7}$$

Here $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. It is easy to prove that

$$\mathbf{A} \triangleq \sum_{p,q\in\mathcal{E}} w_{p,q} l_{p,q} (\mathbf{e}_p - \mathbf{e}_q)(\mathbf{e}_p - \mathbf{e}_q)^\top \tag{3.8}$$

is a Laplacian matrix and hence $\mathbf{M}$ is symmetric and positive-semidefinite. As with any multigrid solver, each row of $\mathbf{U}$ in (3.7) can be solved independently and in parallel.

The RCC algorithm is summarized in Algorithm 3. Note that all updates of $\mathbf{U}$ and $\mathbb{L}$ optimize the same continuous global objective (3.2).

Algorithm 3 Robust Continuous Clustering

```
1:  input: Data samples $\{\mathbf{x}\}_{i=1}^n$.
2:  output: Cluster assignment $\{\hat{c}_i\}_{i=1}^n$.
3:  Construct connectivity structure $\mathcal{E}$.
4:  Precompute $\chi = \|\mathbf{X}\|_2, w_{p,q}, \delta$.
5:  Initialize $\mathbf{u}_i = \mathbf{x}_i, l_{p,q} = 1, \mu \gg \max \|\mathbf{x}_p - \mathbf{x}_q\|_2^2, \lambda = \frac{\chi}{\|\mathbf{A}\|_2}$.
6:  while $|\mathbf{C}^t - \mathbf{C}^{t-1}| < \varepsilon$ or $t <$ maxiterations do
7:      Update $l_{p,q}$ using (3.5) and $\mathbf{A}$ using (3.8).
8:      Update $\{\mathbf{u}_i\}_{i=1}^n$ using (3.7).
9:      Every four iterations, update $\lambda = \frac{\chi}{\|\mathbf{A}\|_2}, \mu = \max\left(\frac{\mu}{2}, \frac{\delta}{2}\right)$.
10: end while
11: Construct graph $\mathcal{G} = (\mathcal{V}, \mathcal{F})$ with $f_{p,q} = 1$ if $\|\mathbf{u}_p^* - \mathbf{u}_q^*\|_2 < \delta$.
12: Output clusters given by the connected components of $\mathcal{G}$.
```

The algorithm employs graduated nonconvexity [71]. It begins with a locally convex approximation of the objective, obtained by setting μ such that the second derivative of the estimator is positive ($\ddot{\rho}(y) > 0$) over the relevant part of the domain. Over the iterations,

μ is automatically decreased, gradually introducing nonconvexity into the objective. The optimization of $\mathbf{U}$ is illustrated in Figure 4.2.

3.2.3 Convergence

Under certain assumptions, such continuation schemes are known to attain solutions that are close to the global optimum [72]. We briefly outline the argument for the convergence of Algorithm 3 which is based on the convergence of alternating minimization for biconvex problems [73, 74]. Due to the duality between robust estimation and line processes, it is sufficient to show convergence w.r.t. objective 3.2. The clustering objective at iteration t is given by

$$\mathbb{E}^t = \mathbf{C}(\mathbf{U}^t, \mathbb{L}^t) = \overbrace{\frac{1}{2}\|\mathbf{X}-\mathbf{U}\|_F^2 + \frac{\lambda}{2}\sum_{(p,q)\in\mathcal{E}} w_{p,q} \underbrace{\left(l_{p,q}\|\mathbf{U}(\mathbf{e}_p - \mathbf{e}_q)\|_2^2 + \Psi(l_{p,q};\mu)\right)}_{\mathrm{II}}}^{\mathrm{I}}. \tag{3.9}$$

For convergence we need to prove that $\mathbb{E}^{t+1} \leq \mathbb{E}^t$.

For a fixed λ, Algorithm 3 alternates between optimizing I and II. Each of these are convex w.r.t. $\mathbf{U}$ and $\mathbb{L}$, respectively. This is due to their Hessians being positive semidefinite:

$$\frac{\partial^2 \mathrm{I}}{\partial \mathbf{U}^2} = \mathbf{M} \succeq 0 \tag{3.10}$$

$$\frac{\partial^2 \mathrm{II}}{\partial {l_{p,q}}^2} = \frac{1}{2 l_{p,q}\sqrt{l_{p,q}}} \geq 0 \tag{3.11}$$

Due to convexity, the stationary points computed for $\mathbf{U}$ and $\mathbb{L}$ using the first order condition at each intermediate time step are nothing but the minima of the functions I and

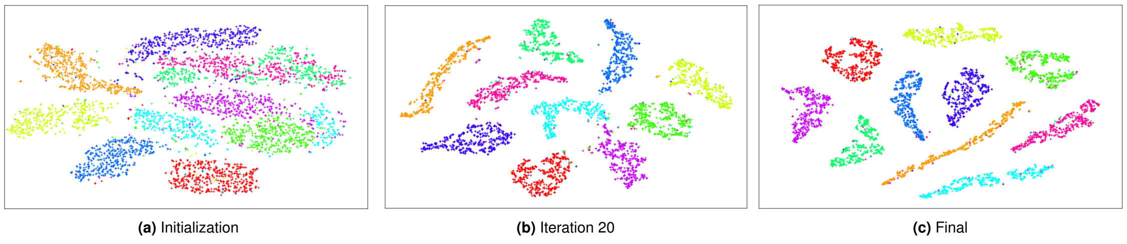


Figure 3.1: Robust Continuous Clustering (RCC) on the MNIST dataset. Each data point $\mathbf{x}_i$ has a corresponding representative $\mathbf{u}_i$. The representatives are optimized to reveal the structure of the data. The different parts of the figure visualize the representation $\mathbf{U}$ using the t-SNE algorithm [75]. Ground-truth clusters are coded by color. (a) The initial state, $\mathbf{U} = \mathbf{X}$. (b) The representation $\mathbf{U}$ after 20 iterations of the optimization. (c) The final representation produced by the algorithm.

II. This gives us

$$\mathbb{E}^{t+\frac{1}{2}} = \mathbf{C}(\mathbf{U}^t, \mathbb{L}^{t+1}) \leq \mathbb{E}^t = \mathbf{C}(\mathbf{U}^t, \mathbb{L}^t) \tag{3.12}$$

$$\mathbb{E}^{t+1} = \mathbf{C}(\mathbf{U}^{t+1}, \mathbb{L}^{t+1}) \leq \mathbb{E}^{t+\frac{1}{2}} = \mathbf{C}(\mathbf{U}^t, \mathbb{L}^{t+1}) \tag{3.13}$$

The above steps show that $\mathbb{E}^{t+1} \leq \mathbb{E}^{t+\frac{1}{2}} \leq \mathbb{E}^t$.

This proves convergence. However, the result is not guaranteed to be a global optimum due to the non-convexity of the problem. We alleviate this issue by starting with a convex approximation and gradually introducing non-convexity. Every update for μ and λ can be considered the start of a new optimization with the initial values for $\mathbf{U}$ and $\mathbb{L}$ given by the previous iterate. This ensures that non-convex subproblems are properly initialized. Recent theoretical analyses of such continuation schemes [72] have shown that they can approach the global optimum of the original non-convex function.

3.2.4 Hyper-parameters

The RCC algorithm has five parameters: $w_{p,q}$, λ, μ, δ, and k for the k-nearest neighbor graph. The value of k is set to 10. All other parameters are set automatically using corresponding formulas. Each formula is justified as follows:

1.
$$w_{p,q} = \frac{\sum_{i=1}^{n} N_i/n}{\sqrt{N_p N_q}}. \tag{3.14}$$

   This is effectively the ratio of the average degree in the graph to the geometric mean of the degrees of nodes p and q.

2. The parameter λ in the RCC objective (3.1) balances the strength of the data terms and pairwise terms. The reformulation of RCC as a linear least-squares problem enables setting λ automatically. Specifically, equation (3.7) suggests that the data terms and pairwise terms can be balanced by setting
$$\lambda = \frac{\|\mathbf{X}\|_2}{\|\mathbf{A}\|_2}. \tag{3.15}$$

   This can be easily obtained by considering the objective in 3.2 and differentiating. The gradient w.r.t. $\mathbf{U}$ is given by,
$$\frac{\partial \mathbf{C}}{\partial \mathbf{U}} = -\mathbf{X} + \mathbf{U} + \lambda \mathbf{U} \left(\sum_{(p,q)\in\mathcal{E}} w_{p,q} l_{p,q} (\mathbf{e}_p - \mathbf{e}_q)(\mathbf{e}_p - \mathbf{e}_q)^T \right) \tag{3.16}$$
$$= -\mathbf{X} + \mathbf{U} + \lambda \mathbf{U}\mathbf{A} \tag{3.17}$$

   In order to prevent any one term from dominating the gradient, one should balance the spectral norm of each of the contributing terms. Neglecting the term $\mathbf{U}$ in (3.17), this balance is approximately achieved by setting the ratio $\frac{\|\mathbf{X}\|_2}{\lambda\|\mathbf{A}\|_2}$ to one. This leads to (3.15). The value of λ is updated automatically according to this formula after every update of μ. An update only involves computing the largest eigenvalue of the Laplacian matrix $\mathbf{A}$. The spectral norm of $\mathbf{X}$ is precomputed at initialization and reused.

3. In order to start RCC algorithm with a locally convex approximation of the objective, μ should be initialize such that the second derivative of the estimator is positive ($\ddot{\rho}(y) > 0$). In our case, the second derivative is given by

$$\ddot{\rho}(y) = \frac{2\mu(\mu - 3y^2)}{(\mu + y^2)^3}. \tag{3.18}$$

   In order to guarantee $\ddot{\rho}(y) > 0$, one should set $\mu > 3y^2$. Given that $\mathbf{u}_i$'s are initialized to $\mathbf{x}_i$, this leads us to initialize $\mu = 3\max\|\mathbf{x}_p - \mathbf{x}_q\|_2^2$.

4. The threshold δ is set to the mean of the lengths of the shortest 1% of the edges in $\mathcal{E}$. We assume that the shortest edge in the original graph $\mathcal{E}$ is spanned by two nodes from the same cluster. The length of this shortest edge is therefore a natural threshold for the connected components in the new graph. However, due to noisy data, sparse features, and data sample replication, this value is unreliable and can be as low as zero. Hence, we set δ to an average over 1% of the shortest edge lengths in $\mathcal{E}$.

The termination conditions are set to maxiterations $= 100$ and $\varepsilon = 0.1$.

3.2.5 Joint Clustering and Dimensionality Reduction

The RCC formulation can be interpreted as learning a graph-regularized embedding $\mathbf{U}$ of the data $\mathbf{X}$. In the algorithm presented in the preceding sections, the dimensionality of the embedding $\mathbf{U}$ is the same as the dimensionality of the data $\mathbf{X}$. However, since RCC optimizes a continuous and differentiable objective, it can be used within end-to-end feature learning pipelines. We now demonstrate this by extending RCC to perform joint clustering and dimensionality reduction. Such joint optimization has been considered in recent work [76, 77]. The algorithm we develop, RCC-DR, learns a linear mapping into a reduced space in which the data is clustered. The mapping is optimized as part of the clustering objective, yielding an embedding in which the data can be clustered most effectively. RCC-DR inherits the appealing properties of RCC: clustering and dimensionality reduction are performed jointly by optimizing a clear continuous objective, the framework supports nonconvex robust estimators that can untangle mixed clusters, and optimization is performed by efficient and scalable numerical methods.

We begin by considering an initial formulation for the RCC-DR objective:

$$\mathbf{C}(\mathbf{U},\mathbf{Z},\mathbf{D}) = \|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_2^2 + \gamma\sum_{i=1}^{n}\|\mathbf{z}_i\|_1 + \nu\left(\sum_{i=1}^{n}\|\mathbf{z}_i-\mathbf{u}_i\|_2^2 + \frac{\lambda}{2}\sum_{(p,q)\in\mathcal{E}} w_{p,q}\rho\left(\|\mathbf{u}_p-\mathbf{u}_q\|_2\right)\right). \tag{3.19}$$

Here $\mathbf{D}\in\mathbb{R}^{D\times d}$ is a dictionary, $\mathbf{z}_i\in\mathbb{R}^d$ is a sparse code corresponding to the i^{th} data sample, and $\mathbf{u}_i\in\mathbb{R}^d$ is the low-dimensional embedding of $\mathbf{x}_i$. For a fixed $\mathbf{D}$, the parameter ν balances the data term in the sparse coding objective with the clustering objective in the reduced space. This initial formulation (3.19) is problematic because in the beginning of the optimization the representation $\mathbf{U}$ can be noisy due to spurious intercluster connections that have not yet been disabled. This had no effect on the convergence of the original RCC objective (3.1), but in formulation (3.19) the contamination of $\mathbf{U}$ can infect the sparse coding system via $\mathbf{Z}$ and corrupt the dictionary $\mathbf{D}$. For this reason, we use a different formulation that has the added benefit of eliminating the parameter ν:

$$\mathbf{C}(\mathbf{U},\mathbf{Z},\mathbf{D}) = \|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_2^2 + \gamma\sum_{i=1}^{n}\|\mathbf{z}_i\|_1 + \sum_{i=1}^{n}\rho_1\left(\|\mathbf{z}_i-\mathbf{u}_i\|_2\right) + \frac{\lambda}{2}\sum_{(p,q)\in\mathcal{E}} w_{p,q}\rho_2\left(\|\mathbf{u}_p-\mathbf{u}_q\|_2\right). \tag{3.20}$$

Here we replaced the ℓ_2 penalty on the data term in the reduced space with a robust penalty. We use the Geman-McClure estimator (3.3) for both ρ_1 and ρ_2.

To optimize objective (3.20), we introduce line processes $\mathbb{L}^1$ and $\mathbb{L}^2$ corresponding to the data and pairwise terms in the reduced space, respectively, and optimize a joint objective over $\mathbf{U}$, $\mathbf{Z}$, $\mathbf{D}$, $\mathbb{L}^1$, and $\mathbb{L}^2$. The optimization is performed by block coordinate descent over these groups of variables. The line processes $\mathbb{L}^1$ and $\mathbb{L}^2$ can be updated in closed form as in (3.5). The variables $\mathbf{U}$ are updated by solving the linear system

$$\mathbf{U}\mathbf{M}_{\text{dr}} = \mathbf{Z}\mathbf{H}, \tag{3.21}$$

where

$$\mathbf{M}_{\text{dr}} = \mathbf{H} + \lambda\sum_{p,q\in\mathcal{E}} w_{p,q} l_{p,q}^2 (\mathbf{e}_p-\mathbf{e}_q)(\mathbf{e}_p-\mathbf{e}_q)^\top \tag{3.22}$$

and $\mathbf{H}$ is a diagonal matrix with $h_{i,i} = l_i^1$.

The dictionary $\mathbf{D}$ and codes $\mathbf{Z}$ are initialized using PCA. (The K-SVD algorithm can also be used for this purpose [78].) The variables $\mathbf{Z}$ are updated by accelerated proximal gradient descent steps [79]:

$$\begin{aligned}\bar{\mathbf{Z}} &= \mathbf{Z}^t + \omega^t(\mathbf{Z}^t - \mathbf{Z}^{t-1}) \\ \mathbf{Z}^{t+1} &= \mathbf{prox}_{\tau\gamma\|\cdot\|_1}\left(\bar{\mathbf{Z}} - \tau\left(\mathbf{D}^\top(-\mathbf{X}+\mathbf{D}\bar{\mathbf{Z}}) + (\bar{\mathbf{Z}}-\mathbf{U})\mathbf{H}\right)\right),\end{aligned} \tag{3.23}$$

where $\tau = \frac{1}{\left\|\mathbf{D}^\top\mathbf{D}\right\|_2 + \|\mathbf{H}\|_2}$ and $\omega^t = \frac{t}{t+3}$. The $\mathbf{prox}_{\varepsilon\|\cdot\|_1}$ operator performs elementwise

Algorithm 4 Joint Clustering and Dimensionality Reduction

```
1: input: Data samples $\{\mathbf{x}\}_{i=1}^{n}$, dimensionality $d$, parameters $\gamma, \xi, \eta$.
2: output: Cluster assignment $\{\hat{c}_i\}_{i=1}^{n}$ and latent factors $\mathbf{D}$.
3: Construct connectivity structure $\mathcal{E}$.
4: Initialize dictionary $\mathbf{D}$ and codes $\mathbf{Z}$.
5: Precompute $w_{p,q}, \delta_1, \delta_2$.
6: Initialize $\mathbf{u}_i = \mathbf{z}_i, l_i^1 = 1, l_{p,q}^2 = 1, \mu_1 = \xi\delta_1, \mu_2 \gg \max \|\mathbf{z}_p - \mathbf{z}_q\|_2^2, \lambda$.
7: while $|\mathbf{C}^t - \mathbf{C}^{t-1}| < \varepsilon$ or $t <$ maxiterations do
8:     Update $l_i^1$ and $l_{p,q}^2$ using (3.5).
9:     Update $\{\mathbf{z}_i\}_{i=1}^{n}$ using (3.23).
10:    Update $\{\mathbf{u}_i\}_{i=1}^{n}$ using (3.21).
11:    Every four iterations, update $\lambda$, $\mu_i = \max\left(\frac{\mu_i}{2}, \frac{\delta_i}{2}\right)$.
12:    Every ten iterations, update $\mathbf{D}$ using (3.26).
13: end while
14: Construct graph $\mathcal{G} = (\mathcal{V}, \mathcal{F})$ with $f_{p,q} = 1$ if $\|\mathbf{u}_p^* - \mathbf{u}_q^*\|_2 < \delta_2$.
15: Output clusters given by the connected components of $\mathcal{G}$.
```

soft thresholding:

$$\mathbf{prox}_{\varepsilon\|.\|_1}(v) = \mathrm{sign}(v) \max\left(0, |v| - \varepsilon\right). \tag{3.24}$$

The variables $\mathbf{D}$ are updated using

$$\bar{\mathbf{D}} = \mathbf{X}\mathbf{Z}^\top \left(\mathbf{Z}\mathbf{Z}^\top + \beta\mathbf{I}\right)^{-1} \tag{3.25}$$

$$\mathbf{D}^{t+1} = \eta\mathbf{D}^t + (1-\eta)\bar{\mathbf{D}}, \tag{3.26}$$

where β is a small regularization value set to $\beta = 10^{-4}\,\mathrm{trace}(\mathbf{Z}\mathbf{Z}^\top)$.

The RCC-DR algorithm is summarized in Algorithm 4. The RCC-DR algorithm has additional sparse coding parameters: d, γ, η, and ξ. They are set to heuristic values that are fixed across all datasets; $d = 100$, $\xi = 8$, $\gamma = 0.2$, and $\eta = 0.9$. In next section, we demonstrates that the RCC-DR algorithms is robust to different settings of these sparse coding parameters. The dictionary is initialized using PCA components. Due to the small input dimension, we set $d = 8$ for the Shuttle, Pendigits, and Mice Protein datasets. The parameters δ_2 and μ_2 in RCC-DR are computed using $\mathbf{Z}$, by analogy to their counterparts in RCC. To set δ_1, we compute the distance r_i of each data point $\mathbf{z}_i$ from the mean of data $\mathbf{Z}$ and set $\delta_1 = \text{mean}(2r_i)$. The initial value of μ_1 is set to $\mu_1 = \xi\delta_1$. The parameter λ is initialize and updated automatically using,

$$\lambda = \frac{\|\mathbf{ZH}\|_2}{\|\mathbf{A}\|_2 + \|\mathbf{H}\|_2}. \tag{3.27}$$

3.3 Experiments

3.3.1 Datasets

We have conducted experiments on datasets from multiple domains. The dimensionality of the data in the different datasets varies from 9 to just below 50,000. `Reuters-21578` is the classic benchmark for text classification, comprising 21,578 articles that appeared on the Reuters newswire in 1987. `RCV1` is a more recent benchmark of 800,000 manually categorized Reuters newswire articles [80]. (Due to limited scalability of some prior algorithms, we use 10,000 random samples from `RCV1`.) `Shuttle` is a dataset from NASA that contains 58,000 multivariate measurements produced by sensors in the radiator subsystem of the Space Shuttle; these measurements are known to arise from seven different conditions of the radiators. `Mice Protein` is a dataset that consists of the expression levels of 77 proteins measured in the cerebral cortex of 8 classes of control and trisomic mice [81]. The last two datasets were obtained from the UCI machine learning repository [82].

`MNIST` is the classic dataset of 70,000 hand-written digits [83]. `Pendigits` is another well-known dataset of hand-written digits [84]. The Extended Yale Face Database B (`YaleB`) contains images of faces of 28 human subjects [85]. The YouTube Faces

Table 3.1: Datasets used in experiments. For each dataset, the table reports the number of instances, number of dimensions, number of ground-truth clusters, and the imbalance, defined as the ratio of the largest and smallest cardinalities of ground-truth clusters.

Name	Instances	Dimensions	Classes	Imbalance
MNIST [83]	70,000	784	10	$\sim$1
Coil-100 [87]	7,200	49,152	100	1
YaleB [85]	2,414	32,256	38	1
YTF [86]	10,036	9,075	40	13
Reuters-21578	9,082	2,000	50	785
RCV1 [80]	10,000	2,000	4	6
Pendigits [84]	10,992	16	10	$\sim$1
Shuttle	58,000	9	7	4,558
Mice Protein [81]	1,077	77	8	$\sim$1

Database (YTF) contains videos of faces of different subjects [86]; we use all video frames from the first 40 subjects sorted in chronological order. Columbia University Image Library (COIL-100) is a classic collection of color images of 100 objects, each imaged from 72 viewpoints [87]. The datasets are summarized in Table 3.1.

3.3.2 Implementation

We use approximate nearest neighbor search to construct the connectivity structure [88] and a conjugate gradient solver for linear systems [89].

3.3.3 Baselines

We compare RCC and RCC-DR to thirteen baselines, which include widely known clustering algorithms as well as recent techniques that were reported to achieve state-of-the-art performance. Our baselines are k-means++ [64], Gaussian mixture models (GMM), fuzzy clustering, mean shift clustering (MS) [67], two variants of agglomerative clustering

(AC-Complete and AC-Ward), normalized cuts (N-Cuts) [59], affinity propagation (AP) [65], Zeta l-links (Zell) [90], spectral embedded clustering (SEC) [91], clustering using local discriminant models and global integration (LDMGI) [92], graph degree linkage (GDL) [93], and path integral clustering (PIC) [94].

3.3.4 Measures

The normalized mutual information (NMI) has emerged as the standard measure for evaluating clustering accuracy in the machine learning community [95]. However, NMI is known to be biased in favor of fine-grained partitions. For this reason, we use the adjusted mutual information (AMI), which removes this bias [96]. This measure is defined as follows:

$$AMI(\mathbf{c}, \hat{\mathbf{c}}) = \frac{\mathrm{MI}(\mathbf{c}, \hat{\mathbf{c}}) - E[\mathrm{MI}(\mathbf{c}, \hat{\mathbf{c}})]}{\sqrt{H(\mathbf{c})H(\hat{\mathbf{c}})} - E[\mathrm{MI}(\mathbf{c}, \hat{\mathbf{c}})]}. \tag{3.28}$$

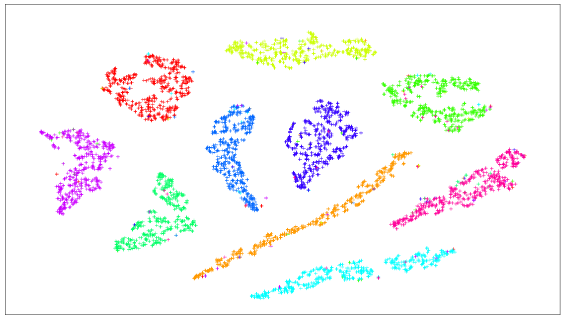
Here $H(\cdot)$ is the entropy, $\mathrm{MI}(\cdot, \cdot)$ is the mutual information, and $\mathbf{c}$ and $\hat{\mathbf{c}}$ are the two partitions being compared.

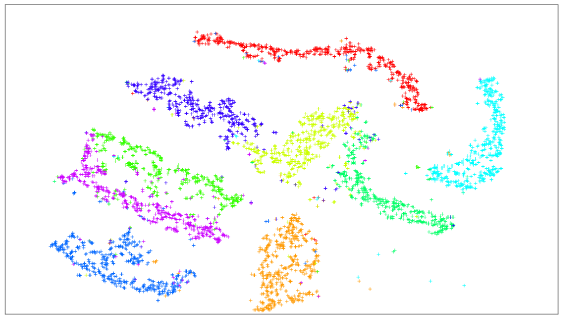
3.3.5 Results

Results on all datasets are reported in Table 4.2. In addition to accuracy on each dataset, the table also reports the average rank of each algorithm across datasets. For example, if an algorithm achieves the third highest accuracy on half of the datasets and the fourth highest on the other half, its average rank is 3.5. If an algorithm did not yield a result on a dataset due to its size, that dataset is not taken into account in computing the average rank of the algorithm.

Table 3.2: Accuracy of all algorithms on all datasets, measured by AMI. Some prior algorithms did not scale to large datasets such as MNIST (70K data points in 784 dimensions). RCC or RCC-DR achieve the highest accuracy on 7 of the 9 datasets. RCC-DR achieves the highest or second highest accuracy on 8 of the 9 datasets. The average rank of RCC-DR across datasets is lower by a multiplicative factor of 3 or more than the average rank of any prior algorithm.

Dataset	k-means++	GMM	fuzzy	MS	AC-C	AC-W	N-Cuts	AP	Zell	SEC	LDMGI	GDL	PIC	RCC	RCC-DR
MNIST	0.500	0.404	0.386	0.264	n/a	0.679	n/a	0.478	n/a	0.469	0.761	n/a	n/a	**0.893**	0.828
COIL-100	0.803	0.786	0.796	0.685	0.703	0.853	0.871	0.761	0.958	0.849	0.888	0.958	**0.965**	0.957	0.957
YTF	0.783	0.793	0.769	0.831	0.673	0.801	0.752	0.751	0.273	0.754	0.518	0.655	0.676	0.836	**0.874**
YaleB	0.615	0.591	0.066	0.091	0.445	0.767	0.928	0.700	0.905	0.849	0.945	0.924	0.941	**0.975**	**0.974**
Reuters	0.516	0.507	0.272	0.000	0.368	0.471	0.545	0.386	0.087	0.498	0.523	0.401	0.057	**0.556**	0.553
RCV1	0.355	0.344	0.205	0.000	0.108	0.364	0.140	0.313	0.023	0.069	0.382	0.020	0.015	0.138	**0.442**
Pendigits	0.679	0.695	0.695	0.694	0.525	0.728	0.813	0.639	0.317	0.741	0.775	0.330	0.467	0.848	**0.854**
Shuttle	0.215	0.266	0.204	0.362	n/a	0.291	0.000	0.322	n/a	0.305	**0.591**	n/a	n/a	0.488	0.513
Mice Protein	0.425	0.385	0.417	0.534	0.315	0.525	0.536	0.554	0.428	0.537	0.527	0.400	0.394	**0.649**	0.638
Rank	7.8	8.6	9.9	9.9	12.4	6.3	6.3	8.1	10.4	7.2	4.9	9.9	10	2.4	**1.6**


(a) RCC


(b) LDMGI

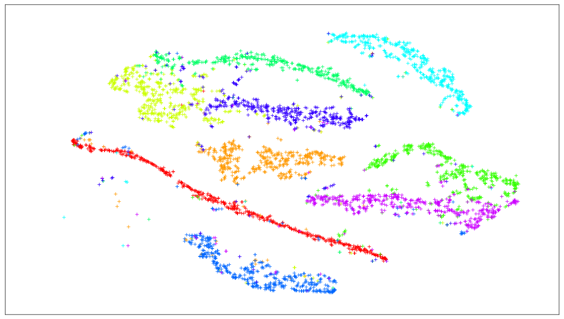

(c) N-Cuts

Table 3.3: Visualization of the representations learned by RCC and the best-performing prior algorithms, LDMGI and N-Cuts. The algorithms are run on 5,000 randomly sampled instances from the MNIST dataset. The learned representations are visualized using t-SNE.

RCC or RCC-DR achieve the highest accuracy on 7 of the 9 datasets. RCC-DR achieves the highest or second highest accuracy on 8 of the 9 datasets and RCC achieves the highest or second highest accuracy on 5 datasets. The average rank of RCC-DR and RCC is 1.6 and 2.4, respectively. The best-performing prior algorithm, LDMGI, has an average rank of 4.9, three times higher than the rank of RCC-DR. This indicates that the performance of prior algorithms is not only lower than the performance of RCC and RCC-DR, it is also inconsistent, since no prior algorithm clearly leads the others across datasets. In contrast, the low average rank of RCC and RCC-DR indicates consistently high performance across datasets.

3.3.6 Clustering gene expression data

We have conducted an additional comprehensive evaluation on a large-scale benchmark that consists of more than thirty cancer gene expression datasets, collected for the purpose of evaluating clustering algorithms [97]. The maximum number of samples across datasets is only 248 and for all but one dataset the dimension $D \gg n$. Since these datasets are statistically very different from those discussed earlier, for each algorithm we retune the major parameters. For both RCC and RCC-DR, we set $k = 9$. For RCC-DR we set $d = 12$ and $\gamma = 0.5$. The author-provided code for GDL breaks on these datasets. The results are reported in Table 3.4. RCC-DR achieves the highest accuracy on 8 of the datasets. Among the prior algorithms, affinity propagation achieves the highest accuracy on 6 of the datasets and all others on fewer. Overall, RCC-DR achieves the highest average AMI across the datasets.

3.4 Analysis

3.4.1 Running time

The execution time of RCC-DR optimization is visualized in Figure 3.2. For reference, we also show the corresponding timings for affinity propagation, a well-known modern clustering algorithm [65], and LDMGI, the baseline that demonstrated the best performance across datasets [92]. Figure 3.2 shows the running time of each algorithm on randomly sampled subsets of the 784-dimensional MNIST dataset. We sample subsets of different sizes to evaluate runtime growth as a function of dataset size. Performance is measured on a workstation with an Intel Core i7-5960x CPU clocked at 3.0 GHz. RCC-DR clusters the whole MNIST dataset within 200 seconds, whereas affinity propagation takes 37 hours and LDMGI takes 17 hours for 40,000 points.

3.4.2 Visualization

We now qualitatively analyze the output of RCC by visualization. We use the MNIST dataset for this purpose. On this dataset, RCC identifies 17 clusters. Nine of these are large clusters with more than 6,000 instances each. The remaining eight are small clusters that encapsulate outlying data points: seven of these contain between 2 and 11 instances, and one contains 148 instances. Figure 3.3(a) shows 10 randomly sampled data points $\mathbf{x}_i$ from each of the large clusters discovered by RCC. Their corresponding representatives $\mathbf{u}_i$ are shown in Figure 3.3(b). Figure 3.3(c) shows 2 randomly sampled data points from each of the small outlying clusters. Additionally, figure 3.4(a) shows 10 randomly sampled data points $\mathbf{x}_i$ from each of 10 clusters randomly sampled from the clusters discovered by RCC on the Coil-100 dataset and figure 3.4(b) shows the corresponding representatives $\mathbf{u}_i$.

Table 3.3 compares the representation $\mathbf{U}$ learned by RCC to representations learned by the best-performing prior algorithms, LDMGI and N-Cuts. We use the MNIST dataset for this purpose and visualize the output of the algorithms on a subset of 5,000 randomly sampled instances from this dataset. Both of the prior algorithms construct Euclidean representations of the data, which can be visualized by dimensionality reduction. We use t-SNE [75] to visualize the representations discovered by the algorithms. As shown in Table 3.3, the representation discovered by RCC cleanly separates the different clusters by significant margins. In contrast, the prior algorithms fail to discover the structure of the data and leave some of the clusters intermixed.

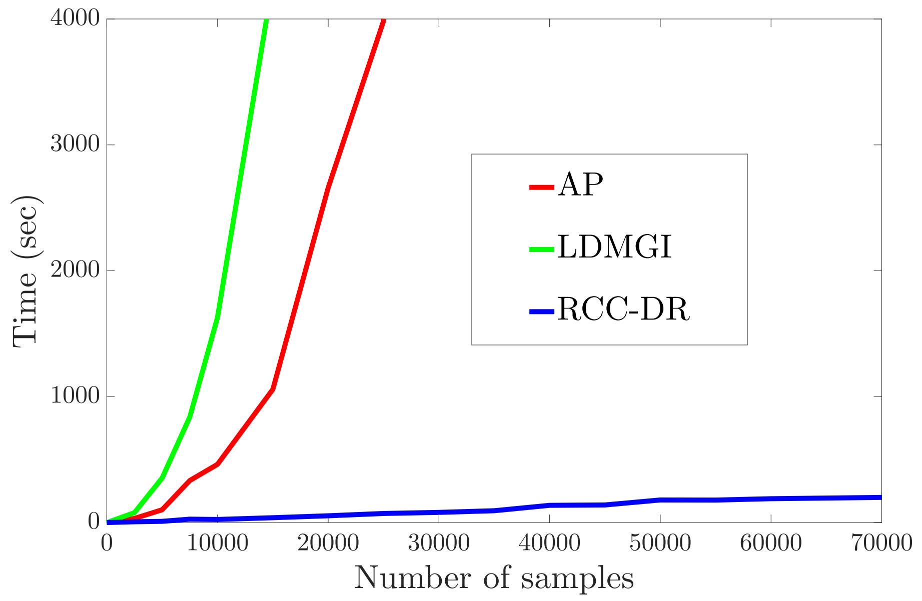


Figure 3.2: Runtime comparison of RCC-DR with AP and LDMGI. Runtime is evaluated as a function of dataset size, using randomly sampled subsets of different sizes from the MNIST dataset.

Table 3.4: AMI on cancer gene expression datasets.

Dataset	k-means++	GMM	fuzzy	MS	AC-C	AC-W	N-Cuts	AP	Zell	SEC	LDMGI	PIC	RCC	RCC-DR
alizadeh-2000-v1	0.340	0.024	0.156	0.000	0.021	0.101	0.096	0.232	0.250	0.238	0.123	0.033	0.000	**0.426**
alizadeh-2000-v2	0.568	0.922	0.570	0.631	0.543	0.922	0.922	0.563	0.922	0.922	0.738	0.922	**1.000**	**1.000**
alizadeh-2000-v3	0.586	0.604	0.591	0.530	0.417	0.616	0.601	0.540	0.702	0.574	0.582	0.625	**0.792**	**0.792**
armstrong-2002-v1	0.372	0.372	0.372	0.202	0.323	0.308	0.372	0.381	0.308	0.323	0.355	0.308	0.528	**0.546**
armstrong-2002-v2	**0.891**	0.803	0.460	0.495	0.775	0.746	0.838	0.586	0.802	**0.891**	0.509	0.802	0.642	0.838
bhattacharjee-2001	0.444	0.406	0.471	0.242	0.389	**0.601**	0.563	0.377	0.496	0.570	0.378	0.378	0.495	**0.600**
bittner-2000	−0.012	−0.002	−0.002	0.000	0.013	0.002	0.042	**0.243**	0.115	−0.002	0.014	0.115	−0.016	0.156
bredel-2005	0.297	0.208	0.297	0.000	0.324	0.384	0.203	0.139	0.278	0.259	0.295	0.278	**0.468**	0.466
chen-2002	0.570	**0.622**	0.570	0.155	0.413	0.441	−0.005	0.347	−0.005	−0.005	0.592	−0.005	0.293	0.326
chowdary-2006	0.764	0.808	0.764	0.488	0.764	**0.859**	**0.859**	0.443	**0.859**	**0.859**	**0.859**	**0.859**	0.360	0.393
dyrskjot-2003	0.507	0.532	0.503	0.063	0.332	0.474	0.303	**0.558**	0.269	0.389	0.385	0.177	0.359	0.383
garber-2001	0.242	0.137	0.156	0.000	0.314	0.210	0.204	**0.274**	0.246	0.200	0.191	0.246	0.240	0.173
golub-1999-v1	0.688	0.583	0.688	0.418	0.044	**0.831**	0.650	0.430	0.615	0.615	0.615	0.615	0.527	0.490
golub-1999-v2	0.680	0.730	0.708	0.571	0.642	**0.737**	0.693	0.516	0.689	0.703	0.600	0.689	0.656	0.597
gordon-2002	0.651	0.669	0.651	0.432	0.646	0.483	0.681	0.304	−0.005	**0.791**	0.669	0.664	0.349	0.343
laiho-2007	−0.007	**0.184**	−0.007	−0.032	−0.017	−0.007	0.030	0.061	0.073	−0.007	0.093	0.044	0.000	0.000
lapointe-2004-v1	0.088	0.141	0.117	0.101	0.039	0.151	**0.179**	0.162	0.151	0.088	0.149	0.151	0.171	0.156
lapointe-2004-v2	0.008	0.013	0.160	0.002	0.173	0.033	0.153	0.210	0.147	0.028	0.118	0.171	0.155	**0.239**
liang-2005	0.301	0.301	0.301	0.078	0.301	0.301	0.301	**0.481**	0.301	0.301	0.301	0.301	0.401	0.419
nutt-2003-v1	**0.171**	0.137	0.082	0.123	0.074	0.159	0.156	0.116	0.109	0.086	0.078	0.113	0.142	0.129
nutt-2003-v2	−0.025	−0.025	−0.025	**0.000**	−0.025	−0.024	−0.025	−0.027	−0.031	−0.025	−0.027	−0.030	−0.030	−0.029
nutt-2003-v3	0.063	**0.259**	0.063	−0.053	0.105	0.004	0.080	−0.002	0.059	0.080	0.174	0.059	0.000	0.000
pomeroy-2002-v1	−0.012	−0.022	−0.012	0.000	0.105	−0.020	−0.006	0.061	−0.020	0.008	−0.026	−0.020	0.111	**0.140**
pomeroy-2002-v2	0.502	0.544	0.580	0.434	0.601	0.591	**0.617**	0.586	0.568	0.577	0.602	0.568	0.582	0.582
ramaswamy-2001	0.618	0.650	0.636	0.009	0.511	0.623	0.651	0.592	0.618	0.620	0.663	0.639	0.635	**0.676**
risinger-2003	0.210	0.194	0.203	0.000	0.114	0.297	0.223	**0.309**	0.201	0.258	0.153	0.201	0.227	0.248
shipp-2002-v1	**0.264**	0.149	0.179	−0.005	0.050	0.208	0.132	0.113	−0.002	0.168	0.203	−0.002	0.134	0.124
singh-2002	0.048	0.029	0.048	0.071	0.069	0.019	0.033	**0.123**	−0.003	0.069	−0.003	0.066	0.034	0.034
su-2001	0.666	0.720	0.660	0.539	0.595	0.662	**0.738**	0.657	0.687	0.650	0.667	0.660	0.725	0.702
tomlins-2006-v2	0.368	0.333	0.261	0.000	0.152	0.215	0.292	0.340	0.226	**0.383**	0.354	0.311	0.348	0.373
tomlins-2006	0.396	0.366	0.568	0.000	0.279	0.454	0.409	0.374	**0.647**	0.469	0.419	0.590	0.485	0.513
west-2001	0.489	0.413	0.489	0.234	0.442	0.489	0.442	0.258	**0.515**	0.489	0.442	**0.515**	0.391	0.391
yeoh-2002-v1	0.914	0.160	0.282	0.000	0.175	0.746	**1.000**	0.336	0.916	0.951	0.857	0.916	0.937	0.430
yeoh-2002-v2	0.385	0.343	0.428	0.000	0.355	0.383	0.479	0.405	0.530	**0.550**	0.337	0.442	0.496	0.465
Mean	0.383	0.362	0.352	0.168	0.296	0.382	0.380	0.326	0.360	0.384	0.366	0.365	0.372	**0.386**

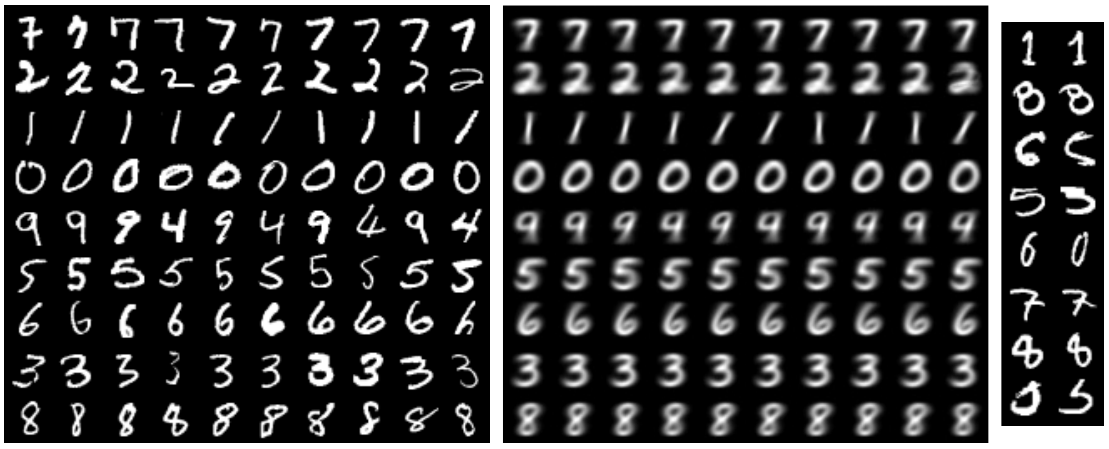

(a) Samples from large clusters discovered by RCC **(b)** Corresponding samples from the learned representation $\mathbf{U}$ **(c)** Outliers

Figure 3.3: Visualization of RCC output on the MNIST dataset. (a) 10 randomly sampled instances $\mathbf{x}_i$ from each large cluster discovered by RCC, one cluster per row. (b) Corresponding representatives $\mathbf{u}_i$ from the learned representation $\mathbf{U}$. (c) 2 random samples from each of the small outlying clusters discovered by RCC.

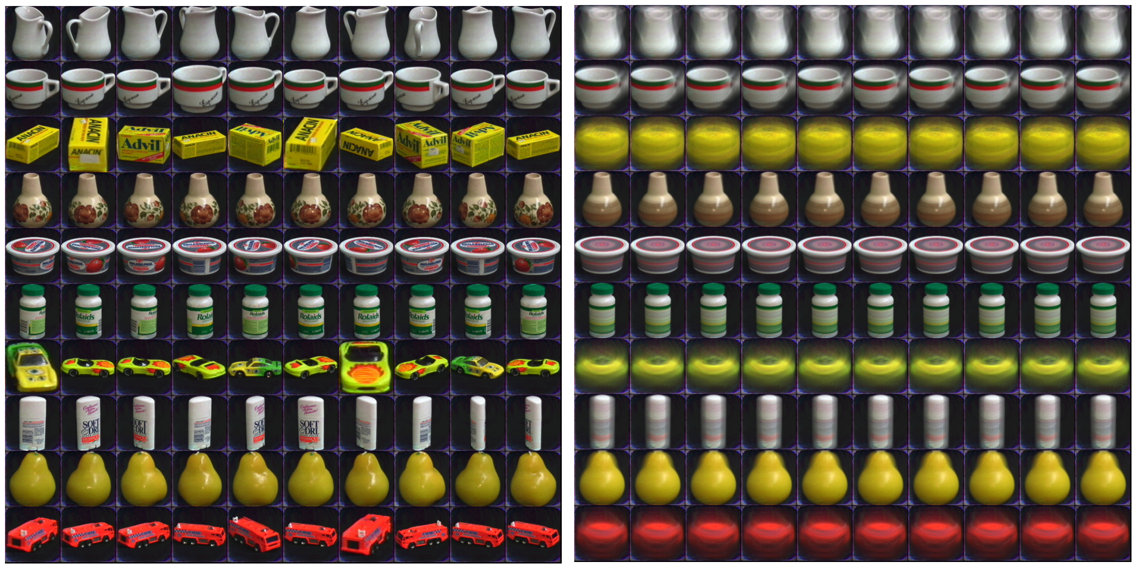

(a) Samples $\mathbf{x}_i$ from 10 random clusters produced by RCC **(b)** Corresponding representatives $\mathbf{u}_i$

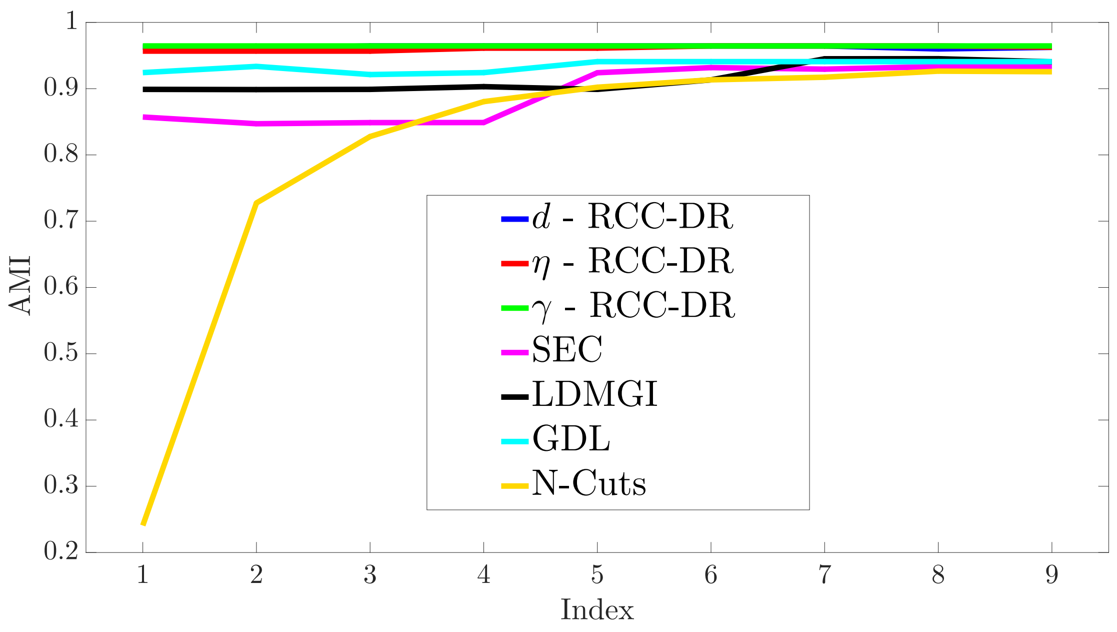
Figure 3.4: Visualization of RCC output on the Coil-100 dataset. (a) 10 randomly sampled instances $\mathbf{x}_i$ from each of 10 clusters randomly sampled from clustered discovered by RCC, one cluster per row. (b) Corresponding representatives $\mathbf{u}_i$ from the learned representation $\mathbf{U}$.

3.4.3 Robustness to hyperparameter settings

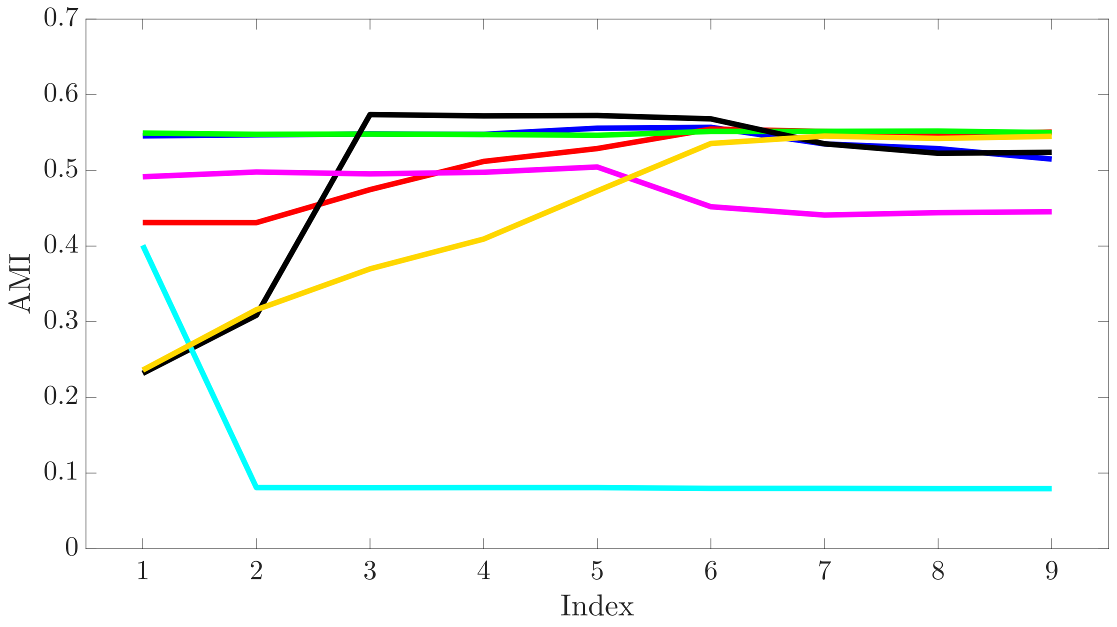
The parameters of the RCC algorithm are set automatically based on the data. The RCC-DR algorithm does have a number of parameters but is largely insensitive to their settings. In the following experiment, we vary the sparse-coding parameters d, η, and γ in the ranges $d = (40 : 20 : 200)$, $\eta = (0.55 : 0.05 : 0.95)$, and $\gamma = (0.1 : 0.1 : 0.9)$. Figures 3.5(a) and 3.5(b) compare the sensitivity of RCC-DR to these parameters with the sensitivity of the best-performing prior algorithms to their key parameters. For each baseline, we use the default search range proposed in their respective papers. The x-axis in Figure 3.5 corresponds to the parameter index. As the figure demonstrates, the accuracy of RCC-DR is robust to hyperparameter settings: the relative change of RCC-DR accuracy in AMI on YaleB is 0.005, 0.008, and 0 across the range of d, η, and γ, respectively. On the other hand, the sensitivity of the baselines is much higher: the relative change in accuracy of SEC, LDMGI, N-Cuts, and GDL is 0.091, 0.049, 0.740, and 0.021, respectively. Moreover, for SEC, LDMGI, and GDL no single parameter setting works best across different datasets.

3.4.4 Robustness to dataset imbalance

We now evaluate the robustness of different approaches to imbalance in class sizes. This experiment uses the MNIST dataset. We control the degree of imbalance by varying a parameter s between 0.1 and 1. The class "0" is sampled with probability s, the class "9" is sampled with probability 1, and the sampling probabilities of other classes vary linearly between s and 1. For each value of s, we sample 10,000 data points and evaluate the



(a) YaleB



(b) Reuters

Figure 3.5: Robustness to hyperparameter settings on the YaleB and Reuters datasets.

accuracy of RCC, RCC-DR, and the top-performing baselines on the resulting dataset. The results are reported in Figure 3.6. The RCC and RCC-DR algorithms retain their accuracy advantage on imbalanced datasets.

