Compressed Sensing: Novel Applications, Challenges, and Techniques

Submitted in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

by

Sabyasachi Ghosh
(Roll No. 174050001)

Supervisors:

Prof. Ajit Rajwade
Prof. Manoj Gopalkrishnan

Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
2023

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Sabyasachi Ghosh

Roll No. 174050001

Abstract

Compressed Sensing (CS) is a widely used technique for efficient signal acquisition, in which a very small number of (possibly noisy) linear measurements of an unknown signal vector are taken via multiplication with a designed 'sensing matrix' in an application-specific manner, and later recovered by exploiting the sparsity of the signal vector in some known orthonormal basis and some special properties of the sensing matrix which allow for such recovery. We study three new applications of CS, each of which poses a unique challenge in a different aspect of it, and propose novel techniques to solve them, advancing the field of CS. Each application involves a unique combination of realistic assumptions on the measurement noise model and the signal, and a unique set of algorithmic challenges.

We frame **Pooled RT-PCR Testing for COVID-19** – wherein RT-PCR (Reverse Transcription Polymerase Chain Reaction) tests for COVID-19 (COronaVIrus Disease of 2019) are saved in a low infection rate setting by performing them on a small number of pooled samples instead of individual samples and inferring the viral loads of the individual samples from the viral loads of the pools – as a CS problem. Our proposed Tapestry algorithm effectively combines traditional group testing (GT) algorithms with some conventional CS algorithms, in such a way that the combination exhibits superior performance to other standalone GT or standalone CS techniques. The combination exploits the inherent heteroscedasticity of the noise in RT-PCR measurements, in particular the fact that negative pooled tests are always noiseless in RT-PCR, unlike positive pooled tests. We demonstrate the superiority of our method over traditional GT and CS via in-silico experiments, validate it in wet lab experiments with oligomers, and prove theoretical guarantees for it.

For **Efficient Automated Image Moderation**, we bring CS methods to the domains of imbalanced binary classification and outlier detection for images. The first task is to use a

neural network to classify images as objectionable or not. We propose the quantitative matrix-pooled neural network, which takes a superposition of multiple images as input and *efficiently* outputs the counts of objectionable images in a small number of pools of these images specified by the rows of a binary matrix. From these (possibly noisy) counts, the classification of the images, as being objectionable or not, is inferred via CS decoding if only a small fraction of input images are objectionable. We empirically demonstrate that computation is saved relative to a network which processes the images separately, while maintaining sufficient accuracy. Our extension of this method to deep outlier detection infers the count of outlier images in a pool by comparing it with a pre-learned distribution of pool-level feature vectors extracted from our network, and is applied to the problem of moderation of off-topic images on topical forums.

Lastly, we propose the **Compressive Perturbed Graph Recovery** problem, in which the signal vector to be recovered from compressive measurements is sparse in the domain of the eigenvectors of the Laplacian matrix of an undirected, unweighted graph known only upto a few edge perturbations. This makes signal recovery challenging due to uncertainty in our knowledge of the orthonormal sparsifying basis. Our method solves this by performing joint signal and graph recovery, using cross-validation error on a held-out set of measurements to disambiguate between candidate graphs generated via a greedy edge selection strategy. We extend our method to solve the problem of recovery of images containing sharp edges (whose locations are not known a priori) from compressive measurements, generating candidate graphs in a structured manner from hypothesis linear image edges. We demonstrate the efficacy of our methods via extensive experiments, and prove theoretical guarantees for a brute-force version of our algorithms.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Compressed Sensing (also known as Compressive Sensing, shortened as CS) is a technique for measuring a signal using far fewer linear measurements of the signal than its number of dimensions, by exploiting its sparsity in some domain [1, 2, 3]. The signal, represented as a vector $\boldsymbol{x}^* \in \mathbb{R}^n$, is measured using a $m \times n$ sensing matrix $\boldsymbol{\Phi}$, with $m \ll n$, with the vector of measurements $\boldsymbol{y} \in \mathbb{R}^m$ given by

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + \boldsymbol{\eta} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}^* + \boldsymbol{\eta}, , \tag{1.1}$$

where $\boldsymbol{\eta} \in \mathbb{R}^m$ is a vector of measurement noise, $\boldsymbol{\Psi}$ is an orthonormal basis ($\boldsymbol{\Psi}^T = \boldsymbol{\Psi}^{-1}$), $\boldsymbol{\theta}^* = \boldsymbol{\Psi}^T\boldsymbol{x}^*$ is the representation of the signal in this basis, with $\|\boldsymbol{\theta}\|_0 = s \ll n$. The signal measurement in 1.1 is implemented in a domain-specific manner, and depending on the domain, is cheaper in terms of either the number hardware elements, the amount of computation, the amount of manual labour, the time of acquisition, or the monetary cost of acquisition, than measuring the full signal $\boldsymbol{x}^*$. The general problem of recovering a vector of n elements from $m < n$ linear measurements is under-determined. However, since $\boldsymbol{x}^*$ is known to be sparse in the orthonormal basis $\boldsymbol{\Psi}$, hence for appropriate measurement matrices $\boldsymbol{\Phi}$, the original signal

$\boldsymbol{x}^*$ may be estimated with good accuracy from $\boldsymbol{y}$ via an appropriate recovery algorithm. The intuition behind why this is possible is as follows: when there is no measurement noise (i.e. $\boldsymbol{\eta} = \mathbf{0}$), then $\boldsymbol{A\theta} = \boldsymbol{y} = \boldsymbol{A\theta}^*$ has a unique s-sparse solution for $\boldsymbol{\theta}$ if no $2s$-sparse vector (other than $\mathbf{0}$) lies in the nullspace of the matrix $\boldsymbol{A} = \boldsymbol{\Phi\Psi}$. For any other s-sparse solution $\tilde{\boldsymbol{\theta}}$, it must be that $\boldsymbol{A}\tilde{\boldsymbol{\theta}} = \boldsymbol{A\theta}^*$, or $\boldsymbol{A}(\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}^*) = 0$. Since $\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}^*$ is a $2s$-sparse vector, and the null-space of $\boldsymbol{A}$ does not contain any $2s$-sparse vectors other than 0, it must be that $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$.