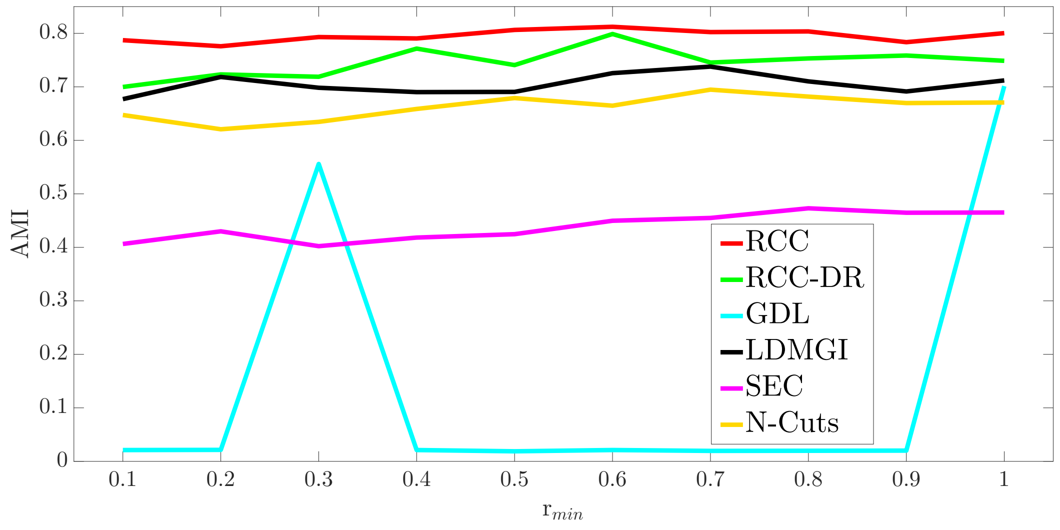


Figure 3.6: Robustness to dataset imbalance.

3.4.5 Learned representation

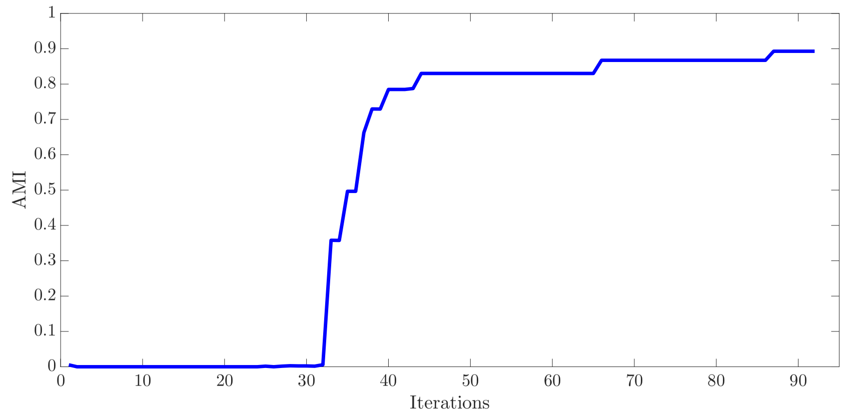
One way to quantitatively evaluate the success of the learned representation $\mathbf{U}$ in capturing the structure of the data in to use it as input to other clustering algorithms and to evaluate whether they are more successful on $\mathbf{U}$ than they are on the original data $\mathbf{X}$. The results of this experiment are reported in Table 3.5. The left part of the table reports the performance of multiple baselines when they are given, as input, the representation $\mathbf{U}$ produced by RCC. The right part of the table reports corresponding results when the baselines are given the representation $\mathbf{U}$ produced by RCC-DR.

The results indicate that the performance of prior clustering algorithms improves significantly when they are run on the representations learned by RCC and RCC-DR. The accuracy improvements for k-means++, AC-Ward, and affinity propagation are particularly notable.

Table 3.5: Success of the learned representation $\mathbf{U}$ in capturing the structure of the data, evaluated by running prior clustering algorithms on $\mathbf{U}$ instead of $\mathbf{X}$. Left: using the representation learned by RCC as input to prior clustering algorithms. Right: using the representation learned by RCC-DR. Accuracy is measured by AMI. The accuracy of prior algorithms increases substantially when a representation learned by RCC or RCC-DR is used as input instead of the original data.

Dataset	RCC						RCC-DR					
	k-means++	AC-W	AP	SEC	LDMGI	GDL	k-means++	AC-W	AP	SEC	LDMGI	GDL
MNIST	**0.879**	**0.879**	0.647	0.866	0.863	n/a	0.808	**0.809**	0.679	0.808	0.808	n/a
Coil-100	0.958	**0.963**	0.956	0.937	0.932	0.919	0.959	**0.960**	0.956	0.930	0.942	0.916
YTF	0.800	0.814	**0.840**	0.737	0.638	0.455	0.803	0.817	**0.879**	0.726	0.689	0.464
YaleB	0.960	0.964	**0.975**	0.957	0.872	0.566	0.967	0.967	**0.974**	0.958	0.872	0.541
Reuters	**0.544**	**0.544**	0.511	0.472	0.372	0.341	**0.545**	**0.545**	0.525	0.492	0.528	0.421
RCV1	0.460	0.425	0.368	**0.461**	0.301	0.018	**0.488**	0.474	0.384	0.455	0.209	0.026
Pendigits	0.750	0.717	**0.759**	0.730	0.526	0.630	0.742	0.729	**0.756**	0.706	0.742	0.676
Shuttle	0.255	0.291	0.338	**0.343**	0.132	n/a	0.275	0.340	0.344	**0.495**	0.327	n/a
Mice Protein	0.584	0.543	**0.641**	0.465	0.312	0.335	0.538	0.539	**0.630**	0.434	0.376	0.261

(a) MNIST



(b) MNIST

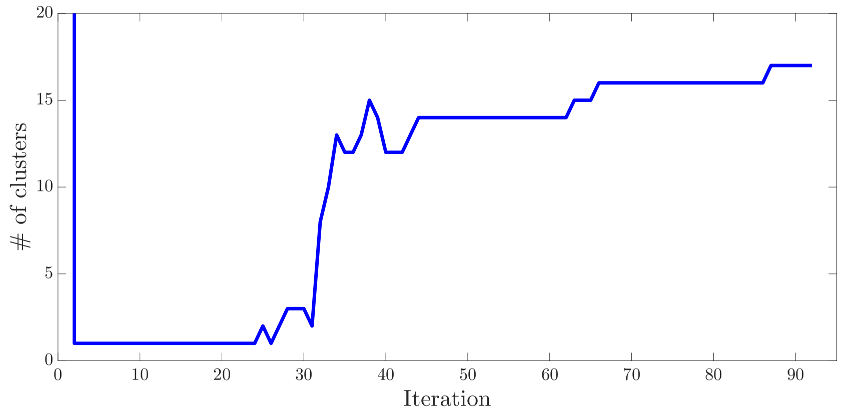


Table 3.6: Progress of the RCC algorithm on the MNIST dataset. (a) Clustering accuracy and (b) number of clusters during the optimization. While clustering accuracy increases monotonically, the number of clusters can both decrease and increase.

3.4.6 Connection with Solution Path and Mean Shift Clustering

The RCC objective (3.1) shares similarity with that of solution path [62] (SPC) and mean-shift [67] (MS) clustering formulation. In this section, we highlight subtle but significant differences which led to establishment of highly efficient - RCC and RCC-DR algorithms.

1. SPC considers a specific family of concave penalties and derives a computationally intensive majorization-minimization scheme for optimizing the objective for this special case. It transforms the original objective to an approximate dual objective using the majorization step. With this approximation, SPC can only optimize for an upper bound of the original objective. In contrast, our work provides a highly efficient general solution and it is based on an exact duality between robust estimation and line processes. As noted in section 3.2.2, solving for the dual variables and substituting them in (3.2) gives back the original objective (3.1). Hence, there is no approximation involved.
2. In SPC, a pairwise penalty is imposed between *every* pair of datapoints. This is a major drawback that prevents SPC from scaling to large datasets. The largest dataset considered in the SPC work is of 5,765 samples and dimensionality 16.
3. In order to speed up the clustering process, SPC merges centroids intermittently and then restarts the process with a new objective. This step establishes SPC to be similar to the existing agglomerative hierarchical clustering algorithms. The only difference w.r.t. hierarchical clustering is that SPC has a different objective and merging criterion. All this was based on an assumption that is at the center

of the SPC formulation: that "with iteration the number of clusters should always decrease". However, this is not necessarily true. In Table 3.6, we plot clustering accuracy and the number of clusters on the MNIST dataset over the course of the RCC algorithm. The clustering accuracy increases monotonically. But the number of clusters (according to the criterion in lines 11-12 of Algorithm 3) is not monotonic: it increases and decreases over time, indicating that clusters split and merge during the optimization. This is also consistent with theory.

4. SPC has two major parameters: λ and δ. These are analogous to λ and μ in RCC. In SPC the penalty function is a function of both λ and δ. This leads to a complex rule for λ updates and for the initialization of δ, requiring three additional user-defined parameters: ω, ϕ, and τ. On the other hand, RCC only has a single user-defined parameter, k.

5. SPC does not provide a single clustering solution. Instead, it provides a set of solutions to choose from. Like hierarchical clustering, SPC is run until all the data points are merged into a single cluster. Any of the intermediate configurations can be chosen as the output of the algorithm. In contrast, RCC optimizes an objective deterministically while yielding a stable non-trivial clustering at the end, which serves as the output.

6. MS is based on the theory of kernel density estimation [67, 98]. In MS, each individual data sample $\mathbf{x}_i$ is associated with its centroid $\mathbf{z}_i$. The MS objective is given by

$$F(\mathbf{Z}) = \sum_{i,j} H(\mathbf{z}_i - \mathbf{z}_j; h), \tag{3.29}$$

where $H(\cdot)$ is a smoothing kernel and h is a bandwidth parameter. The commonly used smoothing kernels are Epanechnikov and Gaussian functions. The $\mathbf{z}_i$'s are initialized with $\mathbf{x}_i$'s and updated every iteration using the above objective. Note that (3.29) is similar to the pairwise term in 3.1. A major difference between the two objectives is that 3.1 has an additional data term. This data term constraints the representatives to remain near their data points. This impedes outliers from merging into legitimate clusters. In comparison, the MS objective does not include this data term and is hence more susceptible to outliers.

7. MS usually employs flat or Gaussian kernels whereas RCC utilizes robust M-estimators. The use of a redescending penalty function is critical to RCC's success, since spurious edge connections are automatically pruned as part of the continuous optimization.
8. It is well known that MS does not scale to large datasets. Like SPC, MS also imposes a pairwise term on every pair of datapoints. This quadratic increase in the number of terms severely limits the scalability of MS. We note that the flat kernel, which is popularly employed in MS, imposes a cut-off distance for the consideration of pairwise terms, but identifying the active pairwise terms is still a procedure with quadratic complexity in the limit, and the number of active terms can be quadratic.
9. In MS, during every iteration each representative $\mathbf{z}_i$ is updated using a different set of neighbors. This is due to the fact that the set of active pairwise terms can change from iteration to iteration. In contrast, in RCC the underlying sparse graph is fixed.
10. In MS, the bandwidth parameter h is fixed throughout the algorithm. In RCC, with the application of graduated non-convexity, the shape of the penalty changes during

the optimization.

3.5 Conclusion

We have presented a clustering algorithm that optimizes a continuous objective based on robust estimation. The objective is optimized using linear least-squares solvers, which scale to large high-dimensional datasets. The robust terms in the objective enable separation of entangled clusters, yielding high accuracy across datasets and domains.

The continuous form of the clustering objective allows it to be integrated into end-to-end feature learning pipelines. We have demonstrated this by extending the algorithm to perform joint clustering and dimensionality reduction. In the next chapter, we further leverage this property of the presented formulation by optimizing compositional nonlinear mappings of the data into Euclidean spaces in which clustering is performed. This supports the use of clustering as an objective for unsupervised end-to-end feature learning.

Chapter 4: Deep Continuous Clustering

4.1 Introduction

Despite decades of progress of clustering algorithms, reliable clustering of noisy high-dimensional datasets remains an open problem. High dimensionality poses a particular challenge because assumptions made by many algorithms break down in high-dimensional spaces [99–101]. One such example is that the interpoint distances can become less informative in high-dimensional spaces.

There are techniques that reduce the dimensionality of data by embedding it in a lower-dimensional space [102]. Such general techniques, based on preserving variance or dissimilarity, may not be optimal when the goal is to discover cluster structure. Dedicated algorithms such as RCC-DR and [103, 104] have been developed that combine dimensionality reduction and clustering by fitting low-dimensional subspaces. Such algorithms can achieve better results than pipelines that first apply generic dimensionality reduction and then cluster in the reduced space. However, frameworks such as subspace clustering and projected clustering operate on *linear* subspaces and are therefore limited in their ability to handle datasets that lie on nonlinear manifolds.

Recent approaches have sought to overcome this limitation by constructing a nonlinear embedding of the data into a low-dimensional space in which it is clustered [105–108].

Ultimately, the goal is to perform nonlinear embedding and clustering jointly, such that the embedding is optimized to bring out the latent cluster structure. These works have achieved impressive results. Nevertheless, they are based on classic center-based, divergence-based, or hierarchical clustering formulations and thus inherit some limitations from these classic methods. In particular, these algorithms require setting the number of clusters a priori. And the optimization procedures they employ involve discrete reconfigurations of the objective, such as discrete reassignments of datapoints to centroids or merging of putative clusters in an agglomerative procedure. Thus it is challenging to integrate them with an optimization procedure that modifies the embedding of the data itself.

We seek a procedure for joint nonlinear embedding and clustering that overcomes some of the limitations of prior formulations. There are a number of characteristics we consider desirable. First, we wish to express the joint problem as optimization of a single continuous objective. Second, this optimization should be amenable to scalable gradient-based solvers such as modern variants of SGD. That is, we wish to largely abstract the objective from the optimization algorithm and its implementation. Third, the formulation should not require setting the number of clusters a priori, since this number is often not known in advance.

While any one of these desiderata can be fulfilled by some existing approaches, the combination is challenging. For example, it has long been known that the k-means objective can be optimized by SGD [109]. But this family of formulations requires positing the number of clusters k in advance. Furthermore, the optimization is punctuated by discrete reassignments of datapoints to centroids, and is thus hard to integrate with continuous embedding of the data.

In this chapter, we present a formulation for joint nonlinear embedding and clustering that possesses all of the aforementioned desirable characteristics. Our approach is rooted in RCC. The basic RCC formulation (3.1) has the characteristics we seek, such as a clear continuous objective and no prior knowledge of the number of clusters. However, integrating it with deep nonlinear embedding is still a challenge. For instance, a RCC-DR formulation for joint *linear* embedding and clustering relies on a complex alternating optimization scheme with linear least-squares subproblems, and does not apply to nonlinear embeddings.

We present an integration of the RCC objective with dimensionality reduction that is simpler and more direct than RCC-DR, while naturally handling deep *nonlinear* embeddings. New formulation avoids alternating optimization and the introduction of auxiliary dual variables. A deep nonlinear embedding of the data into a low-dimensional space is optimized while the data is clustered in the reduced space. The optimization is expressed by a global continuous objective and conducted by standard gradient-based solvers. The code is available at `https://github.com/shahsohil/DCC`.

4.2 Model and Algorithm

4.2.1 Formulation

Let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ be a set of points in $\mathbb{R}^D$ that must be clustered.Generic clustering algorithms that operate directly on $\mathbf{X}$ rely strongly on interpoint distances. When D is high, these distances become less informative [99, 100]. Hence most clustering algorithms do not operate effectively in high-dimensional spaces. To overcome this problem, we embed the data into a lower-dimensional space $\mathbb{R}^d$. The embedding of the dataset into $\mathbb{R}^d$ is denoted by $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$. The function that performs the embedding is denoted by $f_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^d$. Thus $\mathbf{z}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i)$ for all i.

Our goal is to cluster the embedded dataset $\mathbf{Z}$ and to optimize the parameters $\boldsymbol{\theta}$ of the embedding as part of the clustering process. This formulation presents an obvious difficulty: if the embedding $f_{\boldsymbol{\theta}}$ can be manipulated to assist the clustering of the embedded dataset $\mathbf{Z}$, there is nothing that prevents $f_{\boldsymbol{\theta}}$ from distorting the dataset such that $\mathbf{Z}$ no longer respects the structure of the original data. We must therefore introduce a regularizer on $\boldsymbol{\theta}$ that constrains the low-dimensional image $\mathbf{Z}$ with respect to the original high-dimensional dataset $\mathbf{X}$. To this end, we also consider a reverse mapping $g_{\boldsymbol{\omega}} : \mathbb{R}^d \to \mathbb{R}^D$. To constrain $f_{\boldsymbol{\theta}}$ to construct a faithful embedding of the original data, we require that the original data be reproducible from its low-dimensional image [110]:

$$\underset{\Omega}{\text{minimize}} \quad \|\mathbf{X} - G_{\boldsymbol{\omega}}(\mathbf{Z})\|_F^2, \tag{4.1}$$

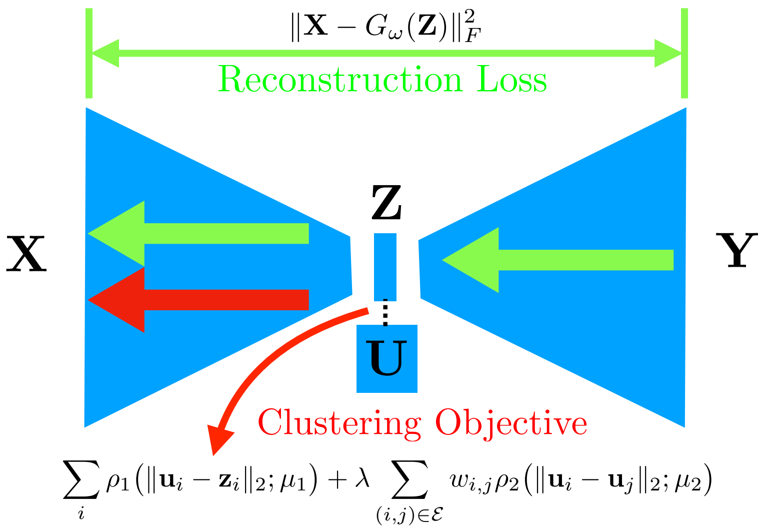
where $\mathbf{Z} = F_{\boldsymbol{\theta}}(\mathbf{X}), \quad \Omega = \{\boldsymbol{\theta}, \boldsymbol{\omega}\}$. Here $F_{\boldsymbol{\theta}}(\mathbf{X}) = [f_{\boldsymbol{\theta}}(\mathbf{x}_1), \dots, f_{\boldsymbol{\theta}}(\mathbf{x}_N)]$, $G_{\boldsymbol{\omega}}(\mathbf{Z}) = [g_{\boldsymbol{\omega}}(\mathbf{z}_1), \dots, g_{\boldsymbol{\omega}}(\mathbf{z}_N)]$, and $\|\cdot\|_F$ denotes the Frobenius norm.

To perform nonlinear embedding and clustering jointly, we wish to integrate the reconstruction objective (4.1) and the RCC objective (3.1). The Deep Continuous Clustering (DCC) algorithm optimizes the following objective:

$$\mathcal{L}(\Omega, \mathbf{U}) = \frac{1}{D} \underbrace{\|\mathbf{X} - G_{\boldsymbol{\omega}}(\mathbf{Z})\|_F^2}_{\text{reconstruction loss}} + \frac{1}{d} \left(\underbrace{\sum_i \rho_1\big(\|\mathbf{u}_i - \mathbf{z}_i\|_2; \mu_1\big)}_{\text{data loss}} + \underbrace{\lambda \sum_{(i,j)\in\mathcal{E}} w_{i,j} \rho_2\big(\|\mathbf{u}_i - \mathbf{u}_j\|_2; \mu_2\big)}_{\text{pairwise loss}} \right)$$

$$\text{where } \mathbf{Z} = F_{\boldsymbol{\theta}}(\mathbf{X}). \tag{4.2}$$

Figure 4.1 summarizes the processing within a deep network.



$\|\mathbf{X} - G_\omega(\mathbf{Z})\|_F^2$
Reconstruction Loss
X
Z
Y
U
Clustering Objective
$\sum_i \rho_1(\|\mathbf{u}_i - \mathbf{z}_i\|_2; \mu_1) + \lambda \sum_{(i,j)\in\mathcal{E}} w_{i,j}\rho_2(\|\mathbf{u}_i - \mathbf{u}_j\|_2; \mu_2)$

Figure 4.1: Visualization of forward pass step with their corresponding loss application and back-propagation step.

This formulation bears some similarity to RCC-DR, but differs in three major respects. First, RCC-DR only operates on a linear embedding defined by a sparse dictionary, while DCC optimizes a more expressive nonlinear embedding parameterized by Ω. Second, RCC-DR alternates between optimizing dictionary atoms, sparse codes, representatives $\mathbf{U}$, and dual line process variables; in contrast, DCC avoids duality altogether and optimizes the global objective directly. Third, DCC does not rely on closed-form or linear least-squares solutions to subproblems; rather, the joint objective is optimized by modern gradient-based solvers, which are commonly used for deep representation learning and are highly scalable.

We now discuss objective (4.2) and its optimization in more detail. The mappings $F_{\boldsymbol{\theta}}$ and $G_{\boldsymbol{\omega}}$ are performed by an autoencoder with fully-connected or convolutional layers and rectified linear units after each affine projection [110, 111]. The parameters $w_{i,j}$, μ_1 and μ_2 are set following their initialization for RCC and RCC-DR. The graph $\mathcal{E}$ is constructed on $\mathbf{X}$ using the mutual kNN criterion [66], augmented by the minimum spanning tree of the kNN graph to ensure connectivity to all datapoints. The role of M-estimators ρ_1 and ρ_2 is to pull the representatives of a true underlying cluster into a single point, while disregarding spurious connections across clusters. For both robust estimators, we use scaled Geman-McClure functions [70]. The parameter λ balances the relative strength of the data loss and the pairwise loss. To balance the different terms, we set $\lambda = \frac{\|\mathbf{Z}\|_2}{1+\|\mathbf{A}\|_2}$, where $\mathbf{A} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top$ and $\|\cdot\|_2$ denotes the spectral norm. This ratio approximately ensures similar maximum curvature for different terms. Since the setting for λ is independent of the reconstruction loss term, the ratio is similar to that considered for RCC-DR. However, in contrast to RCC-DR, the parameter λ need not be updated during the optimization.

4.2.2 Optimization

Objective (4.2) can be optimized using scalable modern forms of stochastic gradient descent (SGD). Note that each $\mathbf{u}_i$ is updated only via its corresponding loss and pairwise terms. On the other hand, the autoencoder parameters Ω are updated via all data samples. Thus in a single epoch, there is bound to be a difference between the update rates for $\mathbf{U}$ and Ω. To deal with this imbalance, an adaptive solver such as Adam should be used [112].

Another difficulty is that the graph $\mathcal{E}$ connects all datapoints such that a randomly sampled minibatch is likely to be connected by pairwise terms to datapoints outside the minibatch. In other words, the objective (4.2), and more specifically the pairwise loss, does not trivially decompose over datapoints. This requires some care in the construction of minibatches. Instead of sampling datapoints, we sample subsets of edges from $\mathcal{E}$. The corresponding minibatch $\mathcal{B}$ is defined by all nodes incident to the sampled edges. However, if we simply restrict the objective (4.2) to the minibatch and take a gradient step, the reconstruction and data terms will be given additional weight since the same datapoint can participate in different minibatches, once for each incident edge. To maintain balance between the terms, we must weigh the contribution of each datapoint in the minibatch. The rebalanced minibatch loss is given by

$$\mathcal{L}_{\mathcal{B}}(\Omega, \mathbf{U}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} w_i\left(\frac{\|\mathbf{x}_i - g_{\boldsymbol{\omega}}(\mathbf{z}_i)\|_2^2}{D} + \frac{\rho_1\big(\|\mathbf{u}_i - \mathbf{z}_i\|_2\big)}{d}\right) + \frac{\lambda}{d|\mathcal{B}|}\sum_{(i,j)\in\mathcal{E}_{\mathcal{B}}} w_{i,j}\rho_2\big(\|\mathbf{u}_i - \mathbf{u}_j\|_2\big)$$

$$\text{where } \mathbf{z}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i) \quad \forall i \in \mathcal{B}. \tag{4.3}$$

Here $w_i = \frac{n_i^{\mathcal{B}}}{n_i}$, where $n_i^{\mathcal{B}}$ is the number of edges connected to the i^{th} node in the subgraph $\mathcal{E}_{\mathcal{B}}$.

The gradients of $\mathcal{L}_{\mathcal{B}}$ with respect to the low-dimensional embedding $\mathbf{Z}$ and the representatives $\mathbf{U}$ are given by

$$\frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial \mathbf{z}_i} = \frac{1}{|\mathcal{B}|}\left(\frac{w_i\mu_1^2(\mathbf{z}_i - \mathbf{u}_i)}{d(\mu_1 + \|\mathbf{u}_i - \mathbf{z}_i\|_2^2)^2} + \frac{2w_i(g_{\boldsymbol{\omega}}(\mathbf{z}_i) - \mathbf{x}_i)}{D}\frac{\partial g_{\boldsymbol{\omega}}(\mathbf{z}_i)}{\partial \mathbf{z}_i}\right) \tag{4.4}$$

$$\frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial \mathbf{u}_i} = \frac{1}{d|\mathcal{B}|}\left(\frac{w_i\mu_1^2(\mathbf{u}_i - \mathbf{z}_i)}{(\mu_1 + \|\mathbf{u}_i - \mathbf{z}_i\|_2^2)^2} + \lambda\mu_2^2\sum_{(i,j)\in\mathcal{E}_{\mathcal{B}}}\frac{w_{i,j}(\mathbf{u}_i - \mathbf{u}_j)}{(\mu_2 + \|\mathbf{u}_i - \mathbf{u}_j\|_2^2)^2}\right) \tag{4.5}$$

These gradients are propagated to the parameters Ω.

4.2.3 Initialization, Continuation, and Termination

Initialization. The embedding parameters Ω are initialized using the stacked denoising autoencoder (SDAE) framework [113]. Each pair of corresponding encoding and decoding layers is pretrained in turn. Noise is introduced during pretraining by adding dropout to the input of each affine projection [114]. Encoder-decoder layer pairs are pretrained sequentially, from the outer to the inner. After all layer pairs are pretrained, the entire SDAE is fine-tuned end-to-end using the reconstruction loss. This completes the initialization of the embedding parameters Ω. These parameters are used to initialize the representatives $\mathbf{U}$, which are set to $\mathbf{U} = \mathbf{Z} = F_{\boldsymbol{\theta}}(\mathbf{X})$.

Continuation. The price of robustness is the nonconvexity of the estimators ρ_1 and ρ_2. One way to alleviate the dangers of nonconvexity is to use a continuation scheme that gradually sharpens the estimator [71, 72]. Following RCC, we initially set μ_i to a high value that makes the estimator ρ_i effectively convex in the relevant range. The value of μ_i is decreased on a regular schedule until a threshold $\frac{\delta_i}{2}$ is reached. We follow the similar setting for δ_1 and δ_2.

Stopping criterion. Once the continuation scheme is completed, DCC monitors the computed clustering. At the end of every epoch, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{F})$ is constructed such that $f_{i,j} = 1$ if $\|\mathbf{u}_i - \mathbf{u}_j\| < \delta_2$. The cluster assignment is given by the connected components of $\mathcal{G}$. DCC compares this cluster assignment to the one produced at the end of the preceding epoch. If more than 0.1% of the edges in $\mathcal{E}$ changed from intercluster to intracluster or vice versa, DCC outputs the computed clustering and terminates.

Complete algorithm. The complete algorithm is summarized in Algorithm 5.

Algorithm 5 Deep Continuous Clustering

```
1:  input: Data samples $\{\mathbf{x}_i\}_i$.
2:  output: Cluster assignment $\{c_i\}_i$.
3:  Construct a graph $\mathcal{E}$ on $\mathbf{X}$.
4:  Initialize $\Omega$ and $\mathbf{U}$.
5:  Precompute $\lambda, w_{i,j}, \delta_1, \delta_2$. Initialize $\mu_1, \mu_2$.
6:  while stopping criterion not met do
7:      Every iteration, construct a minibatch $\mathcal{B}$ defined by a sample of edges $\mathcal{E}_{\mathcal{B}}$.
8:      Update $\{\mathbf{u}_i\}_{i\in\mathcal{B}}$ and $\Omega$.
9:      Every $M$ epochs, update $\mu_i = \max\left(\frac{\mu_i}{2}, \frac{\delta_i}{2}\right)$.
10: end while
11: Construct graph $\mathcal{G} = (\mathcal{V}, \mathcal{F})$ with $f_{i,j} = 1$ if $\|\mathbf{u}_i^* - \mathbf{u}_j^*\|_2 < \delta_2$.
12: Output clusters given by the connected components of $\mathcal{G}$.
```

4.3 Experiments

4.3.1 Datasets

We conduct experiments on six high-dimensional datasets, which cover domains such as handwritten digits, objects, faces, and text. We used the same datasets as for RCC but only the ones that had dimensionality above 100. All features are normalized to the range $[0, 1]$. Note that DCC is an unsupervised learning algorithm. Unlabelled data is embedded and clustered with no supervision. There is thus no train/test split.

4.3.2 Baselines

The presented DCC algorithm is compared to 13 baselines, which include both classic and deep clustering algorithms. The baselines include k-means++ [64], DB-SCAN [115], two variants of agglomerative clustering: Ward (AC-W) and graph degree linkage (GDL) [93], two variants of spectral clustering: spectral embedded clustering (SEC) [116] and local discriminant models and global integration (LDMGI) [92], and two variants of robust continuous clustering: RCC and RCC-DR. We also include an SGD-based implementation of RCC-DR, referred to as RCC-DR (SGD): this baseline uses the same optimization method as DCC, and thus more crisply isolates the improvement in DCC that is due to the nonlinear dimensionality reduction (rather than a different solver).

The deep clustering baselines include four recent approaches that share our basic motivation and use deep networks for clustering: deep embedded clustering (DEC) [106], joint unsupervised learning (JULE) [107], the deep clustering network (DCN) [108], and deep embedded regularized clustering (DEPICT) [105]. These are strong baselines that use deep autoencoders, the same network structure as our approach (DCC). The key difference is in the loss function and the consequent optimization procedure. The prior formulations are built on KL-divergence clustering, agglomerative clustering, and k-means, which involve discrete reconfiguration of the objective during the optimization and rely on knowledge of the number of ground-truth clusters either in the design of network architecture, during the embedding optimization, or in post-processing. In contrast, DCC optimizes a robust continuous loss and does not rely on prior knowledge of the number of clusters.

4.3.3 Implementation

We report experimental results for two different autoencoder architectures: one with only fully-connected layers and one with convolutional layers. This is motivated by prior deep clustering algorithms, some of which used fully-connected architectures and some convolutional.

For fully-connected autoencoders, we use the same autoencoder architecture as DEC [106]. Specifically, for all experiments on all datasets, we use an autoencoder with the following dimensions: D–500–500–2000–d–2000–500–500–D. This autoencoder architecture follows parametric t-SNE [117].

For convolutional autoencoders, the network architecture is modeled on JULE [107]. Table 4.1 summarizes the architecture of the convolutional encoder used for the convolutional configuration of DCC. As in [107], the number of layers depends on image resolution in the dataset and it is set such that the output resolution of the encoder is about 4×4. Convolutional kernels are applied with a stride of two. The input to convolutional layers with 4×4 and 5×5 kernels is zero-padded by one and two pixels, respectively. The encoder is followed by a fully-connected layer with output dimension d and a convolutional decoder with kernel size that matches the output dimension of `conv5`. The decoder architecture mirrors the encoder and the output from each layer is appropriately zero-padded to match the input size of the corresponding encoding layer. All convolutional and transposed convolutional layers are followed by batch normalization and rectified linear units [111, 118].

	MNIST	Coil100	YTF	YaleB
conv1	4×4	4×4	4×4	4×4
conv2	5×5	5×5	5×5	5×5
conv3	5×5	5×5	5×5	5×5
conv4	–	5×5	5×5	5×5
conv5	–	5×5	–	5×5
output	4×4	4×4	4×4	6×6

Table 4.1: Convolutional encoder architecture.

DCC uses three hyperparameters: the embedding dimensionality d, the nearest neighbor parameter k for m-kNN graph construction, and the update period M for graduated nonconvexity. In both architectures and for all datasets, the dimensionality of the reduced space is set to $d = 10$ based on the grid search on MNIST. (It is only varied for controlled experiments that analyze stability with respect to d.) No dataset-specific hyperparameter tuning is done. For fair comparison to RCC and RCC-DR, we fix $k = 10$ and the cosine distance metric is used. The hyperparameter M is architecture specific. We set M to 10 and 20 for convolutional and fully-connected autoencoders respectively and it is varied for varying dimensionality d during the controlled experiment.

For autoencoder initialization, a minibatch size of 256 and dropout probability of 0.2 are used. SDAE pretraining and finetuning start with a learning rate of 0.1, which is decreased by a factor of 10 every 80 epochs. Each layer is pretrained for 200 epochs. Finetuning of the whole SDAE is performed for 400 epochs. For the fully-connected SDAE, the learning rates are scaled in accordance with the dimensionality of the dataset. During the optimization using the DCC objective, the Adam solver is used with its default learning rate of 0.001 and momentum 0.99. Minibatches are constructed by sampling 128 edges. DCC was implemented using the PyTorch library.

For the baselines, we use publicly available implementations. For k-means++, DBSCAN and AC-W, we use the implementations in the SciPy library and report the best results across ten random restarts. For a number of baselines, we performed hyperparameter search to maximize their reported performance. For DBSCAN, we searched over values of Eps, for LDMGI we searched over values of the regularization constant λ, for SEC we searched over values of the parameter μ, and for GDL we tuned the graph construction parameter a. For SGD implementation of RCC-DR the learning rate of 0.01 and momentum of 0.95 were used.

The DCN approach uses a different network architecture for each dataset. Wherever possible, we report results using their dataset-specific architecture. For YTF, Coil100, and YaleB, we use their reference architecture for MNIST. For DEC algorithm, we fix the the tuning parameter λ to be 40. This is supposedly to work best across all datasets, as reported in their paper [106].

4.3.4 Measures

In addition to AMI measure used for evaluating in the previous chapter, we also report ACC measure. Both the measures lies in a range $[0, 1]$. Higher is better. Note that the ACC measure is biased on imbalanced datasets [96]. It favors large classes.

4.3.5 Results

The results are summarized in Table 4.2. Among deep clustering methods that use fully-connected networks, DCN and DEC are not as accurate as fully-connected DCC and are also less consistent: the performance of DEC drops on the high-dimensional image datasets, while DCN is far behind on MNIST and YaleB. Among deep clustering methods that use convolutional networks, the performance of DEPICT drops on COIL100 and YTF, while JULE is far behind on YTF. The GDL algorithm failed to scale to the full MNIST dataset and the corresponding measurement is marked as ‘n/a’. The performance of RCC-DR (SGD) is also inconsistent. Although it performs on par with RCC-DR on image datasets, its performance degrades on text datasets. Note that the reported accuracy for all deep clustering baselines is based on the outcome of an independent run.

4.4 Analysis

4.4.1 Importance of joint optimization

We now analyze the importance of performing dimensionality reduction and clustering jointly, versus performing dimensionality reduction and then clustering the embedded data. To this end, we use the same SDAE architecture and training procedure as fully-connected DCC. We optimize the autoencoder but do not optimize the full DCC objective. This yields a standard nonlinear embedding, using the same autoencoder that is used by DCC, into a space with the same reduced dimensionality d. In this space, we apply a number of clustering algorithms: k-means++, AC-W, DBSCAN, SEC, LDMGI, GDL, and RCC. The results are shown in Table 4.3 (top).

These results should be compared to results reported in Table 4.2. The comparison shows that the accuracy of the baseline algorithms benefits from dimensionality reduction. However, in all cases their accuracy is still lower than that attained by DCC using joint optimization. Furthermore, although RCC and DCC share the same underlying nearest-neighbor graph construction and a similar clustering loss, the performance of DCC far surpasses that achieved by stagewise SDAE embedding followed by RCC. Note also that the relative performance of most baselines drops on Coil100 and YaleB. We hypothesize that the fully-connected SDAE is limited in its ability to discover a good low-dimensional embedding for very high-dimensional image datasets (tens of thousands of dimensions for Coil100 and YaleB).

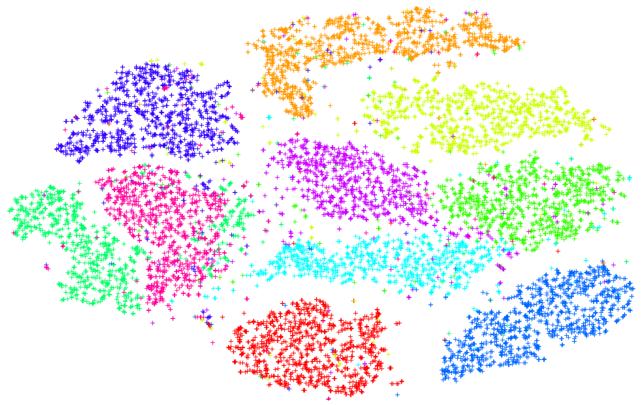
Algorithm	MNIST	Coil100	YTF	YaleB	Reuters	RCV1	MNIST	Coil100	YTF	YaleB	Reuters	RCV1
k-means++	0.500	0.803	0.783	0.615	0.516	0.355	0.532	0.621	0.624	0.514	0.236	0.529
AC-W	0.679	0.853	0.801	0.767	0.471	0.364	0.571	0.697	0.647	0.614	0.261	0.554
DBSCAN	0.000	0.399	0.739	0.456	0.011	0.014	0.000	0.921	0.675	0.632	0.700	0.571
SEC	0.469	0.849	0.745	0.849	0.498	0.069	0.545	0.648	0.562	0.721	0.434	0.425
LDMGI	0.761	0.888	0.518	0.945	0.523	0.382	0.723	0.763	0.332	0.901	0.465	0.667
GDL	n/a	0.958	0.655	0.924	0.401	0.020	n/a	0.825	0.497	0.783	0.463	0.444
RCC	0.893	0.957	0.836	0.975	0.556	0.138	0.876	0.831	0.484	0.939	0.381	0.356
RCC-DR	0.828	0.957	0.874	0.974	0.553	0.442	0.698	0.825	0.579	0.945	0.437	0.676
RCC-DR (SGD)	0.827	0.961	0.830	**0.985**	0.454	0.106	0.696	0.855	0.473	**0.970**	0.372	0.354
Fully-connected												
DCN	0.570	0.810	0.790	0.590	0.430	0.470	0.560	0.620	0.620	0.430	0.220	**0.730**
DEC	0.840	0.611	0.807	0.000	0.397	**0.500**	0.867	0.815	0.643	0.027	0.168	0.683
DCC	**0.912**	0.952	0.877	0.955	**0.572**	**0.495**	**0.962**	0.842	0.605	0.861	**0.596**	0.563
Convolutional												
JULE	0.900	**0.979**	0.574	**0.990***	–	–	0.800	**0.911**	0.342	**0.970***	–	–
DEPICT	**0.919**	0.667	0.785	**0.989***	–	–	**0.968**	0.420	0.586	**0.965***	–	–
DCC	**0.913**	0.962	**0.903**	**0.985***	–	–	**0.963**	0.858	**0.699**	**0.964***	–	–

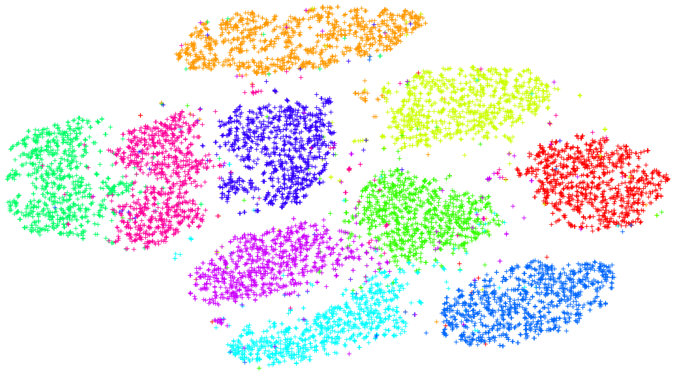
Table 4.2: Clustering accuracy of DCC and 12 baselines, measured by AMI (left) and ACC (right). Higher is better. Methods that do not use deep networks are listed first, followed by deep clustering algorithms that use fully-connected autoencoders (including the fully-connected configuration of DCC) and deep clustering algorithms that use convolutional autoencoders (including the convolutional configuration of DCC). Results that are within 1% of the highest accuracy achieved by any method are highlighted in bold. * indicates that these results were directly obtained on pixel features as against the default DoG features used for YaleB. DCC performs on par or better than prior deep clustering formulations, without relying on a priori knowledge of the number of ground-truth clusters.

Next, we show the performance of the same clustering algorithms when they are applied in the reduced space produced by DCC. These results are reported in Table 4.3 (bottom). In comparison to Table 4.3 (top), the performance of all algorithms improves significantly and some results are now on par or better than the results of DCC as reported in Table 4.2. The improvement for k-means++, Ward, and DBSCAN is particularly striking. This indicates that the performance of many clustering algorithms can be improved by first optimizing a low-dimensional embedding using DCC and then clustering in the learned embedding space.

Dataset	k-means++	AC-W	DBSCAN	SEC	LDMGI	GDL	RCC	DCC
			Clustering in a reduced space learned by SDAE					
MNIST	0.669	0.784	0.115	n/a	0.828	n/a	0.881	0.912
Coil100	0.333	0.336	0.170	0.384	0.318	0.335	0.589	0.952
YTF	0.764	0.831	0.595	0.527	0.612	0.699	0.827	0.877
YaleB	0.673	0.688	0.503	0.493	0.676	0.742	0.812	0.955
Reuters	0.501	0.494	0.042	0.435	0.517	0.488	0.542	0.572
RCV1	0.454	0.430	0.075	0.442	0.060	0.055	0.410	0.495
			Clustering in a reduced space learned by DCC					
MNIST	0.880	0.883	0.890	n/a	0.868	n/a	0.912	0.912
Coil100	0.947	0.947	0.569	0.604	0.919	0.915	0.891	0.952
YTF	0.845	0.841	0.896	0.586	0.762	0.658	0.879	0.877
YaleB	0.811	0.809	0.809	0.584	0.815	0.660	0.814	0.955
Reuters	0.553	0.554	0.560	0.479	0.586	0.401	0.581	0.572
RCV1	0.536	0.472	0.496	0.452	0.178	0.326	0.474	0.495

Table 4.3: Importance of joint optimization. This table shows the accuracy (AMI) achieved by running prior clustering algorithms on a low-dimensional embedding of the data. For reference, DCC results from Table 4.2 are also listed. **Top:** The embedding is performed using the same autoencoder architecture as used by fully-connected DCC, into the same target space. However, dimensionality reduction and clustering are performed separately. Clustering accuracy is much lower than the accuracy achieved by DCC. **Bottom:** Here clustering is performed in the reduced space discovered by DCC. The performance of all clustering algorithms improves significantly.


(a) Raw


(b) SDAE

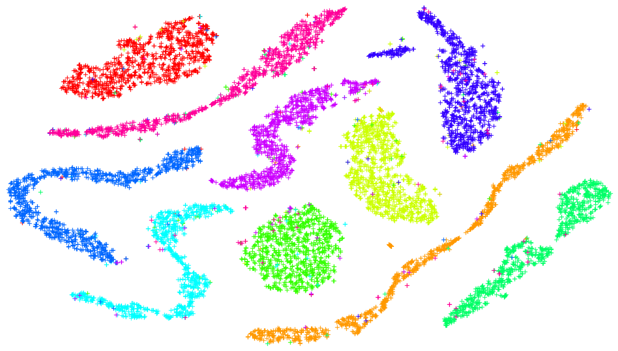
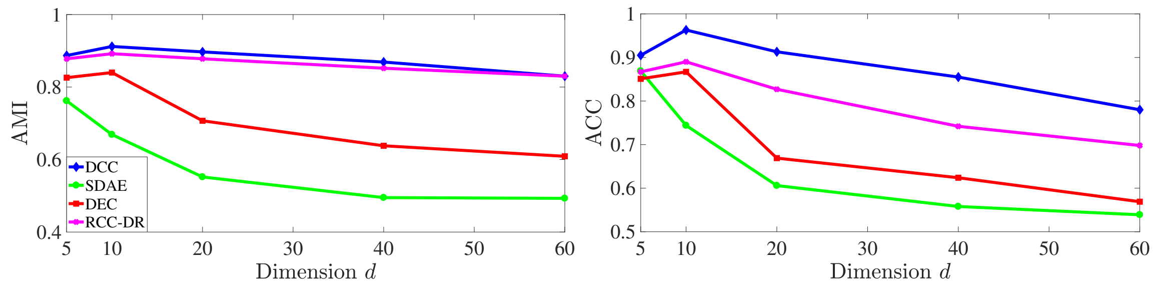

(c) DCC

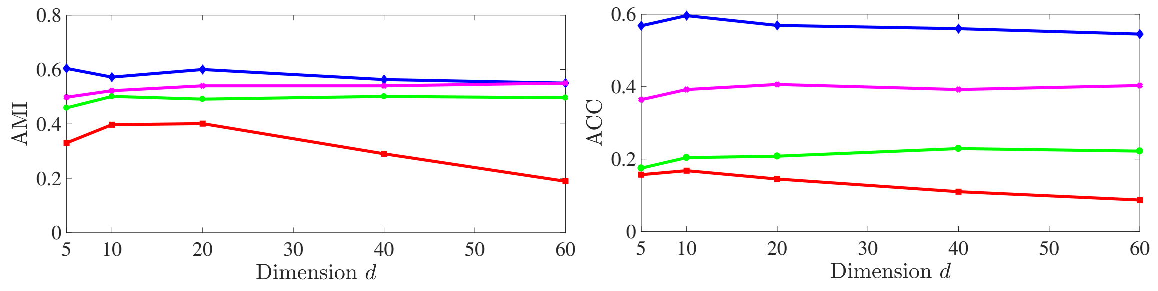
Figure 4.2: Effect of joint dimensionality reduction and clustering on the embedding. (a) A randomly sampled subset of 10K points from the MNIST dataset, visualized using t-SNE. (b) An embedding of these points into $\mathbb{R}^d$, performed by an SDAE that is optimized for dimensionality reduction. (c) An embedding of the same points by the same network, optimized with the DCC objective. When optimized for joint dimensionality reduction and clustering, the network produces an embedding with clearly separated clusters. Best viewed in color.

4.4.2 Visualization

A visualization is provided in Figure 4.2. Here we used Barnes-Hut t-SNE [75, 119] to visualize a randomly sampled subset of 10K datapoints from the MNIST dataset. We show the original dataset, the dataset embedded by the SDAE into $\mathbb{R}^d$ (optimized for dimensionality reduction), and the embedding into $\mathbb{R}^d$ produced by DCC. As shown in the figure, the embedding produced by DCC is characterized by well-defined, clearly separated clusters. The clusters strongly correspond to the ground-truth classes (coded by color in the figure), but were discovered with no supervision.



(a) MNIST



(b) Reuters

Figure 4.3: Robustness to dimensionality of the latent space. Clustering accuracy as a function of the dimensionality d of the latent space. AMI on the left, ACC on the right. Best viewed in color.

4.4.3 Robustness to dimensionality of the latent space

Next we study the robustness of DCC to the dimensionality d of the latent space. For this experiment, we consider fully-connected DCC. We vary d between 5 and 60 and measure AMI and ACC on the MNIST and Reuters datasets. For comparison, we report the performance of RCC-DR, DEC, which uses the same autoencoder architecture, as well as the accuracy attained by running k-means++ on the output of the SDAE, optimized for dimensionality reduction. The results are shown in Figure 4.3.

The results yield two conclusions. First, the accuracy of DCC, RCC-DR, DEC, and SDAE+k-means gradually decreases as the dimensionality d increases. This supports the common view that clustering becomes progressively harder as the dimensionality of the data increases. Second, the results demonstrate that DCC and RCC-DR are more robust to increased dimensionality than DEC and SDAE. For example, on MNIST, as the dimensionality d changes from 5 to 60, the accuracy (AMI) of DEC and SDAE drops by 28% and 35%, respectively, while the accuracy of DCC and RCC-DR decreases only by 9% and 7% respectively. When $d = 60$, the accuracy attained by DCC is higher than the accuracy attained by DEC and SDAE by 27% and 40%, respectively. Given that both DCC and RCC-DR utilize robust estimators and also share similarity in their formulations, it is not surprising that they exhibit similar robustness across datasets and measures.

4.4.4 Running Time

The runtime of DCC is mildly better than DEPICT and more than an order of magnitude better than JULE. For instance, on MNIST (the largest dataset considered), the total runtime of conv-DCC is 9,030 sec. For DEPICT, this runtime is 12,072 sec and for JULE it is 172,058 sec.

4.5 Conclusion

We have presented a clustering algorithm that combines nonlinear dimensionality reduction and clustering. Dimensionality reduction is performed by a deep network that embeds the data into a lower-dimensional space. The embedding is optimized as part of the clustering process and the resulting network produces clustered data. The presented algorithm does not rely on a priori knowledge of the number of ground-truth clusters. Nonlinear dimensionality reduction and clustering are performed by optimizing a global continuous objective using scalable gradient-based solvers.

All algorithms are evaluated on high-dimensional datasets of images and documents. Experiments demonstrate that our formulation performs on par or better than state-of-the-art clustering algorithms across all datasets. This includes recent approaches that utilize deep networks but do not use a global continuous formulation and rely on prior knowledge of the number of ground-truth clusters. Controlled experiments confirm that joint dimensionality reduction and clustering is more effective than a stagewise approach, and that the high accuracy achieved by the presented algorithm is stable across different dimensionalities of the latent space. Moreover, the algorithm performance is architecture agnostic. We hope that the presented continuous formulation can usefully inform future developments in data analysis.

Part II

Model Prior

Chapter 5: Fast SDP Solver

5.1 Introduction

Optimization problems involving either integer-valued vectors or low-rank matrices are ubiquitous in computer vision. Graph-cut methods for image segmentation, for example, involve optimization problems where integer-valued variables represent region labels [23, 120–122]. Problems in multi-camera structure from motion [123], manifold embedding [124], and matrix completion [125] all rely on optimization problems involving matrices with low rank constraints. Since these constraints are non-convex, the design of efficient algorithms that find globally optimal solutions is a difficult task.

For a wide range of applications [124, 126–130], non-convex constraints can be handled by *semidefinite relaxation* (SDR) [126]. In this approach, a non-convex optimization problem involving a vector of unknowns is "lifted" to a higher dimensional convex problem that involves a positive semidefinite (PSD) matrix, which then enables one to solve a SDP [131]. While SDR delivers state-of-the-art performance in a wide range of applications [121, 122, 124–126, 132], the approach significantly increases the dimensionality of the original optimization problem (i.e., replacing a vector with a matrix), which typically results in exorbitant computational costs and memory requirements. Nevertheless, SDR leads to SDPs whose global optimal solution can be found using robust numerical methods.

A growing number of computer-vision applications involve high-resolution images (or videos) that require SDPs with a large number of variables. General-purpose (interior point) solvers for SDPs do not scale well to such problem sizes; the worst-case complexity is $O(N^{6.5}\log(1/\varepsilon))$ for an $N \times N$ problem with ε objective error [133]. In imaging applications, N is often proportional to the number of pixels, which is potentially large.

The prohibitive complexity and memory requirements of solving SDPs exactly with a large number of variables has spawned interest in fast, non-convex solvers that avoid lifting. For example, recent progress in phase retrieval by Netrapalli *et al.* [134] and Candès *et al.* [135] has shown that non-convex optimization methods provably achieve solution quality comparable to exact SDR-based methods with significantly lower complexity. These methods operate on the original dimensions of the (un-lifted) problem, which enables their use on high-dimensional problems. Another prominent example is max-norm regularization by Lee *et al.* [136], which was proposed for solving high-dimensional matrix-completion problems and to approximately perform max-cut clustering. This method was shown to outperform exact SDR-based methods in terms of computational complexity, while delivering acceptable solution quality. While both of these examples outperform classical SDP-based methods, they are limited to very specific problem types, and cannot handle more complex SDPs that typically appear in computer vision.

In this chapter, we introduce a novel framework for approximately solving SDPs with positive semi-definite constraint matrices in a computationally efficient manner and with small memory footprint. The proposed *bi-convex* relaxation (BCR), transforms an SDP into a biconvex optimization problem, which can then be solved in the original, low-dimensional variable space at low complexity. The resulting biconvex problem is solved using a computationally-efficient AM procedure. Since AM is prone to get stuck in local minima, we propose an initialization scheme that enables BCR to start close to the global optimum of the original SDP—this initialization is key for our algorithm to quickly converge to an optimal or near-optimal solution. We showcase the effectiveness of the BCR framework by comparing to highly-specialized SDP solvers for a selected set of problems in computer vision involving image segmentation, co-segmentation, semantic segmentation and metric learning on manifolds. Our results demonstrate that BCR enables high-quality results while achieving speedups ranging from $4\times$ to $35\times$ over state-of-the-art competitor methods [137–141] for the studied applications. The code is available at `https://github.com/shahsohil/biconvex-relaxation`.

5.2 Related Work

We now briefly review semidefinite programs (SDPs) and discuss prior work on fast, approximate solvers for SDPs in computer vision and related applications.

5.2.1 Semidefinite Programs (SDPs)

SDPs find use in a large and growing number of fields, including computer vision, machine learning, signal and image processing, statistics, communications, and control [131]. SDPs can be written in the following general form:

$$\begin{aligned} \underset{\mathbf{Y}\in\mathcal{S}^{+}_{N\times N}}{\text{minimize}} \quad & \langle \mathbf{C}, \mathbf{Y} \rangle \\ \text{subject to} \quad & \langle \mathbf{A}_i, \mathbf{Y} \rangle = b_i, \quad \forall i \in \mathcal{E}, \\ & \langle \mathbf{A}_j, \mathbf{Y} \rangle \leq b_j, \quad \forall j \in \mathcal{B}, \end{aligned} \tag{5.1}$$

where $S^{+}_{N\times N}$ represents the set of $N \times N$ symmetric positive semidefinite matrices, and $\langle \mathbf{C}, \mathbf{Y} \rangle = \mathrm{trace}(\mathbf{C}^T\mathbf{Y})$ is the matrix inner product. The sets $\mathcal{E}$ and $\mathcal{B}$ contain the indices associated with the equality and inequality constraints, respectively; $\mathbf{A}_i$ and $\mathbf{A}_j$ are symmetric matrices of appropriate dimensions.

The key advantages of SDPs are that (i) they enable the transformation of certain non-convex constraints into convex constraints via semidefinite relaxation (SDR) [126] and (ii) the resulting problems often come with strong theoretical guarantees.

In computer vision, a large number of problems can be cast as SDPs of the general form (5.1). For example, [124] formulates image manifold learning as an SDP, [130] uses an SDP to enforce a non-negative lighting constraint when recovering scene lighting and object albedos, [142] uses an SDP for graph matching, [123] proposes an SDP that recovers the orientation of multiple cameras from point correspondences and essential matrices, and [125] uses low-rank SDPs to solve matrix-completion problems that arise in structure-from-motion and photometric stereo.

5.2.2 SDR for Binary-Valued Quadratic Problems

Semidefinite relaxation is commonly used to solve binary-valued labeling problems. For such problems, a set of variables take on binary values while minimizing a quadratic cost function that depends on the assignment of pairs of variables. Such labeling problems typically arise from Markov random fields (MRFs) for which many solution methods exist [143]. Spectral methods, e.g., [23], are often used to solve such binary-valued quadratic problems (BQPs)—the references [120, 121] used SDR inspired by the work of [122] that provides a generalized SDR for the max-cut problem. BQP problems have wide applicability to computer vision problems, such as segmentation and perceptual organization [120, 137, 144], semantic segmentation [145], matching [121, 146], surface reconstruction including photometric stereo and shape from defocus [129], and image restoration [147].

BQPs can be solved by lifting the binary-valued label vector $\mathbf{b} \in \{\pm 1\}^N$ to an N^2-dimensional matrix space by forming the PSD matrix $\mathbf{B} = \mathbf{b}\mathbf{b}^T$, whose non-convex rank-1 constraint is relaxed to PSD matrices $\mathbf{B} \in S^+_{N\times N}$ with an all-ones diagonal [126]. The goal is then to solve a SDP for $\mathbf{B}$ in the hope that the resulting matrix has rank 1; if $\mathbf{B}$ has higher rank, an approximate solution must be extracted which can either be obtained from the leading eigenvector or via randomization methods [126, 148].

5.2.3 Specialized Solvers for SDPs

General-purpose solvers for SDPs, such as SeDuMi [149] or SDPT3 [150], rely on interior point methods with high computational complexity and memory requirements. Hence, their use is restricted to low-dimensional problems. For problems in computer vision, where the number of variables can become comparable to the number of pixels in an image, more efficient algorithms are necessary. A handful of special-purpose algorithms have been proposed to solve specific problem types arising in computer vision. These algorithms fit into two classes: (i) convex algorithms that solve the original SDP by exploiting problem structure and (ii) non-convex methods that avoid lifting.

For certain problems, one can exactly solve SDPs with much lower complexity than interior point schemes, especially for BQP problems in computer vision. Ecker *et al.* [129] deployed a number of heuristics to speed up the Goemans-Williamson SDR [122] for surface reconstruction. Olsson *et al.* [147] proposed a spectral subgradient method to solve BQP problems that include a linear term, but are unable to handle inequality constraints. A particularly popular approach is the SDCut algorithms of Wang *et al.* [137]. This method solves BQP for some types of segmentation problems using dual gradient descent. SDCut leads to a similar relaxation as for BQP problems, but enables significantly lower complexity for graph cutting and its variants. To the best of our knowledge, the method by Wang *et al.* [137] yields state-of-the-art performance—nevertheless, our proposed method is at least an order of magnitude faster, as shown in Section 5.4.

Another algorithm class contains non-convex approximation methods that avoid lifting altogether. Since these methods work with low-dimensional unknowns, they are potentially more efficient than lifted methods. Simple examples include the Wiberg method [151] for low-rank matrix approximation, which uses Newton-type iterations to minimize a non-convex objective. A number of methods have been proposed for SDPs where the objective function is simply the trace-norm of $\mathbf{Y}$ (i.e., problem (5.1) with $\mathbf{C} = \mathbf{I}$) and without inequality constraints. Approaches include replacing the trace norm with the max-norm [136], or using the so-called Wirtinger flow to solve phase-retrieval problems [135]. One of the earliest approaches for non-convex methods are due to Burer and Montiero [9], who propose an augmented Lagrangian method. While this method is able to handle arbitrary objective functions, it does not naturally support inequality constraints (without introducing auxiliary slack variables). Furthermore, this approach uses non-convex methods for which convergence is not well understood and is sensitive to the initialization value.

While most of the above-mentioned methods provide best-in-class performance at low computational complexity, they are limited to very specific problems and cannot be generalized to other, more general SDPs.

5.3 Model and Algorithms

We now present the proposed *biconvex relaxation (BCR)* framework. We then propose an alternating minimization procedure and a suitable initialization method.

5.3.1 Biconvex Relaxation

Rather than solving the general SDP (5.1) directly, we exploit the following key fact: any matrix $\mathbf{Y}$ is symmetric positive semidefinite if and only if it has an expansion of the form $\mathbf{Y} = \mathbf{X}\mathbf{X}^T$. By substituting the factorization $\mathbf{Y} = \mathbf{X}\mathbf{X}^T$ into (5.1), we are able to remove the semidefinite constraint and arrive at the following problem:

$$\begin{aligned} \underset{\mathbf{X}\in\mathbb{R}^{N\times r}}{\text{minimize}} \quad & \operatorname{trace}(\mathbf{X}^T\mathbf{C}\mathbf{X}) \\ \text{subject to} \quad & \operatorname{trace}(\mathbf{X}^T\mathbf{A}_i\mathbf{X}) = b_i, \quad \forall i \in \mathcal{E}, \\ & \operatorname{trace}(\mathbf{X}^T\mathbf{A}_j\mathbf{X}) \leq b_j, \quad \forall j \in \mathcal{B}, \end{aligned} \tag{5.2}$$

where $r = \text{rank}(\mathbf{Y})$.[1] Note that any symmetric semi-definite matrix $\mathbf{A}$ has a (possibly complex-valued) square root $\mathbf{L}$ of the form $\mathbf{A} = \mathbf{L}^T\mathbf{L}$. Furthermore, we have $\text{trace}(\mathbf{X}^T\mathbf{A}\mathbf{X}) = \text{trace}(\mathbf{X}^T\mathbf{L}^T\mathbf{L}\mathbf{X}) = \|\mathbf{L}\mathbf{X}\|_F^2$, where $\|\cdot\|_F$ is the Frobenius (matrix) norm. This formulation enables us to rewrite (5.2) as follows:

$$\begin{aligned} \underset{\mathbf{X}\in\mathbb{R}^{N\times r}}{\text{minimize}} \quad & \text{trace}(\mathbf{X}^T\mathbf{C}\mathbf{X}) \\ \text{subject to} \quad & \mathbf{Q}_i = \mathbf{L}_i\mathbf{X}, \quad \|\mathbf{Q}_i\|_F^2 = b_i, \quad \forall i \in \mathcal{E}, \\ & \mathbf{Q}_j = \mathbf{L}_j\mathbf{X}, \quad \|\mathbf{Q}_j\|_F^2 \leq b_j, \quad \forall j \in \mathcal{B}. \end{aligned} \tag{5.3}$$

If the matrices $\{\mathbf{A}_i\}$, $\{\mathbf{A}_j\}$, and $\mathbf{C}$ are themselves PSDs, then the objective function in (5.3) is convex and quadratic, and the inequality constraints in (5.3) are convex—non-convexity of the problem is only caused by the equality constraints. The core idea of BCR explained next is to relax these equality constraints. Here, we assume that the factors of these matrices are easily obtained from the underlying problem structure. For some applications, where these factors are not readily available this could be a computational burden (worst case $\mathcal{O}(N^3)$) rather than an asset.

In the formulation (5.3), we have lost convexity. Nevertheless, whenever $r < N$, we achieved a (potentially large) dimensionality reduction compared to the original SDP (5.1). We now relax (5.3) in a form that is biconvex, i.e., convex with respect to a group of variables when the remaining variables are held constant. By relaxing the convex problem in biconvex form, we retain many advantages of the convex formulation while maintaining low dimensionality and speed. In particular, we propose to approximate (5.3) with the

[1] Straightforward extensions of our approach allow us to handle constraints of the form $\text{trace}(\mathbf{X}^T\mathbf{A}_k\mathbf{X}) \geq b_k, \forall k \in \mathcal{A}$, as well as complex-valued matrices and vectors.

following *biconvex relaxation (BCR)*:

$$\begin{aligned} \underset{\mathbf{X},\mathbf{Q}_i, i\in\{\mathcal{B}\cup\mathcal{E}\}}{\text{minimize}} \;\; & \text{trace}(\mathbf{X}^T\mathbf{C}\mathbf{X}) + \frac{\alpha}{2}\sum_{i\in\{\mathcal{E}\cup\mathcal{B}\}} \|\mathbf{Q}_i - \mathbf{L}_i\mathbf{X}\|_F^2 - \frac{\beta}{2}\sum_{j\in\mathcal{E}} \|\mathbf{Q}_j\|_F^2 \\ \text{subject to} \;\; & \|\mathbf{Q}_i\|_F^2 \le b_i, \quad \forall i \in \{\mathcal{B}\cup\mathcal{E}\}, \end{aligned} \tag{5.4}$$

where $\alpha > \beta > 0$ are relaxation parameters (discussed in detail below). In this BCR formulation, we relaxed the equality constraints $\|\mathbf{Q}_i\|_F^2 = b_i$, $\forall i \in \mathcal{E}$, to inequality constraints $\|\mathbf{Q}_i\|_F^2 \le b_i$, $\forall i \in \mathcal{E}$, and added negative quadratic penalty functions $-\frac{\beta}{2}\|\mathbf{Q}_i\|$, $\forall i \in \mathcal{E}$, to the objective function. These quadratic penalties attempt to force the inequality constraints in $\mathcal{E}$ to be satisfied exactly. We also replaced the constraints $\mathbf{Q}_i = \mathbf{L}_i\mathbf{X}$ and $\mathbf{Q}_j = \mathbf{L}_j\mathbf{X}$ by quadratic penalty functions in the objective function.

The relaxation parameters are chosen by freezing the ratio α/β to 2, and following a simple, principled way of setting β. Unless stated otherwise, we set β to match the curvature of the penalty term with the curvature of the objective i.e., $\beta = \|\mathbf{C}\|_2$, so that the resulting bi-convex problem is well-conditioned.

Our BCR formulation (5.4) has some important properties. First, if $\mathbf{C} \in \mathcal{S}^+_{N\times N}$ then the problem is biconvex, i.e., convex with respect to $\mathbf{X}$ when the $\{\mathbf{Q}_i\}$ are held constant, and vice versa. Furthermore, consider the case of solving a constraint feasibility problem (i.e., problem (5.1) with $\mathbf{C} = \mathbf{0}$). When $\mathbf{Y} = \mathbf{X}\mathbf{X}^T$ is a solution to (5.1) with $\mathbf{C} = \mathbf{0}$, the problem (5.4) assumes objective value $-\frac{\beta}{2}\sum_j b_j$, which is the global minimizer of the BCR formulation (5.4). Likewise, it is easy to see that any global minimizer of (5.4) with objective value $-\frac{\beta}{2}\sum_j b_j$ must be a solution to the original problem (5.1).

5.3.2 Alternating Minimization (AM) Algorithm

One of the key benefits of biconvexity is that (5.4) can be globally minimized with respect to $\mathbf{Q}$ or $\mathbf{X}$. Hence, it is natural to compute approximate solutions to (5.4) via alternating minimization. Note the convergence of AM for biconvex problems is well understood [152, 153]. The two stages of the proposed method for BCR are detailed next.

Stage 1: Minimize with respect to $\{\mathbf{Q}_i\}$. The BCR objective in (5.4) is quadratic in $\{\mathbf{Q}_i\}$ with no dependence between matrices. Consequently, the optimal value of $\mathbf{Q}_i$ can be found by minimizing the quadratic objective, and then reprojecting back into a unit Frobenius-norm ball of radius $\sqrt{b_i}$. The minimizer of the quadratic objective is given by $\frac{\alpha}{\alpha-\beta_i}\mathbf{L}_i\mathbf{X}$, where $\beta_i = 0$ if $i \in \mathcal{B}$ and $\beta_i = \beta$ if $i \in \mathcal{E}$. The projection onto the unit ball then leads to the following *expansion–reprojection* update:

$$\mathbf{Q}_i \leftarrow \frac{\mathbf{L}_i\mathbf{X}}{\|\mathbf{L}_i\mathbf{X}\|_F} \min\left\{\sqrt{b_i},\ \frac{\alpha}{\alpha-\beta_i}\|\mathbf{L}_i\mathbf{X}\|_F\right\}. \tag{5.5}$$

Intuitively, this expansion–reprojection update causes the matrix $\mathbf{Q}_i$ to expand if $i \in \mathcal{E}$, thus encouraging it to satisfy the relaxed constraints in (5.4) with equality.

Stage 2: Minimize with respect to $\mathbf{X}$. This stage solves the least-squares problem:

$$\mathbf{X} \leftarrow \underset{\mathbf{X}\in\mathbb{R}^{N\times r}}{\arg\min}\ \operatorname{trace}(\mathbf{X}^T\mathbf{C}\mathbf{X}) + \frac{\alpha}{2}\sum_{i\in\{\mathcal{E}\cup\mathcal{B}\}} \|\mathbf{Q}_i - \mathbf{L}_i\mathbf{X}\|_F^2. \tag{5.6}$$

The optimality conditions for this problem are linear equations, and the solution is

$$\mathbf{X} \leftarrow \left(\mathbf{C} + \alpha \sum_{i\in\{\mathcal{E}\cup\mathcal{B}\}} \mathbf{L}_i^T\mathbf{L}_i\right)^{-1} \left(\sum_{i\in\{\mathcal{E}\cup\mathcal{B}\}} \mathbf{L}_i^T\mathbf{Q}_i\right), \tag{5.7}$$

where the matrix inverse (one-time computation) may be replaced by a pseudo-inverse if necessary. Alternatively, one may perform a simple gradient-descent step with a suitable

Algorithm 6 AM for Biconvex Relaxation

```
1: inputs: $\mathbf{C}$, $\{\mathbf{L}_i\}$, $b_i$, $\alpha$, and $\beta$, output: $\mathbf{X}$
2: Compute an initializer for $\mathbf{X}$ as in Section 5.3.3
3: Precompute $\mathrm{M} = \left(\mathbf{C} + \alpha \sum_{i\in\{E\cup\mathcal{B}\}} \mathbf{L}_i^T \mathbf{L}_i\right)^{-1}$
4: while not converged do
5:     $\mathbf{Q}_i \leftarrow \frac{\mathbf{L}_i\mathbf{X}}{\|\mathbf{L}_i\mathbf{X}\|_F} \min\left\{\sqrt{b_i}, \frac{\alpha}{\alpha-\beta_i}\|\mathbf{L}_i\mathbf{X}\|_F\right\}$
6:     $\mathbf{X} \leftarrow \mathrm{M}\left(\sum_{i\in\{\mathcal{E}\cup\mathcal{B}\}} \mathbf{L}_i^T \mathbf{Q}_i\right)$,
7: end while
```

step size, which avoids the inversion of a potentially large-dimensional matrix.

The resulting AM algorithm for the proposed BCR (5.4) is summarized in Algorithm 6.

5.3.3 Initialization

The problem (5.4) is biconvex and hence, a global minimizer can be found with respect to either $\{\mathbf{Q}_i\}$ or $\mathbf{X}$, although a global minimizer of the joint problem is not guaranteed. We hope to find a global minimizer at low complexity using the AM method, but in practice AM may get trapped in local minima, especially if the variables have been initialized poorly. We now propose a principled method for computing an initializer for $\mathbf{X}$ that is often close to the global optimum of the BCR problem—our initializer is key for the success of the proposed AM procedure and enables fast convergence.

The papers [134, 135] have considered optimization problems that arise in phase retrieval where $\mathcal{B} = \varnothing$ (i.e., there are only equality constraints), $\mathbf{C} = \mathbf{I}$ being the identity, and $\mathbf{Y}$ being rank one. For such problems, the objective of (5.1) reduces to $\mathrm{trace}(\mathbf{Y})$. By setting $\mathbf{Y} = \mathbf{x}\mathbf{x}^T$, we obtain the following formulation:

$$\underset{\mathbf{x}\in\mathbb{R}^N}{\text{minimize}} \ \|\mathbf{x}\|_2^2 \quad \text{subject to} \ \mathbf{q}_i = \mathbf{L}_i\mathbf{x}, \quad \|\mathbf{q}_i\|_2^2 = b_i, \quad \forall i \in \mathcal{E}. \tag{5.8}$$

Netrapali *et al.* [134] proposed an iterative algorithm for solving (5.8), which has been initialized by the following strategy. Define

$$\mathbf{Z} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} b_i \mathbf{L}_i^T \mathbf{L}_i. \tag{5.9}$$

Let $\mathbf{v}$ be the leading eigenvector of $\mathbf{Z}$ and λ the leading eigenvalue. Then $\mathbf{x} = \lambda \mathbf{v}$ is an accurate approximation to the true solution of (5.8). In fact, if the matrices $\mathbf{L}_i$ are sampled from a random normal distribution, then it was shown in [134, 135] that $\mathbb{E}\|\mathbf{x}^\star - \lambda\mathbf{v}\|_2^2 \to 0$ (in expectation) as $|\mathcal{E}| \to \infty$, where $\mathbf{x}^\star$ is the true solution to (5.8).

We are interested in a good initializer for the general problem in (5.3) where $\mathbf{X}$ can be rank one or higher. We focus on problems with equality constraints only—note that one can use slack variables to convert a problem with inequality constraints into the same form [131]. Given that $\mathbf{C}$ is a symmetric positive definite matrix, it can be decomposed into $\mathbf{C} = \mathbf{U}^T\mathbf{U}$. By the change of variables $\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{X}$, we can rewrite (5.1) as follows:

$$\underset{\mathbf{X} \in \mathbb{R}^{N \times r}}{\text{minimize}} \ \|\widetilde{\mathbf{X}}\|_F^2 \quad \text{subject to} \ \langle \widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T \rangle = b_i, \quad \forall i \in \mathcal{E}, \tag{5.10}$$

where $\widetilde{\mathbf{A}}_i = \mathbf{U}^{-T}\mathbf{A}_i\mathbf{U}^{-1}$, and we omitted the inequality constraints. To initialize the proposed AM procedure in Algorithm 6, we make the change of variables $\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{X}$ to transform the BCR formulation into the form of (5.10). Analogously to the initialization procedure in [134] for phase retrieval, we then compute an initializer $\widetilde{\mathbf{X}}_0$ using the leading r eigenvectors of $\mathbf{Z}$ scaled by the leading eigenvalue λ. Finally, we calculate the initializer for the original problem by reversing the change of variables as $\mathbf{X}_0 = \mathbf{U}^{-1}\widetilde{\mathbf{X}}_0$. For most problems the initialization time is a small fraction of the total runtime.

5.3.4 Advantages of Biconvex Relaxation

The proposed framework has numerous advantages over other non-convex methods. First and foremost, BCR can be applied to general SDPs. Specialized methods, such as Wirtinger flow [135] for phase retrieval and the Wiberg method [151] for low-rank approximation are computationally efficient, but restricted to specific problem types. Similarly, the max-norm method [136] is limited to solving trace-norm-regularized SDPs. The method of Burer and Montiero [9] is less specialized, but does not naturally support inequality constraints. Furthermore, since BCR problems are biconvex, one can use numerical solvers with guaranteed convergence. Convergence is guaranteed not only for the proposed AM least-squares method in Algorithm 6 (for which the objective decreases monotonically), but also for a broad range of gradient-descent schemes suitable to find solutions to biconvex problems [154]. In contrast, the method in [9] uses augmented Lagrangian methods with non-linear constraints for which convergence is not guaranteed.

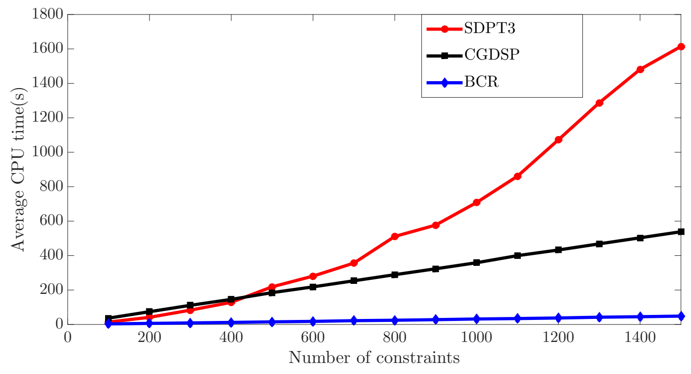
5.4 Experiments

We now evaluate our solver using both synthetic and real-world data. We begin with a brief comparison showing that biconvex solvers outperform both interior-point methods for general SDPs and also state-of-the-art low-rank solvers. Of course, specialized solvers for specific problem forms achieve superior performance to classical interior point schemes. For this reason, we evaluate our proposed method on three important computer vision applications, i.e., segmentation, co-segmentation, semantic segmentation and manifold metric learning, using public datasets, and we compare our results to state-of-the-art methods. These applications are ideal because (i) they involve large scale SDPs and (ii) customized solvers are available that exploit problem structure to solve these problems efficiently. Hence, we can compare our BCR framework to powerful and optimized solvers.

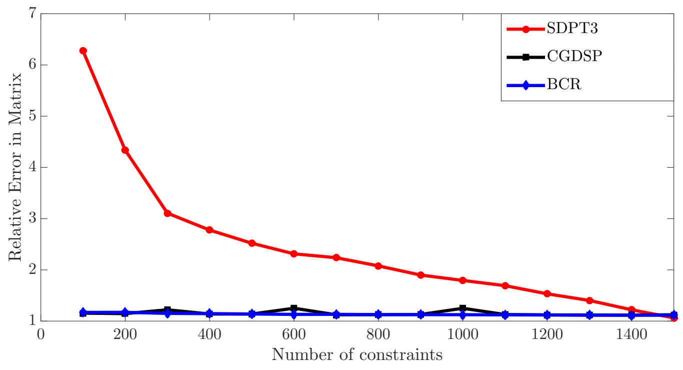
5.4.1 General-Form Problems

We briefly demonstrate that BCR performs well on general SDPs by comparing to the widely used SDP solver, SDPT3 [150] and the state-of-the-art, low-rank SDP solver CGDSP [10]. Note that SDPT3 uses an interior point approach to solve the convex problem in (5.1) whereas the CGDSP solver uses gradient-descent to solve a non-convex formulation. For fairness, we initialize both algorithms using the proposed initializer and the gradient descent step in CGDSP was implemented using various acceleration techniques [155]. Since CGDSP cannot handle inequality constraints we restrict our comparison to equality constraints only.

Experiments: We randomly generate a 256×256 rank-3 data matrix of the form $\mathbf{Y}_{\text{true}} = \mathbf{x}_1\mathbf{x}_1^T + \mathbf{x}_2\mathbf{x}_2^T + \mathbf{x}_3\mathbf{x}_3^T$, where $\{\mathbf{x}_i\}$ are standard normal vectors. We generate a standard normal matrix $\mathbf{L}$ and compute $\mathbf{C} = \mathbf{L}^T\mathbf{L}$. Gaussian matrices $\mathbf{A}_i \in \mathbb{R}^{250\times 250}$ form equality constraints. We report the relative error in the recovered solution $\mathbf{Y}_{\text{rec}}$ measured as $\|\mathbf{Y}_{\text{rec}} - \mathbf{Y}_{\text{true}}\|/\|\mathbf{Y}_{\text{true}}\|$. Average runtimes for varying numbers of constraints are shown in Figure 5.1(a), while Figure 5.1(b) plots the average relative error. Figure 5.1(a) shows that our method has the best runtime of all the schemes. Figure 5.1(b) shows convex interior point methods do not recover the correct solution for small numbers of constraints. With few constraints, the full lifted SDP is under-determined, allowing the objective to go to zero. In contrast, the proposed BCR approach is able to enforce an additional rank-3 constraint, which is advantageous when the number of constraints is low.



(a) Average solver runtime



(b) Average relative error

Figure 5.1: Results on synthetic data for varying number of linear constraints.

5.4.2 Image Segmentation

Consider an image of N pixels. Segmentation of foreground and background objects can be accomplished using graph-based approaches, where graph edges encode the similarities between pixel pairs. Such approaches include normalized cut [23] and ratio cut [156]. The graph cut problem can be formulated as an NP-hard integer program [122]

$$\underset{\mathbf{x}\in\{-1,1\}^N}{\text{minimize}}\ \mathbf{x}^T\mathbf{L}\mathbf{x}, \tag{5.11}$$

where $\mathbf{L}$ encodes edge weights and $\mathbf{x}$ contains binary region labels, one for each pixel. This problem can be "lifted" to the equivalent higher dimensional problem

$$\underset{\mathbf{X}\in S^{+}_{N\times N}}{\text{minimize}} \;\; \operatorname{trace}(\mathbf{L}^T\mathbf{X}) \quad \text{subject to} \;\; \operatorname{diag}(\mathbf{X}) = \mathbf{1}, \;\; \operatorname{rank}(\mathbf{X}) = 1. \tag{5.12}$$

After dropping the non-convex rank constraint, (5.12) becomes an SDP that is solvable using convex optimization [120, 132, 146]. The SDP approach is computationally intractable if solved using off-the-shelf SDP solvers (such as SDPT3 [150] or other interior point methods). Furthermore, exact solutions cannot be recovered when the solution to the SDP has rank greater than 1. In contrast, BCR is computational efficient for large problems and can easily incorporate rank constraints, leading to efficient spectral clustering.

BCR is also capable of incorporating annotated foreground and background pixel priors [157] using linear equality and inequality constraints. We consider the SDP based segmentation presented in [157], which contains three grouping constraints on the pixels: $(\mathbf{t}_f^T\mathbf{P}\mathbf{x})^2 \geq \kappa\|\mathbf{t}_f^T\mathbf{P}\mathbf{x}\|_1^2$, $(\mathbf{t}_b^T\mathbf{P}\mathbf{x})^2 \geq \kappa\|\mathbf{t}_b^T\mathbf{P}\mathbf{x}\|_1^2$ and $((\mathbf{t}_f-\mathbf{t}_b)^T\mathbf{P}\mathbf{x})^2 \geq \kappa\|(\mathbf{t}_f-\mathbf{t}_b)^T\mathbf{P}\mathbf{x}\|_1^2$, where $\kappa \in [0,1]$. $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ is the normalized pairwise affinity matrix and $\mathbf{t}_f$ and $\mathbf{t}_b$ are indicator variables denoting the foreground and background pixels. These constraints enforce that the segmentation respects the pre-labeled pixels given by the user, and also pushes high similarity pixels to have the same label. The affinity matrix $\mathbf{W}$ is given by

$$W_{i,j} = \begin{cases} \exp\left(-\frac{\|\mathbf{f}_i-\mathbf{f}_j\|_2^2}{\gamma_f^2} - \frac{d(i,j)^2}{\gamma_d^2}\right), & \text{if } d(i,j) < r \\ 0, & \text{otherwise,} \end{cases} \tag{5.13}$$

where $\mathbf{f}_i$ is the color histogram of the ith super-pixel and $d(i,j)$ is the spatial distance between i and j. Considering these constraints and letting $\mathbf{X} = \mathbf{Y}\mathbf{Y}^T$, (5.12) can be written in the form of (5.2) as follows:

$$\begin{aligned}
\underset{\mathbf{Y}\in\mathbb{R}^{N\times r}}{\text{minimize}} \quad & \operatorname{trace}(\mathbf{Y}^T\mathbf{L}\mathbf{Y}) \\
\text{subject to} \quad & \operatorname{trace}(\mathbf{Y}^T\mathbf{A}_i\mathbf{Y}) = 1, \quad \forall i = 1,\dots,N \\
& \operatorname{trace}(\mathbf{Y}^T\mathbf{B}_2\mathbf{Y}) \geq \kappa\|\mathbf{t}_f^T\mathbf{P}\mathbf{x}\|_1^2,\ \operatorname{trace}(\mathbf{Y}^T\mathbf{B}_3\mathbf{Y}) \geq \kappa\|\mathbf{t}_b^T\mathbf{P}\mathbf{x}\|_1^2 \\
& \operatorname{trace}(\mathbf{Y}^T\mathbf{B}_4\mathbf{Y}) \geq \kappa\|(\mathbf{t}_f-\mathbf{t}_b)^T\mathbf{P}\mathbf{x}\|_1^2,\ \operatorname{trace}(\mathbf{Y}^T\mathbf{B}_1\mathbf{Y}) = 0.
\end{aligned} \tag{5.14}$$

Here, r is the rank of the desired solution, $\mathbf{B}_1 = \mathbf{1}\mathbf{1}^T$, $\mathbf{B}_2 = \mathbf{P}\mathbf{t}_f\mathbf{t}_f^T\mathbf{P}$, $\mathbf{B}_3 = \mathbf{P}\mathbf{t}_b\mathbf{t}_b^T\mathbf{P}$, $\mathbf{B}_4 = \mathbf{P}(\mathbf{t}_f-\mathbf{t}_b)(\mathbf{t}_f-\mathbf{t}_b)^T\mathbf{P}$, $\mathbf{A}_i = \mathbf{e}_i\mathbf{e}_i^T$, $\mathbf{e}_i \in \mathbb{R}^n$ is an elementary vector with a 1 at the ith position. After solving (5.14) using BCR (5.4), the final binary solution is extracted from the score vector using the swept random hyperplanes method [148].

We compare the performance of BCR with the highly customized BQP solver SDCut [137] and biased normalized cut (BNCut) [138]. BNCut is an extension of the Normalized cut algorithm [23] whereas SDCut is currently the most efficient and accurate SDR solver but limited only to solving BQP problems. Also, BNCut can support only one quadratic grouping constraint per problem.

Experiments: We consider the Berkeley image segmentation dataset [52]. Each image is segmented into super-pixels using the VL-Feat [158] toolbox. For SDCut and BNCut, we use the publicly available code with hyper-parameters set to the values suggested in [137]. For BCR, we set $\beta = \lambda/\sqrt{|\mathcal{B}\cup\mathcal{E}|}$, where λ controls the coarseness of the segmentation by mediating the tradeoff between the objective and constraints, and would typically be chosen from $[1, 10]$ via cross validation. For simplicity, we just set $\lambda = 5$ in all experiments reported here.

We compare the runtime and quality of each algorithm. Figure 5.2 shows the

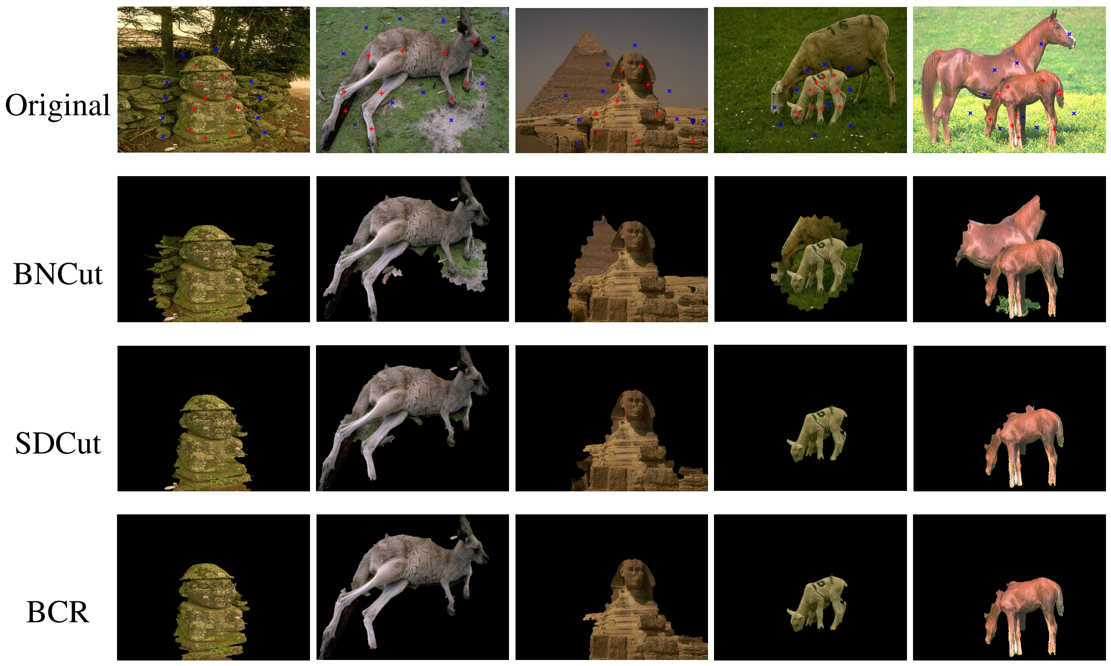


Figure 5.2: Image segmentation results on the Berkeley dataset. The red and blue marker indicates the annotated foreground and background super-pixels, respectively.

segmentation results while the quantitative results are displayed in Table 5.1. For all the considered images, our approach gives superior foreground object segmentation compared to SDCut and BNCut. Moreover, as seen in Table 5.1, our solver is $35\times$ faster than SDCut and yields lower objective energy. Segmentation using BCR is achieved using only rank 2 solutions whereas SDCut requires rank 7 solutions to obtain results of comparable accuracy.[2] Note that while BNCut with rank 1 solutions is much faster than SDP based methods, the BNCut segmentation results are not on par with SDP approaches.

5.4.3 Co-segmentation

We next consider image co-segmentation, in which segmentation of the same object is jointly computed on multiple images simultaneously. Because co-segmentation involves

[2]The optimal solutions found by SDCut all had rank 7 except for one solution of rank 5.

multiple images, it provides a testbed for large problem instances. Co-segmentation balances a tradeoff between two criteria: (i) color and spatial consistency within a single image and (ii) discrimination between foreground and background pixels over multiple images. We closely follow the work of Joulin *et al.* [144], whose formulation is given by

$$\underset{\mathbf{x}\in\{\pm 1\}^N}{\text{minimize}}\ \mathbf{x}^T\mathbf{A}\mathbf{x} \quad \text{subject to} \ \ (\mathbf{x}^T\delta_i)^2 \le \lambda^2, \quad \forall i = 1,\dots,M, \tag{5.15}$$

where M is the number of images and $N = \sum_{i=1}^{M} N_i$ is the total number of pixels over all images. The matrix $\mathbf{A} = \mathbf{A}_b + \frac{\mu}{N}\mathbf{A}_w$, where $\mathbf{A}_w$ is the intra-image affinity matrix and $\mathbf{A}_b$ is the inter-image discriminative clustering cost matrix computed using the χ^2 distance between SIFT features in different images (see [144] for a details).

To solve this problem with BCR, we re-write (5.15) in the form (5.2) to obtain

$$\begin{aligned} \underset{\mathbf{X}\in\mathbb{R}^{N\times r}}{\text{minimize}} \quad & \text{trace}(\mathbf{X}^T\mathbf{A}\mathbf{X}) \\ \text{subject to:} \quad & \text{trace}(\mathbf{X}^T\mathbf{Z}_i\mathbf{X}) = 1, \quad \forall i = 1,\dots,N \\ & \text{trace}(\mathbf{X}^T\boldsymbol{\Delta}_i\mathbf{X}) \le \lambda^2, \ \forall i = 1,\dots,M, \end{aligned} \tag{5.16}$$

where $\boldsymbol{\Delta}_i = \delta_\mathbf{i}\delta_\mathbf{i}{}^T$ and $\mathbf{Z}_i = \mathbf{e}_i\mathbf{e}_i^T$. Finally, (5.16) is solved using BCR (5.4), following which one can recover the optimal score vector $\mathbf{x}_p^*$ as the leading eigenvector of $\mathbf{X}^*$. The final binary solution is extracted by thresholding $\mathbf{x}_p^*$ to obtain integer-valued labels [139].

Experiments: We compare BCR to two well-known co-segmentation methods, namely low-rank factorization [139] (denoted LR) and SDCut [137]. We use publicly available code for LR and SDCut. We test on the Weizman horses[3] and MSRC[4] datasets with a total of four classes (horse, car-front, car-back, and face) containing $6 \sim 10$

[3]www.msri.org/people/members/eranb/

[4]www.research.microsoft.com/en-us/projects/objectclassrecognition/

Table 5.1: Results on image segmentation. Numbers are the mean over the images in Fig. 5.2. Lower numbers are better. The proposed algorithm and the best performance are highlighted.

Method	BNCut	SDCut	**BCR**
Time (s)	**0.08**	27.64	0.97
Objective	10.84	6.40	**6.34**
Rank	1	7	2

Table 5.2: Co-segmentation results. The proposed algorithm and the best performance is highlighted.

		Test Cases			
Dataset		horse	face	car-back	car-front
Number of images		10	10	6	6
Variables in BQPs		4587	6684	4012	4017
Time (s)	LowRank	2647	1614	724	749
	SDCut	220	274	180	590
	BCR	**18.8**	**61.8**	**46.7**	**44.7**
Objective	LowRank	4.84	4.48	5.00	4.17
	SDCut	5.24	4.94	4.53	4.27
	BCR	**4.64**	**3.29**	**4.36**	**3.94**
Rank	LowRank	18	11	7	10
	SDCut	3	3	3	3
	BCR	**2**	**2**	**2**	**2**

images per class. Each image is over-segmented to $400 \sim 700$ SLIC superpixels using the VLFeat [158] toolbox, giving a total of around $4000 \sim 7000$ super-pixels per class. Relative to image segmentation problems, this application requires $10\times$ more variables.

Qualitative results are presented in Figure 5.3 while Table 5.2 provides a quantitative comparison. From Table 5.2, we observe that on average our method converges $\sim 9.5\times$ faster than SDCut and $\sim 60\times$ faster than LR. Moreover, the optimal objective value achieved by BCR is significantly lower than that achieved by both SDCut and LR methods. Figure 5.3 displays the visualization of the final score vector $\mathbf{x}_p^*$ for selected images, depicting that in general SDCut and BCR produce similar results. Furthermore, the optimal BCR score vector $\mathbf{x}_p^*$ is extracted from a rank-2 solution, as compared to rank-3 and rank-7 solutions needed to get comparable results with SDCut and LR.

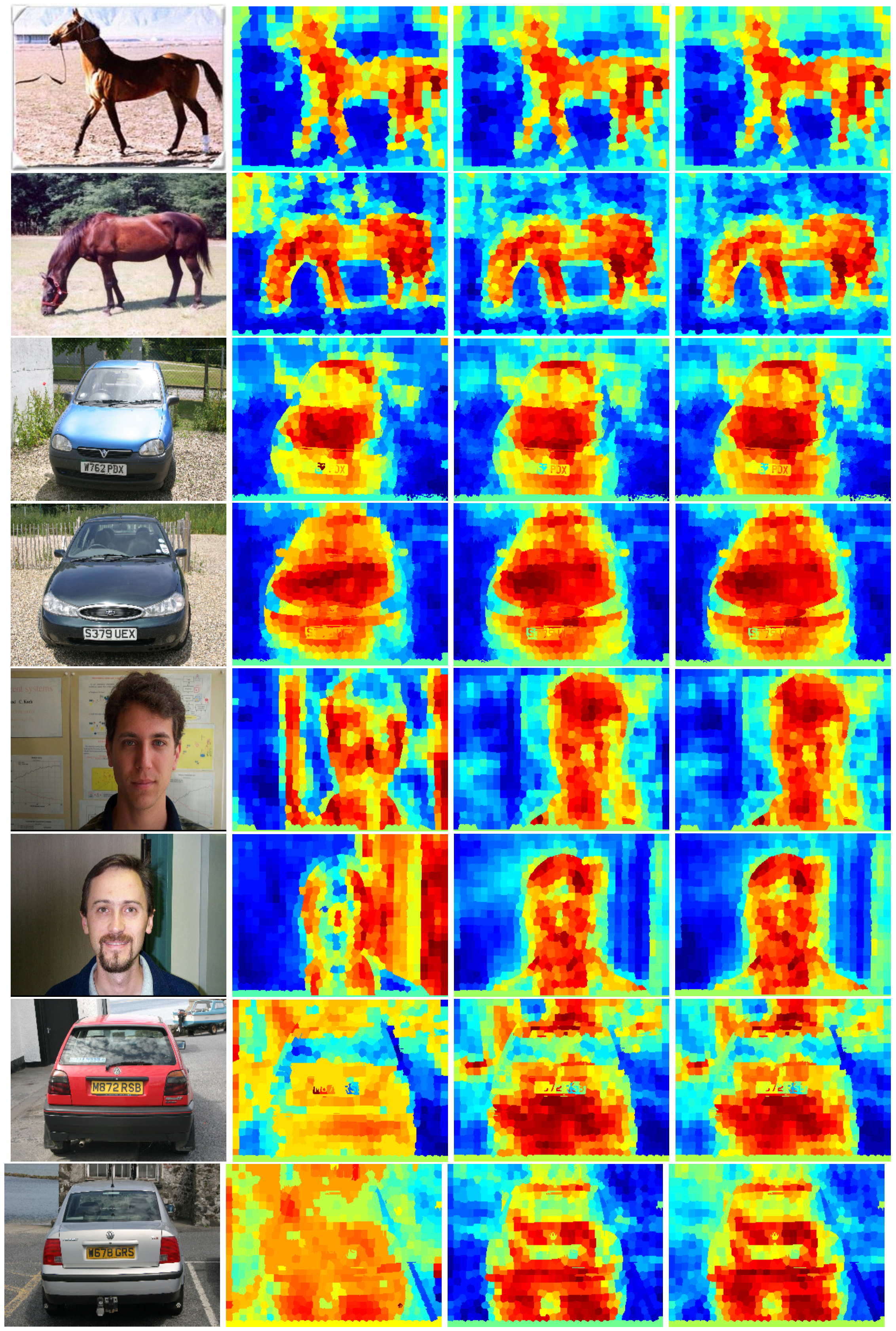


Figure 5.3: Co-segmentation results on the Weizman horses and MSRC datasets. From left to right: the original images, the results of LR, SDCut, and BCR, respectively.

5.4.4 Conditional Random Fields

Next, we consider solving maximum a posteriori(MAP) inference on fully connected CRFs for semantic segmentation [1]. Unlike previously discussed segmentation approaches, in this model each node is connected to every other node in an image and the label assignment is non-binary. Through this application, we wish to demonstrate that unlike other SDP solvers our proposed method is scalable and it can efficiently handle large number of variables. A fully connected CRF on N nodes, denoted by random variables $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, consists of N^2 edges and each variable can be assigned discrete labels from the set $\mathcal{L} = \{1, \ldots, L\}$. The MAP inference problem is given by,

$$\underset{\mathbf{x}\in\mathcal{L}^N}{\text{minimize}} \;\; E(\mathbf{x}) = \sum_i \phi_i(x_i) + \sum_{i,j,i<j} \phi_{i,j}(x_i, x_j) \tag{5.17}$$

where ϕ_i denotes unary potential and $\phi_{i,j} = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$ denotes pairwise potential. Here, $\mathbf{f}_i$ is the feature vector corresponding to variable x_i, $\mu(x_i, x_j) = \delta(x_i \neq x_j)$ is Potts model based label compatibility function, k is the kernel function and w indicates the weights. Following [145], the (5.17) can be expressed as BQP,

$$\begin{aligned} &\underset{\mathbf{X}\in\{0,1\}^{N\times L}}{\text{minimize}} && \text{trace}(\mathbf{H}^T\mathbf{X}) - \frac{1}{2}\text{trace}(\mathbf{X}^T\mathbf{K}\mathbf{X}) + \frac{1}{2}\mathbf{1}^T\mathbf{K}\mathbf{1} \\ &\text{subject to:} && \sum_{l=1}^{L} X_{i,l} = 1 \quad \forall\, i \in N \end{aligned} \tag{5.18}$$

where $H_{i,l} = \phi_i(l)$ and $K_{i,j} = \sum_{m=1}^{M} w^{(m)} k^{(m)}(f_i, f_j)$. We solve this problem in the continuous label space using BCR. We introduce an additional equality constraint in (5.18) given by,

$$\text{trace}(\mathbf{X}^T\mathbf{Z}_i\mathbf{X}) = 1 \quad \forall\, i = 1, \ldots, N \tag{5.19}$$

where $\mathbf{Z}_i = \mathbf{e}_i \mathbf{e}_i^T$. This constraint in combination with the ones in (5.18) forces solution to be one of the vertices of simplex. Once solved, the discrete label assignment $\mathbf{x}_i^*$ is obtained by assigning to the maximum indices along each row.

Experiments: We compare our algorithm with mean-field approximation [1] on the test data of MSRC 21-class database. The unary potentials are obtained from the corresponding author of [1]. Following the procedure in [145], we approximate the kernel $\mathbf{K}$ using the Nystrom method. Each of the image contains around 60,000 pixels and hence the large number of variables. The quantitative result is summarized in Table 5.3 whereas Figure 5.4 displays the qualitative results. The runtime comparison does not seem fair as our MATLAB implementation runtime is compared against that of C++ implementation of [1]. The code for [145] is not released and hence cannot be compared.

Method	Standard GT	Accurate GT	Time (s)
Unary	82.94	83.70	-
BCR	85.35	89.34	268
MF+filter [1]	86	88.2	4.5

Table 5.3: Comparison of quantitative results on MSRC-21 test dataset for fully connected CRF's.

5.4.5 Metric Learning on Manifolds

Large SDPs play a central role in manifold methods for classification and dimensionality reduction on image sets and videos [140, 141, 159]. Manifold methods rely heavily on covariance matrices, which accurately characterize second-order statistics of variation between images. Typical methods require computing distances between matrices along a Riemannian manifold—a task that is expensive for large matrices and limits the applicability of these techniques. It is of interest to perform dimensionality reduction on

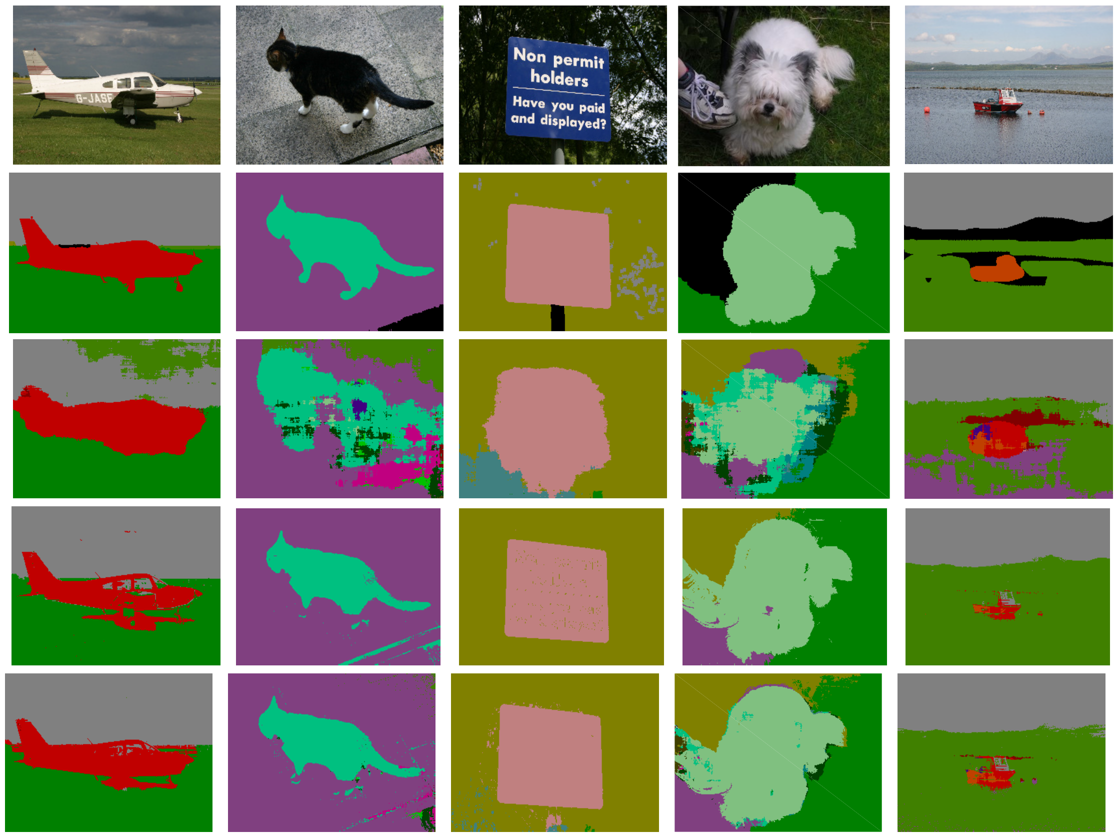


Figure 5.4: Qualitative results of Image segmentation on MSRC-21 dataset using fully connected CRF's. From top to bottom: original image, groundtruth segmentation, unary potential, mean-field and ours

SPD matrices, thus enabling the use of covariance methods on very large problems.

In this section, we discuss dimensionality reduction on manifolds of SPD matrices using BCR, which is computationally much faster than the state-of-the-art while achieving comparable (and often better) performance. Consider a set of high-dimensional SPD matrices $\{\mathbf{S}_1, \dots, \mathbf{S}_n\}$ where $\mathbf{S}_i \in S^+_{N\times N}$. We can project these onto a low-dimensional manifold of rank $K < N$ by solving

$$\begin{aligned} \underset{\mathbf{X}\in S^+_{N\times N}, \eta_{ij}\geq 0}{\text{minimize}} \quad & \text{trace}(\mathbf{X}) + \mu \textstyle\sum_{i,j} \eta_{ij} \\ \text{subject to} \quad & \mathbb{D}_X(\mathbf{S}_i, \mathbf{S}_j) \leq u + \eta_{ij}, \quad \forall (i,j) \in \mathcal{C} \\ & \mathbb{D}_X(\mathbf{S}_i, \mathbf{S}_j) \geq l - \eta_{ij}, \quad \forall (i,j) \in \mathcal{D} \end{aligned} \tag{5.20}$$

where $\mathbf{X}$ is a (low-dimensional) SPD matrix, $\mathbb{D}_X$ is Riemannian distance metric, and η_{ij} are slack variables. The sets $\mathcal{C}$ and $\mathcal{D}$ contain pairs of similar/dissimilar matrices labeled by the user, and the scalars u and l are given upper and lower bounds. For simplicity, we measure distance using the log-Euclidean metric (LEM) defined by [140]

$$\mathbb{D}(\mathbf{S}_i, \mathbf{S}_j) = \| \log(\mathbf{S}_i) - \log(\mathbf{S}_j) \|_F^2 = \operatorname{trace}\big((\mathbf{R}_i - \mathbf{R}_j)^T (\mathbf{R}_i - \mathbf{R}_j)\big) , \tag{5.21}$$

where $\mathbf{R}_i = \log(\mathbf{S}_i)$ is a matrix logarithm. When $\mathbf{X}$ has rank K, it is a transformation onto the space of rank K covariance matrices, where the new distance is given by [140]

$$\mathbb{D}_X(\mathbf{S}_i, \mathbf{S}_j) = \operatorname{trace}\big(\mathbf{X}(\mathbf{R}_i - \mathbf{R}_j)^T (\mathbf{R}_i - \mathbf{R}_j)\big). \tag{5.22}$$

We propose to solve the semi-definite program (5.20) using the representation $\mathbf{X} = \mathbf{Y}\mathbf{Y}^T$ which puts our problem in the form (5.2) with $\mathbf{A}_{ij} = (\mathbf{R}_i - \mathbf{R}_j)^T (\mathbf{R}_i - \mathbf{R}_j)$. This problem is then solved using BCR, where the slack variables $\{\eta_{ij}\}$ are removed and instead a hinge loss penalty approximately enforces the inequality constraints in (5.4). In our experiments we choose $u = \rho - \xi\tau$ and $l = \rho + \xi\tau$, where ρ and τ are the mean and standard deviation of the pairwise distances between $\{S_i\}$ in the original space, respectively. The quantities ξ and μ are treated as hyper-parameters.

Experiments: We analyze the performance of our approach (short BCRML) against state-of-the-art manifold metric learning algorithms using three image set classification databases: ETH-80, YouTube Celebrities (YTC), and YouTube Faces (YTF) [86]. The ETH-80 database consists of a 10 image set for each of 8 object categories. YTC contains 1,910 video sequences for 47 subjects from YouTube. YTF is a face verification database containing 3,425 videos of 1,595 different people. Features were extracted from images as described in [140]. Faces were cropped from each dataset using bounding boxes, and scaled to size 20×20 for the ETH and YTC datasets. For YTF we used a larger 30×30 scaling, as larger images were needed to replicate the results reported in [140].

We compare BCR to three state-of-the-art schemes: LEML [140] is based on a log-Euclidean metric, and minimizes the logdet divergence between matrices using Bregman projections. SPDML [141] optimizes a cost function on the Grassmannian manifold while making use of either the affine-invariant metric (AIM) or Stein metric. We use publicly available code for LEML and SPDML and follow the details in [140, 141] to select algorithm specific hyper-parameters using cross-validation. For BCRML, we fix α to be $1/\sqrt{|\mathcal{C} \cup \mathcal{D}|}$ and μ as $\alpha/2$. The ξ is fixed to 0.5, which performed well under cross-validation. For SPDML, the dimensionality of the target manifold K is fixed to 100. In LEML, the dimension cannot be reduced and thus the final dimension is the same as the original. Hence, for a fair comparison, we report the performance of BCRML using full target dimension (BCRML-full) as well as for $K = 100$ (BCRML-100).

Table 5.4 summarizes the classification performance on the above datasets. We observe that BCRML performs almost the same or better than other ML algorithms. One can apply other algorithms to gain a further performance boost after projecting onto the low-dimensional manifold. Hence, we also provide a performance evaluation for LEML and BCRML using the LEM based CDL-LDA recognition algorithm [159]. The last three columns of Table 5.4 display the runtime measured on the YTC dataset. We note that BCRML-100 trains roughly $2\times$ faster and overall runs about $3.5\times$ faster than the next fastest method. Moreover, on testing using CDL-LDA, the overall computation time is approximately $5\times$ faster in comparison to the next-best performing approach.

Method	ETH-80	YTC	YTF	Train (s)	Test (s)	Total (s)
AIM	89.25 ± 1.69	62.77 ± 2.89	59.82 ± 1.63	-	5.189	1463.3
Stein	89.00 ± 2.42	62.02 ± 2.71	57.56 ± 2.17	-	3.593	1013.3
LEM	90.00 ± 2.64	62.06 ± 3.04	59.78 ± 1.69	-	1.641	462
SPDML-AIM [141]	91.00 ± 3.39	65.32 ± 2.77	61.64 ± 1.46	3941	0.227	4005
SPDML-Stein [141]	90.75 ± 3.34	66.10 ± 2.92	61.66 ± 2.09	1447	**0.024**	1453.7
LEML [140]	92.00 ± 2.18	62.13 ± 3.13	60.92 ± 1.95	93	1.222	437.7
BCRML-full	92.00± 3.12	64.40 ± 2.92	60.58 ± 1.75	189	1.222	669.7
BCRML-100	92.25 ± 3.78	64.61 ± 2.65	**62.42 ± 2.14**	**45**	0.291	127
CDL-LDA [159]	**94.25 ± 3.36**	72.94 ± 1.81	N/A	-	1.073	302.7
LEML+CDL-LDA [140]	94.00 ± 3.57	73.01 ± 1.67	N/A	93	0.979	369
BCRML-100 +CDL-LDA	93.75 ± 3.58	**73.48 ± 1.83**	N/A	45	0.045	**57.7**

Table 5.4: Image set classification results for state-of-the-art metric learning algorithms. The last three columns report computation time in seconds. The last 3 rows report performance using CDL-LDA after dimensionality reduction. Methods using the proposed BCR are listed in bold.

5.5 Conclusion

We have presented a novel biconvex relaxation framework (BCR) that enables the solution of general semidefinite programs (SDPs) at low complexity and with a small memory footprint. We have provided an alternating minimization (AM) procedure along with a new initialization method that, together, are guaranteed to converge, computationally efficient (even for large-scale problems), and able to handle a variety of SDPs. Comparisons of BCR with state-of-the-art methods for specific computer vision problems, such as segmentation, co-segmentation, and metric learning, show that BCR provides similar or better solution quality with significantly lower runtime.

Acknowledgement: This work was done jointly in collaboration with Abhay Yadav.

Chapter 6: Stabilizing Adversarial Nets

6.1 Introduction

Adversarial networks play an important role in a variety of applications, including image generation [160, 161], style transfer [161–164], domain adaptation [163, 165, 166], imitation learning [167], privacy [168, 169], fair representation [168, 170], etc. One particularly motivating application of adversarial nets is their ability to form generative models, as opposed to the classical discriminative models [171–174].

While adversarial networks have the power to attack a wide range of previously unsolved problems, they suffer from a major flaw: they are difficult to train. This is because adversarial nets try to accomplish two objectives simultaneously; weights are adjusted to maximize performance on one task while minimizing performance on another. Mathematically, this corresponds to finding a *saddle point* of a loss function - a point that is minimal with respect to one set of weights, and maximal with respect to another.

Conventional neural networks are trained by marching down a loss function until a minimizer is reached (Figure 6.1(a)). In contrast, adversarial training methods search for saddle points rather than a minimizer, which introduces the possibility that the training path "slides off" the objective functions and the loss goes to $-\infty$ (Figure 6.1(b)), resulting in "collapse" of the adversarial network. As a result, many authors suggest using early

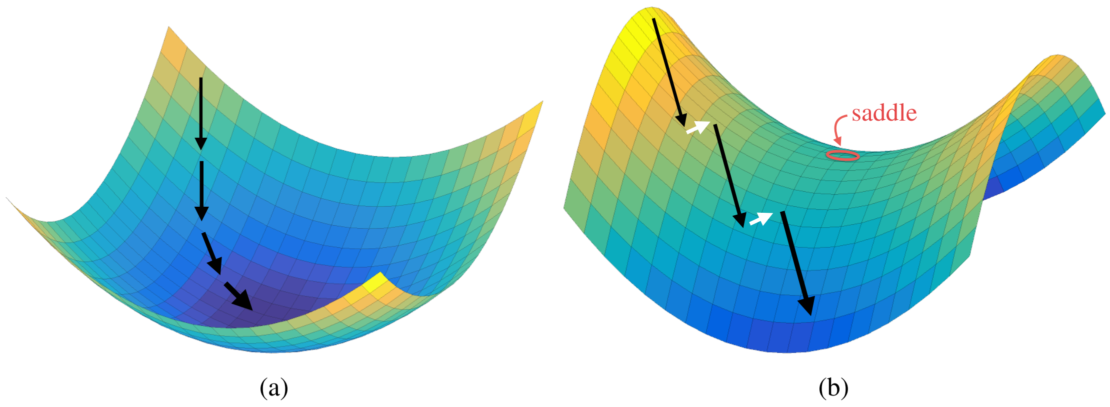


Figure 6.1: A schematic depiction of gradient methods. (a) Classical networks are trained by marching down the loss function until a minimizer is reached. Because classical loss functions are bounded from below, the solution path gets stopped when a minimizer is reached, and the gradient method remains stable. (b) Adversarial net loss functions may be unbounded from below, and training alternates between minimization and maximization steps. If minimization (or, conversely, maximization) is more powerful, the solution path "slides off" the loss surface and the algorithm becomes unstable, resulting in a sudden "collapse" of the network.

stopping, gradients/weight clipping [175], or specialized objective functions [171,175,176] to maintain stability.

In this chapter, we present a simple "prediction" step that is easily added to many training algorithms for adversarial nets. Finally, we use a wide range of experiments to show that prediction enables faster training of adversarial networks using large learning rates without the instability problems that plague conventional training schemes. The code is available at `https://github.com/shahsohil/stableGAN`.

6.2 Related Work

6.2.1 Adversarial Networks as a Saddle-Point Problem

We now discuss a few common adversarial network problems and their saddle-point formulations.

Generative Adversarial Networks (GANs) fit a generative model to a dataset using a game in which a generative model competes against a discriminator [171]. The generator, $\mathbf{G}(\mathbf{z};\theta_g)$, takes random noise vectors $\mathbf{z}$ as inputs, and maps them onto points in the target data distribution. The discriminator, $\mathbf{D}(\mathbf{x};\theta_d)$, accepts a candidate point $\mathbf{x}$ and tries to determine whether it is really drawn from the empirical distribution (in which case it outputs 1), or fabricated by the generator (output 0). During a training iteration, noise vectors from a Gaussian distribution $\mathcal{G}$ are pushed through the generator network $\mathbf{G}$ to form a batch of generated data samples denoted by $\mathcal{D}_{fake}$. A batch of empirical samples, $\mathcal{D}_{real}$, is also prepared. One then tries to adjust the weights of each network to solve a saddle point problem, which is popularly formulated as,

$$\min_{\theta_g} \max_{\theta_d} \quad \mathbb{E}_{x\sim\mathcal{D}_{real}}\, f(\mathbf{D}(\mathbf{x};\theta_d)) + \mathbb{E}_{z\sim\mathcal{G}}\, f(1-\mathbf{D}(\mathbf{G}(\mathbf{z};\theta_g);\theta_d)). \tag{6.1}$$

Here $f(.)$ is any monotonically increasing function. Initially, [171] proposed using $f(x) = \log(x)$.

Domain Adversarial Networks (DANs) [166, 168, 177] take data collected from a "source" domain, and extract a feature representation that can be used to train models that generalize to another "target" domain. For example, in the domain adversarial neural network (DANN [166]), a set of feature layers maps data points into an embedded feature space, and a classifier is trained on these embedded features. Meanwhile, the adversarial discriminator tries to determine, using only the embedded features, whether the data points belong to the source or target domain. A good embedding yields a better task-specific objective on the target domain while fooling the discriminator, and is found by solving

$$\min_{\theta_f, \theta_{y^k}} \max_{\theta_d} \quad \sum_k \alpha_k \mathcal{L}_{y^k} \left(\mathbf{x}_s; \theta_f, \theta_{y^k}\right) - \lambda \mathcal{L}_d \left(\mathbf{x}_s, \mathbf{x}_t; \theta_f, \theta_d\right). \tag{6.2}$$

Here $\mathcal{L}_d$ is any adversarial discriminator loss function and $\mathcal{L}_{y^k}$ denotes the task specific loss. θ_f, θ_d, and θ_{y^k} are network parameter of feature mapping, discriminator, and classification layers.

6.2.2 Stabilizing saddle point solvers

It is well known that alternating stochastic gradient methods are unstable when using simple logarithmic losses. This led researchers to explore multiple directions for stabilizing GANs; either by adding regularization terms [175, 176, 178, 179], a myriad of training "hacks" [180, 181], re-engineering network architectures [176], and designing different solvers [12]. Specifically, the Wasserstein GAN (WGAN) [175] approach modifies the original objective by replacing $f(x) = \log(x)$ with $f(x) = x$. This led to a training scheme in which the discriminator weights are "clipped." However, as discussed in [175], the WGAN training is unstable at high learning rates, or when used with popular momentum based solvers such as Adam. Currently, it is known to work well only with RMSProp [175].

The unrolled GAN [12] is a new solver that can stabilize training at the cost of more expensive gradient computations. Each generator update requires the computation of multiple extra discriminator updates, which are then discarded when the generator update is complete. While avoiding GAN collapse, this method requires increased computation and memory.

In the convex optimization literature, saddle point problems are more well studied.

One popular solver is the primal-dual hybrid gradient (PDHG) method [182, 183], which has been popularized by Chambolle and Pock [184], and has been successfully applied to a range of machine learning and statistical estimation problems [185]. PDHG relates closely to the method proposed here - it achieves stability using the same prediction step, although it uses a different type of gradient update and is only applicable to bi-linear problems.

Stochastic methods for convex saddle-point problems can be roughly divided into two categories: stochastic coordinate descent [186–192] and stochastic gradient descent [193, 194]. Similar optimization algorithms have been studied for reinforcement learning [195, 196]. Recently, a "doubly" stochastic method that randomizes both primal and dual updates was proposed for strongly convex bilinear saddle point problems [197]. For general saddle point problems, "doubly" stochastic gradient descent methods are discussed in [198], [199], in which primal and dual variables are updated simultaneously based on the previous iterates and the current gradients.

6.3 Model and Algorithm

As discussed above, saddle-point optimization problems have the general form

$$\min_u \max_v \mathcal{L}(u, v) \tag{6.3}$$

for some loss function $\mathcal{L}$ and variables u and v. Most authors use the alternating stochastic gradient method to solve saddle-point problems involving neural networks. This method alternates between updating u with a stochastic gradient *descent* step, and then updating v with a stochastic gradient *ascent* step. When simple/classical SGD updates are used, the steps of this method can be written

$$\begin{aligned} u^{k+1} &= u^k - \alpha_k \mathcal{L}'_u(u^k, v^k) && | \quad \text{gradient descent in } u\text{, starting at } (u^k, v^k) \\ v^{k+1} &= v^k + \beta_k \mathcal{L}'_v(u^{k+1}, v^k) && | \quad \text{gradient ascent in } v\text{, starting at } (u^{k+1}, v^k)\ . \end{aligned} \tag{6.4}$$

Here, $\{\alpha_k\}$ and $\{\beta_k\}$ are learning rate schedules for the minimization and maximization steps, respectively. The vectors $\mathcal{L}'_u(u, v)$ and $\mathcal{L}'_v(u, v)$ denote (possibly stochastic) gradients of $\mathcal{L}$ with respect to u and v. In practice, the gradient updates are often performed by an automated solver, such as the Adam optimizer [112], and include momentum updates.

We propose to stabilize the training of adversarial networks by adding a *prediction* step. Rather than calculating v^{k+1} using u^{k+1}, we first make a prediction, $\bar{u}^{k+1}$, about where the u iterates will be in the future, and use this predicted value to obtain v^{k+1}.

The proposed *Prediction SDG* method is given by (6.5).

Prediction Method

$$\begin{aligned} u^{k+1} &= u^k - \alpha_k \mathcal{L}'_u(u^k, v^k) && | \quad \text{gradient descent in } u\text{, starting at } (u^k, v^k) \\ \bar{u}^{k+1} &= u^{k+1} + (u^{k+1} - u^k) && | \quad \textit{predict} \text{ future value of } u \\ v^{k+1} &= v^k + \beta_k \mathcal{L}'_v(\bar{u}^{k+1}, v^k) && | \quad \text{gradient ascent in } v\text{, starting at } (\bar{u}^{k+1}, v^k)\ . \end{aligned} \tag{6.5}$$

The Prediction step (6.5) tries to estimate where u is going to be in the future by assuming its trajectory remains the same as in the current iteration.

6.4 Interpretations of the prediction step

We present three ways to explain the effect of prediction: an intuitive, non-mathematical perspective, a more analytical viewpoint involving dynamical systems, and finally a rigorous proof-based approach.

6.4.1 An intuitive viewpoint

The standard alternating SGD switches between minimization and maximization steps. In this algorithm, there is a risk that the minimization step can overpower the maximization step, in which case the iterates will "slide off" the edge of saddle, leading to instability (Figure 6.1(b)). Conversely, an overpowering maximization step will dominate the minimization step, and drive the iterates to extreme values as well.

The effect of prediction is visualized in Figure 6.2. Suppose that a maximization step takes place starting at the red dot. Without prediction, the maximization step has no knowledge of the algorithm history, and will be the same regardless of whether the previous minimization update was weak (Figure 6.2(a)) or strong (Figure 6.2(b)). Prediction allows the maximization step to exploit information about the minimization step. If the previous minimizations step was weak (Figure 6.2(a)), the prediction step (dotted black arrow) stays close to the red dot, resulting in a weak predictive maximization step (white arrow). But if we arrived at the red dot using a strong minimization step (Figure 6.2(b)), the prediction moves a long way down the loss surface, resulting in a stronger maximization step (white arrows) to compensate.

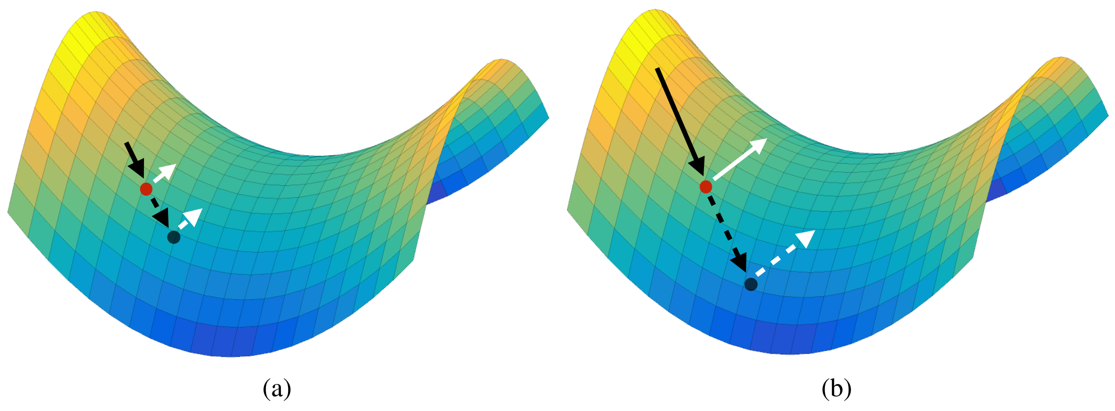


Figure 6.2: A schematic depiction of the prediction method. When the minimization step is powerful and moves the iterates a long distance, the prediction step (dotted black arrow) causes the maximization update to be calculated further down the loss surface, resulting in a more dramatic maximization update. In this way, prediction methods prevent the maximization step from getting overpowered by the minimization update.

6.4.2 A more mathematical perspective

To get stronger intuition about prediction methods, let's look at the behavior of Algorithm (6.5) on a simple bi-linear saddle of the form

$$\mathcal{L}(u, v) = v^T K u \tag{6.6}$$

where K is a matrix. When exact (non-stochastic) gradient updates are used, the iterates follow the path of a simple dynamical system with closed-form solutions. We give here a sketch of this argument: a detailed derivation is provided in the Supplementary Material.

When the (non-predictive) gradient method (6.4) is applied to the linear problem (6.6), the resulting iterations can be written

$$\frac{u^{k+1} - u^k}{\alpha} = -K^T v^k, \qquad \frac{v^{k+1} - v^k}{\alpha} = (\beta/\alpha) K u^{k+1}.$$

When the stepsize α gets small, this behaves like a discretization of the system of differential equations

$$\dot{u} = -K^T v, \qquad \dot{v} = \beta/\alpha K u$$

where $\dot{u}$ and $\dot{v}$ denote the derivatives of u and v with respect to time. These equations describe a simple harmonic oscillator, and the closed form solution for u is

$$u(t) = C \cos(\Sigma^{1/2} t + \phi)$$

where Σ is a diagonal matrix, and the matrix C and vector ϕ depend on the initialization. We can see that, for small values of α and β, the non-predictive algorithm (6.4) approximates an undamped harmonic motion, and the solutions orbit around the saddle without converging.

The prediction step (6.5) improves convergence because it produces *damped* harmonic motion that sinks into the saddle point. When applied to the linearized problem (6.6), we get the dynamical system

$$\dot{u} = -K^T v, \qquad \dot{v} = \beta/\alpha K(u + \alpha \dot{u}) \tag{6.7}$$

which has solution

$$u(t) = UA \exp(-\frac{t\alpha}{2}\sqrt{\Sigma}) \sin(t\sqrt{(1-\alpha^2/4)\Sigma} + \phi).$$

From this analysis, we see that the damping caused by the prediction step causes the orbits to converge into the saddle point, and the error decays exponentially fast.

6.4.3 A rigorous perspective

While the arguments above are intuitive, they are also informal and do not address issues like stochastic gradients, non-constant stepsize sequences, and more complex loss functions. We now provide a rigorous convergence analysis that handles these issues.

We assume that the function $\mathcal{L}(u, v)$ is convex in u and concave in v. We can then measure convergence using the "primal-dual" gap, $P(u, v) = \mathcal{L}(u, v^\star) - \mathcal{L}(u^\star, v)$ where $(u^\star, v^\star)$ is a saddle. Note that $P(u, v) > 0$ for non-optimal (u, v), and $P(u, v) = 0$ if (u, v) is a saddle. Using these definitions, we formulate the following convergence result. The proof for the same can be found in [11].

Theorem 1 *Suppose the function $\mathcal{L}(u, v)$ is convex in u, concave in v, and that the partial gradient $\mathcal{L}'_v$ is uniformly Lipschitz smooth in u ($\|\mathcal{L}'_v(u_1, v) - \mathcal{L}'_v(u_2, v)\| \leq L_v\|u_1 - u_2\|$). Suppose further that the stochastic gradient approximations satisfy $\mathbb{E}\|\mathcal{L}'_u(u, v)\|^2 \leq G_u^2$, $\mathbb{E}\|\mathcal{L}'_v(u, v)\|^2 \leq G_v^2$ for scalars G_u and G_v, and that $\mathbb{E}\|u^k - u^\star\|^2 \leq D_u^2$, and $\mathbb{E}\|v^k - v^\star\|^2 \leq D_v^2$ for scalars D_u and D_v.*

If we choose decreasing learning rate parameters of the form $\alpha_k = \frac{C_\alpha}{\sqrt{k}}$ and $\beta_k = \frac{C_\beta}{\sqrt{k}}$, then the SGD method with prediction converges in expectation, and we have the error bound

$$\mathbb{E}[P(\hat{u}^l, \hat{v}^l)] \leq \frac{1}{2\sqrt{l}}\left(\frac{D_u^2}{C_\alpha} + \frac{D_v^2}{C_\beta}\right) + \frac{\sqrt{l+1}}{l}\left(\frac{C_\alpha G_u^2}{2} + C_\alpha L_v G_u^2 + C_\alpha L_v D_v^2 + \frac{C_\beta G_v^2}{2}\right)$$

where $\hat{u}^l = \frac{1}{l}\sum_{k=1}^{l} u^k$, $\hat{v}^l = \frac{1}{l}\sum_{k=1}^{l} v^k$.

6.5 Experiments

We present a wide range of experiments to demonstrate the benefits of the proposed prediction step for adversarial nets. We consider a saddle point problem on a toy dataset constructed using MNIST images, and then move on to consider state-of-the-art models for three tasks: GANs, domain adaptation, and learning of fair classifiers.

6.5.1 MNIST Toy problem

We consider a classic MNIST digits dataset [200] consisting of 60,000 training images and 10,000 testing images each of size 28×28. For simplicity, let us consider a task (T1) of classifying into odd and even numbered images. Let's say, that $\sim 50\%$ of data instances were corrupted using salt and pepper noise of probability 0.2 and this distortion process was biased. Specifically, only 30% of even numbered images were distorted as against the 70% of odd-numbered images. We have observed that any feature representation network θ_f (for example, LeNet network [200]) trained using the binary classification loss function for task T1 also encodes noise bias within it. However, when a noise vs no-noise classifier is trained on the deep features generated by LeNet, it gets 100% accuracy.

The goal of this task is to force LeNet to ignore the noise when making decisions. This lead us to design of simple adversarial network to "unlearn" the noise bias from the feature learning pipeline. We create an adversarial model of the form (6.2) in which $\mathcal{L}_d$ is a softmax loss for the task (T2) of classifying whether the input sample is noisy or not. $\mathcal{L}_y$ is a softmax loss for task T1 and $\lambda = 1$. A LeNet network [200] is considered for training on task T1 while a two-layer MLP is used for training on task T2. LeNet consist of two convolutional (conv) layers followed by two fully connected (FC) layers at the top. The parameters of conv layers form θ_f while that of FC and MLP layers forms θ_y and θ_d respectively. We train the network in three stages. Following the training on task T1, θ_f were fixed and MLP is trained independently on task T2. The default learning schedule of the LeNet implementation in Caffe [201] were followed for both the tasks. The total training iterations on each task were set to $10,000$. After pretraining, the whole network is jointly finetuned using the adversarial approach. (6.2) is alternatively minimized w.r.t. $\theta_{\mathbf{f}}, \theta_{\mathbf{y}}$ and maximized w.r.t. $\theta_{\mathbf{d}}$. The predictive steps were only used during the finetuning phase.

Figure 6.3 summarizes our findings. In this experiment, we considered applying prediction to both the classifier and discriminator. We note that our task is to retain good classification accuracy while preventing the discriminator from doing better than the trivial strategy of classifying odd digits as noisy and even digits as non-noisy. This means that the discriminator accuracy should ideally be ~ 0.7. As shown in Figure 6.3(a), the prediction step hardly makes any difference when evaluated at the small learning rate of 10^{-4}. However, when evaluated at higher rates, Figures 6.3(b) and 6.3(c) show that the prediction solvers are very stable while one without prediction collapses (blue solid line is flat) very early. Figure 6.3(c) shows that the default learning rate (10^{-3}) of the Adam solver is unstable unless prediction is used. Finally, Figure 6.3(d) provides head-to-head comparison of two popular solvers Adam and SGD using the predictive step. Not surprisingly, the Adam solver shows relatively better performance and convergence even with an additional predictive step. This also suggests that the default hyper-parameter for the Adam solver can be retained and utilized for training this networks without resorting to any further hyper-parameter tuning (as it is currently in practice).

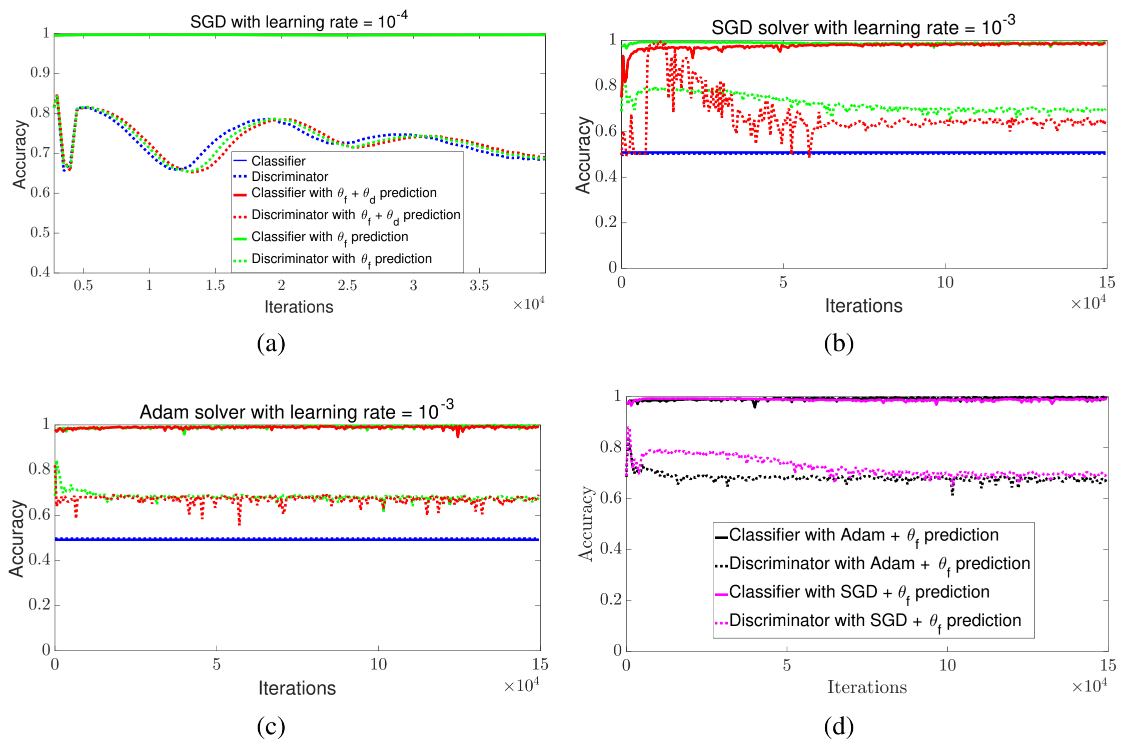


Figure 6.3: Comparison of the classification accuracy (digit parity) and discriminator (noisy vs. no-noise) accuracy using SGD and Adam solver with and without prediction steps. θ_f and θ_d refers to variables in eq. (6.2). (a) Using SGD with learning rate $lr = 10^{-4}$. Note that the solid lines of red, blue and green are overlapped. (b) SGD solver with higher learning rate of $lr = 10^{-3}$, (c) using Adam solver with its default parameter and (d) Comparison of the classification accuracy of parity classification and noise discrimination using the SGD and Adam solvers with and without prediction step.

6.5.2 Domain Adaptation

Next, we consider the domain adaptation task [165, 166, 202] wherein the representation learned using the source domain samples is altered so that it can also generalize to samples from the target distribution. We use the problem setup and hyper-parameters as described in [166]using the OFFICE dataset [202]. OFFICE is a small scale dataset consisting of images collected from three distinct domains: AMAZON, DSLR and WEBCAM. For such a small scale dataset, it is non-trivial to learn features from images of a single domain. For instance, consider the largest subset AMAZON, which contains only 2,817 labeled images spread across 31 different categories. However, one can leverage the power of domain adaptation to improve cross domain accuracy. We follow the protocol listed in [166] and the same network architecture is used. Caffe [201] is used for implementation. The training procedure from [166] is kept intact except for the additional prediction step.

In Table 6.1, comparisons are drawn with respect to target domain accuracy on six pairs of source-target domain tasks. The test accuracy is reported at the end of 50,000 training iterations. We observe that the prediction step has mild benefits on the "easy" adaptation tasks with very similar source and target domain samples. However, on the transfer learning tasks of AMAZON-to-WEBCAM, WEBCAM-to-AMAZON, and DSLR-to-AMAZON which has noticeably distinct data samples, an extra prediction step gives an absolute improvement of $1.3 - 6.9\%$ in predicting target domain labels.

Method	Source	AMAZON	WEBCAM	DSLR	WEBCAM	AMAZON	DSLR
	Target	WEBCAM	AMAZON	WEBCAM	DSLR	DSLR	AMAZON
DANN [166]		73.4	51.6	95.5	**99.4**	**76.5**	51.7
DANN + prediction		**74.7**	**58.5**	**96.1**	99.0	73.5	**57.6**

Table 6.1: Comparison of target domain accuracy on OFFICE dataset.

6.5.3 Fair Classifier

Finally, we consider a task of learning fair feature representations [168, 170, 203] such that the final learned classifier does not discriminate with respect to a sensitive variable. As proposed in [168] one way to measure fairness is using discrimination,

$$y_{disc} = \left| \frac{1}{N_0} \sum_{i:s_i=0} \eta(x_i) - \frac{1}{N_1} \sum_{i:s_i=1} \eta(x_i) \right| . \tag{6.8}$$

Here s_i is a binary sensitive variable for the i^{th} data sample and N_k denotes the total number of samples belonging to the k^{th} sensitive class. Similar to the domain adaptation task, the learning of each classifier can be formulated as a minimax problem in (6.2) [168, 170]. Unlike the previous example though, this task has a model selection component. From a pool of hundreds of randomly generated adversarial deep nets, for each value of t, one selects the model that maximizes the difference

$$y_{t,Delta} = y_{acc} - t * y_{disc}. \tag{6.9}$$

The "Adult" dataset from the UCI machine learning repository is used, which consists of census data from $\sim 45,000$ people. The task (y_{acc}) is to classify whether a person earns $\geq \$50k$/year. The person's gender is chosen to be the sensitive variable. We binarize all the category attributes, giving us a total of 102 input features per sample. We randomly split data into 35,000 samples for training, 5000 for validation and 5000 for testing. The result reported here is an average over five such random splits. To demonstrate the advantage of using prediction for model selection, we follow the protocol developed in [168]. In this work, the search space is restricted to a class of models that consist of a fully connected autoencoder, one task specific discriminator, and one adversarial discriminator. The encoder output from the autoencoder acts as input to both the discriminators. In our experiment, 100 models are randomly selected. During the training of each adversarial model, $\mathcal{L}_d$ is a cross-entropy loss while $\mathcal{L}_y$ is a linear combination of reconstruction and cross-entropy loss. Once all the models are trained, the best model for each value of t is selected by evaluating (6.9) on the validation set.

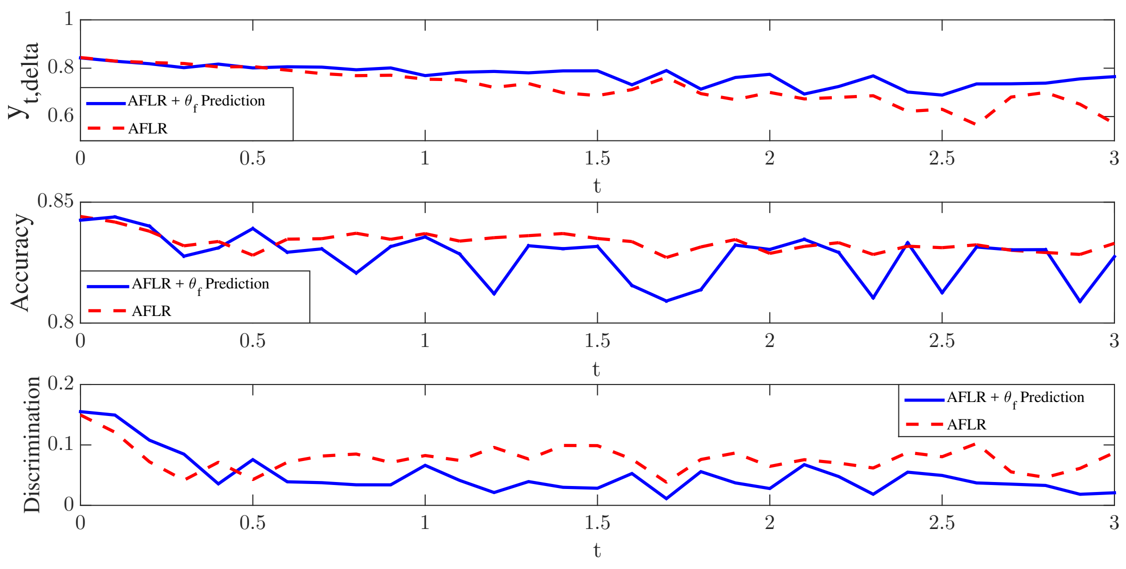
Figure 6.4(a), 6.4(b) respectively plots the results on the test and validation set for the AFLR approach with and without prediction steps in their default Adam solver. For each value of t, Figure 6.4(c), 6.4(d) also compares the number of layers in the selected encoder and discriminator networks. When using prediction for training, relatively stronger encoder models are produced and selected during validation, and hence the prediction results generalize better on the test set.

6.5.4 Generative Adversarial Networks

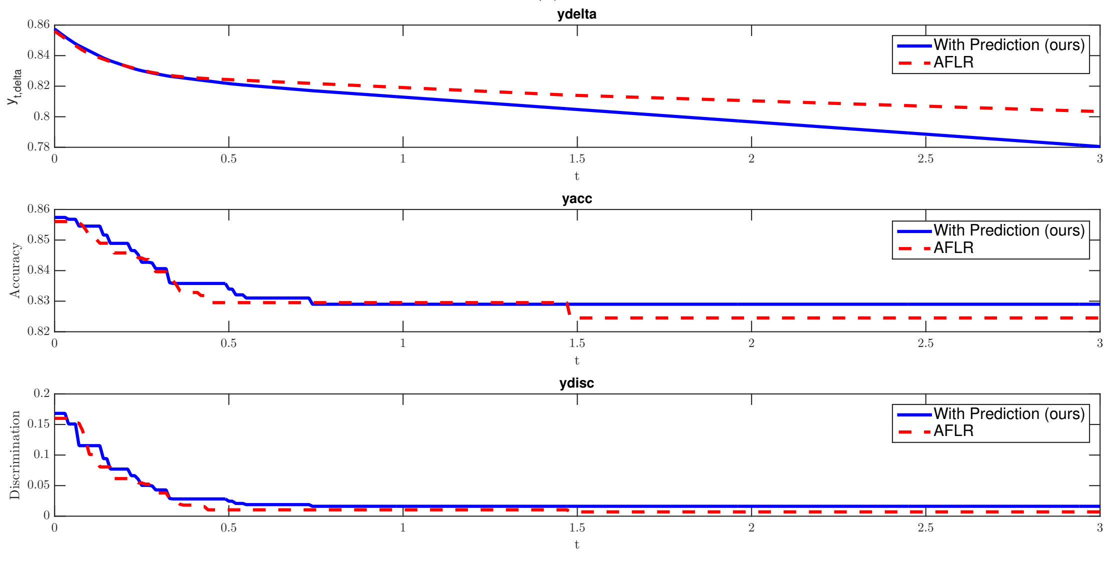
Next, we test the efficacy and stability of our proposed predictive step on generative adversarial networks (GAN), which are formulated as saddle point problems (6.1) and are popularly solved using a heuristic approach [171]. We consider an image modeling task using CIFAR-10 [204] on the recently popular convolutional GAN architecture, DCGAN [172]. We compare our predictive method with that of DCGAN and the unrolled GAN [12] using the training protocol described in [172]. Note that we compared against the unrolled GAN with stop gradient switch[1] and $K = 5$ unrolling steps. All the approaches were trained for five random seeds and 100 epochs each.

We start with comparing all three methods using the default solver for DCGAN (the Adam optimizer) with learning rate=0.0002 and β_1=0.5. Figure 6.5 compares the generated sample images (at the 100^{th} epoch) and the training loss curve for all approaches. The discriminator and generator loss curves in Figure 6.5(e) show that without prediction, the DCGAN collapses at the 45^{th} and 57^{th} epochs. Similarly, Figure 6.5(f) shows that the training for unrolled GAN collapses in at least three instances. The training procedure using

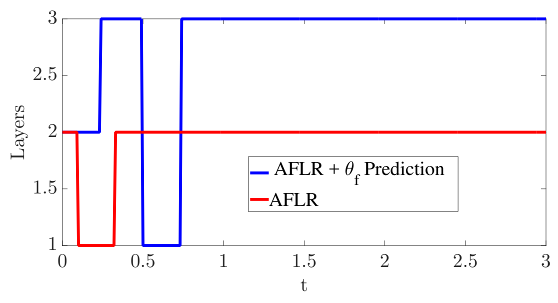
[1]We found the unrolled GAN without stop gradient switch as well as for smaller values of K collapsed when used on the DCGAN architecture.



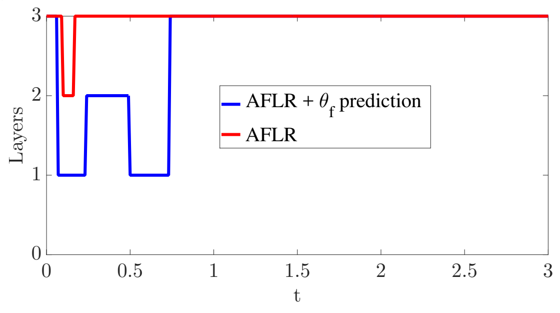
(a)



(b)



(c)



(d)

Figure 6.4: Model selection for learning a fair classifier. (a) Comparison of $y_{t,delta}$ (higher is better), and also y_{disc} (lower is better) and y_{acc} on the test set using AFLR with and without predictive steps. (b) On validation set. (c) Number of encoder layers in the selected model. (d) Number of discriminator layers (both adversarial and task-specific) in the selected model.

predictive steps never collapsed during any epochs. Qualitatively, the images generated using prediction are more diverse than the DCGAN and unrolled GAN images.

Figure 6.6 compares all approaches when trained with $5\times$ higher learning rate (0.001) (the default for the Adam solver). As observed in [172], the standard and unrolled solvers are very unstable and collapse at this higher rate. However, as shown in Figure 6.6(d), & 6.6(a), training remains stable when a predictive step is used, and generates images of reasonable quality. The training procedure for both DCGAN and unrolled GAN collapsed on all five random seeds. The results on various additional intermediate learning rates using DCGAN as well as on high resolution Imagenet dataset using ACGAN [14] can be found in [11]. Overall, of the 25 training settings we ran on (each of five learning rates for five random seeds), the DCGAN training procedure collapsed in 20 such instances while unrolled GAN collapsed in 14 experiments (not counting the multiple collapse in each training setting). On the contrary, we find that our simple predictive step method collapsed only once.

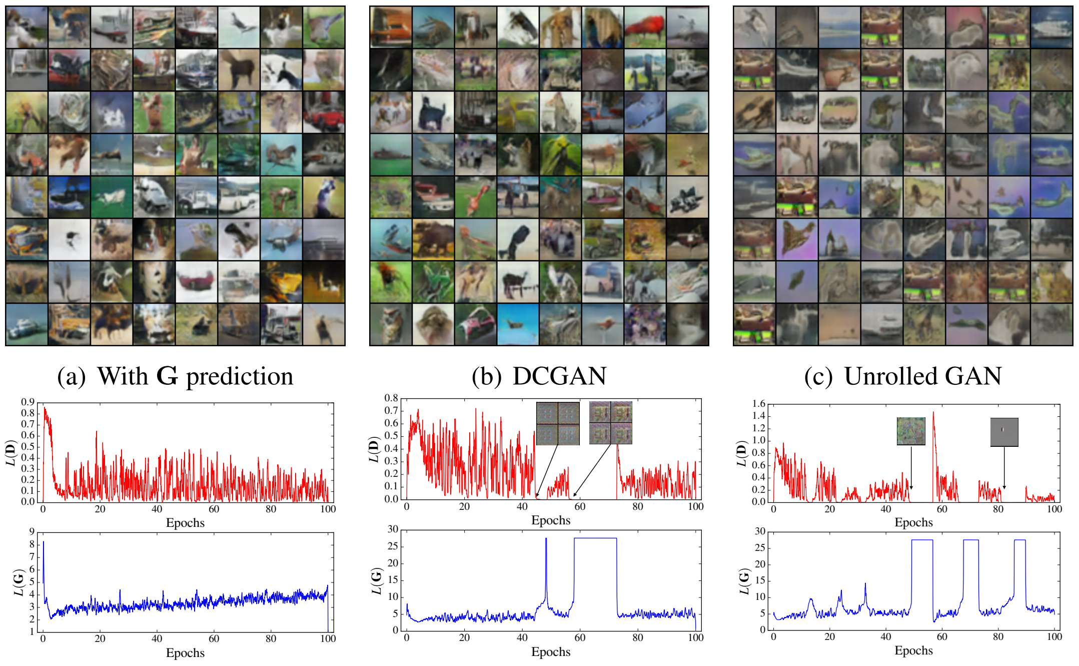
Note that prediction adds trivial cost to the training algorithm. Using a single TitanX Pascal, a training epoch of DCGAN takes 35 secs. With prediction, an epoch takes 38 secs. The unrolled GAN method, which requires extra gradient steps, takes 139 secs/epoch.

Finally, we draw quantitative comparisons based on the inception score [180], which is a widely used metric for visual quality of the generated images. For this purpose, we consider the current state-of-the-art Stacked GAN [13] architecture. Table 6.2 lists the inception scores computed on the generated samples from Stacked GAN trained $(200$ epochs) with and without prediction at different learning rates. The joint training of Stacked GAN collapses when trained at the default learning rate of adam solver (i.e., 0.001).

However, reasonably good samples are generated if the same is trained with prediction on both the generator networks. The right end of Table 6.2 also list the inception score measured at fewer number of epochs for higher learning rates. It suggest that the model trained with prediction methods are not only stable but also allows faster convergence using higher learning rates. For reference the inception score on real images of CIFAR-10 dataset is 11.51 ± 0.17.

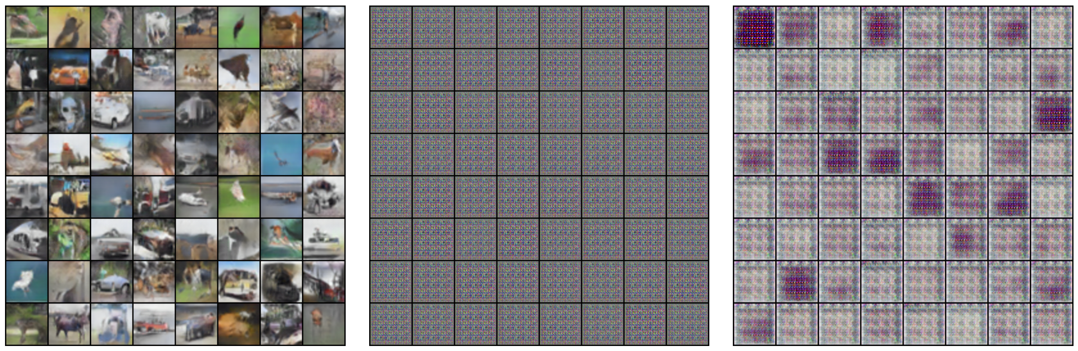
Learning rate	0.0001	0.0005	0.001	0.0005 (40)	0.001 (20)
Stacked GAN	8.44 ± 0.11	7.90 ± 0.08	1.52 ± 0.01	5.80 ± 0.15	1.42 ± 0.01
Stacked GAN + prediction	$\mathbf{8.55} \pm 0.12$	$\mathbf{8.13} \pm 0.09$	$\mathbf{7.96} \pm 0.11$	$\mathbf{8.10} \pm 0.10$	$\mathbf{7.79} \pm 0.07$

Table 6.2: Comparison of Inception Score on Stacked GAN network with and w/o **G** prediction.

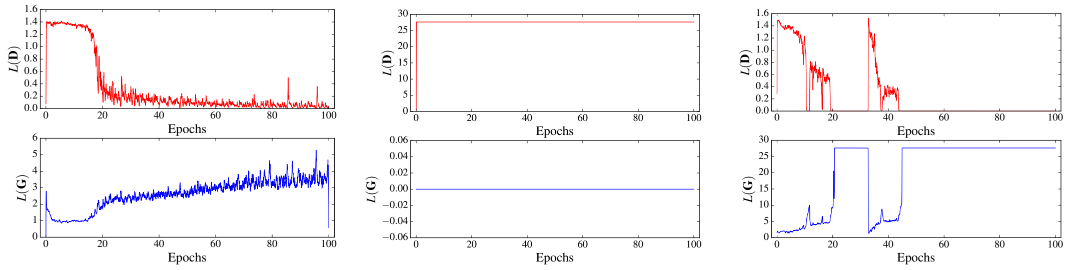


(d) With **G** prediction (e) DCGAN (f) Unrolled GAN

Figure 6.5: Comparison of GAN training algorithms for DCGAN architecture on Cifar-10 image datasets. Using default parameters of DCGAN; $lr = 0.0002, \beta_1 = 0.5$.



(a) With **G** prediction (b) DCGAN (c) Unrolled GAN



(d) With **G** prediction (e) DCGAN (f) Unrolled GAN

Figure 6.6: Comparison of GAN training algorithms for DCGAN architecture on Cifar-10 image datasets at higher learning rate, $lr = 0.001, \beta_1 = 0.5$.

6.6 Conclusion

We present a simple modification to the alternating SGD method, called a prediction step, that improves the stability of adversarial networks. We present theoretical results showing that the prediction step is asymptotically stable for solving saddle point problems. We show, using a variety of test problems, that prediction steps prevent network collapse and enable training with a wider range of learning rates than plain SGD methods.

Acknowledgement: This work was done jointly in collaboration with Abhay Yadav.

Chapter 7: Stacked U-Nets

7.1 Introduction

Semantic segmentation methods decompose an image into groups of pixels, each representing a common object class. While the output of a segmentation contains object labels assigned at the local (pixel) level, each label much have a global *field of view*; each such label depends on global information about the image, such as textures, colors, and object boundaries that may span large chunks of the image. Simple image classification algorithms consolidate global information by successively pooling features until the final output is a single label containing information from the entire image. In contrast, segmentation methods must output a full-resolution labeled image (rather than a single label). Thus, a successful segmentation method must address this key question: how can we learn long-distance contextual information while at the same time retaining high spatial resolution at the output for identifying small objects and sharp boundaries?

For natural image processing, most research has answered this question using one of two approaches. One approach is to use very few pooling layers, thus maintaining resolution (although methods may still require a small number of deconvolution layers [205–209]). Large fields of view are achieved using dilated convolutions, which span large regions. By maintaining resolution at each layer, this approach preserves substantial amounts of signal about smaller and less salient objects. However, this is achieved at the cost of computationally expensive and memory exhaustive training/inference. The second related approach is to produce auxiliary context aggregation blocks [1, 210–218] that contain features at different distance scales, and then merge these blocks to produce a final segmentation. This category includes many well-known techniques such as dense CRF [1] (conditional random fields) and spatial pyramid pooling [210].

These approaches suffer from the following challenges:

1. Deconvolutional (i.e., encoder-decoder) architectures perform significant nonlinear computation at low resolutions, but do very little processing of high-resolution features. During the convolution/encoding stage, pooling layers can move information over large distances, but information about small objects is often lost. During the deconvolution/decoding stage, high- and low-resolution features are merged to produce upsampled feature maps. However, the high-resolution inputs to deconvolutional layers come from relatively shallow layers that do not effectively encode semantic information.

2. Image classification networks are parameter heavy (44.5M parameters for ResNet-101), and segmentation methods built on top of these classification networks are often even more burdensome. For example, on top of the resnet-101 architecture, PSP-Net [217] uses 22M *additional* parameters for context aggregation, while the ASPP and Cascade versions of the Deeplab network utilize 14.5M [210] and 40M [218] *additional* parameters, respectively.

A popular and simple approach to segmentation is u-nets, which perform a chain of convolutional/downsampling operations, followed by a chain of deconvolutional/upsampling layers that see information from both low- and high-resolution scales. These u-net architectures are state-of-the art for medical image segmentation [219], but they do not perform well when confronted with the complex color profiles, lighting effects, perspectives, and occlusions present in natural images.

In this chapter, we expand the power of u-nets by stacking u-net blocks into deep architectures. This addresses the two challenges discussed above: As data passes through multiple u-net blocks, high-resolution features are mixed with low-resolution context information and processed through many layers to produce informative high-resolution features. Furthermore, stacked U-net models require fewer feature maps per layer than conventional architectures, and thus achieve higher performance with far fewer parameters. Our smallest model exceeds the performance of ResNet-101 on the PASCAL VOC 2012 semantic segmentation task by 4.5% mIoU, while having $\sim 7\times$ fewer parameters. The code is available at `https://github.com/shahsohil/sunets`.

7.2 Related Work

Many models [205, 207, 209, 210, 217, 218, 220, 221] have boosted the performance of semantic segmentation networks. These gains are mainly attributed to the use of pre-trained models, dilated convolutional layers [210, 216] and fully convolutional architectures (DCNN) [222]. These works employ a range of strategies to tap contextual information, which fall into three major categories.

Context Aggregation Modules: These architectures place a special module on top of a pre-trained network that integrates context information at different distance scales. The development of fast and efficient algorithm for DenseCRF [1] led to the development of numerous algorithms [210–213] incorporating it on top of the output belief map. Moreover, the joint training of CRF and CNN parameters was made possible by [214, 215]. In [216], context information was integrated by processing a belief map using a cascade of dilated layers operating at progressively increasing dilation rates, and [221] proposed a hybrid dilation convolution framework to alleviate gridding artifacts. ParseNet [223] exploits image-level feature information at each layer to learn global contextual information. In contrast, [210, 217, 218] realized substantial performance improvements by employing parallel layers of spatial pyramid pooling. The work [217] spatially pools output feature maps at different scales, while [210, 218] advocates applying dilated convolution at varying dilation rates.

Image Pyramid: The networks proposed in [224, 225] learn context information by simultaneously processing inputs at different scales and merging the output from all scales. An attention mechanism was used to perform fusion of output maps in [225], while [224] concatenates all the feature maps produced by blocks of parallel layers, each learned exclusively for differently scaled inputs. Recently, [226] developed a deformable network that adaptively determines an object's scale and accordingly adjusts the receptive field size of each activation function. On the other hand, [227] proposed a new training paradigm for object detection networks that trains each object instance only using the proposals closest to the ground truth scale.

Encoder-Decoder: These models consist of an encoder network and one or many blocks of decoder layers. The decoder fine-tunes the pixel-level labels by merging the contextual information from feature maps learned at all the intermediate layers. Usually, a popular bottom-up pre-trained classification network such as ResNet [228], VGG [229] or DenseNet [230] serves as an encoder model. U-net [219] popularly employs skip connections between an encoder and its corresponding decoding layers. On a similar note, the decoder in Segnet [231] upsamples the lower resolution maps by reusing the pooling indices from the encoder. Deconvolution layers were stacked in [206–208], whereas [209] uses a Laplacian pyramid reconstruction network to selectively refine the low resolution maps. Refinenet [205] employs sophisticated decoder modules at each scale on top of the ResNet encoder, while [232] utilizes a simple two-level decoding of feature maps from the Xception network [233]. In short, the structure in [209, 232] is a hybrid of decoding and context aggregation modules. Any task that requires extracting multi-scale information from inputs can benefit from an encoder-decoder structure. The recent works on object detection [234, 235] also utilize this structure.

7.2.1 Use of Pre-Trained Nets

Many of the networks described above make extensive use of image classification networks that were pre-trained for other purposes. ResNet employs an identity mapping [236] which, along with batch-normalization layers, facilitates efficient learning of very deep models. VGG was popular before the advent of ResNet. Although parameter heavy, much fundamental work on segmentation (FCN [222], dilated nets [216], u-nets [219] and CRF [214]) was built on VGG. All these architectures share common origins in that they were designed for the ImageNet competition and features are processed bottom-up. This prototype works well when the network has to identify only a single object without exact pixel localization. However, when extended to localization tasks such as segmentation and object detection, it is not clear whether the complete potential of these networks has been properly tapped. Recent work on object detection [227] also echoes a similar concern.

7.3 U-Nets Revisited

The original u-net architecture was introduced in [219], and produced almost perfect segmentation of cells in biomedical images using very little training data. The structure of u-nets makes it possible to capture context information at multiple scales and propagate them to the higher resolution layers. These higher order features have enabled u-nets to outperform previous deep models [222] on various tasks including semantic segmentation [208, 237, 238], depth-fusion [239], image translation [240] and human-pose estimation [241]. Moreover, driven by the initial success of u-nets, many recent works on semantic segmentation [205–209, 231] and object detection [234, 235] also propose an encoder-decoder deep architecture.

The u-net architecture evenly distributes its capacity among the encoder and decoder modules. Moreover, the complete network can be trained in an end-to-end setting. In contrast, the more recent architectures reviewed in Section 7.2 do not equally distribute the processing of top-down and bottom-up features. Since these architectures are built on top of pre-trained feature extractors [228, 229, 233], the decoder modules are trained separately and sometimes in multiple stages. To overcome these drawbacks, [208] proposed an equivalent u-net based on the Densenet [230] architecture. However, DenseNet is memory intensive, and adding additional decoder layers leads to a further increase in memory usage.

Given these drawbacks, the effectiveness of these architectures on different datasets and applications is unclear.

The goal of this work is to realize the benefits of u-nets (small size, easy trainability, high performance) for complex natural image segmentation problems.

Specifically, we propose a new architecture composed of multiple stacks of u-nets. The network executes repeated processing of both top-down as well as bottom-up features and captures long-distance spatial information at multiple resolutions. The network is trained end-to-end on image classification tasks and can be seamlessly applied to semantic segmentation without any additional modules on top (except for replacing the final classifier).

Our stacked u-net (SUNet) architecture shares some similarity with other related stacked encoder-decoder structures [207, 241]. Fu *et al.* [207] uses multiple stacks of de-convolutional networks (maximum of three) on top of a powerful encoder (DenseNet) while [241] applies multiple stacks of u-net modules for human-pose estimation. However, the processing of features inside each u-net module in [241] differs from ours. [241] replaces each convolutional block with a residual module and utilizes nearest-neighbor up-sampling for deconvolution. In contrast, SUNets retain the basic u-net structure from [219]. Also, SUNet operates without any intermediate supervision and processes features by progressively downsampling while [241] operates at fix resolution.

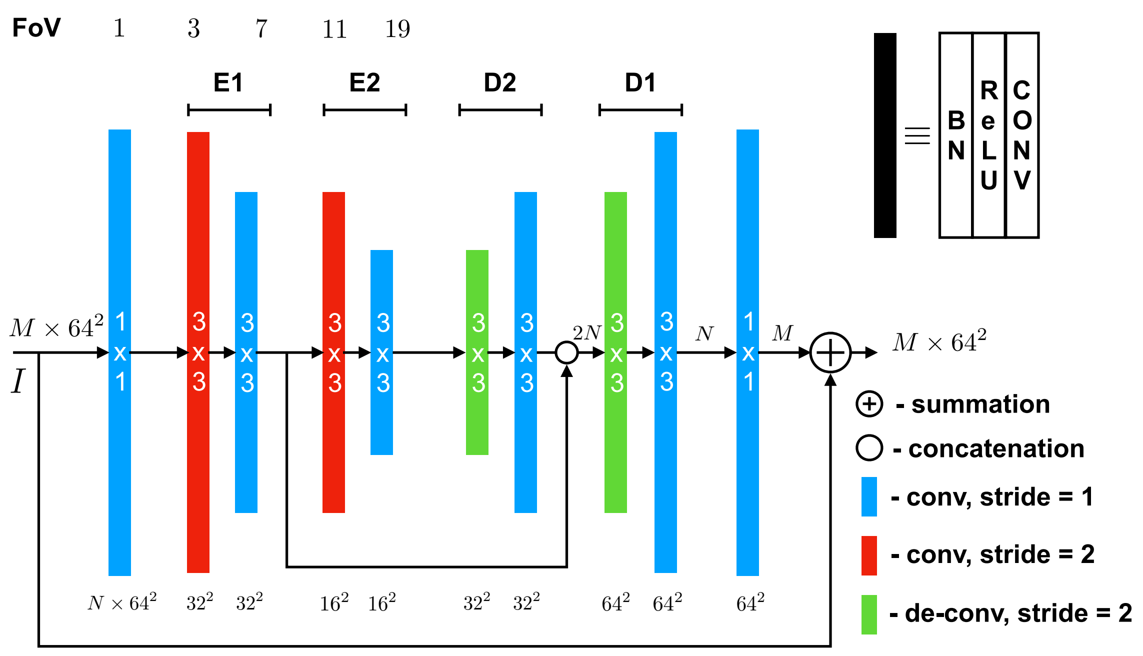


Figure 7.1: A typical u-net module with outer residual connection. M is the number of input features. Across the u-net module, each layer has the same number of output feature maps (except for the final 1×1 filter), which we denote N. For better understanding the figure also includes the field of view (FoV) of each convolutional kernel (top) and the feature map size at the output of each filter (bottom), assuming a 64×64 input I. Best viewed in color.

7.3.1 U-Net Module Implementation

Figure 7.1 illustrates the design of the u-net module employed in our stacked architecture. Each module is composed of 10 pre-activated convolutional blocks each preceded by a batch-normalization and a ReLU non-linearity. The pooling/unpooling operation, handled by the strided convolutional/deconvolutional layers, facilitates information exchange between the lower and the higher resolution features. A skip connection branches off at the output of the first encoder block, $E1$. Following this, the $E2$ and $D2$ blocks capture long distance context information using lower resolution feature maps and merge the information back with the high resolution features from $E1$ at the output of $D2$. Every layer (except for the bottleneck layers) uses 3×3 kernels, and outputs a fixed number of feature maps, N. To mitigate high frequency noise from the sampling operation, each strided conv/de-conv layer is followed by a convolution. Unlike traditional u-nets, the design of convolutional layers in our u-net module helps in retaining the original size of the feature maps at its output (no crop operation takes place). Consequently, multiple u-net modules can be stacked without loosing resolution.

In the following, we briefly highlight some of the design choices of the architecture. In comparison to traditional u-nets, the max-pooling operation is replaced with strided convolution for SUNets. The use of strided convolutions enables different filters in each u-net module to operate at different resolutions (see the discussion in Section 7.5). Moreover, the repeated use of max-pooling operations can cause gridding artifacts in dilated networks [242].

Unlike the u-nets of [219, 241], our u-net module is comprised of only two levels of depth. We considered two major factors in choosing the depth: field of view (FoV) of the innermost conv filter and the total number of parameters in a single u-net module. The number of parameters influences the total number of stacks in SUNets. While keeping the total parameters of SUNet approximately constant, we experimented with a higher depth of three and four. We found that the increase in depth indeed led to a decline in performance for the image classification task. This may not be surprising, given that a SUNet with depth of two is able to stack more u-net modules. Moreover, deeper u-net modules make it harder to train the inner-most convolutional layers due to the vanishing gradients problem [243]. For instance, in our current design, the maximum length of the gradient path is six. The popular classification networks [228, 230] are known to operate primarily on features with 28^2 and 14^2 resolution. At this scale, the effective FoV of 19 is more than sufficient to capture long-distance contextual information. Moreover, the stacking of multiple u-net modules will also serve to increase the effective FoV of higher layers.

SUNets train best when there is sufficient gradient flow to the bottom-most u-net layers. To avoid vanishing gradients, we include a skip connection [228, 230] around each u-net module. Also, inspired by the design of bottleneck blocks [228], we also include 1×1 convolutional layers. Bottleneck layers restricts the number of input features to a small number (N), avoiding parameter inflation.

When stacking multiple u-nets it makes sense for each u-net module to reuse the raw feature maps from all the preceding u-net modules. Thus we also explored replacing the identity connection with dense connectivity [230]. This new network is memory intensive[1] which in turn prevented proper learning of the batch-norm parameters. Instead, we chose to utilize dense connectivity only within each u-net, i.e., while reusing feature maps from $E1$ at $D1$. Thus the proposed u-net module leverages skip connectivity without getting burdened.

7.4 SUNets: Stacked U-Nets for Classification

Before addressing segmentation, we describe a stacked u-net (SUNet) architecture that is appropriate for image classification. Because the amount of labeled data available for classification is much larger than for segmentation, classification tasks are often used

[1]Restricted by the current implementation of deep learning packages

Layers	Output Size	SUNet-64	SUNet-128	SUNet-7-128
Convolution	112×112	7×7 conv, 64, stride 2		
Residual Block	56×56	$\begin{bmatrix} 3 \times 3 \text{ conv, 128, stride 2} \\ 3 \times 3 \text{ conv, 128, stride 1} \end{bmatrix} \times 1$		
UNets Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv, 64} \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, 256} \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 512} \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 512} \end{bmatrix} \times 2$
Transition Layer	28×28	2×2 average pool, stride 2		
UNets Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv, 64} \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, 512} \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 1024} \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 1280} \end{bmatrix} \times 7$
Transition Layer	14×14	2×2 average pool, stride 2		
UNets Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv, 64} \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, 768} \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 1536} \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, 2048} \end{bmatrix} \times 7$
Transition Layer	7×7	2×2 average pool, stride 2		
UNets Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv, 64} \\ \text{U-Net}^{+}\text{, N=64} \\ 1 \times 1 \text{ conv, 1024} \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net}^{+}\text{, N=128} \\ 1 \times 1 \text{ conv, 2048} \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv, 128} \\ \text{U-Net}^{+}\text{, N=128} \\ 1 \times 1 \text{ conv, 2304} \end{bmatrix} \times 1$
Classification Layer	1×1	7×7 global average pool		
		1000D fully-connected, softmax		
Total Layers		110	110	170
Params		$6.9M$	$24.6M$	$37.7M$

Table 7.1: SUNet architectures for ImageNet. N denotes the number of filters per convolutional layer. Note that the building block in bracket refers to the integrated u-net module shown in Figure 7.1.

to pre-train feature extraction networks, which are then adapted to perform segmentation.

The network design of SUNets for ImageNet classification is summarized in Table 7.1. Note that each "conv" layer shown in the table corresponds to a sequence of "BN-ReLU-Conv" layers. The three listed configurations mainly differ in the number of output feature maps N of each convolutional layer and the total number of stacks in blocks 2 and 3. Input images are processed using a 7×7 conv filter followed by a residual block. Inspired by the work on dilated resnet [242], the conventional max-pooling layer at this stage is replaced by a strided convolutional layer inside the residual block. Subsequently, the feature maps are processed bottom-up as well as top-down by multiple stacks of u-nets at different scales and with regularly decreasing resolutions. The feature map input size to block 4 is 7×7 and is further reduced to 2×2 at the input to the encoder $E2$ of the u-net module. At this resolution, it is not possible to have $E2$ and $D2$ layers, and hence a trimmed version of u-nets (u-net$^+$) are employed in block 4. The u-net$^+$ includes a single level of encoder and decoder ($E1, D1$) processing. Towards the end of block 4, a batch normalization is performed and a ReLU non-linearity is applied. Following this, a global average pooling is performed on features and transferred to a softmax classifier.

The residual connection in all but the first u-net in each block is implemented as an identity mapping. In the first u-nets the skip connection is implemented using an expansion layer i.e., a 1×1 conv filter. The number of feature map outputs from each block approximately equates to the total number of feature maps generated by all the preceding u-net modules. This arrangement allows flexibility for the network to retain all the raw feature maps of the preceding modules. Moreover among all other possibilities, the above architectures were picked because their performance on the image classification task is roughly equivalent to the ResNet-18, 50 and 101 network architectures (discussed in Section 7.5.3), albeit with fewer parameters. However, in contrast to the work on residual net [244], our experimentation with wider nets (i.e., $N > 128$) did not yield any performance improvements on ImageNet.

As in ResNet [228] and DenseNet [230], most of the processing in SUNet is performed at the feature scale of 14×14 (46 conv layers) and 7×7 (44 conv layers). However, the order at which the local information is processed can lead to a substantial gap in performance between ResNet and SUNet when extending these popular architectures to object localization, detection, and image segmentation tasks. All these task demands pixel-level localization and hence require a deep architecture that can efficiently integrate local and global cues. The development of SUNet is a first step towards achieving this objective. Intuitively, multiple stacks of u-nets can be seen as multiple iterations of the message passing operation in a CRF [214].

7.5 Dilated SUNets for Segmentation

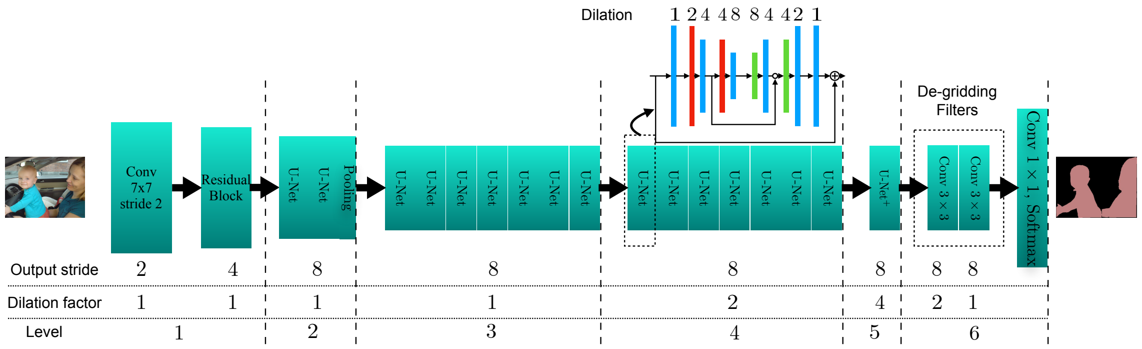


Figure 7.2: Dilated SUNet-7-128 network for segmentation at $output\_stride = 8$. For dilation > 1, the feature maps are processed with a varying range of dilation factors inside each u-net module (for example, see inset). The de-gridding filters smooth out aliasing artifacts that occur during deconvolution.

We now explain how pre-trained SUNet models can be adapted to perform semantic segmentation (see Section 7.5.3). One can directly extend SUNet to segmentation by removing a global average pooling layer (to increase output resolution) and operating the network in fully convolutional mode. Akin to other works on semantic segmentation [205, 217, 218], the output feature maps are rescaled using bilinear interpolation to the input image size before passing into the softmax layer with multi-class cross-entropy loss.

7.5.1 Dilation

For an input image of 512×512, the output map size at the softmax is 16×16 i.e., subsampled by a factor of 32. This is insufficient to preserve precise pixel-level localization information at its output. The precision can be improved by increasing the output map size of the network. This is realized by dropping the pooling stride at the transition layer. Merely eliminating stride leads to the reduction in the receptive field of the subsequent layers by a factor of two. Consequently, this reduces the influence of long-distance context information on the output prediction. Nevertheless, the receptive field is restored to that of the original network by operating each convolutional filter in the subsequent layers at a dilation factor of 2 [210, 216].

7.5.2 Multigrid

Figure 7.2 shows a sample dilated SUNet architecture used for the semantic segmentation task. Similar to [218], we define $output\_stride$ to be the ratio of resolution of an input image to that of its output feature map.

To sample at an $output\_stride$ of 8 the pooling layers preceding blocks (3) and (4) are discarded. Following this, the dilation factor for each u-net module in blocks 3 and 4 is fixed at 2 and 4, respectively. In each subsequent u-net module the 3×3 conv layers are operated with $stride = 1$. To keep the receptive field of the low-resolution layers in these modules constant, a dilation is applied. This arrangement facilitates the network to preserve spatial information learned from the prior modules (because there is no downsampling in the final u-net block) while preserving the distance scale of features within each block. As an example, the inset in Figure 7.2 displays the effective dilation rate for each layer in the u-net module at block 3. Similarly, the dilation rate of each layer (except for bottleneck layers) in the u-net^{+} module will be twice that of the corresponding layers in block 3. The steady increase and decrease of dilation factors inside each u-net module is analogous to multigrid solvers for linear systems [245, 246], which use grids at different scales to move information globally. Many recent works [218, 221, 226] on deep networks advocate the use of special structures for information globalization. In SUNet, the multigrid structure is baked into the model, and no further "frills" are needed to globalize information.

7.5.3 De-gridding Filters

By adopting dilated SUNets, we observe a vast improvement in segmentation performance. However, for $output\_stride = 8$ the segmentation map displays gridding artifacts [221, 242]. This aliasing artifact is introduced when the sampling rate of the dilated layer is lower than the high-frequency content of input feature maps. The final 3×3 conv filter of u-net^{+} operates at the dilation factor of 4. Directly transferring u-net^{+}'s feature map output to a classification layer can cause gridding artifacts. Following [242], the u-net^{+} module is followed by two layers of de-gridding filters with progressively decreasing dilation factor. Each filter is a 3×3 conv layer and outputs 512 feature maps.

SUNet does not require any additional post-hoc structural changes popularized by recent works such as decoding layers [205, 207], appending context aggregation blocks [210, 216–218] and learning conditional random fields [213, 214]. Hence we regard SUNet as a “no-frills” network.

7.6 Experiments

7.6.1 ImageNet Classification

In this section, we evaluate three SUNet architectures on the ILSVRC-2012 classification dataset, which contains $1.28M$ training images and $50,000$ images for validation, with labels distributed over 1000 classes. Training utilized the same data augmentation scheme used for ResNet [228] and DenseNet [230]. Following common practice [228,236], we apply a 224×224 center crop on test images and report Top-1 and Top-5 error on the validation set.

Implementation Details: All the models were implemented using the PyTorch deep learning framework and trained using four P6000 GPUs on a single node. We use SGD with a batch size of 256. For our largest model, 7-128, we were limited to a batch size of 212, due to the GPUs memory constraints. The initial learning rate was set to 0.01 and decreased by a factor of 10 every 30 epochs. We use a weight decay of $5e^{-4}$ and Nesterov momentum of 0.9 without dampening. The weights were initialized as in [247] and all the models were trained from scratch for a total of 100 epochs.

Table 7.2 compares the performance of SUNet against other classification networks. The comparison is restricted to only ResNet and DenseNet models as most recent work on segmentation builds on top of them. The notable point about the result is that the repeated top-down and the bottom-up processing of features performs equivalently to state-of-the-art classification networks.

We emphasize that our objective is not to surpass classification accuracy but instead to build a better architecture for segmentation by pre-training on a classification task. Indeed, each SUNet model was selected such that it is the counterpart for the corresponding ResNet model.

Model	Top-1	Top-5	Depth	Params
ResNet-18†	30.24	10.92	18	$11.7M$
ResNet-50†	23.85	7.13	50	$25.6M$
ResNet-101†	22.63	6.44	101	$44.5M$
DenseNet-201†	22.80	6.43	201	$20M$
DenseNet-161†	22.35	6.20	161	$28.5M$
SUNet-64	29.28	10.21	111	$6.9M$
SUNet-128	23.64	7.56	111	$24.6M$
SUNet-7-128	22.47	6.85	171	$37.7M$

Table 7.2: Error rates for classification networks on the ImageNet 2012 validation set. '†' denotes error rates from the official PyTorch implementation.

Model	mIoU
ResNet-101 [218]	68.39
SUNet-64	72.85
SUNet-128	77.16
SUNet-7-128	78.95

Table 7.3: The semantic segmentation performance of dilated SUNet and ResNet-101 networks on PASCAL VOC 2012 validation set trained with $output\_stride = 16$. Relative to the ResNet-101 network, all SUNets perform very well.

7.6.2 Semantic Segmentation

Semantic segmentation networks were built using the dilated version of the ImageNet pre-trained SUNet models (Section 7.5). We evaluate on the PASCAL VOC 2012 semantic segmentation benchmark [248] and urban scene understanding Cityscape [249] datasets. The performance on each of these datasets is reported using intersection-over-union (IoU) averaged over all classes.

7.6.3 Datasets

PASCAL VOC 2012: This dataset contains $1{,}464$ train, $1{,}449$ validation and $1{,}456$ test images. The pixel-level annotation for 20 objects and one background class is made available for the train and validation set. Following common practice, the train set is augmented with additional annotated data from [250] which finally provides a total of $10{,}582$ (trainaug) training images.

Cityscape: This dataset consists of finely annotated images of urban scenes covering multiple instances of cars, roads, pedestrians, buildings, etc. In total, it contains 19 classes on $2,975$ finely annotated training and 500 validation images.

7.6.4 Implementation Details

We use the SGD optimizer with a momentum of 0.95 and weight decay of 10^{-4}. Each model is fine-tuned starting with an initial learning rate of 0.0002 which is decreased every iteration by a factor of $0.5 \times \left(1 + \cos\left(\pi \frac{\text{iter}}{\text{max iters}}\right)\right)$ [251]. The batch-norm parameters are learned with a decay rate of 0.99 and the input crop size for each training image is set to 512×512. We train each model using two P6000 GPUs and the batch size 22. On PASCAL VOC, each model is trained for 45K iterations while for Cityscapes we use 90K iterations.

Unless mentioned, for all our experiments we set $output\_stride = 16$. This means only the u-net modules in the final block (4) operate at dilation factor of two; all other modules use the same stride as in the original classification model. Furthermore, $output\_stride = 16$ enables larger batch sizes than smaller stride choices, and hence leads to efficient learning of batch norm parameters. Also, training and inference are $2\times$ faster with $output\_stride = 16$ rather than 8.

Data Augmentation: To prevent overfitting during the training process, each training image is resized with a random scale from 0.5 to 2 following which the input image is randomly cropped. Additionally, the input is randomly flipped horizontally and also randomly rotated between $-10°$ to $10°$.

7.6.5 Ablation Study

We experiment with different SUNet variants on the PASCAL VOC 2012 dataset.

SUNets vs ResNet-101: We compare the performance of the dilated SUNet architecture on semantic segmentation against the popular dilated ResNet-101 model. Models were fine-tuned on the "trainaug" set without the degridding layers and evaluated on the validation set.

The performance of the plain dilated SUNets surpasses that of ResNet-101 by a wide margin when trained with $output\_stride = 16$ (Table 7.3). In fact, the smallest SUNet model, SUNet-64 with $6.7M$ parameters, beats ResNet-101 (with $44.5M$) by an absolute margin of 4.5% IoU while SUNet-7-128, the counterpart network to ResNet-101, improves by over 10.5% IoU. This is substantial, given that the gap between the ResNet-101 and VGG-16 models is $\sim 3\%$ [210] (at $output\_stride = 8$). This contrasts with the negligible performance differences observed on classification, and suggests that specialized segmentation network architectures can surpass architectures adapted from classification.

Finally, we note that, although SUNets were designed for pixel-level localization tasks, the selected models were chosen only based on their classification performance. By linking the model selection process to the primary task (segmentation and object detection) there is a possibility of improving performance.

OS	Strided conv	Multigrid
8	65.99	78.64
16	78.25	78.95

Table 7.4: Performance comparison of multigrid dilation against strided convolution inside each u-net module, using the SUNet-7-128 model and evaluated using mean IoU. **OS** denotes *output_stride* during training.

train OS	eval OS	DL	MS	Flip	mIoU
32	32				76.03
32	32		✓		77.58
32	32		✓	✓	77.57
16	16				78.95
16	16	✓			78.10
16	16		✓		80.22
16	16		✓	✓	80.40
8	8				78.64
8	8	✓			78.88
8	8		✓		80.37
8	8		✓	✓	80.50

Table 7.5: Performance comparison at various *output_stride* and inference strategies. **MS:** Multi-scale, **DL:** with Degridding Layers

Multigrid vs Downsampling: We compare the performance of multigrid dilation (as shown in Figure 7.2) inside each u-net against the usual downsampling (Figure 7.1). We consider a dilated SUNet-7-128 network and report performance at different training *output_stride*. For $output\_stride = 8$, the network was trained with a batch size of 12. The result is summarized in Table 7.4. For a dilated network, replacing strided convolutional layers with the corresponding dilated layers is more logical as well as beneficial. This is because, when operating dilated convolutional layers with $stride > 1$, alternate feature points are dropped without being processed by any of the filters, leading to high frequency noise at the decoder output. Furthermore, due to a skip connection, the features from the lower layers are also corrupted. Due to error propagation, this effect is more prominent in a network with many dilated modules (for eg., $output\_stride = 8$).

Output Stride and Inference Strategy: Finally, we experiment with three different training output strides (8,16,32) and multi-scale inference at test time. For $output\_stride = 32$, none of the layers are dilated and hence de-gridding layers were not used. Training with an $output\_stride = 32$ is equivalent to fine-tuning a classification network with a global pooling layer removed. For multi-scale inference, each input image is scaled and tested using multiple scales $\{0.5, 0.75, 1.0, 1.25\}$ and its left-right flipped image. The average over all output maps is used in the final prediction. See results in Table 7.5. We note that:

1. The network trained with $OS = 32$ performs 0.7 IoU better (with single scale) than the Resnet-101 and Resnet-152 models [252] each trained at $OS = 8$. This is significant, since the SUNet output contains $16\times$ fewer pixels. This leads to $4\times$ faster training/inference without a performance drop.
2. The degridding layers do not improve performance at $OS = 16$. In this case, there is only a small change in dilation factor between the final layer of SUNet and the classification layer, so aliasing is not problematic.
3. The margin of performance improvement decreases with increase in training OS. Given this and the above fact, subsequently we only report performance for models trained at $OS = 16$ without any degridding layers.

Pretraining with MS-COCO: Following common practice [217, 218, 232], we pretrain SUNet-7-128 with the MS-COCO dataset [253]. The MS-COCO dataset contains pixel-level annotation for 80 object classes. Except for the PASCAL VOC classes, the pixel annotation for all other classes is set to the background class. Following this, we use all the images from the MS-COCO "trainval" set except for those having < 1000 annotated pixel labels. This yields $90,000$ training images. After pretraining, the model is fine-tuned on "trainaug" for 5K iterations with $10\times$ smaller initial learning rate. In the end, our model achieves 83.27% mIoU on the validation set. This performance is slightly better than the ResNet+ASPP model [218] (82.70%) and equivalent to Xception+ASPP+Decoder model [232] (83.34%).

Methods	Validation mIoU
Resnet+ASPP	82.70
Xception+ASPP+Decoder	83.34
SUNet-7128	83.27

Table 7.6: Performance comparison on PASCAL VOC 2012 validation set. All networks were pretrained with MS-COCO.

7.6.6 Results on Test set

PASCAL VOC 2012: Before submitting test set output to an evaluation server, the above model was further fine-tuned on the "trainval" set with batch-norm parameters frozen and at $10\times$ smaller initial learning rate. Table 7.7 compares the test set results[2] against other state-of-the-art methods. PSPNet performs slightly better than SUNet, but at the cost of $30M$ more parameters while training at an $output\_stride = 8$. Figure 7.3, 7.4 displays some qualitative results on validation and test sets.

Cityscapes: A similar training strategy as in PASCAL is adopted except that the multi-scale inference is performed on additional scales $\{1.5, 1.75, 2.0, 2.25, 2.5\}$. Only the *finer*

[2]http://host.robots.ox.ac.uk:8080/anonymous/KT7JGJ.html

https://www.cityscapes-dataset.com/method-details/?submissionID=1151

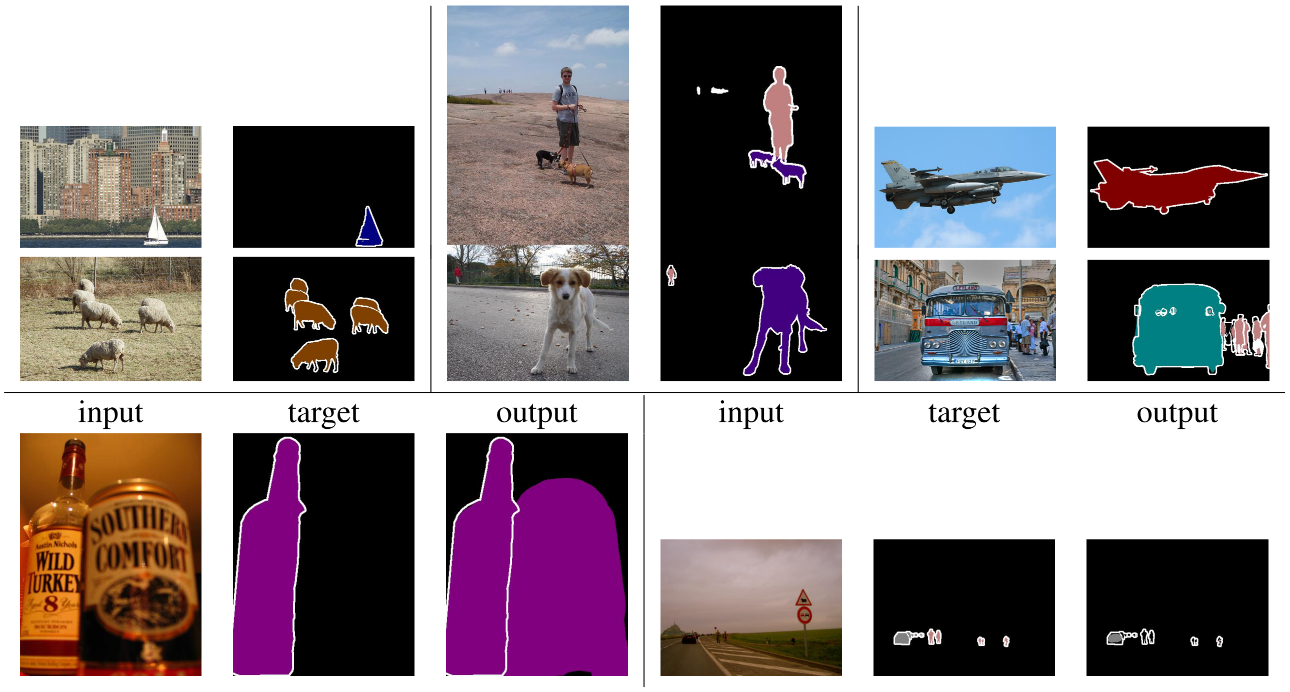


Figure 7.3: Visualization of the segmentation output on PASCAL VOC 2012 *val* set when trained at an $output\_stride = 16$ using SUNet-7-128 network + MS-COCO. Final row shows couple of failure case which happens due to, ambiguous annotation and inability in detecting low resolution objects.

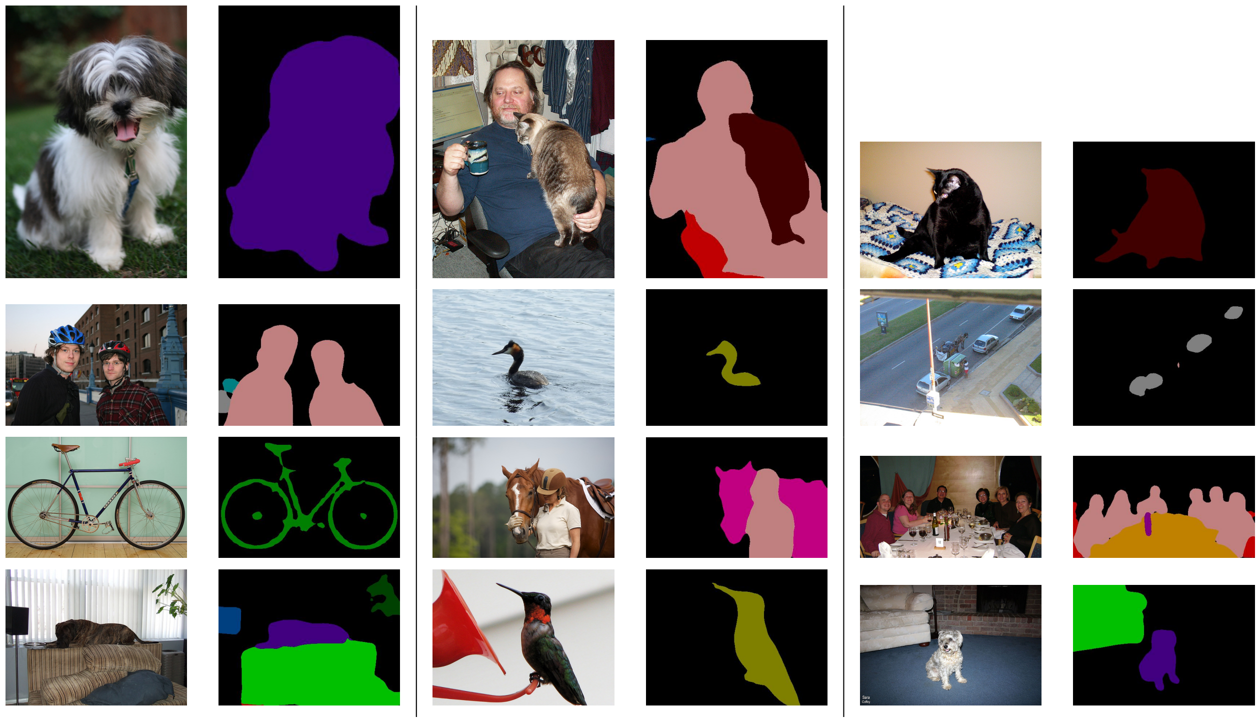


Figure 7.4: Visualization of the segmentation output on PASCAL VOC 2012 *test* set.

Methods	mIoU
Piecewise (VGG16) [224]	78.0
LRR+CRF [209]	77.3
DeepLabv2+CRF [210]	79.7
Large-Kernel+CRF [220]	82.2
Deep Layer Cascade* [254]	82.7
Understanding Conv [221]	83.1
RefineNet [205]	82.4
RefineNet-ResNet152 [205]	83.4
PSPNet [217]	**85.4**
SUNet-7-128	84.3

Table 7.7: Performance comparison on PASCAL VOC 2012 test set. For fair comparison, only the methods pre-trained using MS-COCO are displayed.

Methods	mIoU
LRR (VGG16) [209]	69.7
DeepLabv2+CRF [210]	70.4
Deep Layer Cascade* [254]	71.1
Piecewise (VGG16) [224]	71.6
RefineNet [205]	73.6
Understanding Conv [221]	77.6
PSPNet [217]	**78.4**
SUNet-7-128	75.3

Table 7.8: Performance comparison on Cityscapes *test* set. All methods were trained only using the "fine" set. All nets utilize ResNet-101 as a base network, except if specified or marked with *.

annotation set was used for training. The comparison on the Cityscapes test set results are displayed in Table 7.8.

7.6.7 Activation Maps

Figure 7.5 shows the activation map recorded at the end of each level (as indicated in figure 7.2) for an example input image of an "Aeroplane." As noted earlier, the inclusion of strided convolutions instead of multigrid dilations leads to noisy feature maps (see col 3; rows 4-6). The addition of de-gridding layers serves to produce a coherent prediction map at the output (see col 2; row 6).

7.7 Conclusion

The fundamental structure of conventional bottom-up classification networks limits their efficacy on secondary tasks involving pixel-level localization or classification. To

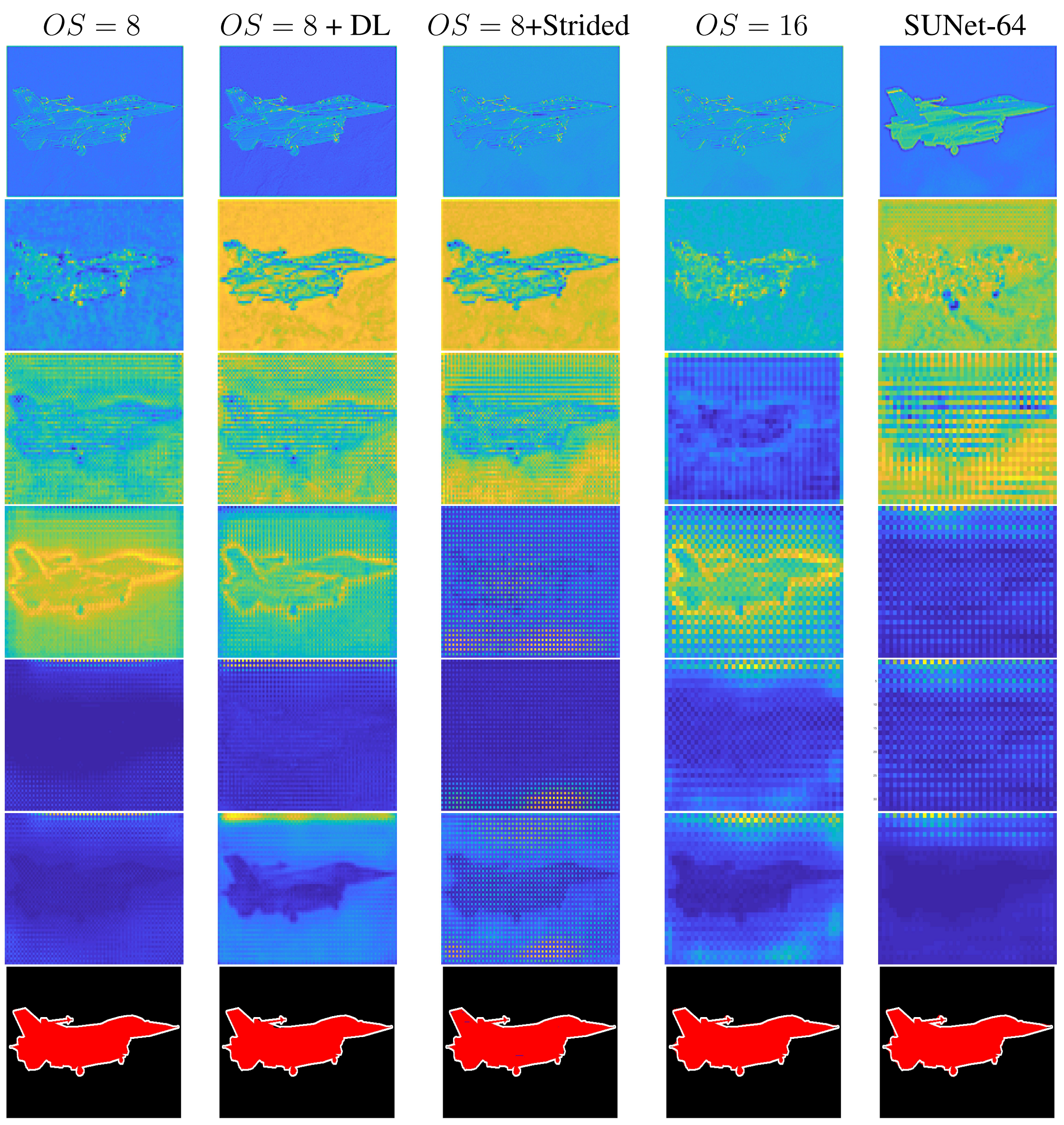


Figure 7.5: Activation map recorded at the end of each level of the dilated SUNet for an example input image of an 'Aeroplane'. The activation map with total highest magnitude were selected from among all feature map outputs at the corresponding layer. **Top** to **Bottom**: output at end of level $1 - 6$ followed by classification output. The level 6 output is simply a prediction map before bilinear interpolation.

overcome this drawback, a new network architecture, stacked u-nets (SUNets), is discussed in this chapter. SUNets leverage the information globalization power of u-nets in a deeper network architecture that is capable of handling the complexity of natural images. SUNets perform exceptionally well on semantic segmentation tasks while achieving fair performance on ImageNet classification.

There are several directions for future research that may improve upon the performance achievable using a simple SUNet. It may be advantageous to replace each convolution block by their corresponding depthwise separable convolution [233], as done in [232, 255–257]. The inclusion of post-hoc context [217, 218] or decoder networks [232] on top of SUNets may also help. Given the huge margin of improvement over ResNet models for semantic segmentation tasks, it is obvious to extend SUNets to object detection tasks. Finally, as suggested, rigorous hyper-parameter search and numerical analysis [258] on SUNets may yield better generic as well as task-specific models.

Part III

Label Prior

Chapter 8: Weakly-Supervised Co-Clustering

8.1 Introduction

Watching and sharing videos on social media has become an integral part of everyday life. We are often intrigued by the textual description of the videos and attempt to fast-forward to the segments of interest without watching the entire video. However, these textual descriptors usually do not specify the exact segment of the video associated with a particular description. For example, someone describing a movie clip as "head-on collision between cars while Chris Cooper is driving" neither provide the time-stamps for the collision or driving events nor the spatial locations of the cars or Chris Cooper. Such descriptions are referred to as 'weak labels'. For efficient video navigation and consumption, it is important to automatically determine the spatio-temporal locations of these concepts (such as 'collision' or 'cars'). However, it is prohibitively expensive to train concept-specific models for all concepts of interest in advance and use them for localization. This shortcoming has triggered a great amount of interest in *jointly* learning concept-specific classification models as well as localizing concepts from multiple weakly labeled images [259–261] or videos [262, 263].

Video descriptions include concepts which may refer to persons, objects, scenes and/or actions and thus a typical description is a combination of heterogeneous concepts.

In the running example, extracted heterogeneous concepts are ‘car’ (object), ‘head-on collision’ (action), ‘Chris Cooper’ (person) and ‘driving’ (action). Learning classifiers for these heterogeneous concepts along with localization is an extremely challenging task because: (a) the classifiers for different kinds of concepts are required to be learned simultaneously, e.g., a face classifier, an object classifier, an action classifier etc., and (b) the learning model must take into account the spatio-temporal location constraints imposed by the descriptions while learning these classifiers. For example, the concepts ‘head-on collision’ and ‘cars’ should spatio-temporally co-occur at least once and there should be at least one car in the video.

Recently there has been growing interest to jointly learn concept classifiers from weak labels [259, 263]. Bojanowski *et al* [263] proposed a discriminative clustering framework to jointly learn person and action models from movies using weak supervision provided by the movie scripts. Since weak labels are extracted from scripts, each label can be associated with a particular shot in the movie, which may last only for a few seconds, i.e., the labels are well localized and that makes the overall learning easier. However, in real world videos, one does not have access to such shot-level labels but only to video-level labels. Therefore in our work, we do not assume availability of such well localized labels, and tackle the more general problem of learning concepts from weaker video-level labels. The framework in [263], when extended to long videos, does not give satisfactory results (see section 8.4). Such techniques, which are based on a linear mapping from features to labels and model background using only a single latent factor, are usually inadequate to capture all the inter-class and intra-class variations. Shi *et al* [259] jointly learn object and attribute classifiers from images using weakly supervised Indian

Buffet Process (IBP). Note that IBP [264, 265] allows observed features to be explained by a countably infinite number of latent factors. However, the framework in [259] is not designed to handle heterogeneous concepts and location constraints, which leads to a significant degradation in performance (section 8.4.3). [266] and [267] propose IBP based cross-modal categorization/query image retrieval models which learn semantically meaningful abstract features from multimodal (image, speech and text) data. However, these unsupervised approaches do not incorporate any location constraints which naturally arise in the weakly supervised setting with heterogeneous labels.

In this chapter, we propose a novel Bayesian Non-parametric (BNP) approach called WSC-SIIBP (Weakly Supervised, Constrained & Stacked Integrative IBP) to jointly learn heterogeneous concept classifiers and localize these concepts in videos. BNP models are a class of Bayesian models where the hidden structure that may have generated the observed data is not assumed to be fixed. Instead, a framework is provided that allows the complexity of the model to increase as more data is observed [268]. Specifically, we propose a IBP model to jointly learn heterogeneous concepts which incorporates weakly supervised spatio-temporal location constraints in the learning procedure. The posterior inference is derived using mean-field approximation.

We assume that the weak video labels come in the form of tuples: in the running example, the extracted heterogeneous concept tuples are ({car, head-on collision}, {Chris Cooper, driving}). The experiments on two video datasets (a) the Casablanca movie dataset [263] and (b) the A2D dataset [269] show that the proposed approach WSC-SIIBP outperforms several state-of-the-art methods for heterogeneous concept classification and localization in a weakly supervised setting. For example, WSC-SIIBP leads to a relative

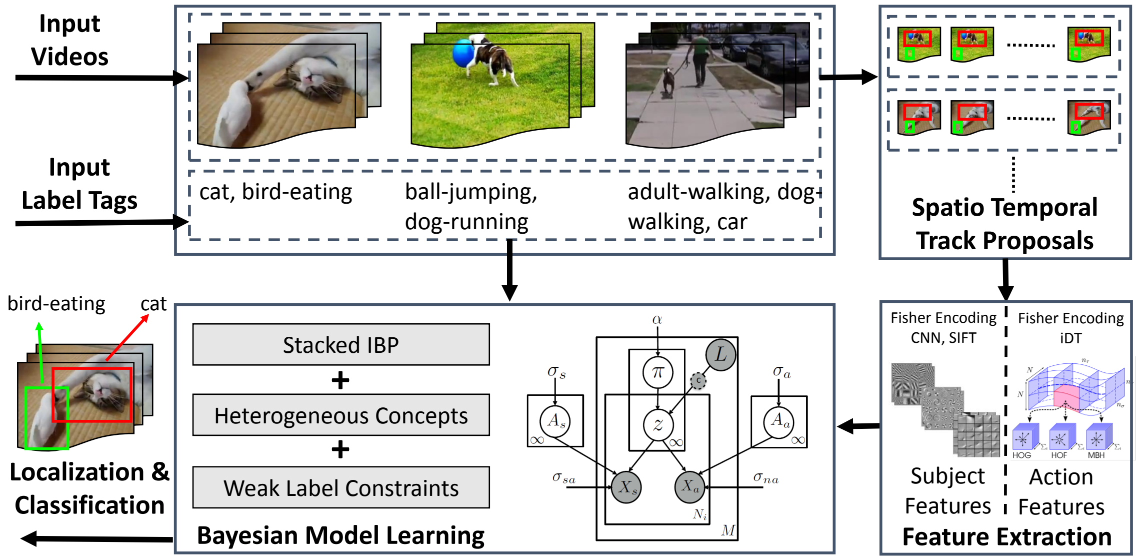


Figure 8.1: Pipeline of WSC-SIIBP. Multiple videos with heterogeneous weak labels are provided as input and localization and classification of the concepts are performed in these videos.

improvement of 7%, 5% and 24% on person, action and pairwise classification accuracies, respectively, over the most competitive baselines on the Casablanca dataset. Similarly, the relative improvement on localization accuracy is 9% over the next best approach on the A2D dataset. The code is available at `https://github.com/shahsohil/WSC-SIIBP`.

8.2 Related Work

Weakly Supervised Learning: Localizing concepts and learning classifiers from weakly annotated data is an active research topic. Researchers have learned models for various concepts from weakly labeled videos using Multi-Instance Learning (MIL) [270, 271] for human action recognition [272], visual tracking [273] etc. Cour *et al* [274] uses a novel convex formulation to learn face classifiers from movies and TV series using multimodal features which are obtained from finely aligned screenplay, speech and video data. In [262, 275], the authors propose discriminative clustering approaches for aligning videos with temporally ordered text descriptions or predefined tags and in the process also learn action classifiers. In our approach, we consider weak labels which are neither ordered nor aligned to any specific video segment. [276] proposes a method for learning object class detectors from real world web videos known to contain only the target class by formulating the problem as a domain adaptation task. [277] learns weakly supervised object/action classifiers using a latent-SVM formulation where the objects or actions are localized in training images/videos using latent variables. We note that - both [276, 277] consider only a single weak label per video and, unlike our approach, do not jointly learn the heterogeneous concepts. The authors in [278, 279] use dialogues, scene and character identification to find an optimal mapping between a book and movie shots using shortest path or CRF approach. However, these approaches neither jointly model heterogeneous concepts nor spatio-temporally localized them. Although [280] proposes a discriminative clustering model for coreference resolution in videos, only faces are considered in their experiments.

Heterogeneous concept learning: There are prior works on automatic image [281–284] and video [285–287] caption generation, where models are trained on pairs of image/video and text that contain heterogeneous concept descriptions to predict captions for novel images/videos. While most of these approaches rely on deep learning methods to learn a mapping between an image/video and the corresponding text description, [283] uses MIL to learn visual concept detectors (spatial localization in images) for nouns, verbs and adjectives. However, none of these approaches spatio-temporally localize points of interests in videos. Perhaps the available video datasets are not large enough to train such a weakly supervised deep learning model.

To the best of our knowledge there is no prior work that jointly classifies and localizes heterogeneous concepts in weakly supervised videos.

8.3 WSC-SIIBP: Model and Algorithm

In this section, we describe the details of WSC-SIIBP (see figure 8.1 for the pipeline). We first introduce notations and motivate our approach in sections 8.3.1 and 8.3.2 respectively. This is followed by section 8.3.3 where we introduce stacked non-parametric graphical model - IBP and its corresponding posterior computation. In sections 8.3.4 and 8.3.5, we formulate an extension of the stacked IBP model which can generalize to heterogeneous concepts as well as incorporate the constraints obtained from weak labels. In section 8.3.6, we briefly describe the inference procedure using truncated mean-field variational approximation and summarize our entire algorithm. Finally, we discuss how one can classify and localize concepts in new test videos using WSC-SIIBP.

8.3.1 Notation

Assume we are given a set of weakly labeled videos denoted by $\mathbf{\Lambda} = \left\{(i, \Gamma^{(i)})\right\}$, where i indicates a video and $\Gamma^{(i)}$ denotes the heterogeneous weak labels corresponding to the i-th video. Although the proposed approach can be used for any number of heterogeneous concepts, for readability, we restrict ourselves to two concepts and call them subjects and actions. We also have a closed set of class labels for these heterogeneous concepts: for subjects $\mathcal{S} = (s_1, \ldots, s_{K_s})$ and for actions $\mathcal{A} = (a_1, \ldots, a_{K_a})$. Let $K_s = |\mathcal{S}|$, $K_a = |\mathcal{A}|$, $\Gamma^{(i)} = \{(s_l, a_l) : s_l \in \mathcal{S} \cup \emptyset, a_l \in \mathcal{A} \cup \emptyset,\ \ 1 \leq l \leq |\Gamma^{(i)}|\}$, $\emptyset$ indicate that the corresponding subject or action class label is not present and $M = |\mathbf{\Lambda}|$ represents the number of videos. The video-level annotation simply indicates that the paired concepts $\Gamma^{(i)}$ can occur anywhere in the video and at multiple locations.

Assume that N_i spatio-temporal tracks are extracted from each video *i* where each track *j* is represented as an aggregation of multiple local features, $\mathbf{x}_j^{(i)}$. The spatio-temporal tracks could be face tracks, 3-D object proposals or action proposals (see section 8.4.1 for more details). We associate the j^{th} track in video *i* to an infinite binary latent coefficient vector $\mathbf{z}_j^{(i)}$ [259, 264]. Each video *i* is represented by a bag of spatio-temporal tracks $\mathbf{X}^{(i)} = \{\mathbf{x}_j^{(i)}, j = 1, \ldots, N_i\}$. Similarly, $\mathbf{Z}^{(i)} = \{\mathbf{z}_j^{(i)}, j = 1, \ldots, N_i\}$.

8.3.2 Motivation

Our objective is to learn (a) a mapping between each of the N_i tracks in video i and the labels in $\Gamma^{(i)}$ and (b) the appearance model for each label identity such that the tracks from new test videos can be classified. To achieve these objectives, it is important for any model to discover the latent factors that can explain similar tracks across a set of videos with a particular label. In general, the number of latent factors are not known apriori and must be inferred from the data. In Bayesian framework, IBP treats this number as a random variable that can grow with new observations, thus letting the model to effectively explain the unbounded complexity in the data. Specifically, IBP defines a prior distribution over an equivalence class of binary matrices of bounded rows (indicating spatio-temporal tracks) and infinite columns (indicating latent coefficients). To achieve our goals, we build on IBP and introduce the WSC-SIIBP model which can effectively learn the latent factors corresponding to each heterogeneous concept and utilize prior location constraints to reduce the ambiguity in learning through the knowledge of other latent coefficients.

8.3.3 Indian Buffet Process (IBP)

The spatio-temporal tracks in the videos $\mathbf{\Lambda}$ are obtained from an underlying generative process. Specifically, we consider a stacked IBP model [259] as described below.

- For each latent factor $k \in 1 \dots \infty$,
    1. Draw an appearance distribution with mean $\mathbf{a}_k \sim \mathcal{N}(0, \sigma_A^2 \mathbf{I})$
- For each video $i \in 1 \dots M$,
    1. Draw a sequence of i.i.d. random variables, $v_1^{(i)}, v_2^{(i)} \dots \sim \text{Beta}(\alpha, 1)$
    2. Construct the prior on the latent factors, $\pi_k^{(i)} = \prod_{t=1}^{k} v_t^{(i)}, \forall k \in 1 \dots \infty$,
    3. For j^{th} subject track in i^{th} video, where $j \in 1 \dots N_i$,
        (a) Sample state of each latent factor, $z_{jk}^{(i)} \sim \text{Bern}(\pi_k^{(i)})$,
        (b) Sample track appearance, $\mathbf{x}_j^{(i)} \sim \mathcal{N}\left(\mathbf{z}_j^{(i)} \mathbf{A}, \sigma_n^2 \mathbf{I}\right)$

where α is the prior controlling the sparsity of latent factors, σ_A^2 and σ_n^2 are the prior appearance and noise variance shared across all factors, respectively. Each $\mathbf{a}_k$ forms k^{th} row of $\mathbf{A}$ and the value of the latent coefficient $z_{jk}^{(i)}$ indicates whether data $\mathbf{x}_j^{(i)}$ contains the k^{th} latent factor or not. In the above model, we have used stick-breaking construction [288] to generate the $\pi_k^{(i)}$s.

Posterior: Now, we describe how the posterior is obtained for the above graphical model. Let $\mathbf{Y} = \left\{\pi^{(1)} \ldots \pi^{(M)}, \mathbf{Z}^{(1)} \ldots \mathbf{Z}^{(M)}, \mathbf{A}\right\}$ and $\boldsymbol{\Theta} = \{\alpha, \sigma_A^2, \sigma_n^2\}$ denote hidden variables and prior parameters, respectively. $\mathbf{X}$ denotes the concatenation of all the spatio-temporal tracks in all M videos, $\left\{\mathbf{X}^{(1)} \ldots \mathbf{X}^{(M)}\right\}$. Given prior distribution $\Psi(\mathbf{Y}|\boldsymbol{\Theta})$ and likelihood function $p(\mathbf{x}_j^{(i)}|\mathbf{Y}, \boldsymbol{\Theta})$, the posterior probability is given by,

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\Theta}) &= \frac{\Psi(\mathbf{Y}|\boldsymbol{\Theta}) \prod_{i=1}^{M} \prod_{j=1}^{N_i} p(\mathbf{X}_j^{(i)}|\mathbf{Y}, \boldsymbol{\Theta})}{p(\mathbf{X}|\boldsymbol{\Theta})} \\ \Psi(\mathbf{Y}|\boldsymbol{\Theta}) &= \prod_{k=1}^{\infty} \left(\prod_{i=1}^{M} p(\pi_k^{(i)}|\alpha) \prod_{j=1}^{N_i} p(z_{jk}^{(i)}|\pi_k^{(i)}) \right) p(\mathbf{a}_{k\cdot}|\sigma_A^2). \end{aligned} \tag{8.1}$$

where $p(\mathbf{X}|\boldsymbol{\Theta})$ is the marginal likelihood. For simplicity, we denote $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\Theta})$ as $q(\mathbf{Y})$. Apart from the significance of inferring $\mathbf{Z}^{(i)}$ for identifying track-level labels, inferring prior $\pi_k^{(i)}$ for each video helps to identify video-level labels, while the inference of appearance model $\mathbf{A}$ will be used to classify new test samples (see section 8.3.6). Thus, learning in our model requires computing the full posterior distribution over $\mathbf{Y}$.

Regularized posterior: We note that it is difficult to infer the regularized posterior distributions using (8.1). However, it is known [289, 290] that the posterior distribution in (8.1) can also be obtained as the solution $q(\mathbf{Y})$ of the following optimization problem,

$$\begin{aligned} &\underset{q(\mathbf{Y})}{\text{minimize}} \ \mathrm{KL}\left(q(\mathbf{Y})||\Psi(\mathbf{Y}|\boldsymbol{\Theta})\right) - \sum_{i=1}^{M} \sum_{j=1}^{N_i} \int \log p(\mathbf{x}_j^{(i)}|\mathbf{Y}, \boldsymbol{\Theta}) q(\mathbf{Y}) d\mathbf{Y} \\ &\text{subject to} \quad q(\mathbf{Y}) \in P_{prob} \end{aligned} \tag{8.2}$$

where KL(.) denotes the Kullback-Liebler divergence and P_{prob} is the probability simplex. As we will see later, this procedure enables us to learn the posterior distribution using a constrained optimization framework.

8.3.4 Integrative IBP

Our objective is to model heterogeneous concepts (such as subjects and actions) using a graphical model. However, the IBP model described above can not handle multiple concepts because it is highly unlikely that the subject and the action features can be explained by the same statistical model. Hence, we propose an extension of stacked IBP for heterogeneous concepts, where different concept types are modeled using different appearance models.

Let the subject and action types corresponding to the spatio-temporal track *j* in video *i* be denoted by ${\mathbf{x}^{\mathbf{s}}}^{(i)}_j$ and ${\mathbf{x}^{\mathbf{a}}}^{(i)}_j$, respectively, with each having different dimensions D^e ($e \in \{s, a\}$)[1]. Unlike the IBP model, ${\mathbf{X}^{\mathbf{s}}}^{(i)}_j$ and ${\mathbf{X}^{\mathbf{a}}}^{(i)}_j$ are now represented using two different gaussian noise models $\mathcal{N}(\mathbf{z}^{(i)}_j \mathbf{A}^s, \sigma^2_{ns}\mathbf{I})$ and $\mathcal{N}(\mathbf{z}^{(i)}_j \mathbf{A}^a, \sigma^2_{na}\mathbf{I})$ respectively where σ^2_{ne} denotes prior noise variance and $\mathbf{A}^e$ are $K \times D^e$ matrices (K $\to \infty$). The mean of the subject and action appearance models for each latent factor are also sampled independently from gaussian distributions of different variances σ^2_{Ae}. The new posterior probability is given by,

$$
\begin{aligned}
\tilde{q}(\mathbf{Y}) &= \frac{\Psi(\mathbf{Y}|\boldsymbol{\Theta}) \prod_{i=1}^{M} \prod_{j=1}^{N_i} \prod_{e\in\{s,a\}} p({\mathbf{x}^{\mathbf{e}}}^{(i)}_j | \mathbf{Z}, \mathbf{A}^e, \boldsymbol{\Theta})}{p(\mathbf{X}|\boldsymbol{\Theta})} \\
\Psi(\mathbf{Y}|\boldsymbol{\Theta}) &= \prod_{k=1}^{\infty} \left(\prod_{i=1}^{M} p(\pi^{(i)}_k|\alpha) \prod_{j=1}^{N_i} p(z^{(i)}_{jk}|\pi^{(i)}_k) \right) \prod_{e\in\{s,a\}} p(\mathbf{a}^e_k|\sigma^2_{Ae}\mathbf{I}).
\end{aligned}
\tag{8.3}
$$

[1]We often use e as a replacement for s and a

8.3.5 Integrative IBP with Constraints

Although the graphical model described above is capable of handling heterogeneous features, the location constraints inferred from the weak labels still need to be incorporated into the graphical model. As motivated in section 8.1, the concepts 'head-on collision' and 'cars' should spatio-temporally co-occur at least once and there should be at least one car in the full video. Imposing these location constraints in the inference algorithm can lead to more accurate parameter estimation of the graphical model and faster convergence of the inference procedure. These constraints can be generalized as follows,

1. Every label tuple in $\Gamma^{(i)}$, is associated with at least one spatio-temporal track (i.e., the event occurs in the video).
2. Spatio-temporal tracks should be assigned a label only from the list of weak labels assigned to the video. Concepts present in the video but not in the label will be subsumed in the background models.

Ideally, in the case of noiseless labels, these constraints should be strictly followed. However, we assume that real-world labels could be noisy and noise is independent of the videos. Hence, we allow constraints to be violated but penalize the violations using additional slack variables.

We associate the first K_s and the following K_a latent factors (the rows of $\mathbf{A}$) to the subject and action classes in $\mathcal{S}$ and $\mathcal{A}$ respectively. The inferred values of their corresponding latent coefficients in $\mathbf{z}_j^{(i)}$ are used to determine the presence/absence of the associated concept in a particular spatio-temporal track. The remaining unbounded number of latent factors are used to explain away the background tracks from unknown action and subject classes in a video. With these assignments, we enforce the following constraints on latent factors which are sufficient to satisfy the conditions mentioned earlier.

To satisfy 1, we introduce the following constraints, $\forall i \in 1 \dots M$, and $\forall j \in 1 \dots N_i$,

$$\sum_{j=1}^{N_i} z_{js}^{(i)} z_{ja}^{(i)} \geq 1 - \xi_{(s,a)}^{(i)}, \quad \forall (s,a) \in \Gamma^{(i)}, \tag{8.4}$$

$$\sum_{j=1}^{N_i} z_{js}^{(i)} \geq 1 - \xi_{(s,\emptyset)}^{(i)}, \quad \forall (s,\emptyset) \in \Gamma^{(i)}, \tag{8.5}$$

$$\sum_{j=1}^{N_i} z_{ja}^{(i)} \geq 1 - \xi_{(\emptyset,a)}^{(i)}, \quad \forall (\emptyset,a) \in \Gamma^{(i)}, \tag{8.6}$$

where ξ is the slack variable, z_{js} and z_{ja} are the latent factor coefficients corresponding to subject class s and action class a respectively.

To satisfy 2, we use the following constraints, $\forall i \in 1 \dots M$ and $\forall j \in 1 \dots N_i$,

$$z_{js}^{(i)} = 0, \text{if } \nexists(s,\emptyset) \in \Gamma^{(i)} \text{ and } \nexists(s,a) \in \Gamma^{(i)}, \forall a \in \mathcal{A}, \tag{8.7}$$

$$z_{ja}^{(i)} = 0, \text{if } \nexists(\emptyset,a) \in \Gamma^{(i)} \text{ and } \nexists(s,a) \in \Gamma^{(i)}, \forall s \in \mathcal{S}. \tag{8.8}$$

The constraints defined in (8.4)-(8.8) have been used in the context of discriminative clustering [263, 280]. However, our model is the first to use these constraints in a Bayesian setup. In their simplest form, they can be enforced using the point estimate of z e.g., MAP estimation. However, $\mathbf{Z}^{(i)}$ is defined over the entire probability space. To enforce the above constraints in a Bayesian framework, we need to account for the uncertainty in $\mathbf{Z}^{(i)}$. Following [291, 292], we define effective constraints as an expectation of the original constraints in (8.4)-(8.8), where the expectation is computed w.r.t. the posterior distribution in (8.3)

$$\forall i \in 1 \ldots M,$$

$$\sum_{j=1}^{N_i} \mathbb{E}_{\tilde{q}} \left[z_{js}^{(i)} z_{ja}^{(i)} \right] \geq 1 - \xi_{(s,a)}^{(i)}, \quad \forall (s,a) \in \Gamma^{(i)} \tag{8.9}$$

$$\sum_{j=1}^{N_i} \mathbb{E}_{\tilde{q}} \left[z_{js}^{(i)} \right] \geq 1 - \xi_{(s,\emptyset)}^{(i)}, \quad \forall (s,\emptyset) \in \Gamma^{(i)} \tag{8.10}$$

$$\sum_{j=1}^{N_i} \mathbb{E}_{\tilde{q}} \left[z_{ja}^{(i)} \right] \geq 1 - \xi_{(\emptyset,a)}^{(i)}, \quad \forall (\emptyset,a) \in \Gamma^{(i)} \tag{8.11}$$

$$\forall i \in 1 \ldots M \text{ and } \forall j \in 1 \ldots N_i,$$

$$\mathbb{E}_{\tilde{q}} \left[z_{js}^{(i)} \right] = 0, \text{ if } \nexists (s,\emptyset) \in \Gamma^{(i)} \text{ and } \nexists (s,a) \in \Gamma^{(i)}, \forall a \in \mathcal{A} \tag{8.12}$$

$$\mathbb{E}_{\tilde{q}} \left[z_{ja}^{(i)} \right] = 0, \text{ if } \nexists (\emptyset,a) \in \Gamma^{(i)} \text{ and } \nexists (s,a) \in \Gamma^{(i)}, \forall s \in \mathcal{S} \tag{8.13}$$

The proposed graphical model, incorporating heterogeneous concepts as well as the location constraints provided by the weak labels, is shown in Figure 8.2.

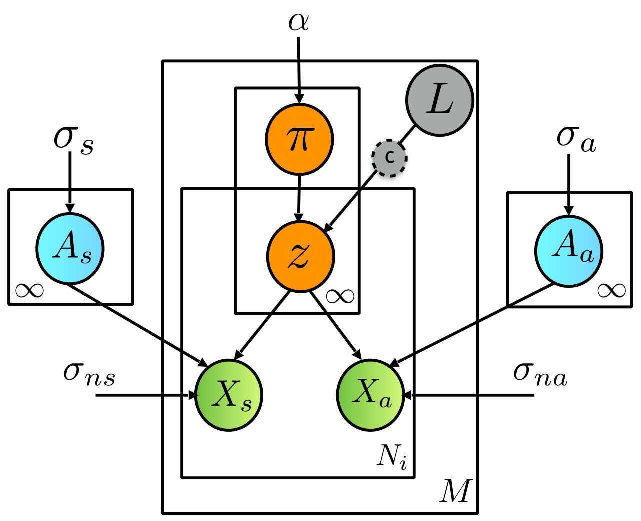
We restrict the search space for the posterior distribution in Equation (8.3) by using the expectation constraints in (8.9)-(8.13). In order to obtain the regularized posterior distribution of the proposed model, we solve the following optimization problem under these expectation constraints,

$$\underset{\tilde{q}(\mathbf{Y}),\xi^{(i)}}{\text{minimize}} \; \mathrm{KL}\left(\tilde{q}(\mathbf{Y}) || \tilde{\Psi}(\mathbf{Y}|\mathbf{\Theta})\right) - \sum_{i=1}^{M} \sum_{j=1}^{N_i} \int \left(\sum_{e \in \{s,a\}} \log p \left(\mathbf{X}_j^{\mathbf{e}^{(i)}} | \mathbf{Y}, \mathbf{\Theta} \right) \right) \tilde{q}(\mathbf{Y}) d\mathbf{Y} + C \sum_{i=1}^{M} \sum_{J \in \Gamma^{(i)}} \xi_J^{(i)}$$

$$\text{subject to} \quad \tilde{q}(\mathbf{Y}) \in P_{prob}$$

8.3.6 Learning and Inference

Note that the variational inference for true posterior $\tilde{q}(\mathbf{Y})$ (in Equation (8.3)) is intractable over the general space of probability functions. To make our problem easier to



M: Total number of videos
$\alpha, \sigma_a, \sigma_s, \sigma_{na}, \sigma_{ns}$: Hyperparameters
$\mathbf{A}_a, \mathbf{A}_s \sim N(0, \sigma_*^2\mathbf{I})$: Appearance model

For each video $i \in 1, \dots, M$,
$\pi^{(i)} \sim Stick(\alpha)$: Video prior
$L_k^{(i)} \in \{0, 1\}$: Weak input labels for kth concept

For each track $j \in 1, \dots, N_i$,
$z_{jk} \sim Bern\left(\pi_k^{(i)} L_k^{(i)} | \mathbf{z}_j^{(i)}, c\right)$: Latent coefficient
$\mathbf{x}_{j*}^{(i)} \sim N\left(\mathbf{z}_{j*}^{(i)} \mathbf{A}_*, \sigma_{n*}^2 \mathbf{I}\right)$: Feature vector

Figure 8.2: WSC-SIIBP: Graphical model using two heterogeneous concepts, subjects and actions. Each video (described by video-level labels L) is independently modeled using latent factor prior π and contains N_i tracks. Each track is represented using subject and action features X_s and X_a respectively, which are modeled using Gaussian appearance models A_s and A_a. z are the binary latent variables indicating the presence or absence of the latent factors in each track. c denotes the set of location constraints extracted from the video labels.

solve, we establish *truncated* mean-field variational approximation [288] to the desired posterior $\tilde{q}(\mathbf{Y})$, such that the search space P_{prob} is constrained by the following tractable parametrised family of distributions,

$$\tilde{w}(\mathbf{Y}) = \prod_{i=1}^{M} \left(\prod_{k=1}^{K_{max}} p(v_k^{(i)} | \tau_{k1}^{(i)}, \tau_{k2}^{(i)}) \prod_{j=1}^{N_i} p(z_{jk}^{(i)} | \nu_{jk}^{(i)}) \right) \prod_{k=1}^{K_{max}} \prod_{e \in \{s,a\}} p(\mathbf{a}_k^e | \mathbf{\Phi}_k^e, \sigma_{ke}^2 \mathbf{I}). \quad (8.14)$$

where $p(v_k^{(i)} | \tau_{k1}^{(i)}, \tau_{k2}^{(i)}) = \text{Beta}(v_k^{(i)}; \tau_{k1}^{(i)}, \tau_{k2}^{(i)})$, $p(z_{jk}^{(i)} | \nu_{jk}^{(i)}) = \text{Bern}(z_{jk}^{(i)}; \nu_{jk}^{(i)})$ and $p(\mathbf{a}_k^e | \mathbf{\Phi}_k^e, \sigma_{ke}^2 \mathbf{I}) = \mathcal{N}(\mathbf{a}_k^e; \mathbf{\Phi}_k^e, \sigma_{ke}^2 \mathbf{I})$. In Equation (8.14), we note that all the latent variables are modeled independently of all other variables, hence simplifying the inference procedure. The constraint in equation (8.9)-(8.13) simplifies to,

$$\forall i \in 1 \ldots M,$$

$$\sum_{j=1}^{N_i} \nu_{js}^{(i)} \nu_{ja}^{(i)} \geq 1 - \xi_{(s,a)}^{(i)}, \quad \forall (s,a) \in \Gamma^{(i)} \tag{8.15}$$

$$\sum_{j=1}^{N_i} \nu_{js}^{(i)} \geq 1 - \xi_{(s,\emptyset)}^{(i)}, \quad \forall (s,\emptyset) \in \Gamma^{(i)} \tag{8.16}$$

$$\sum_{j=1}^{N_i} \nu_{ja}^{(i)} \geq 1 - \xi_{(\emptyset,a)}^{(i)}, \quad \forall (\emptyset,a) \in \Gamma^{(i)} \tag{8.17}$$

$$\forall i \in 1 \ldots M \text{ and } \forall j \in 1 \ldots N_i,$$

$$\nu_{js}^{(i)} = 0, \text{if } \nexists (s,\emptyset) \in \Gamma^{(i)} \text{ and } (s,a) \in \Gamma^{(i)}, \forall a \in \mathcal{A} \tag{8.18}$$

$$\nu_{ja}^{(i)} = 0, \text{if } \nexists (\emptyset,a) \in \Gamma^{(i)} \text{ and } (s,a) \in \Gamma^{(i)}, \forall s \in \mathcal{S} \tag{8.19}$$

The truncated stick breaking process of $\pi_k^{(i)}$'s is bounded at K_{max}, wherein $\pi_k = 0$ for $k > K_{max}\mathbf{g}K_s + K_a + K_{bg}$. K_{bg} indicates the number of latent factors chosen to explain background tracks.

The optimization problem in Equation (8.14) is solved using the posterior distribution from Equation (8.14). We obtain the parameters (see appendix for details) σ_{ke}^2, $\mathbf{\Phi^e}_k$, $\tau_{ke}^{(i)}$ and $\nu_{jk}^{(i)}$ for the optimal posterior distribution $\tilde{q}(\mathbf{Y})$ using iterative update rules as summarized in Algorithm 7. We note that this algorithm is similar to other IBP learning algorithms [259, 288]. The complexity of Algorithm 7 is $\mathcal{O}(MN_{max}D_{max}K_{max})$, the same as [259]. The mean of binary latent coefficients z_{jk}, denoted by ν_{jk}, has an update rule which will lead to several interesting observations.

$$\nu_{jk}^{(i)} = \frac{L_k^{(i)}}{1 + e^{-\varsigma_{jk}^{(i)}}}. \tag{8.20}$$

$$
\begin{aligned}
\zeta_{jk}^{(i)} &= \sum_{j=1}^{k} \left(\Psi(\tau_{j1}^{(i)}) - \Psi(\tau_{j1}^{(i)} + \tau_{j2}^{(i)}) \right) - \mathcal{L}_k - \sum_{e\in\{s,a\}} \frac{1}{2\sigma_{ne}^2} \left(D^e \sigma_{ke}^2 + \mathbf{\Phi^e}_k \mathbf{\Phi^e}_k^T \right) \\
&+ \sum_{e\in\{s,a\}} \frac{1}{\sigma_{ne}^2} \mathbf{\Phi^e}_k \left(\mathbf{x}_j^{(i)} - \sum_{l\neq k} \nu_{jl}^{(i)} \mathbf{\Phi^e}_l \right)^T + C \underbrace{\sum_{\substack{J\in\Gamma^{(i)} \\ J=(k,a)}} \mathbb{I}_{\left\{\sum_{l=1}^{N_i} \nu_{lk}^{(i)} \nu_{la}^{(i)} < 1\right\}} \nu_{ja}^{(i)}}_{\text{(i)}} \qquad (8.21) \\
&+ C \overbrace{\sum_{\substack{J\in\Gamma^{(i)} \\ J=(s,k)}} \mathbb{I}_{\left\{\sum_{l=1}^{N_i} \nu_{ls}^{(i)} \nu_{lk}^{(i)} < 1\right\}} \nu_{js}^{(i)}}^{\text{(ii)}} + C \overbrace{\mathbb{I}_{\left\{\sum_{l=1}^{N_i} \nu_{lk}^{(i)} < 1, k \leq K_a + K_s\right\}}}^{\text{(iii)}}.
\end{aligned}
$$

where $\Psi(.)$ is the digamma function, $\mathbb{I}$ is an indicator function, $L_k^{(i)}$ is an indicator variable and $\mathcal{L}_k$ is a lower bound for $\mathbb{E}_{\tilde{w}}[\log(1 - \prod_{j=1}^{k} v^{(i)})]$. The $L_k^{(i)}$ indicates whether a concept (action/subject) k is part of the i^{th} video label set $\Gamma^{(i)}$ or not. If $L_k^{(i)} = 0$, all the corresponding binary latent coefficients $z_{jk}^{(i)}, j = \{1, \ldots, N_i\}$, are forced to 0, which is equivalent to enforcing the constraints in Equation (8.7) and (8.8). Note that the value of $\nu_{jk}^{(i)}$ increases with $\zeta_{jk}^{(i)}$. The terms (i)-(iii) in the update rule for $\zeta_{jk}^{(i)}$ (Equation (8.36)), which are obtained due to the location constraints in Equation (8.4)-(8.6), act as the coupling terms between $\nu_{je}^{(i)}$'s. For example, for any action concept, term (ii) suggests that if the location constraints are not satisfied, better localization of all the coupled subject concepts (high value of $\nu_{js}^{(i)}$) will drive up the value of $\zeta_{ja}^{(i)}$. This implies that the strong localization of one concept can lead to better localization of other concepts.

The hyperparameter σ_{ne}^2 and σ_{Ae}^2 can be set apriori or estimated from data. Similar to the maximization step of EM algorithm, their empirical estimation can easily be obtained by maximizing the expected log-likelihood.

Given the input features $\mathbf{X_s}$ and $\mathbf{X_a}$, the inferred latent coefficients $\nu_{je}^{(i)}$ estimate presence/absence of associated classes in a video. One can classify each spatio-temporal

Algorithm 7 Learning Algorithm of WSC-SIIBP

```
1:  **Input:** data $\mathbf{\Lambda} = \{(i, \mathbf{\Gamma}^{(i)})\}_{i \in 1 \dots M}$, constant $\alpha, K_{max}, C$
2:  **Output:** distribution $p(\mathbf{v}), p(\mathbf{Z}), p(\mathbf{A}^s), p(\mathbf{A}^a)$ and hyper-parameters $\sigma^2_{ns}, \sigma^2_{na}, \sigma^2_{As}$ and $\sigma^2_{Aa}$
3:  **Initialize:** $\tau^{(i)}_{k1} = \alpha, \tau^{(i)}_{k2} = 1, \nu^{(i)}_{jk} = 0.5, \mathbf{\Phi}^s_k = \mathbf{\Phi}^a_k = 0, \sigma^2_{ks} = \sigma^2_{ka} = \sigma^2_{ns} = \sigma^2_{na} = \sigma^2_{As} = \sigma^2_{Aa} = 1$
4:  **repeat**
5:      **repeat**
6:          update $\sigma^2_{ke}$ and $\mathbf{\Phi^e}_{k.}$ , $\forall 1 \le k \le K_{max}, e \in \{s, a\}$;
7:          update $\tau^{(i)}_{k1}$ and $\tau^{(i)}_{k2}$ , $\forall 1 \le k \le K_{max}$ and $i \in$ 1 to M;
8:          update $\nu^{(i)}_{jk}$ using Equation (8.35) and (8.36), $\forall 1 \le k \le K_{max}, 1 \le j \le N_i$ and $i \in$ 1 to M;
9:      **until** T iterations or $\frac{\|L(t-1)-L(t)\|}{L(t)} \le 1e^{-3}$
10:     update the hyperparameters $\sigma^2_{As}, \sigma^2_{Aa}, \sigma^2_{ns}, \sigma^2_{na}$
11: **until** T' iterations or $\frac{\|L(t'-1)-L(t')\|}{L(t')} \le 1e^{-4}$
```

track by estimating the track-level labels using $L^*_j = \arg\max_k \nu_{jk}$. Here the maximization is over the latent coefficients corresponding to either the subject or action concepts depending upon the label which we are interested in extracting. For the concept localization task in a video with label pair (s, a), the best track in the video is selected using $j^* = \arg\max_j \nu_{js} \times \nu_{ja}$.

Test Inference: Although the above formulation is proposed for concept classification and localization in a given set of videos (transductive setting), the same algorithm can also be applied to unseen test videos. The latent coefficients for the tracks of test videos can be learned alongside the training data except that the parameters σ^2_{ke}, $\mathbf{\Phi^e}_{k.}$, σ^2_{Ae} and σ^2_{ne} are updated only using training data. In the case of free annotation, i.e., absence of labels for test video i, we run the proposed approach by setting $L^{(i)}_k = 1$ in eq (8.35), indicating that the tracks in a video i can belong to any of the classes in $\mathcal{S}$ or $\mathcal{A}$ (i.e., no constraints as defined by (8.4)-(8.8) are enforced).

8.4 Experiments

In this section, we present an evaluation of WSC-SIIBP on two real-world databases: Casablanca movie and A2D dataset, which represent typical 'in-the-wild' videos with weak labels on heterogeneous concepts.

8.4.1 Datasets

Casablanca dataset: This dataset, introduced in [263], has 19 persons (movie actors) and three action classes (sitdown, walking, background). The heterogeneous concepts used in this dataset are persons and actions. The Casablanca movie is divided into shorter segments of duration either 60 or 120 seconds. We manually annotate all the tracks in each video segment which may contain multiple persons and actions. Given a video segment and the corresponding video-level labels (extracted from all ground truth track labels), our algorithm maps each of these labels to one or more tracks in that segment, i.e., converts the weak labels to strong labels. Our main objective of evaluation on this dataset is to compare the performance of various algorithms in classifying tracks from videos of varying length.

For our setting, we consider face and action as the two heterogeneous concepts and thus it is required to extract the face and the corresponding action track features. We extract 1094 facial tracks from the full 102 minute Casablanca video. The face tracks are extracted by running the multi-view face detector from [293] in every frame and associating detections across frames using point tracks [294]. We follow [295] to generate the face track feature representations: Dense rootSIFT features are extracted for each face in the track followed by PCA and video-level Fisher vector encoding. The action tracks corresponding to 1094 facial tracks are obtained by extrapolating the face bounding-boxes using linear transformation [263]. For action features, we compute Fisher vector encoding on dense trajectories [296] extracted from each action track.

On an average, each 60 sec. segment contains 11 face-action tracks and 4 face-action annotations while each 120 sec. video contains 21 tracks and 6 annotations. Note that, our experimental setup is more difficult compared to the experimental setting considered in [263]. In [263], the Casablanca movie is divided into numerous bags based on the movie script, where on average each segment is of duration 31 sec. containing only 6.27 face-action tracks.

A2D dataset: This dataset [269] contains 3782 YouTube videos (on average 7-10 sec. long) covering seven objects (bird, car etc.) performing one of nine actions (fly, jump etc.). The heterogeneous concepts considered are objects and actions. This dataset provides the bounding box annotations for every video label pair of object and action. Using the A2D dataset, we aim to analyze the track localization performance on weakly labeled videos as well as the track classification accuracy on a held-out test dataset.

We use the method proposed in [297] to generate spatio-temporal object track proposals. For computational purpose, we consider only 10 tracks per video and use the Imagenet pretrained VGG CNN-M network [298] to generate object feature representation. We extract convolutional layer conv-4 and conv-5 features for each track image followed by PCA and video-level Fisher vector encoding. In this dataset, the corresponding action tracks are kept similar to the object tracks (proposals) and the action features are extracted using the same approach as used for the Casablanca dataset.

8.4.2 Baselines

We compare WSC-SIIBP to several state-of-the-art approaches using the same features.

1. **WS-DC [263]**: This approach uses similar weak constraints as in (8.4)-(8.6), but in a discriminative setup where the constraints are incorporated in a biconvex optimization framework.
2. **WS-SIBP [259]**: This is a weakly supervised stacked IBP model which does not consider integrative framework for heterogeneous data and only enforces constraints equivalent to (8.7)-(8.8). For each spatio-temporal track, the features extracted for heterogeneous concepts are concatenated while using this approach.
3. **WS-S / WS-A**: This is similar to WS-SIBP except that instead of concatenating features from multiple concepts they are treated independently in two different IBP. WS-S(WS-A) is used to model only the person/object(action) features.
4. **WS-SIIBP**: This model integrates WS-SIBP with heterogeneous concepts.
5. **WSC-SIBP**: This model is similar to WS-SIBP, but unlike WS-SIBP, it additionally enforces the location constraints obtained from weak labels.

Implementation details: For each dataset, the Fisher encoded features are PCA reduced to an appropriate dimension, D^e. We select the best feature length and other algorithm specific hyper-parameters for each algorithm using cross-validation on a small set of input videos. For the IBP based models, the cross-validation range for hyper-parameters are $K_{max} := K_a + K_s : 10 : K_a + K_s + 100$, $\alpha := 3K_{max} : 10 : 4K_{max}$ and $C := 0 : 0.5 : 5$. For all IBP based models, the parameters D^e, α, K_{max} and C are set as 32, 100, 30 and 0.5 respectively for the Casablanca dataset and as 128, 160, 50 and 5 respectively for the A2D dataset. For WS-DC, D^e is set as 1024.

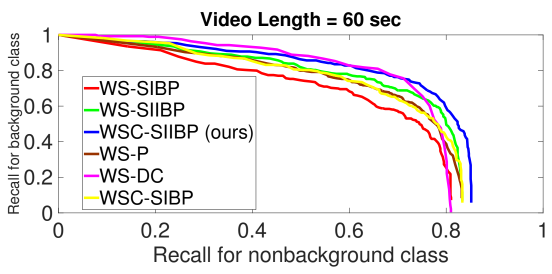
8.4.3 Results on Casablanca

The track-level classification performance is compared in Figure 8.3. From Figures 8.3(c) and 8.3(d), it can be seen that WSC-SIIBP significantly outperforms other methods for person and action classification in almost all of the scenarios. For instance, in the 120 second video segments, person classification improves by 4% (relative improvement is 7%) compared to the most competitive approach WS-SIIBP. We also compare pairwise label accuracy to gain insight into the importance of the constraints in eq (8.4)-(8.6). For any given track with non-background person and action label, the classification is assumed to be correct only if both person and action labels are correctly assigned. Even in this scenario WSC-SIIBP performs 8.1% better (24% relative improvement) than the most competitive baseline. Since we combine the heterogeneous concepts along with location constraints in an integrated framework, WSC-SIIBP outperforms all other baselines. The weak results of WS-DC in pairwise classification, though surprising, can be attributed to their action classification results which are significantly biased towards one particular action 'sitdown' (figure 8.3(d), note that WS-DC performs very poorly in 'walking' classification). Indeed, it should be noted that nearly 40% and 89% of person and action labels respectively belong to the background class. Thus, for fair evaluation of both background and non-background classes, we also plot the recall of background class against the recall of nonbackground classes for person and action classification in Figure 8.3(a), 8.3(b), 8.3(e), 8.3(f). These curves were obtained by simultaneously computing recall for background and non-background classes over a range of threshold values on score, ν. The mean average precision (mAP) of WSC-SIIBP along with all other baselines are plotted in Figure 8.3(g) and 8.3(h). The mAP values also clearly demonstrate the effectiveness of the proposed approach. From the performance of WS-SIIBP (integrative concepts, no constraints) and WSC-SIBP (no integrative concepts, constraints) (Figure 8.3(c) and 8.3(d)), it is clear that the improvement in performance in the WSC-SIIBP can be attributed to both addition of integrative concepts and the location constraints. Finally, the person class confusion matrix is shown in Figure 8.4. It exhibits that our approach learns each person appearance model with high accuracy and it can learn from as less as 15 weakly annotated samples.

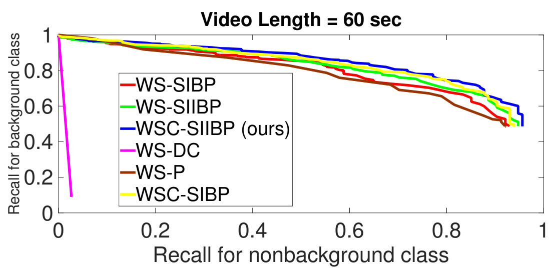
Effect of constraints (8.7)**,** (8.8)**:** We note that, regardless of other differences, every weakly supervised IBP model considered here enforces constraints (8.7), (8.8). However, these constraints are not part of the original WS-DC. To make a fair comparison between WS-DC and WSC-SIIBP, we analyze the effect of these constraints in Figure 8.3(i). Although, these additional constraints improve WS-DC performance, they do not supersede the performance of WSC-SIIBP. Further we observe that these constraints have improved the performance of all the weakly supervised IBP models.

8.4.4 Results on A2D

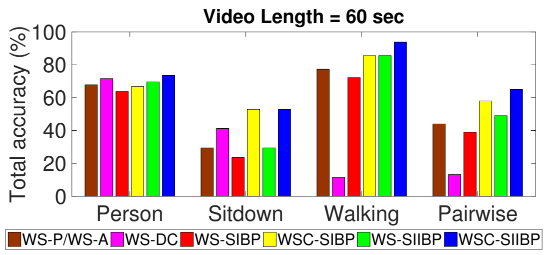
First, we evaluate localization performance on the full A2D dataset. We experiment with 37,820 tracks extracted from 3,782 videos with around 5000 weak labels. For every given object-action label pair our algorithm selects the best track from the corresponding



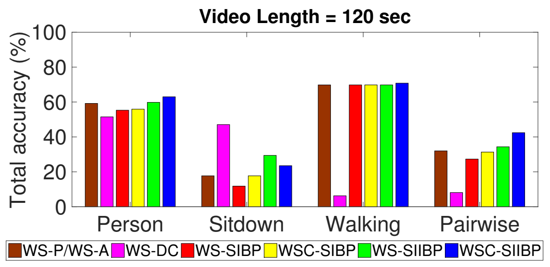
(a)



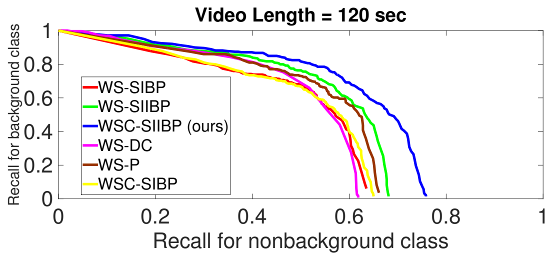
(b)



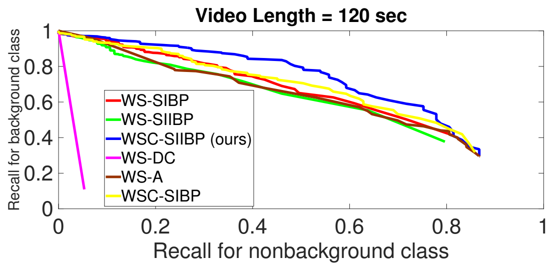
(c)



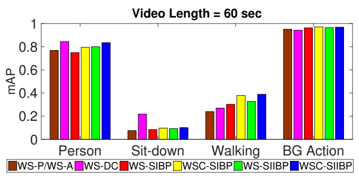
(d)



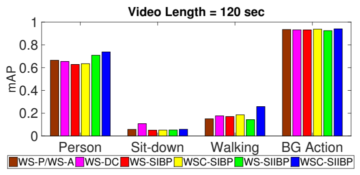
(e)



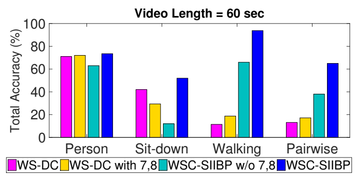
(f)



(g)

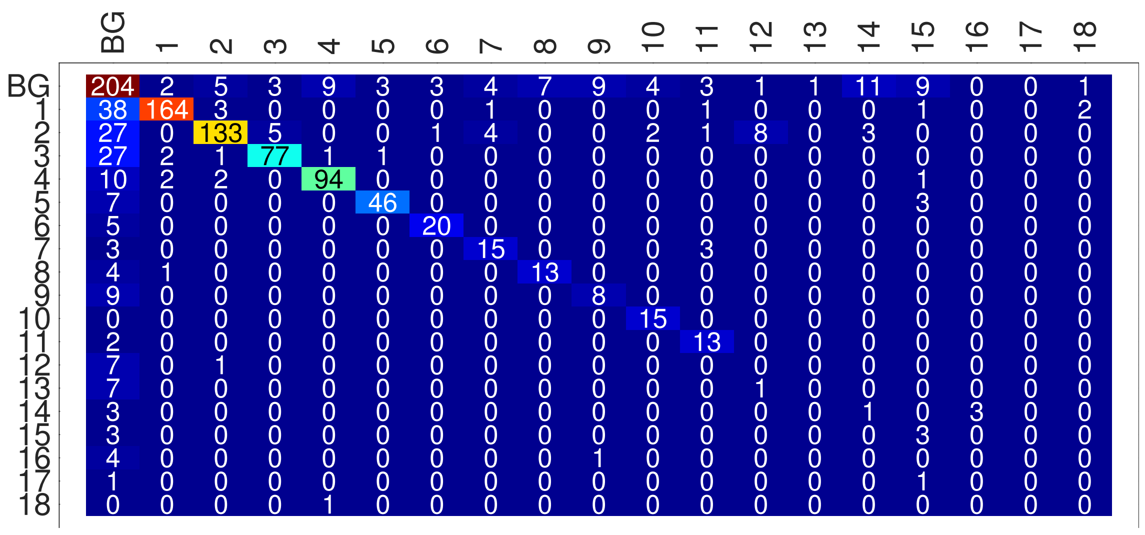


(h)



(i)

Figure 8.3: Comparison of results for the Casablanca movie dataset. (a) Classification accuracy for 60 sec. segments. (b) Recall for background vs non-background class (60 sec., person). (c) Recall for background vs non-background (60 sec., action). (d) Classification accuracy for 120 sec. segments. (e) Recall for background vs non-background class (120 sec., person). (f) Recall for background vs non-background (120 sec., action). (g),(h) Mean Average Precision for 60, 120 sec. segments. (i) Classification accuracy obtained with and without constraints (8.7) and (8.8)



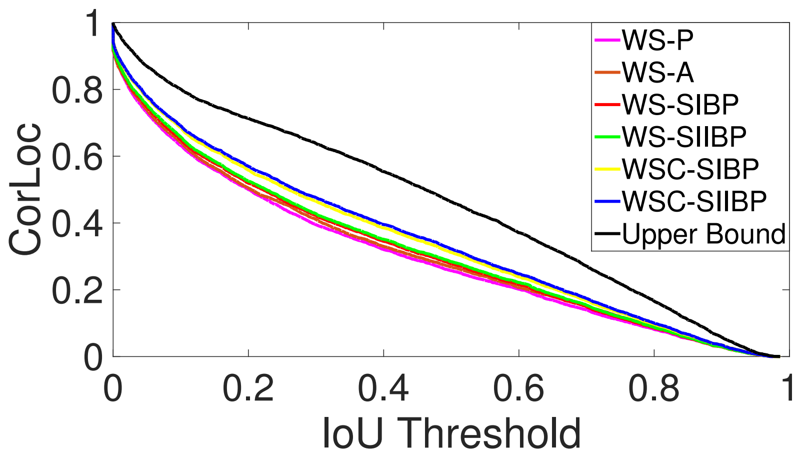
	BG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
BG	204	2	5	3	9	3	3	4	7	9	4	3	1	1	11	9	0	0	1
1	38	164	3	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	2
2	27	0	133	5	0	0	1	4	0	0	2	1	8	0	3	0	0	0	0
3	27	2	1	77	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10	2	2	0	94	0	0	0	0	0	0	0	0	0	0	1	0	0	0
5	7	0	0	0	0	46	0	0	0	0	0	0	0	0	0	3	0	0	0
6	5	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
7	3	0	0	0	0	0	0	15	0	0	0	3	0	0	0	0	0	0	0
8	4	1	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
9	9	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0
11	2	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
12	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
14	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3	0	0
15	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
16	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
18	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8.4: Person class confusion matrix. BG denotes the background class which can represent any unknown face.

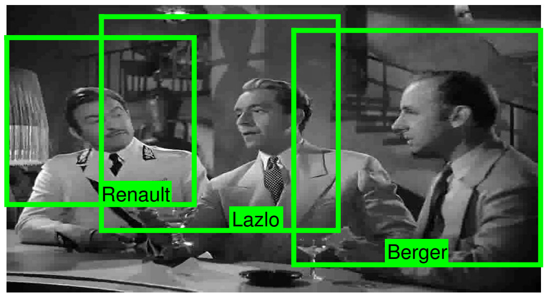
video using the approach outlined in section 8.3.6. The localization accuracy is measured by calculating the average IoU (Intersection over Union) of the selected track (3-D bounding box) with the ground truth bounding box. The class-wise IoU accuracy and the mean IoU accuracy for all classes are tabulated in Table 8.1 and 8.2 respectively. In this task WSC-SIIBP also leads to a relative improvement of 9% above the next best baseline. We also evaluate how accurately the extracted object proposals match with the ground truth bounding boxes to estimate an upper bound on the localization accuracy (referred as Upper Bound in Table 8.1, 8.2). In this case, the track maximizing the average IoU with the ground truth annotation is selected and the corresponding IoU is reported. We plot the correct localization accuracy with varying IoU thresholds in Figure 8.5(a), which also shows the effectiveness of the proposed approach. Figure 8.5(b), 8.5(c), 8.6 shows some qualitative track localization results using the proposed approach on the selected frame.

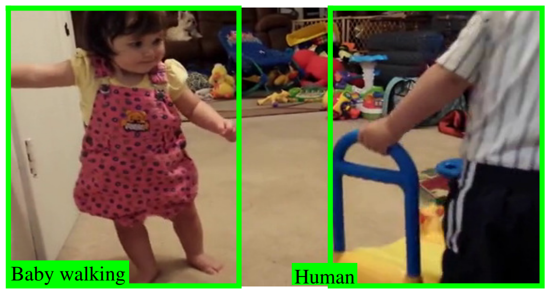
<u>Test Inference</u>: We evaluate the classification performance on held-out test samples using the same train/test partition as in [269]. We consider two setups for the evaluation,

(a) using video-level labels for the test samples and (b) free annotation where no test video labels are provided. The proposed approach is compared with GT-SVM, which is a fully supervised linear SVM that uses ground truth bounding boxes and their corresponding strong labels during training. The results are tabulated in Table 8.3. Note that the performance of WSC-SIIBP is close to that of the fully supervised setup.



(a)





(b) (c)

Figure 8.5: (a) Correct localization accuracy at various IOU thresholds. (b) and (c) Qualitative results: green boxes show the concept localization using our proposed approach.

	adult	baby	ball	bird	car	cat	dog
WSC-SIIBP	28.4	43.6	9.8	37.8	37.4	40.8	42.0
Upper Bound	39.9	53.9	16.4	48.2	48.7	52.8	51.4

	climb	crawl	eat	fly	jump	roll	run	walk
WSC-SIIBP	37.5	47.6	46.1	24.5	29.4	50.9	25.6	37.2
Upper Bound	50.0	59.2	57.2	33.9	41.0	59.1	38.1	47.9

Table 8.1: Per class mean IoU on A2D dataset.

	Random	WS-P	WS-A	WS-SIBP	WS-SIIBP	WSC-SIBP	WSC-SIIBP	Upper Bound
IoU	25.5	29.7	30.43	31.1	31.55	31.69	**34.38**	45.05

Table 8.2: Average IoU comparison with other approaches on A2D dataset.

	WSC-SIIBP		GT-SVM	
Setup	Obj	Act	Obj	Act
Using video Labels	94.77	90.68	98.20	94.92
Free Annotation	76.62	64.77	85.18	73.26

Table 8.3: mAP classification test accuracy on A2D dataset.

8.5 Conclusion

We developed a Bayesian non-parametric approach that integrates the Indian Buffet Process with heterogeneous concepts and spatio-temporal location constraints arising from weak labels. We report experimental results on two recent datasets containing heterogeneous concepts such as persons, objects and actions and show that our approach outperforms the best state of the art method.

Appendix 8.A Derivation of Posterior Update Equations

Now, note that the constraints in (8.15)-(8.17) can be rewritten as hinge loss function and added as part of the objective function in equation (8.14). Hence the final formulation is given by,

$$\begin{aligned}
\min_{\substack{\nu^{(i)},\tau^{(i)},\\ \boldsymbol{\Phi}_k^*,\sigma_{k*}^2}} \quad & KL\left(\tilde{w}(\mathbf{Y})||\tilde{\Psi}(\mathbf{Y}|\boldsymbol{\Theta})\right) - \sum_{i=1}^{M}\sum_{j=1}^{N_i}\int\left(\sum_{e\in\{s,a\}}\log p\left(\mathbf{X}^{\mathbf{e}}{}_j^{(i)}|\mathbf{Y},\boldsymbol{\Theta}\right)\right)\tilde{w}(\mathbf{Y})d\mathbf{Y} \\
& + C\sum_{i=1}^{M}\left(\sum_{\substack{J\in\Gamma^{(i)}\\ J=(s,a)}}\max\left(0, 1-\sum_{j=1}^{N_i}\nu_{js}^{(i)}\nu_{ja}^{(i)}\right) + \sum_{\substack{J\in\Gamma^{(i)}\\ J=(s,\emptyset)}}\max\left(0, 1-\sum_{j=1}^{N_i}\nu_{js}^{(i)}\right)\right. \\
& \left. + \sum_{\substack{J\in\Gamma^{(i)}\\ J=(\emptyset,a)}}\max\left(0, 1-\sum_{j=1}^{N_i}\nu_{ja}^{(i)}\right)\right) \\
& s.t. \quad \forall i \in 1\ldots M, \text{ and } \forall j\in 1\ldots N_i, \quad (8.18), (8.19)
\end{aligned} \tag{8.22}$$

The objective function in eq. (8.22) can be rewritten as,

$$L(\nu^{(i)}, \tau^{(i)}, \boldsymbol{\Phi}_k^*, \sigma_{k*}^2) = \mathcal{L} - \sum_{i=1}^{M}\sum_{j=1}^{N_i}\left(L_{ij} - C\sum_{k=1}^{K_a+K_s}H_{jk}^{(i)}\right) \tag{8.23}$$

where $\mathcal{L}$ represent KL-divergence term, L_{ij} denote the likelihood term and H_{jk} is the term corresponding to hinge loss function for $\nu_{jk}^{(i)}$. Expanding L_{ij}, we get,

$$L_{ij} \triangleq \mathbb{E}_{\tilde{w}}\left[\log p(\mathbf{X}^{\mathbf{s}}{}_j^{(i)}|\mathbf{Y},\boldsymbol{\Theta}) + \log p(\mathbf{X}^{\mathbf{a}}{}_j^{(i)}|\mathbf{Y},\boldsymbol{\Theta})\right] \tag{8.24}$$

$$\begin{aligned}
= & -\frac{\mathbf{x}^{\mathbf{s}}{}_j^{(i)T}\mathbf{x}^{\mathbf{s}}{}_j^{(i)} - 2\mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)}\mathbf{A}^{\mathbf{s}}]\mathbf{x}^{\mathbf{s}}{}_j^{(i)} + \mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)}\mathbf{U}^{\mathbf{s}}\mathbf{z}_{j.}^{(i)T}]}{2\sigma_{ns}^2} - \frac{D^s\log(2\pi\sigma_{ns}^2)}{2} \\
& -\frac{\mathbf{x}^{\mathbf{a}}{}_j^{(i)T}\mathbf{x}^{\mathbf{a}}{}_j^{(i)} - 2\mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)}\mathbf{A}^{\mathbf{a}}]\mathbf{x}^{\mathbf{a}}{}_j^{(i)} + \mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)}\mathbf{U}^{\mathbf{a}}\mathbf{z}_{j.}^{(i)T}]}{2\sigma_{na}^2} - \frac{D^a\log(2\pi\sigma_{na}^2)}{2}
\end{aligned} \tag{8.25}$$

where $\mathbf{U}^* = \mathbb{E}_{\tilde{w}}[\mathbf{A}^*\mathbf{A}^{*T}]$ is $K_{max} \times K_{max}$ matrix, $U^*_{jk} = \mathbf{\Phi}^*{}_{j.}\mathbf{\Phi}^*{}^T_{k.}$; $\mathbb{E}_{\tilde{w}}[\mathbf{z}^{(i)}_{j.}\mathbf{A}^*]\mathbf{x}^*{}^{(i)}_j = \left(\sum_k \nu^{(i)}_{jk}\mathbf{\Phi}^*{}_{k.}\right)\mathbf{x}^*{}^{(i)}_j$; and

$$\mathbb{E}_{\tilde{w}}[\mathbf{z}^{(i)}_{n.}\mathbf{U}^*\mathbf{z}^{(i)}_{n.}{}^T] = 2\sum_{j<k}\nu^{(i)}_{nj}\nu^{(i)}_{nk}U^*_{jk} + \sum_k \nu^{(i)}_{nk}\left(D^*\sigma^2_{k*} + \mathbf{\Phi}^*{}_{k.}\mathbf{\Phi}^*{}^T_{k.}\right) \tag{8.26}$$

For KL-divergence term, we get $\mathrm{KL}\left(\tilde{w}(\mathbf{Y})||\tilde{\Psi}(\mathbf{Y}|\mathbf{\Theta})\right) = \mathrm{KL}\left(\tilde{w}(\mathbf{v})||\tilde{\Psi}(\mathbf{v}|\mathbf{\Theta})\right) + \mathrm{KL}\left(\tilde{w}(\mathbf{Z})||\tilde{\Psi}(\mathbf{Z}|\mathbf{\Theta})\right) +$ $\mathrm{KL}\left(\tilde{w}(\mathbf{A^s})||\tilde{\Psi}(\mathbf{A^s}|\mathbf{\Theta})\right) + \mathrm{KL}\left(\tilde{w}(\mathbf{A^a})||\tilde{\Psi}(\mathbf{A^a}|\mathbf{\Theta})\right)$, where the individual terms are,

$$\begin{aligned}
\mathrm{KL}\left(\tilde{w}(\mathbf{v})||\tilde{\Psi}(\mathbf{v}|\mathbf{\Theta})\right) =& \sum_{i=1}^{M}\left(\sum_{k=1}^{K_{max}}\left((\tau^{(i)}_{k1} - \alpha)(\Psi(\tau^{(i)}_{k1}) - \Psi(\tau^{(i)}_{k1} + \tau^{(i)}_{k2}))\right.\right.\\
&+ (\tau^{(i)}_{k2} - 1)(\Psi(\tau^{(i)}_{k2}) - \Psi(\tau^{(i)}_{k1} + \tau^{(i)}_{k2}))\\
&\left.\left.- \log\frac{\Gamma(\tau^{(i)}_{k1})\Gamma(\tau^{(i)}_{k2})}{\Gamma(\tau^{(i)}_{k1} + \tau^{(i)}_{k2})}\right) - K_{max}\log\alpha\right)
\end{aligned} \tag{8.27}$$

$$\begin{aligned}
\mathrm{KL}\left(\tilde{w}(\mathbf{Z})||\tilde{\Psi}(\mathbf{Z}|\mathbf{\Theta})\right) =& \sum_{i=1}^{M}\left(\sum_{j=1}^{N_i}\sum_{k=1}^{K_{max}}\left(-\nu^{(i)}_{jk}\sum_{j=1}^{K_{max}}(\Psi(\tau^{(i)}_{k1}) - \Psi(\tau^{(i)}_{k1} + \tau^{(i)}_{k2}))\right.\right.\\
&- (1 - \nu^{(i)}_{jk})\mathbb{E}_{\tilde{w}}[\log(1 - \prod_{j=1}^{k} v^{(i)})]\\
&\left.\left.+\nu^{(i)}_{jk}\log\nu^{(i)}_{jk} + (1 - \nu^{(i)}_{jk})\log(1 - \nu^{(i)}_{jk})\right)\right)
\end{aligned} \tag{8.28}$$

$$\mathrm{KL}\left(\tilde{w}(\mathbf{A}^*)||\tilde{\Psi}(\mathbf{A}^*|\mathbf{\Theta})\right) = \sum_{k=1}^{K_{max}}\left(\frac{D^*\sigma^2_{k*} + \mathbf{\Phi}^*{}_k\mathbf{\Phi}^*{}^T_k}{2\sigma^2_{A*}} - \frac{D^*\left(1 + \log\frac{\sigma^2_{k*}}{\sigma^2_{A*}}\right)}{2}\right) \tag{8.29}$$

where $\Psi(.)$ is the digamma function. As shown for original IBP in [288], the term $\mathbb{E}_{\tilde{w}}[\log(1 - \prod_{j=1}^{k} v^{(i)})]$ is approximated by its lower bound,

$$\begin{aligned}
\mathbb{E}_{\tilde{w}}[\log(1 - \prod_{j=1}^{k} v^{(i)})] \geq& \sum_{m=1}^{k} q_{km}\Psi(\tau^{(i)}_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k} q_{kn}\right)\Psi(\tau^{(i)}_{m1})\\
&- \sum_{m=1}^{k}\left(\sum_{n=m}^{k} q_{kn}\right)\Psi(\tau^{(i)}_{m1} + \tau^{(i)}_{m2}) + \mathcal{H}(q_{k.})\\
&= \mathcal{L}_k
\end{aligned} \tag{8.30}$$

where the variational parameter $q_{k.} = (q_{k1} \dots q_{kk})$ is k-point probability mass function and $\mathcal{H}(q_{k.})$ denotes entropy of $q_{k.}$. The tightest upper bound is obtained by setting,

$$q_{km} = \frac{1}{Z_k} \exp\left(\Psi(\tau_{m2}^{(i)}) + \sum_{n=1}^{m-1} \Psi(\tau_{n1}^{(i)}) - \sum_{n=1}^{m} \Psi(\tau_{n1}^{(i)} + \tau_{n2}^{(i)}) \right)$$

where Z_k is the normalization factor to enable $q_{k.}$ to be a distribution. On replacing the term $\mathbb{E}_{\tilde{w}}[\log(1 - \prod_{j=1}^{k} v_j^{(i)})]$ with its lower bound $\mathcal{L}_k$, we have an upper bound for $\mathrm{KL}\left(\tilde{w}(\mathbf{Y}) || \tilde{\Psi}(\mathbf{Y}|\mathbf{\Theta})\right)$.

On substituting equation (8.24)-(8.30) in (8.23), the optimum value for parameters of mean-field variational approximate posterior distribution (8.14) are obtained by setting the derivative of (8.23) w.r.t. those parameters to zero and simultaneously solving for all parameters (using KKT conditions). We derive the following equations which are iteratively solved,

$$\sigma_{ke}^2 = \left(\frac{1}{\sigma_{Ae}^2} + \frac{1}{\sigma_{ne}^2} \sum_{i=1}^{M} \sum_{j=1}^{N_i} \nu_{jk}^{(i)} \right)^{-1}, \ \forall e \in \{s, a\} \tag{8.31}$$

$$\mathbf{\Phi^e}_k = \left(\frac{1}{\sigma_{ne}^2} \sum_{i=1}^{M} \sum_{j=1}^{N_i} \nu_{jk}^{(i)} \left(\mathbf{x^e}_j^{(i)} - \sum_{l:l \neq k} \nu_{jl}^{(i)} \mathbf{\Phi^e}_l \right) \right) \sigma_{ke}^2, \ \forall e \in \{s, a\} \tag{8.32}$$

$$\tau_{k1}^{(i)} = \alpha + \sum_{m=k}^{K_{max}} \sum_{j=1}^{N_i} \nu_{jm}^{(i)} + \sum_{m=k+1}^{K_{max}} \left(N_i - \sum_{j=1}^{N_i} \nu_{jm}^{(i)} \right) \left(\sum_{s=k+1}^{m} q_{ms}^{(i)} \right) \tag{8.33}$$

$$\tau_{k2}^{(i)} = 1 + \sum_{m=k}^{K_{max}} \left(N_i - \sum_{j=1}^{N_i} \nu_{jm}^{(i)} \right) q_{mk}^{(i)} \tag{8.34}$$

Above equations are somewhat similar to those given by variational approximation on IBP [288]. The update equation for ν differs completely and it is given by,

$$\nu_{jk}^{(i)} = \frac{L_k^{(i)}}{1 + e^{-\zeta_{jk}^{(i)}}} \tag{8.35}$$

$$\zeta_{jk}^{(i)} = \sum_{t=1}^{k} \left(\Psi(\tau_{t1}^{(i)}) - \Psi(\tau_{t1}^{(i)} + \tau_{t2}^{(i)}) \right) - \mathcal{L}_k$$
$$- \frac{1}{2\sigma_{ns}^2} \left(D^s \sigma_{ks}^2 + \mathbf{\Phi^s}_k \mathbf{\Phi^s}_k^T \right) - \frac{1}{2\sigma_{na}^2} \left(D^a \sigma_{ka}^2 + \mathbf{\Phi^a}_k \mathbf{\Phi^a}_k^T \right)$$
$$+ \frac{1}{\sigma_{ns}^2} \mathbf{\Phi^s}_k \left(\mathbf{x}_j^{(i)} - \sum_{l \neq k} \nu_{jl}^{(i)} \mathbf{\Phi^s}_l \right)^T + \frac{1}{\sigma_{na}^2} \mathbf{\Phi^a}_k \left(\mathbf{x}_j^{(i)} - \sum_{l \neq k} \nu_{jl}^{(i)} \mathbf{\Phi^a}_l \right)^T$$
$$+ C \sum_{\substack{J \in \Gamma^{(i)} \\ J=(k,a)}} \mathbb{I}_{\left\{ \sum_{l=1}^{N_i} \nu_{lk}^{(i)} \nu_{la}^{(i)} < 1 \right\}} \nu_{ja}^{(i)} + C \sum_{\substack{J \in \Gamma^{(i)} \\ J=(s,k)}} \mathbb{I}_{\left\{ \sum_{l=1}^{N_i} \nu_{ls}^{(i)} \nu_{lk}^{(i)} < 1 \right\}} \nu_{js}^{(i)}$$
$$+ C \mathbb{I}_{\left\{ \sum_{l=1}^{N_i} \nu_{lk}^{(i)} < 1, k \leq K_a + K_s \right\}} \tag{8.36}$$

where $L_k^{(i)}$ and $\mathbb{I}$ is an indicator variable. $L_k^{(i)}$ indicates whether an entity (action / subject) k is part of i^{th} video label set $\Gamma^{(i)}$ or not. This inturn enforces $\nu = 0$ for all ν satisfying eq. (8.18) and eq. (8.19).

The hyperparameter σ_{n*}^2 and σ_{A*}^2 can be set apriori or estimated from the data. The empirical estimation can be easily derived by maximizing the expected log-likelihood, which is similar to maximization step of EM algorithm. The closed form solution is given by,

$$\sigma_{A*}^2 = \frac{\sum_{k=1}^{K_{max}} D^* \sigma_{k*}^2 + \mathbf{\Phi}^*{}_k \mathbf{\Phi}^*{}_k^T}{K_{max} D^*} \tag{8.37}$$

$$\sigma_{n*}^2 = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N_i} \left(\mathbf{x}^*{}_j^{(i)T} \mathbf{x}^*{}_j^{(i)} - 2\mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)} \mathbf{A}^*] \mathbf{x}^*{}_j^{(i)} + \mathbb{E}_{\tilde{w}}[\mathbf{z}_{j.}^{(i)} \mathbf{U}^* \mathbf{z}_{j.}^{(i)T}] \right)}{(\sum_{i=1}^{M} N_i) D} \tag{8.38}$$

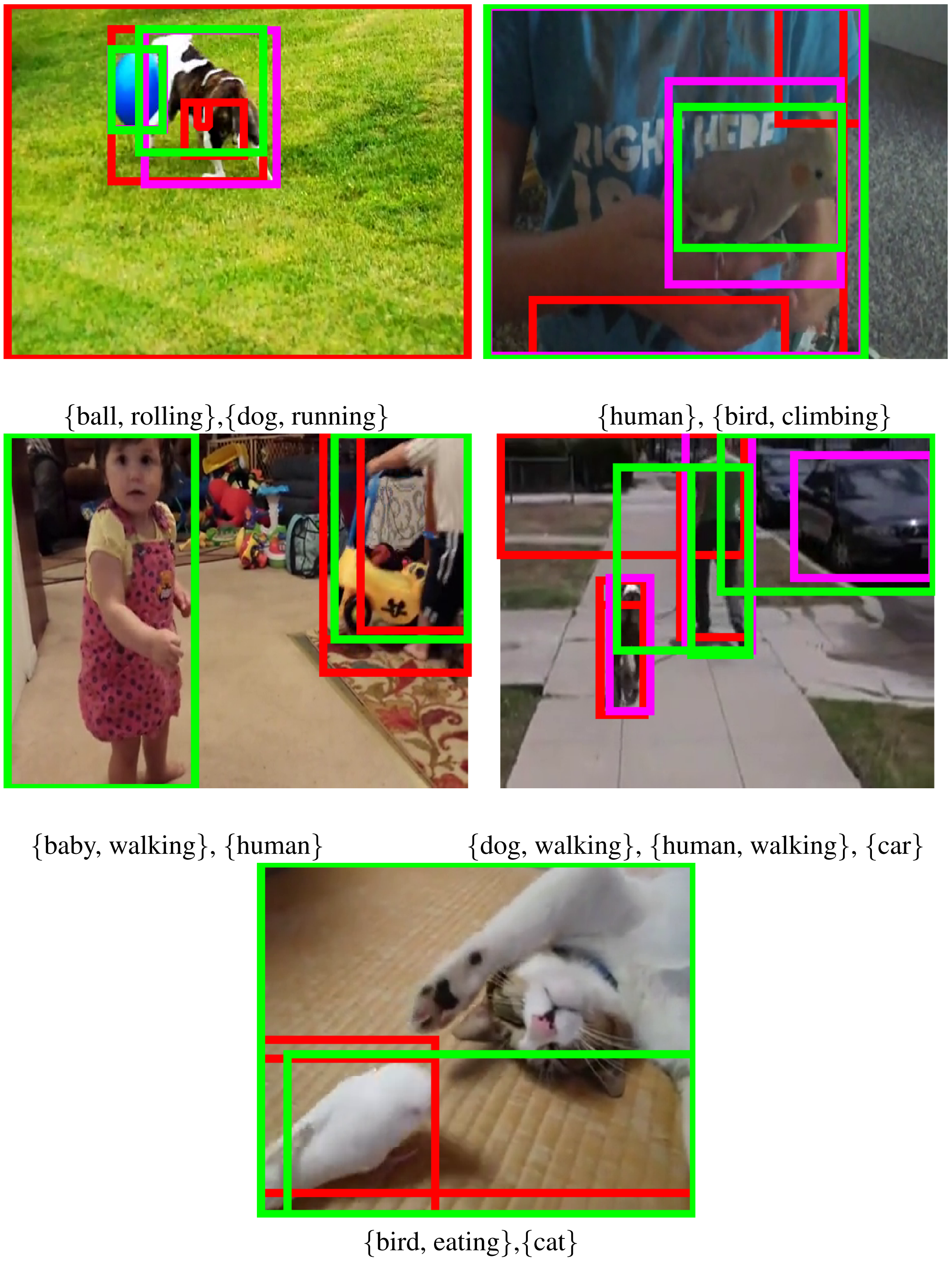

{ball, rolling},{dog, running}
{human}, {bird, climbing}
{baby, walking}, {human}
{dog, walking}, {human, walking}, {car}
{bird, eating},{cat}

Figure 8.6: Qualitative results of weakly supervised concept localization on A2D dataset using WSC-SIIBP algorithm. Tags are weak paired label input for the video. The red boxes represents generated proposals, green boxes represents the selected proposals using WSC-SIIBP algorithm and magenta boxes represents the groundtruth annotation. In case of overlapping boxes (proposals), only the last plotted rectangular box is visible. Boxes were plotted in the following order: red (first), magenta, green (last).

Part IV

Inverse Approach: Learning Priors

Chapter 9: Density Estimation using GAN

9.1 Introduction

In this chapter, we consider completely reverse approach of learning data prior from a given trained model. Generative adversarial networks are trained to generate data which is indistinguishable from the input training samples. Such adversarially trained networks are known to generate high quality samples without the need for explicitly specifying the likelihood function. Among the generative models, GANs are widely popular due to it simpler training and sampling procedure.

On the other hand, models such as RBMs [19] and DBMs [20], which are trained by explicitly maximizing the data likelihood, suffers from complex training and sampling procedure due to the use of mean field inference and MCMC. The variational autoencoder, which simultaneously learns a generative model and approximate inference, produces blurry samples and it is limited in its application to low-dimensional deep representation. Ideally, a good generative model is one which generates perceptually high quality samples while employing simpler training, exact and efficient sampling, exact and efficient inference of latent variables and exact log-likelihood computation.

Recently, there has been a renewed interest in developing highly efficient generative models. [299] proposed a new model using a set of real-valued non-volume preserving (real

NVP) transformations which are easily invertible as well as learnable. Subsequently, [300] and [301] independently developed adversarial training procedure for real NVPs in which the generator of GAN is replaced by real NVP transformations. [301] further demonstrated hybrid training procedure of real NVPs using GAN and MLE objective. The overall performance is shown to be better than the stand-alone training procedure. In summary, these new models closes high-quality sample generation gap with GANs while at the same time it can exactly evaluate the log-likelihood scores.

In this chapter, we develop a fast and efficient procedure to evaluate the log-likelihood of the generated and test data samples using GAN. We investigate this by constructing the jacobian of the generator transformation function. We demonstrate that this approximate density function can be employed for outlier detection and data augmentation for training a classifier.

9.2 Log-Likelihood Evaluation in Generative Models

GANs are a class of generative models consisting of a generator and discriminator network. The generator network $G_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^D$ samples low-dimensional $\mathbf{z} \in \mathbb{R}^d$ from a fixed tractable distribution $p_z(\mathbf{z})$ and transforms deterministically into higher dimensional image sample $\hat{\mathbf{x}} = G_\theta(\mathbf{z})$. The generated $\hat{\mathbf{X}}$ and the true $\mathbf{X}$ data samples are passed on to a discriminator which learns to distinguish fake samples from the true data by minimizing the negative cross-entropy loss. At the same time, the generator is trained to produce realistically looking fake samples. The overall objective function optimizes a minmax formulation.

Now, let us consider the following setting: Given an i.i.d data $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^M$ sampled from a training (approximately) distribution p_{data}, we are interested in computing the parametric density $p_\theta(\mathbf{x})$, where θ denotes the parameters of the GAN model. Evaluating the likelihood of a sample using GAN is challenging because the density p_θ is only defined implicitly using the prior density $p_z(\mathbf{z})$ and generator transformation function. However, if G_θ was an invertible transformation i.e., $\mathbf{z} = F_\theta(\mathbf{x})$ where $F_\theta = G_\theta^{-1}$ then, using change-of-variables, the parametric density can be given by,

$$p_\theta(\mathbf{x}) = p_z(\mathbf{z}) \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right| \tag{9.1}$$

$$= p_z(F_\theta(\mathbf{x})) \left| \det \frac{\partial F_\theta(\mathbf{x})}{\partial \mathbf{x}} \right| \tag{9.2}$$

$$\log p_\theta(\mathbf{x}) = \log p_z(F_\theta(\mathbf{x})) + \log \left| \det \frac{\partial F_\theta(\mathbf{x})}{\partial \mathbf{x}} \right| \tag{9.3}$$

where $p_z(.)$ is typically chosen to be isotropic Gaussian and $\frac{\partial F_\theta(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{D \times d}$ is the jacobian J of function F_θ at $\mathbf{x}$. We note that requiring generator function G_θ to be reversible imposes a constraint on the dimensionality of latent variable $\mathbf{z}$ i.e., it requires $d = D$. Furthermore, the determinant in (9.2) is non-zero only if J is a square matrix. However, in state-of-the-art GAN models, the generator functions are not invertible and $d << D$. In the following section we list simple mechanism to overcome above mentioned limitation such that the log-likelihood (9.2) can be evaluated using GANs.

9.3 Tractable Log-Likelihood in GAN

9.3.1 Computing determinant of non-square Jacobian matrix

The determinant of non-square matrix J cannot be computed directly. One way to actually compute this is by replacing det$|J|$ with $\sqrt{\det(J^T J)}$. Thus log-likelihood is now given by,

$$\log p_\theta(\mathbf{x}) = \log p_z(F_\theta(\mathbf{x})) + \log \left| \sqrt{\det(J^T J)} \right| \tag{9.4}$$

$$= \log p_z(F_\theta(\mathbf{x})) + \log \left| \sqrt{\prod_{j=1}^{d} \sigma_j^2} \right| \tag{9.5}$$

$$= \log p_z(F_\theta(\mathbf{x})) + \sum_{j=1}^{d} \log |\sigma_j| \tag{9.6}$$

where σ_j's are the singular values of jacobian matrix J. The singular value decomposition of J involving G_θ with high-dimensional latent variables $\mathbf{z}$ is computationally expensive to compute. Instead QR decomposition can be used efficiently to calculate the product of the singular value of matrix. This gives,

$$\log p_\theta(\mathbf{x}) = \log p_z(F_\theta(\mathbf{x})) + \sum_{j=1}^{d} \log |r_{ii}| \tag{9.7}$$

where r_{ii}'s are the diagonal entries of R matrix.

9.3.2 Inferring Latent Variable z

In order to compute log-likelihood of external data sample $\mathbf{x}$ using (9.7), one need to also infer the latent variable $\mathbf{z} = F_\theta(\mathbf{x})$. However, in the absence of inverse transformable generator function there are three possible ways to compute them approximately either explicitly/implicitly.

1. **Using encoder network:** Similar to variational autoencoder (VAE), we propose including an additional encoder network $F_\omega : \mathbb{R}^D \rightarrow \mathbb{R}^d$ which can be trained alongside the generator and the discriminator networks. Such bi-directional training mechanism was recently proposed in ALI [302] and Bi-GAN [303] work. The main difference between the two is that Bi-GAN uses deterministic F_ω network whereas ALI uses stochastic network. [303] further proves that under certain assumption the training procedure learns an encoder networks which is an exact inverse of the generator network. We propose to utilize a modified Bi-GAN model wherein the generator, encoder and discriminator networks are adversarially trained using the following objective,

$$\begin{aligned} \max_{\Theta} \quad & \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\mathcal{F}\left(D_\Theta(\mathbf{x}, F_\omega(\mathbf{x}))\right) + \frac{1}{2}\mathcal{F}\left(-D_\Theta(G_\theta(F_\omega(\mathbf{x})), F_\omega(\mathbf{x}))\right)\right] \\ & + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\mathcal{F}\left(-D_\Theta(G_\theta(\mathbf{z}), \mathbf{z})\right) \qquad (9.8) \\ \min_{\omega,\theta} \quad & \frac{1}{k}\sum_{j=1}^{k}\left|\mathbb{E}\left[\bar{D}_\Theta(\mathbf{x}, F_\omega(\mathbf{x}))_j\right] - \mathbb{E}\left[\bar{D}_\Theta(G_\theta(\mathbf{z}), \mathbf{z})_j\right]\right| \qquad (9.9) \end{aligned}$$

where $\mathcal{F}$ denotes softplus activation function and D represents discriminator network. The notation $\bar{D}$ represent the features extracted from the penultimate layer of the discriminator network. The discriminator network is trained using (9.8) whereas the generator and the encoder networks are updated using (9.9). The network is trained using similar training procedure as described in [303]. Once trained, the log-likelihood of any new data samples is easily computed by evaluating 9.7 using the encoder network.

2. **Optimizing for z:** Similar to Flow-GAN [301] approach, the above methodology can be only applied to restricted class of GAN models. Since many state-of-the-

art GAN models do not utilize an encoder network, we propose another simpler strategy to infer $\mathbf{z}$. Given a trained generator network G_θ and data sample $\mathbf{x}$, the corresponding latent variable $\mathbf{z}$ can be inferred by optimizing a following objective,

$$\min_{z} \quad \|G_\theta(\mathbf{z}) - \mathbf{x}\|_F^2 \tag{9.10}$$

(9.10) is easily solved using standard gradient descent solvers by back propagating through the fixed generator network and updating the input $\mathbf{z}$. Once $\mathbf{z}$ is estimated, the log-likelihood is easily evaluated using jacobian of generator transformation and applying,

$$\log p_\theta(\mathbf{x}) = \log p_z(\mathbf{z}) - \sum_{j=1}^{d} \log |r_{ii}| \tag{9.11}$$

The disadvantage of this approach is that it violates prior density model $p_z(.)$ and hence it is not clear how far (9.7) is applicable. Moreover, it seems computationally expensive because for every input data sample it requires number of back-propagation steps to infer $\mathbf{z}$.

3. **Using regression network:** This is a simplest approach in which a new regression network is trained independently to learn log-likelihood function using large number of samples and their corresponding log-likelihood. The training data is sampled from the generator network (of GAN in question) and their log-likelihood value is computed using (9.11) and input latent noise $\mathbf{z}$. Once trained, the log-likelihood evaluation for any new input data sample $\mathbf{x}$ is executed by a single forward pass through the regression network. Given that an objective is only to compute log-likelihood, this method do away with an exact inference of latent variables. In comparison to previous approaches, this methodology is strikingly fast and generalizes to any GAN

model. The architecture of regression network is detailed in section 9.4. Figure 9.1 summarizes our methodology.

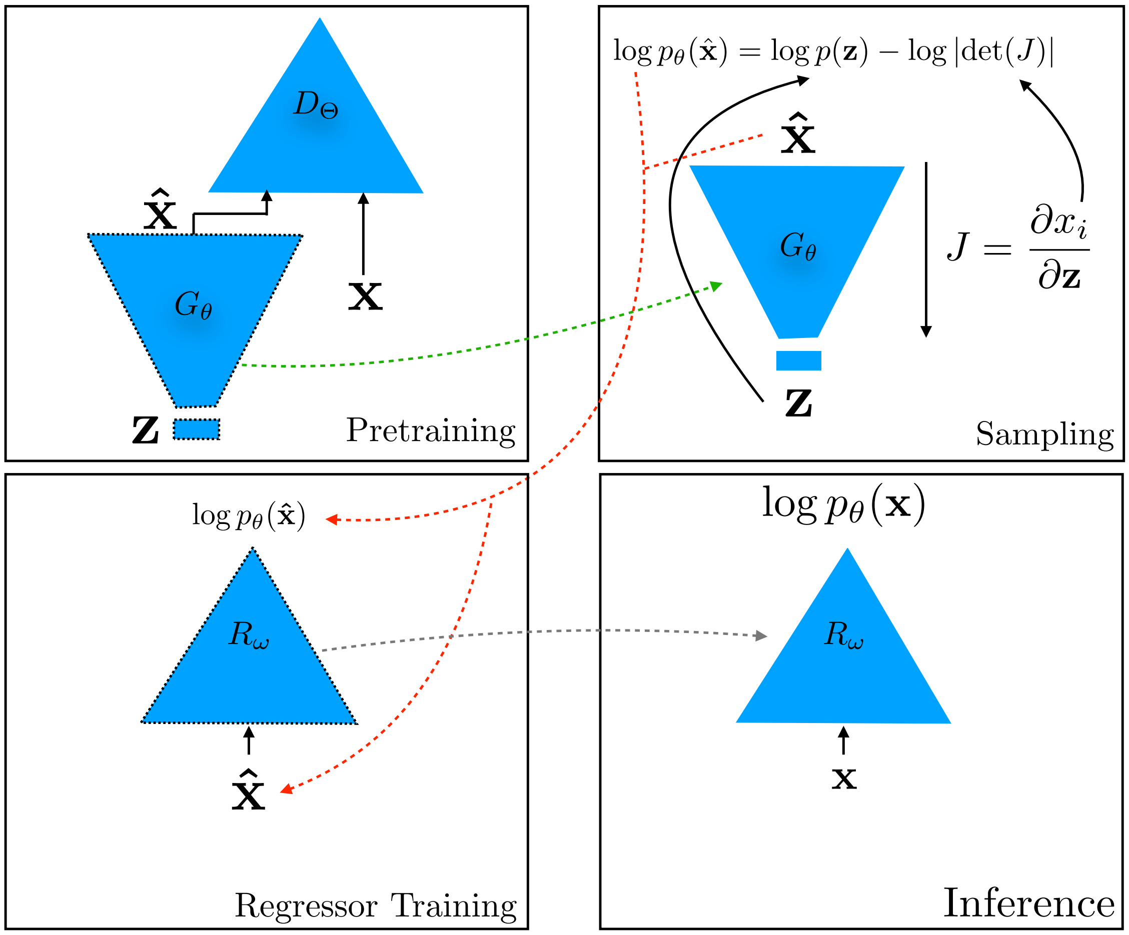

D_Θ
$\hat{\mathbf{x}}$
G_θ
$\mathbf{x}$
$\mathbf{z}$
Pretraining
$\log p_\theta(\hat{\mathbf{x}}) = \log p(\mathbf{z}) - \log|\det(J)|$
$J = \frac{\partial x_i}{\partial \mathbf{z}}$
Sampling
$\log p_\theta(\hat{\mathbf{x}})$
R_ω
Regressor Training
$\log p_\theta(\mathbf{x})$
Inference

Figure 9.1: Step-by-step procedure for learning and inferring log-likelihood function using regression network

9.3.3 Computing Jacobian

Given a trained GAN model, we can choose to estimate jacobian of generator/encoder transformation function either using finite difference or back-propagation.

Back-propagation: The jacobian matrix of a network can be computed iteratively one row after another. During each iteration, the gradient of a fix scalar output f_i w.r.t.

each and every input is computed using the back-propagation algorithm. Subsequently, the i^{th} row of J is updated. The number of back-propagation step is equal to the total number of scalar outputs. This suggest, for a generator network, the log-likelihood computation time quadratically increases with an output image size. On the other hand, an encoder network typically requires small computation time due to low dimensional latent variables.

Finite Difference Approximation: The gradients of a scalar function w.r.t. each input can be approximately computed by two forward pass using second order central finite difference scheme, four forward pass using fourth order scheme and so on [304]. The second order approximation scheme is given by,

$$\frac{\partial G_\theta(\mathbf{z})_i}{z_j} \approx \frac{G_\theta(\mathbf{z}+\epsilon\delta_{\mathbf{j}})_i - G_\theta(\mathbf{z}-\epsilon\delta_{\mathbf{j}})_i}{2\epsilon} \tag{9.12}$$

where ϵ is step-size and δ_i is an identity vector. Using finite differences, the jacobian of a generator network is computed using $2d$ forward passes as against $D(>> d)$ backward step using back-propagation mechanism. Furthermore, unlike the backward step, forward pass can be parallelized across input dimension and data samples thus providing enormous speed-up. However, this speed-up is obtained at the cost of accuracy. As discussed in [305] as well in section 9.4, the log-likelihood of data samples varies vastly with changing ϵ.

9.4 Experiments

We start with discussing experimental results on data sampled from mixture of eight gaussian following which we present results on MNIST and Cifar-10 dataset.

9.4.1 Mixture of Gaussians

To illustrate the advantage of our proposed methodology, we experiment on a simple GAN architecture with fully connected layers using the toy dataset. The constructed toy example is inspired by the one presented in [12]. The two dimensional data is sampled from the mixture of eight Gaussians with their means equally spaced around the circle of radius 2 centered at $(0, 0)$. The standard deviation of each Gaussian is set at 0.01. The two dimensional latent vector $\mathbf{z}$ is sampled from the multivariate Gaussian distribution.

We start with experimenting on bi-directional GAN which includes an encoder network. The generator and discriminator networks consist of four fully connected hidden layers, each with 200 and 400 hidden units respectively followed by tanh activation units. Similarly, the encoder network consists of two FC layers each with 200 hidden units with tanh activations. The final layer of each entity has linear activation. The network is optimized using minmax objective of type (6.1). We use adam solver with its default parameters (i.e., learning rate $= 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) and with input batch size of 512.

The generated two dimensional samples are plotted in Figure 9.2(a). Furthermore, Figure 9.2(b) displays top 98% points of the same generated data which were picked based on the computed log-likelihood scores while Figure 9.2(c) displays reconstructed samples corresponding to input test samples (sampled from training distribution). These results simply suggest that the generator and encoder network fits well onto the training data and only generate samples which approximately belongs to modes of training data.

Although, the trained model works exceedingly well for in-sample, the above results do not provide any hint about its performance on out-of-sample data. In order to assess this, we evaluate log-likelihood of million test samples uniformly sampled from $[-3, 3]$ grid in each dimension. Figure 9.2(d) and 9.2(e) plots the output when evaluated using encoder and generator transformation function (using regressor) respectively. The result suggest that an encoder transformation function assigns unusual high probability to samples closer to center $(0, 0)$. On the other hand, the results are relatively better when evaluated using generator transformation function.

We also compare results obtained using Flow-GAN model which trains GAN using invertible transformable generator networks. We consider two training objective - MLE and hybrid of MLE and GAN and compare the log-likelihood scores in Figure 9.3. Unlike the training with MLE objective, hybrid model outputs bizare results.

9.4.1.1 Outlier Detection

Here we demonstrate that our proposed log-likelihood evaluation approach assigns high score only to relevant samples from training modes and low scores to samples from missing mode. In order to experiment this, Bi-GAN model is trained only with samples from seven modes. Figure 9.4 plots the generated samples and log-likelihood evaluation on uniform grid.

9.4.1.2 Regressor model

Since the evaluation using regressor model is relatively better than using encoder function, it is inessential to continue using Bi-GAN model. Instead, we experiment with generic GAN model, i.e., model without an encoder networks, by additionally training a regressor network. The generator and discriminator networks consist of two fully connected hidden layers, each with 128 hidden units followed by tanh activation units. On the other hand, the regressor networks consist of five fully connected layers with the following number of hidden units: $16, 32, 64, 128, 256$ each followed by tanh activation units. Since regressor network only sees (generated) samples which are spread around the training modes, it is not clear how the network will learn to interpolate log-likelihood output for out-of-sample data. Ideally, one would expect regressor network to learn to assign very low scores to such samples. But there is no guarantee that neural network would behave accordingly. In order to prevent interpolation towards the higher end for an unseen data, we proposed a simple activation function for the final layer,

$$\mathcal{F}(x) = x - \alpha \max(x - \beta, 0) \tag{9.13}$$

$$= x - \alpha \text{ReLU}(x - \beta) \tag{9.14}$$

Figure 9.5 illustrate an example of $\mathcal{F}(x)$ for $\alpha = \beta = 2$.

Similar to earlier experiment, we consider GAN trained with missing mode. Figure 9.6 plots log-likelihood evaluation on test data sampled from original MoG and uniform grid respectively. These results are indeed superior to that in Figure 9.4 clearly indicating that with well-designed regressor network, we can achieve fast and efficient inference.

9.4.2 MNIST

Next, we test our proposed approach on GANs trained with real datasets such as MNIST and Cifar-10. The popular DC-GAN [172] architecture was considered and jacobian is computed on deep features. Deep features are extracted from any pretrained classifier network. For our experiment, we consider a classifier network with three FC layers respectively consisting of $625, 625, 10$ hidden units and followed by ReLU activation. Same regressor network as above is utilized for log-likelihood evaluation. Figure 9.7 displays top and lowest 100 test samples from each class based on the log-likelihood scores.

9.4.2.1 Without Deep Features

In contrast to procedure followed for MoG experiments, we chose to compute jacobian on deep feature for high-dimensional dataset. Our initial experiments using image features produced mixed results in that the perceptually good test samples were scored lowest. Figure 9.8 displays the 100 top and lowest scored test samples while Figure 9.9 displays the same for generated samples.

9.4.2.2 With Finite Difference

As we discussed earlier, finite difference is the most efficient and fastest approach to compute the jacobian matrix. In this section, we demonstrate that the log-likelihood computation is sensitive to the step-size ϵ. We use the same setting as in Figure 9.7 except that the jacobian is now computed using finite differences. Figure 9.10 compares best generated output sample at different step-size settings. It can be easily seen that with decreasing step-size many bizarre samples are scored high. Furthermore, the disparity in histogram plot of log-likelihood advocates that ϵ value should be carefully chosen for different dataset and deep features.

9.4.2.3 Outlier Detection

Similar to experimental setting for MoG, we consider evaluating log-likelihood score on samples from class not seen during training (of both classifier and GANs). The highest and the lowest scored samples for three different scenario for missing class is displayed in Figure 9.11(a). Visual results exhibits successful isolation of test samples belonging to missing class. The histogram plot in Figure 9.11(b) indicates the shift in distribution of sample log-likelihood of missing class '9'. Finally, Figure 9.11(c) quantitatively compares the performance of outlier detection using the computed log-likelihood scores as against using logit output of classifier. Log-likelihood score leads to moderate increase in AUC measure.

9.4.3 Cifar-10

Figure 9.12 displays the best and the lowest scored Cifar-10 test samples. We note that images with high-frequency content are scored lowest. This might be due to the fact that such images usually occupies isolated regions on the manifold. Furthermore, most images with sky blue background were scored highest.

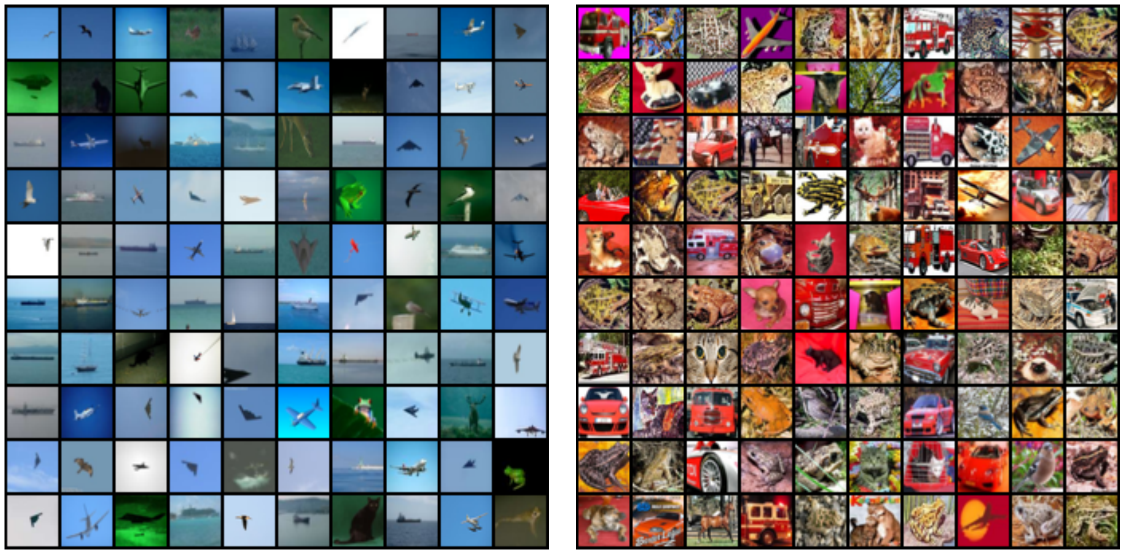
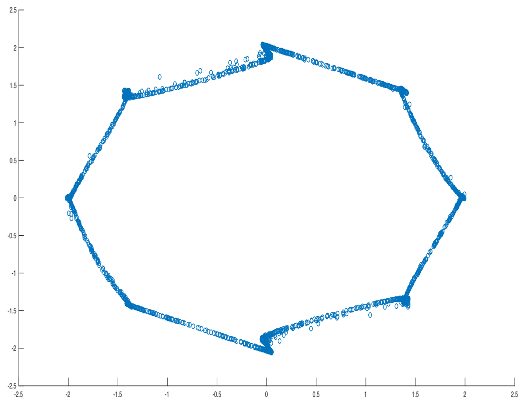


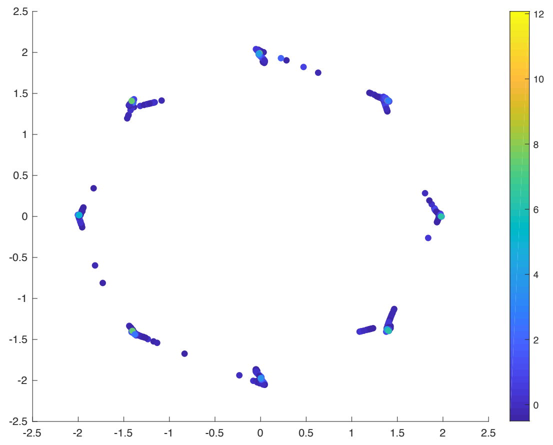
Figure 9.12: Log-likehood evaluation on Cifar test data. Based on log-likelihood score **Left**: Top 100 samples, **Right**: Lowest 100 samples.

9.5 Conclusion

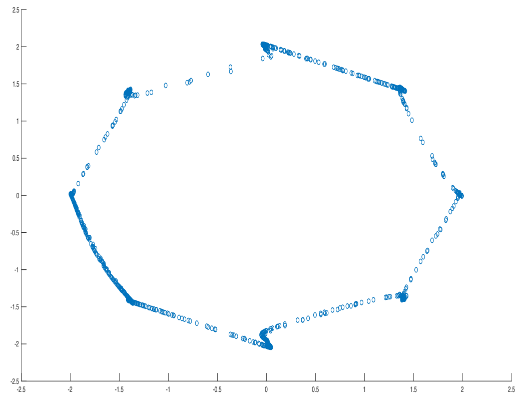
In this chapter, we proposed simple and efficient approach for log-likelihood evaluation of external sample using GANs. The inference procedure easily scales to high-dimension and large data samples. This mechanism can also be applied to other recently developed implicit generative models such as adversarial autoencoder [177]. In the future, we wish to investigate the stability/sensitivity issues of finite difference mechanism which can lead to further speed-up in sampling procedure. We would also like to draw meaningful quantitative comparison against MLE and Flow-GAN training models. Finally, based on log-likelihood scores we wish to carefully explore new scheme for data augmentation in order to boost conventional classifier performance.



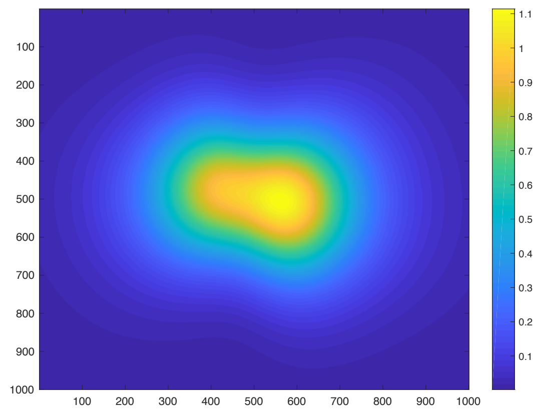
(a)



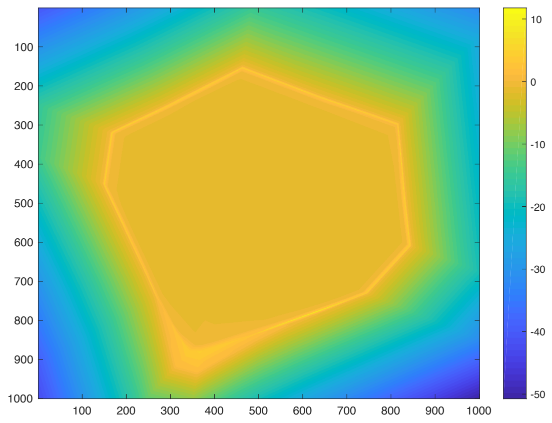
(b)



(c)

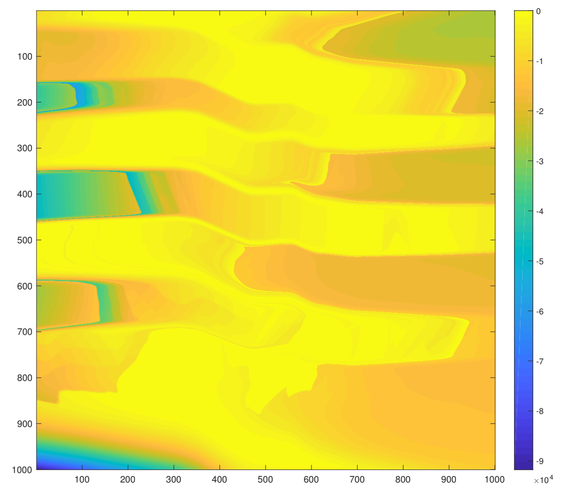


(d)

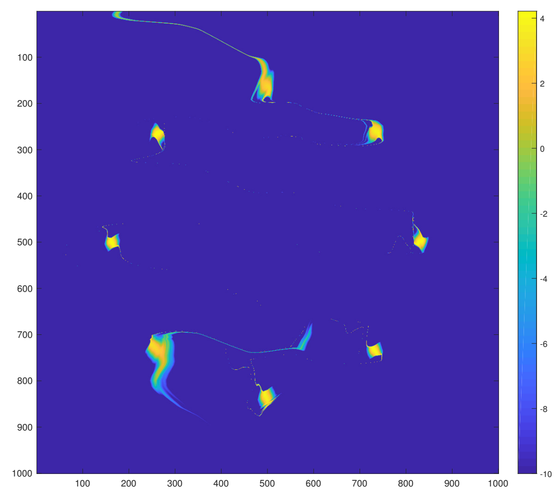


(e)

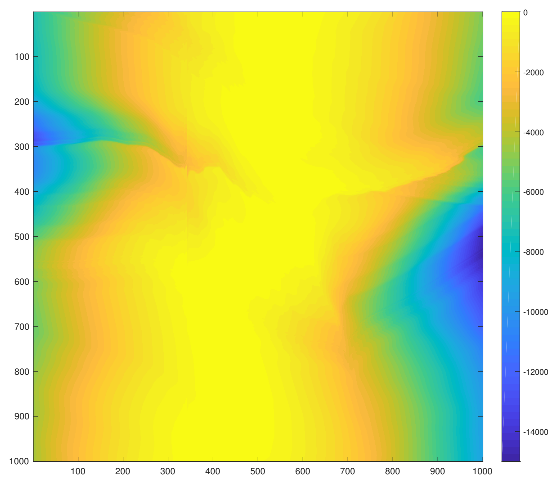
Figure 9.2: BiGAN model outputs. (a) Generated samples. (b) Top generated samples based on log-likelihood score computed using (9.11). and (c) Reconstructed test samples. Log-likelihood evaluation of test samples sampled from uniform grid using, (d) encoder network and (e) generator network + regressor training.



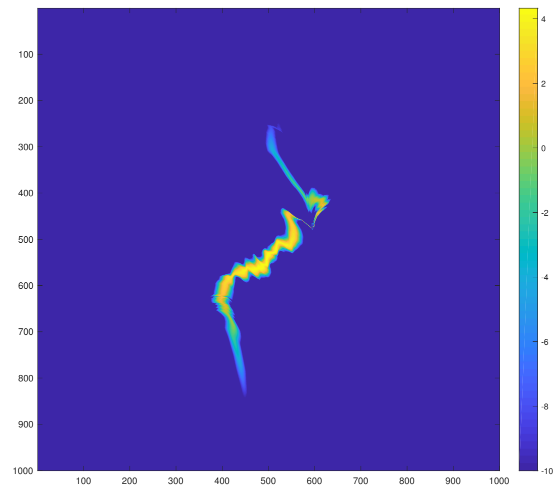
(a)



(b)

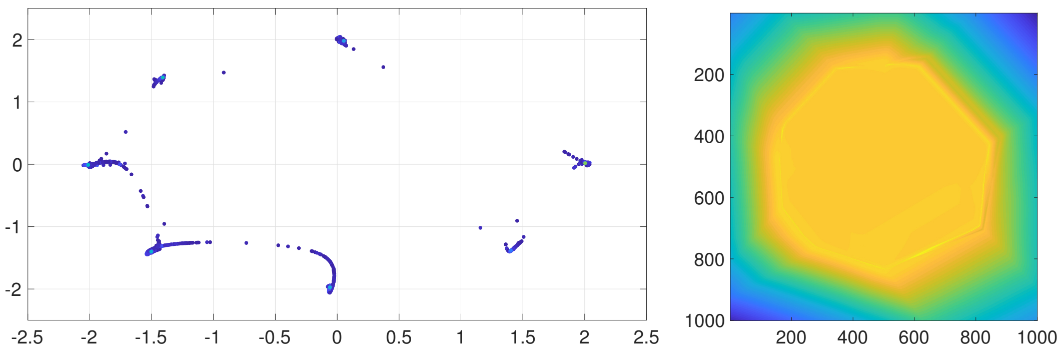


(c)



(d)

Figure 9.3: Flow-GAN model outputs. Log-likelihood output when evaluated on test samples uniformly sampled from fixed grid. Trained using (a), (b) MLE objective and (c),(d) Hybrid of MLE and GAN objective. For better understanding and visualization, (b) and (d) displays the same log-likelihood plot which is thresholded from bottom at particular value.



(a) (b)

Figure 9.4: BiGAN model trained without a gaussian mode. (a) Top 98% generated samples based on log-likelihood score and (b) Log-likelihood evaluation on uniform grid using regressor model.

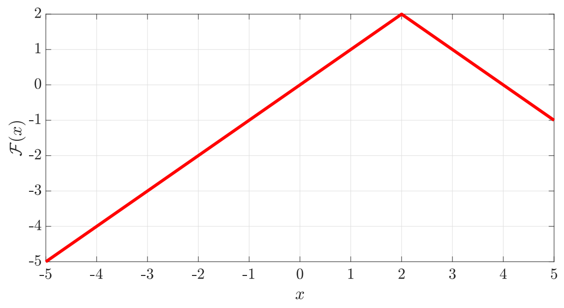
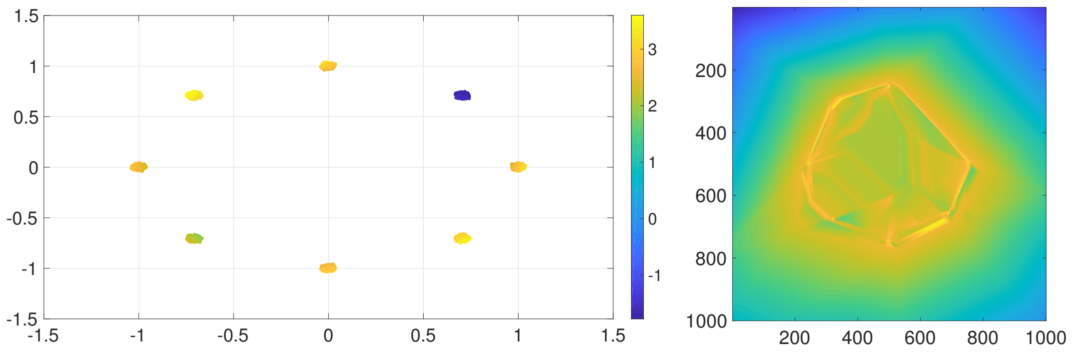


Figure 9.5: Illustrative example of activation function used in the final layer of regressor network.



(a) (b)

Figure 9.6: GAN model trained without a mode and evaluated using regressor model. (a) Log-likelihood score of test data sampled from original GAN and (b) Log-likelihood evaluation on uniform grid.

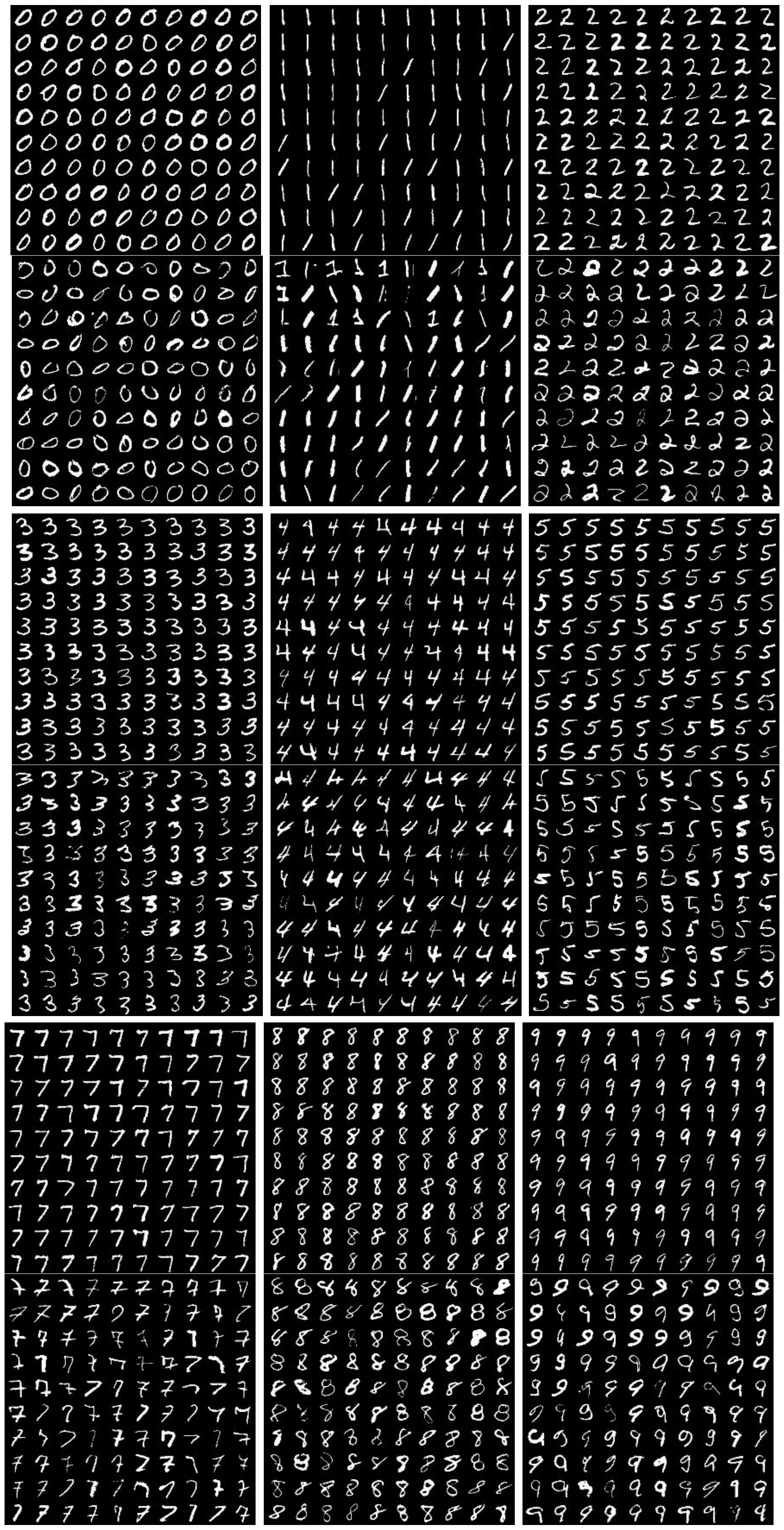


Figure 9.7: Log-likehood evaluation on MNIST test data using deep features. Based on log-likelihood score, for each class it displays **Top**: Top 100 samples, **Bottom**: Lowest 100 samples

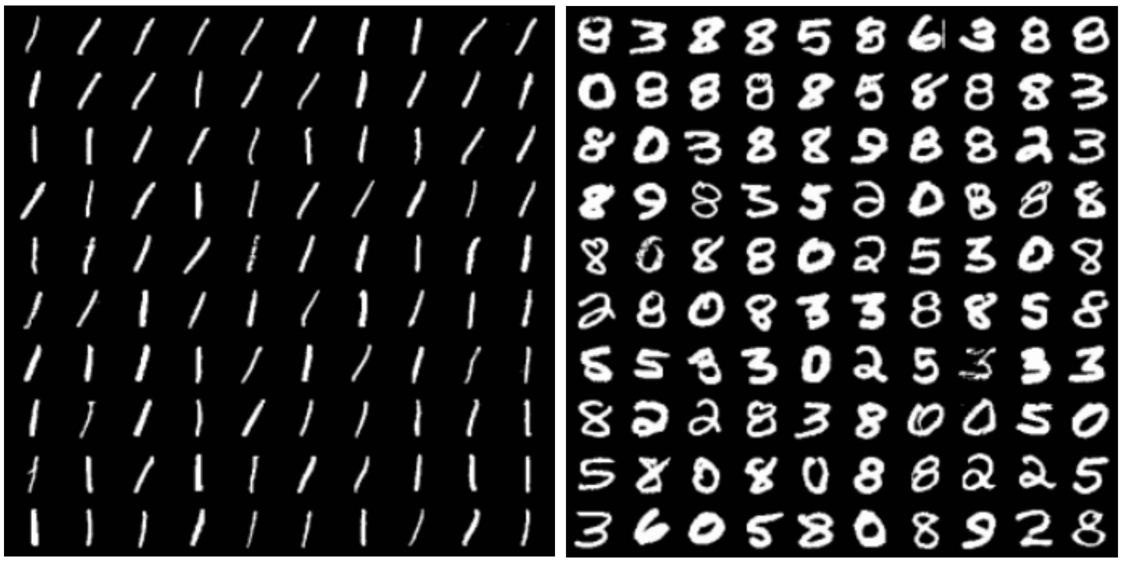


Figure 9.8: Log-likehood evaluation on MNIST test data using image features. Based on log-likelihood score **Left**: Top 100 samples, **Right**: Lowest 100 samples

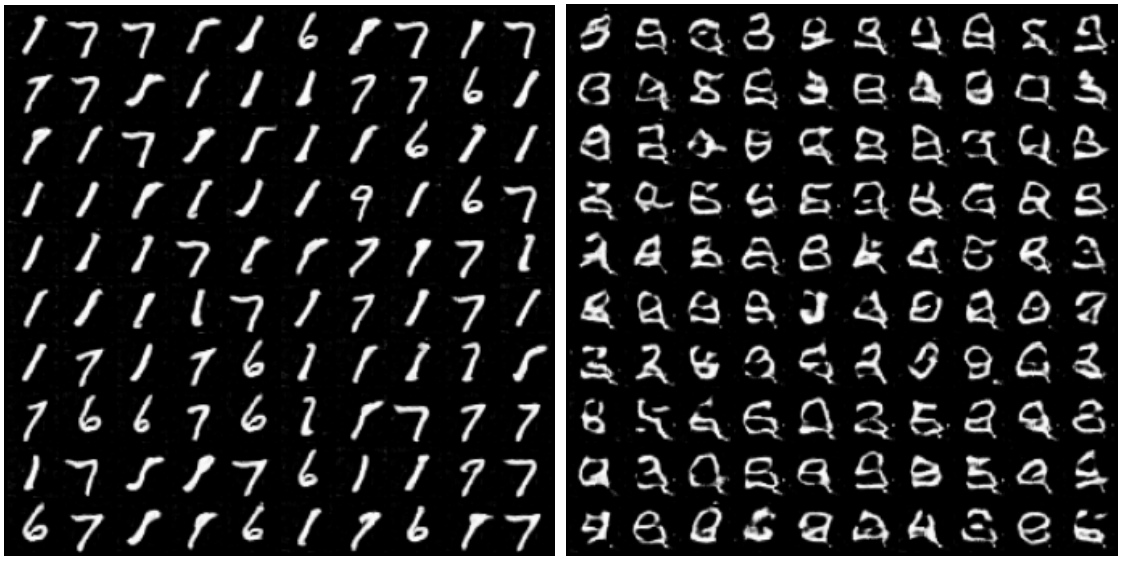
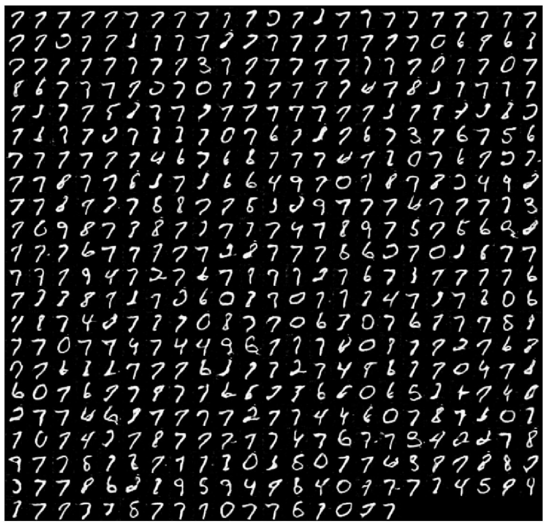
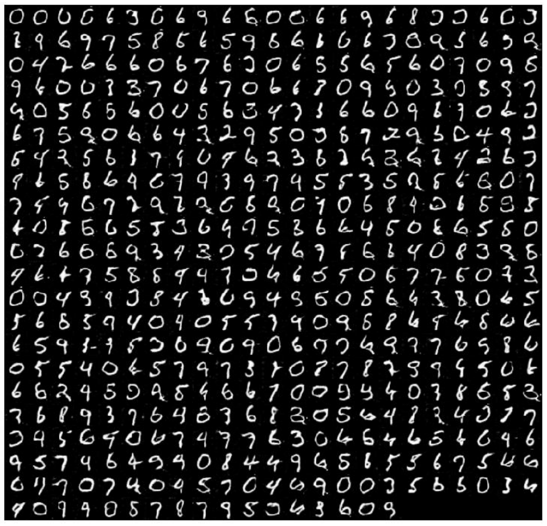


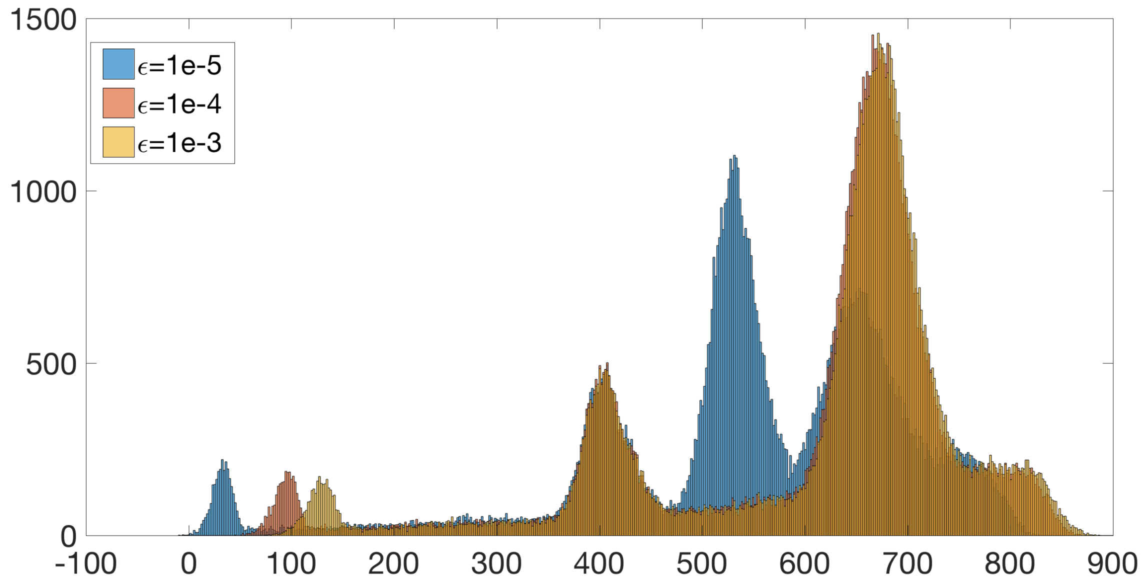
Figure 9.9: Log-likehood evaluation on generated MNIST data. Based on log-likelihood score **Left**: Top 100 samples, **Right**: Lowest 100 samples



(a) $\epsilon = 10^{-3}$

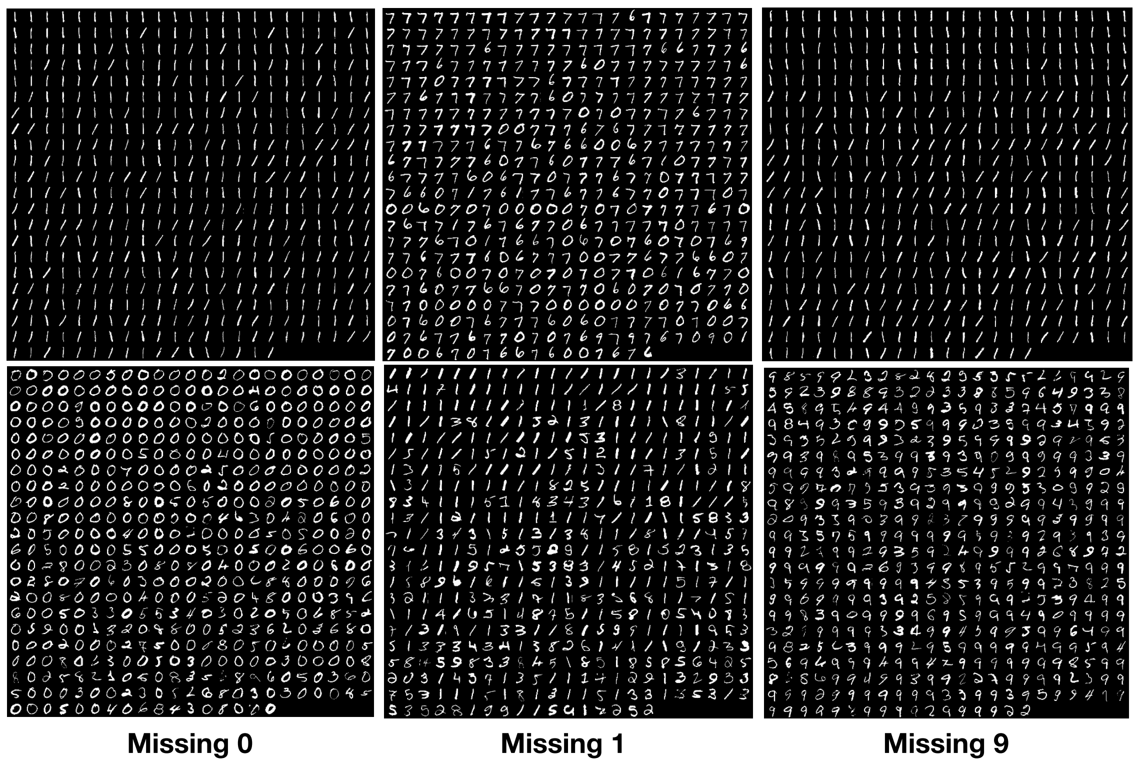


(b) $\epsilon = 10^{-5}$

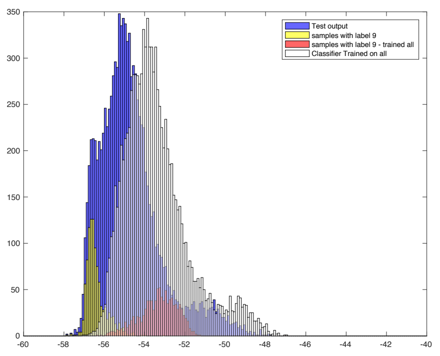


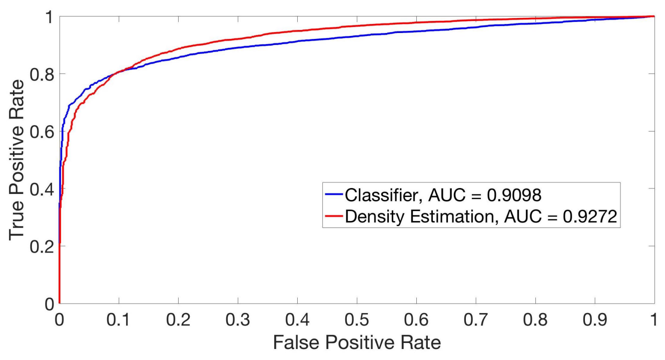
(c)

Figure 9.10: Log-likehood evaluation on generated MNIST data at different epsilon settings. (a), (b) displays best samples and (c) histogram of log-likelihood outputs.



(a)





(b) (c)

Figure 9.11: Log-likehood evaluation on MNIST test data when trained without a missing class. (a) Based on log-likelihood score **Top**: Highest scored samples, **Bottom**: Lowest scored samples. (b) Histogram of log-likelihood score when trained with missing class 9. The score of test samples from class 9 is lowest. and (c) RoC plot compares outlier detection based on estimated log-likelihood scores as against the logit scores of classifier.

Bibliography

[1] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[2] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[3] Volkan Cevher, Marco F Duarte, Chinmay Hegde, and Richard Baraniuk. Sparse signal recovery using markov random fields. In *NIPS*, pages 257–264, 2009.

[4] Sohil Shah, Tom Goldstein, and Christoph Studer. Estimating sparse signals with smooth support via convex programming and block sparsity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5906–5915, 2016.

[5] Liviu I Nicolaescu. *Lectures on the Geometry of Manifolds*. World Scientific, 2007.

[6] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences (PNAS)*, 114(37), 2017.

[7] Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv:1803.01449*, 2018.

[8] Sohil Shah, Abhay Kumar Yadav, Carlos D Castillo, David W Jacobs, Christoph Studer, and Tom Goldstein. Biconvex relaxation for semidefinite programming in computer vision. In *European Conference on Computer Vision*, pages 717–735. Springer, 2016.

[9] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[10] Qinqing Zheng and John Lafferty. A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements. In *Neural Information Processing Systems (NIPS)*. 2015.

[11] Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. In *International Conference on Learning Representations (ICLR)*, 2018.

[12] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.

[13] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *CVPR*, 2017.

[14] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICLR*, 2017.

[15] Sohil Atul Shah, Pallabi Ghosh, Larry S. Davis, and Tom Goldstein. Stacked generative adversarial networks. In *under review ECCV*, 2018.

[16] Sohil Shah, Kuldeep Kulkarni, Arijit Biswas, Ankit Gandhi, Om Deshmukh, and Larry S Davis. Weakly supervised learning of heterogeneous concepts in videos. In *European Conference on Computer Vision*, pages 275–293. Springer, 2016.

[17] Yikang Li, Wanli Ouyang, and Xiaogang Wang. Vip-cnn: A visual phrase reasoning convolutional neural network for visual relationship detection. *arXiv preprint arXiv:1702.07191*, 2017.

[18] Bohan Zhuang, Lingqiao Liu, Chunhua Shen, and Ian Reid. Towards context-aware interaction recognition. *arXiv preprint arXiv:1703.06246*, 2017.

[19] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.

[20] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[21] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[22] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[24] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[25] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *JMLR*, 12:3371–3412, 2011.

[26] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *JMLR*, 12:2777–2824, 2011.

[27] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, pages 19–53, 2011.

[28] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *ICML*, pages 433–440. ACM, 2009.

[29] László A Jeni, András LHorincz, Zoltán Szabó, Jeffrey F Cohn, and Takeo Kanade. Spatio-temporal event classification using time-series kernel based structured sparsity. In *ECCV 2014*, pages 135–150.

[30] Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. *arXiv preprint arXiv:0909.1440*, 2009.

[31] Michael Leigsnering, Farhan Ahmad, Moeness Amin, and Abdelhak Zoubir. Multipath exploitation in through-the-wall radar imaging using sparse reconstruction. *Aerospace and Electronic Systems, IEEE Transactions on*, 50(2):920–939, 2014.

[32] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[33] Cho-Chun Cheng, Guan-Ju Peng, and Wen-Liang Hwang. Subband weighting with pixel connectivity for 3-d wavelet coding. *IEEE Transactions on Image Processing*, 18(1):52–62, 2009.

[34] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[35] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

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

[37] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[38] Tom Goldstein and Stanley Osher. The split Bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[39] Xavier Bresson and Tony F Chan. Fast dual minimization of the vectorial total variation norm and applications to color image processing. *Inverse Problems and Imaging*, 2(4):455–484, 2008.

[40] Shunsuke Ono and Isao Yamada. Decorrelated vectorial total variation. In *CVPR 2014*, pages 4090–4097.

[41] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[42] Yangmuzi Zhang, Zhuolin Jiang, and Larry S Davis. Learning structured low-rank representations for image classification. In *CVPR, 2013*, pages 676–683.

[43] Wei Deng, Wotao Yin, and Yin Zhang. Group sparse optimization by alternating direction method. In *SPIE Optical Engineering+ Applications*, pages 88580R–88580R. International Society for Optics and Photonics, 2013.

[44] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[45] Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. In *NIPS*, pages 352–360, 2011.

[46] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *PAMI*, 35(1):171–184, 2013.

[47] Tom Goldstein, Christoph Studer, and Richard G. Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *CoRR*, abs/1411.3406, 2014.

[48] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[49] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[50] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE CVPR*, 2013.

[51] Elaine T Hale, Wotao Yin, and Yin Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*, 2007.

[52] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[53] Liyuan Li, Weimin Huang, IY-H Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

[54] Chih-Fan Chen, Chia-Po Wei, and Yu-Chiang Frank Wang. Low-rank matrix recovery with structural incoherence for robust face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2618–2625. IEEE, 2012.

[55] Long Ma, Chunheng Wang, Baihua Xiao, and Wen Zhou. Sparse representation for face recognition based on discriminative low-rank dictionary learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2586–2593. IEEE, 2012.

[56] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2009.

[57] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010.

[58] Zhuolin Jiang, Zhe Lin, and Larry S Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1697–1704. IEEE, 2011.

[59] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence (PAMI)*, 22(8), 2000.

[60] Toby Hocking, Jean-Philippe Vert, Francis R. Bach, and Armand Joulin. Clusterpath: an algorithm for clustering using convex fusion penalties. In *Proc. International Conference on Machine Learning (ICML)*, 2011.

[61] Eric C. Chi and Kenneth Lange. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24, 2015.

[62] Yuliya Marchetti and Qing Zhou. Solution path clustering with adaptive concave penalty. *Electronic Journal of Statistics*, 8(1), 2014.

[63] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[64] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proc. Symposium on Discrete Algorithms (SODA)*, 2007.

[65] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814), 2007.

[66] M.R. Brito, E.L. Chávez, A.J. Quiroz, and J.E. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35, 1997.

[67] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5), 2002.

[68] Michael J. Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 19(1), 1996.

[69] Peter J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society, Series B*, 46(2), 1984.

[70] Stuart Geman and Donald E. McClure. Statistical methods for tomographic image reconstruction. *Bulletin of the International Statistical Institute*, 52, 1987.

[71] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[72] Hossein Mobahi and John W. Fisher III. A theoretical analysis of optimization by Gaussian continuation. In *Proc. Conference on Artificial Intelligence (AAAI)*, 2015.

[73] Amir Beck. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015.

[74] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2), 2010.

[75] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-SNE. *JMLR*, 9, 2008.

[76] Zhangyang Wang, Yingzhen Yang, Shiyu Chang, Jinyan Li, Simon Fong, and Thomas S. Huang. A joint optimization framework of sparse coding and discriminative clustering. In *Proc. IJCAI*, 2015.

[77] Nicolas Flammarion, Balamurugan Palanisamy, and Francis R. Bach. Robust discriminative clustering with sparse regularizers. *arXiv:1608.08052*, 2016.

[78] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Transactions on Signal Processing*, 54(11), 2006.

[79] Neal Parikh and Stephen P. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3), 2014.

[80] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5, 2004.

[81] Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of Down syndrome. *PLoS ONE*, 10(6), 2015.

[82] Moshe Lichman. UCI machine learning repository, 2013. `http://archive.ics.uci.edu/ml`.

[83] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[84] Fevzi Alimoglu and Ethem Alpaydin. Combining multiple representations and classifiers for pen-based handwritten digit recognitio. In *Proc. ICDAR*, 1997.

[85] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence (PAMI)*, 23(6), 2001.

[86] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.

[87] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (COIL-100). Technical Report CUCS-006-96, Columbia University, 1996.

[88] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *PAMI*, 36(11), 2014.

[89] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010. `http://eigen.tuxfamily.org`.

[90] Deli Zhao and Xiaoou Tang. Cyclizing clusters via zeta function of a graph. In *Proc. NIPS*, 2008.

[91] Feiping Nie, Dong Xu, Ivor W. Tsang, and Changshui Zhang. Spectral embedded clustering. In *Proc. IJCAI*, 2009.

[92] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10), 2010.

[93] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *Proc. ECCV*, 2012.

[94] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11), 2013.

[95] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – A knowledge reuse framework for combining multiple partitions. *JMLR*, 3, 2002.

[96] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *JMLR*, 11, 2010.

[97] Marcilio CP de Souto, Ivan G. Costa, Daniel SA de Araujo, Teresa B. Ludermir, and Alexander Schliep. Clustering cancer gene expression data: A comparative study. *BMC Bioinformatics*, 9, 2008.

[98] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Transactions on Information Theory*, 21(1), 1975.

[99] Keith Ball. An elementary introduction to modern convex geometry. In *Flavors of Geometry*. 1997.

[100] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory (ICDT)*, 1999.

[101] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*. 2004.

[102] Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[103] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1), 2009.

[104] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2), 2011.

[105] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *International Conference on Computer Vision (ICCV)*, 2017.

[106] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. ICML*, 2016.

[107] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proc. CVPR*, 2016.

[108] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning (ICML)*, 2017.

[109] Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Neural Information Processing Systems (NIPS)*, 1994.

[110] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006.

[111] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

[112] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[113] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, 11, 2010.

[114] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 2014.

[115] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining (KDD)*, 1996.

[116] Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11), 2011.

[117] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[118] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[119] Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research (JMLR)*, 15, 2014.

[120] Jens Keuchel, Christoph Schno, Christian Schellewald, and Daniel Cremers. Binary partitioning, perceptual grouping, and restoration with semidefinite programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1364–1379, 2003.

[121] Philip HS Torr. Solving markov random fields using semi definite programming. In *Artificial Intelligence and Statistics*, 2003.

[122] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[123] Mica Arie-Nachimson, Shahar Z Kovalsky, Ira Kemelmacher-Shlizerman, Amit Singer, and Ronen Basri. Global motion estimation from point matches. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 81–88. IEEE, 2012.

[124] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[125] Kaushik Mitra, Sameer Sheorey, and Rama Chellappa. Large-scale matrix factorization with missing data under additional constraints. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1651–1659, 2010.

[126] Zhi-Quan Luo, Wing-kin Ma, Anthony Man-Cho So, Yinyu Ye, and Shuzhong Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, May 2010.

[127] Jean B Lasserre. An explicit exact sdp relaxation for nonlinear 0-1 programs. In *Integer Programming and Combinatorial Optimization*, pages 293–303. Springer, 2001.

[128] Stephen Boyd and Lieven Vandenberghe. Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization. In *Communications, Computation, Control, and Signal Processing*, pages 279–287. Springer, 1997.

[129] Ady Ecker, Allan D Jepson, and Kiriakos N Kutulakos. Semidefinite programming heuristics for surface reconstruction ambiguities. In *Computer Vision–ECCV 2008*, pages 127–140. Springer, 2008.

[130] Sameer Shirdhonkar and David W Jacobs. Non-negative lighting and specular object recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1323–1330. IEEE, 2005.

[131] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, July 1996.

[132] Matthias Heiler, Jens Keuchel, and Christoph Schnörr. Semidefinite clustering for image segmentation with a-priori knowledge. In *Pattern Recognition*, pages 309–317. Springer, 2005.

[133] Chunhua Shen, Junae Kim, and Lei Wang. A scalable dual approach to semidefinite metric learning. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2601–2608. IEEE, 2011.

[134] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase retrieval using alternating minimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2796–2804, 2013.

[135] Emmanuel J Candès, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *Information Theory, IEEE Transactions on*, 61(4):1985–2007, 2015.

[136] Jason D Lee, Ben Recht, Nathan Srebro, Joel Tropp, and Ruslan R Salakhutdinov. Practical large-scale optimization for max-norm regularization. In *Advances in Neural Information Processing Systems*, pages 1297–1305, 2010.

[137] Peng Wang, Chunhua Shen, and Anton van den Hengel. A fast semidefinite approach to solving binary quadratic problems. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[138] Subhransu Maji, Nisheeth K Vishnoi, and Jitendra Malik. Biased normalized cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2057–2064. IEEE, 2011.

[139] Michel Journée, F Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[140] Zhiwu Huang, Ruiping Wang, Shiguang Shan, Xianqiu Li, and Xilin Chen. Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 720–729, 2015.

[141] Mehrtash T Harandi, Mathieu Salzmann, and Richard Hartley. From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices. In *European Conference on Computer Vision*, pages 17–32. Springer, 2014.

[142] Xiao Bai, Hang Yu, and Edwin R Hancock. Graph matching using spectral embedding and alignment. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 398–401. IEEE, 2004.

[143] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Computer Vision and Image Understanding*, 117(11):1610–1627, 2013.

[144] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[145] Peng Wang, Chunhua Shen, and Anton van den Hengel. Efficient SDP inference for fully-connected CRFs based on low-rank decomposition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[146] Christian Schellewald and Christoph Schnörr. Probabilistic subgraph matching based on convex relaxation. In *Energy minimization methods in computer vision and pattern recognition*, pages 171–186. Springer, 2005.

[147] Carl Olsson, Anders P Eriksson, and Fredrik Kahl. Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[148] Kevin Lang. Fixing two weaknesses of the spectral method. In *Advances in Neural Information Processing Systems (NIPS)*, pages 715–722, 2005.

[149] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[150] K. C. Toh, M.J. Todd, and R.H. Tutuncu. Sdpt3 - a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1998.

[151] Takayuki Okatani and Koichiro Deguchi. On the wiberg algorithm for matrix factorization in the presence of missing components. *International Journal of Computer Vision*, 72(3):329–337, 2007.

[152] John C. Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[153] Jim Douglas and James E Gunn. A general formulation of alternating direction methods. *Numerische Mathematik*, 6(1):428–453, 1964.

[154] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[155] Tom Goldstein, Christoph Studer, and Richard Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406, 2014.

[156] Song Wang and Jeffrey Mark Siskind. Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:675–690, 2003.

[157] Stella X Yu and Jianbo Shi. Segmentation given partial grouping constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):173–183, 2004.

[158] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms (2008), 2012.

[159] Ruiping Wang, Huimin Guo, Larry S Davis, and Qionghai Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2496–2503. IEEE, 2012.

[160] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[161] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, pages 318–335, 2016.

[162] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017.

[163] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.

[164] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[165] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *ICLR Workshop*, 2017.

[166] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1180–1189, 2015.

[167] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pages 2760–2769, 2016.

[168] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. In *ICLR*, 2016.

[169] Martín Abadi and David G Andersen. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016.

[170] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *NIPS*, pages 5041–5049, 2016.

[171] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[172] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[173] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[174] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[175] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[176] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *ICLR*, 2017.

[177] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. In *ICLR*, 2016.

[178] Yujia Li, Kevin Swersky, and Richard S Zemel. Generative moment matching networks. In *ICML*, pages 1718–1727, 2015.

[179] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. In *ICLR*, 2017.

[180] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pages 2234–2242, 2016.

[181] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[182] Mingqiang Zhu and Tony Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA CAM Report*, pages 08–34, 2008.

[183] Ernie Esser, Xiaoqun Zhang, and Tony Chan. A general framework for a class of first order primal-dual algorithms for tv minimization. *UCLA CAM Report*, pages 09–67, 2009.

[184] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[185] Tom Goldstein, Min Li, and Xiaoming Yuan. Adaptive primal-dual splitting methods for statistical learning and image processing. In *NIPS*, pages 2089–2097, 2015.

[186] Cong Dang and Guanghui Lan. Randomized first-order methods for saddle point optimization. *arXiv preprint arXiv:1409.8625*, 2014.

[187] Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *arXiv preprint arXiv:1507.02000*, 2015.

[188] Yuchen Zhang and Xiao Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, pages 353–361, 2015.

[189] Zhanxing Zhu and Amos J Storkey. Adaptive stochastic primal-dual coordinate descent for separable saddle point problems. In *ECML-PKDD*, pages 645–658, 2015.

[190] Zhanxing Zhu and Amos J Storkey. Stochastic parallel block coordinate descent for large-scale saddle point problems. In *AAAI*, 2016.

[191] Jialei Wang and Lin Xiao. Exploiting strong convexity from data with primal-dual first-order algorithms. *ICML*, 2017.

[192] Atsushi Shibagaki and Ichiro Takeuchi. Stochastic primal dual coordinate method with non-uniform sampling based on optimality violations. *arXiv preprint arXiv:1703.07056*, 2017.

[193] Yunmei Chen, Guanghui Lan, and Yuyuan Ouyang. Optimal primal-dual methods for a class of saddle point problems. *SIAM Journal on Optimization*, 24(4):1779–1814, 2014.

[194] Linbo Qiao, Tianyi Lin, Yu-Gang Jiang, Fan Yang, Wei Liu, and Xicheng Lu. On stochastic primal-dual hybrid gradient approach for compositely regularized minimization. In *ECAI*, 2016.

[195] Mengdi Wang and Yichen Chen. An online primal-dual method for discounted markov decision processes. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 4516–4521. IEEE, 2016.

[196] Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. *ICML*, 2017.

[197] Adams Wei Yu, Qihang Lin, and Tianbao Yang. Doubly stochastic primal-dual coordinate method for empirical risk minimization and bilinear saddle-point problem. *arXiv preprint arXiv:1508.03390*, 2015.

[198] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[199] Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *NIPS*, pages 1408–1416, 2016.

[200] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[201] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014.

[202] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. *ECCV*, pages 213–226, 2010.

[203] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. In *ICLR*, 2016.

[204] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[205] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[206] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[207] Jun Fu, Jing Liu, Yuhang Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *arXiv preprint arXiv:1708.04943*, 2017.

[208] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017.

[209] Golnaz Ghiasi and Charless C Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, pages 519–534. Springer, 2016.

[210] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[211] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representations*, 2015.

[212] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *European Conference on Computer Vision*, pages 402–418. Springer, 2016.

[213] Siddhartha Chandra, Nicolas Usunier, and Iasonas Kokkinos. Dense and low-rank gaussian crfs using deep embeddings. In *ICCV 2017-International Conference on Computer Vision*, 2017.

[214] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[215] Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[216] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[217] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[218] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[219] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[220] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2017.

[221] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. *arXiv preprint arXiv:1702.08502*, 2017.

[222] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[223] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[224] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

[225] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016.

[226] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 764–773, 2017.

[227] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection-snip. *arXiv preprint arXiv:1711.08189*, 2017.

[228] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[229] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[230] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[231] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[232] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[233] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[234] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[235] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[236] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[237] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 565–571. IEEE, 2016.

[238] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[239] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *Proceedings of the International Conference on 3D Vision*, 2017.

[240] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[241] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[242] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition*, volume 1, 2017.

[243] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[244] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[245] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[246] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[247] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[248] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[249] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[250] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[251] Ilya Loshchilov and Frank Hutter. Sgdr: stochastic gradient descent with restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[252] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016.

[253] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[254] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3193–3202, 2017.

[255] Haozhi Qi, Zheng Zhang, Bin Xiao, Han Hu, Bowen Cheng, Yichen Wei, and Jifeng Dai. Deformable convolutional networks – coco detection and segmentation challenge 2017 entry. *ICCV COCO Challenge Workshop*, 2017.

[256] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[257] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[258] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[259] Zhiyuan Shi, Yongxin Yang, Timothy M Hospedales, and Tao Xiang. Weakly supervised learning of objects, attributes and their associations. In *Computer Vision–ECCV 2014*, pages 472–487. Springer, 2014.

[260] Thomas Leung, Yang Song, and John Zhang. Handling label noise in video classification via multiple instance learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2056–2063. IEEE, 2011.

[261] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[262] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *Computer Vision–ECCV 2014*, pages 628–643. Springer, 2014.

[263] Piotr Bojanowski, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Finding actors and actions in movies. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2280–2287. IEEE, 2013.

[264] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the indian buffet process. In *Advances in neural information processing systems*, pages 475–482, 2005.

[265] Thomas L Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *The Journal of Machine Learning Research*, 12:1185–1224, 2011.

[266] Bahadir Ozdemir and Larry S Davis. A probabilistic framework for multimodal retrieval using integrative indian buffet process. In *Advances in Neural Information Processing Systems*, pages 2384–2392, 2014.

[267] Ilker Yildirim and Robert A Jacobs. A rational analysis of the acquisition of multisensory representations. *Cognitive science*, 36(2):305–332, 2012.

[268] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[269] Chenliang Xu, Shao-Hang Hsieh, Caiming Xiong, and Jason J Corso. Can humans fly? action understanding with multiple classes of actors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2264–2273, 2015.

[270] Cha Zhang, John C Platt, and Paul A Viola. Multiple instance boosting for object detection. In *Advances in neural information processing systems*, pages 1417–1424, 2005.

[271] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Proceedings of Advances in Neural Information Processing Systems*, pages 561–568, 2002.

[272] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):288–303, 2010.

[273] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[274] Timothee Cour, Chris Jordan, Eleni Miltsakaki, and Ben Taskar. Movie/script: Alignment and parsing of video and text transcription. In *Proceedings of 10th European Conference on Computer Vision, Marseille, France*, 2008.

[275] Piotr Bojanowski, Rémi Lagugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. Weakly-supervised alignment of video with text. In *ICCV*. IEEE, 2015.

[276] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3282–3289. IEEE, 2012.

[277] Hakan Bilen, Vinay P Namboodiri, and Luc J Van Gool. Object and action classification with latent window parameters. *International Journal of Computer Vision*, 106(3):237–251, 2014.

[278] Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. Book2movie: Aligning video scenes with book chapters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1827–1835, 2015.

[279] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27, 2015.

[280] Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. Linking people in videos with "their" names using coreference resolution. In *Computer Vision–ECCV 2014*, pages 95–110. Springer, 2014.

[281] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[282] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2048–2057, 2015.

[283] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482, 2015.

[284] Chen Sun, Chuang Gan, and Ram Nevatia. Automatic concept discovery from parallel text and visual corpora. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2596–2604, 2015.

[285] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[286] Anna Rohrbach, Marcus Rohrbach, and Bernt Schiele. The long-short story of movie description. In *Pattern Recognition*, pages 209–221. Springer, 2015.

[287] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *Multimedia, IEEE Transactions on*, 17(11):1875–1886, 2015.

[288] Finale Doshi, Kurt Miller, Jurgen V Gael, and Yee W Teh. Variational inference for the indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, pages 137–144, 2009.

[289] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[290] Arnold Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 42(4):278–280, 1988.

[291] Jun Zhu, Ning Chen, and Eric P Xing. Bayesian inference with posterior regularization and applications to infinite latent svms. *The Journal of Machine Learning Research*, 15(1):1799–1847, 2014.

[292] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010.

[293] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation and landmark estimation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[294] Mark Everingham, Josef Sivic, and Andrew Zisserman. Hello! my name is... buffy"–automatic naming of characters in tv video. In *BMVC*, volume 2, page 6, 2006.

[295] Omkar M Parkhi, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. A compact and discriminative face track descriptor. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1693–1700. IEEE, 2014.

[296] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. IEEE, 2013.

[297] Dan Oneata, Jérôme Revaud, Jakob Verbeek, and Cordelia Schmid. Spatio-temporal object detection proposals. In *Computer Vision–ECCV 2014*, pages 737–752. Springer, 2014.

[298] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[299] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[300] Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, and Peter Dayan. Comparison of maximum likelihood and gan-based training of real nvps. *arXiv preprint arXiv:1705.05263*, 2017.

[301] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Bridging implicit and prescribed learning in generative models. *arXiv preprint arXiv:1705.08868*, 2017.

[302] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[303] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2016.

[304] `http://yyy.rsmas.miami.edu/users/miskandarani/Courses/MSC321/lectfiniteDifference.pdf`.

[305] `http://www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf`.