A typical recovery problem P0 consists of optimizing the following cost function:

$$\min \|\boldsymbol{\theta}\|_0 \text{ s.t. } \|\boldsymbol{y} - \boldsymbol{\Phi\Psi\theta}\|_2 \leq \varepsilon, \tag{1.2}$$

where ε is an upper bound (possibly a high probability upper bound) on $\|\boldsymbol{\eta}\|_2$, and $\|\boldsymbol{\theta}\|_0$ is the number of non-zero elements in $\boldsymbol{\theta}$. In the absence of noise, a unique and exact solution to this problem is possible with as few as $2s$ measurements in $\boldsymbol{y}$ if $\boldsymbol{\theta}$ has s non-zero elements [4]. Unfortunately, this optimization problem P0 is NP-Hard and the algorithm requires brute-force subset enumeration. Some examples of practical recovery algorithms are Basis Pursuit Denoising (BPDN) [4], LASSO (Least Absolute Shrinkage and Selection Operator) [5, 6], Sparse Bayesian Learning (SBL) [7, 8], Orthogonal Matching Pursuit (OMP) [9], etc. These typically act as approximation algorithms for the NP-Hard P0 problem, and also require $O(s \log \frac{n}{s})$ measurements for successful recovery. A BPDN estimate is given by a convex relaxation of P0, which gives the following quadratically constrained ℓ_1-norm minimization problem:

$$\hat{\boldsymbol{\theta}}_{\text{bpdn}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_1, \quad \text{s.t. } \|\boldsymbol{y} - \boldsymbol{\Phi\Psi\theta}\|_2 \leq \varepsilon. \tag{1.3}$$

A LASSO estimate is given by a penalized form of the above problem:

$$\hat{\boldsymbol{\theta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{\Phi\Psi\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1, \tag{1.4}$$

where λ is an appropriately chosen parameter.

Various classes of random matrices exist such that if $\boldsymbol{\Phi}$ is drawn from such a class then for any orthonormal $\boldsymbol{\Psi}$, $\boldsymbol{A}$ satisfies a condition such as the Restricted Isometry Property [4], Restricted Eigenvalue Condition [6], Robust Nullspace Property [10], etc, with high probability.

These conditions are sufficient for recovery of $\boldsymbol{\theta}^*$ from noisy measurements as in Eqn. 1.1 via BPDN, LASSO, or other recovery algorithms, with appropriate bounds on the error of the recovered vector. Typically, for random matrices only $m = O(s \log \frac{n}{s})$ measurements are needed for $\boldsymbol{A}$ to satisfy such a condition. Some examples of these classes of matrices include matrices whose entries are independent and identically distributed (i.i.d.) random variables from any sub-Gaussian probability distribution with mean 0 [11], or a Bernoulli with some probability p of being 1 and $1 - p$ of being 0 [12]. In case $\boldsymbol{\Psi}$ is the identity matrix, making $\boldsymbol{A} = \boldsymbol{\Phi}$, explicit deterministic constructions of matrices satisfying some of these conditions also exist – for example matrices constructed from Steiner Systems [13] have the RIP in ℓ_1-norm [14]. In some cases, such as if $\boldsymbol{\Psi}$ is a (discrete) Fourier basis matrix, $\boldsymbol{\Phi}$ may even be chosen to a subset of the identity matrix [15]. Due to the simplicity in the scheme of measurement which arises naturally in many problems (a matrix-vector product), the flexibility in the design of sensing matrices, and the inherent sparsity of the signal being measured in some basis in these problems, the technique of Compressed Sensing finds applicability in a number of domains, such as image or video acquisition (computational photography), Magnetic Resonance Imaging (MRI), radar, signal processing, error correcting codes, etc [15, Chapter 1].

In this thesis, we employ Compressed Sensing to solve problems in three new domains – COVID-19 Testing, image moderation, and graph signal processing, and present the following novel applications:

1. Pooled RT-PCR Testing for COVID-19 Detection (Chapter 2) [14, 16],
2. Efficient Automated Image Moderation (Chapter 3) [17], and
3. Compressive perturbed graph recovery (Chapter 4) [18].

Each of these applications presents a unique challenge in a different aspect of compressed sensing, and new techniques needed to be developed to address these challenges. Fig. 1.1 summarizes the different aspects of compressed sensing worked on in this thesis for each application.

First, in Chapter 2 we consider a case where some additional information about the noise vector $\boldsymbol{\eta}$ is available. In particular, we consider the case when the vector $\boldsymbol{x}^*$ is sparse in the canonical basis and $\boldsymbol{x}^*$ as well as the measurements $\boldsymbol{y}$ have only non-negative entries, with the sensing matrix $\boldsymbol{\Phi}$ having binary entries (0 or 1). While there may be noise in the non-zero

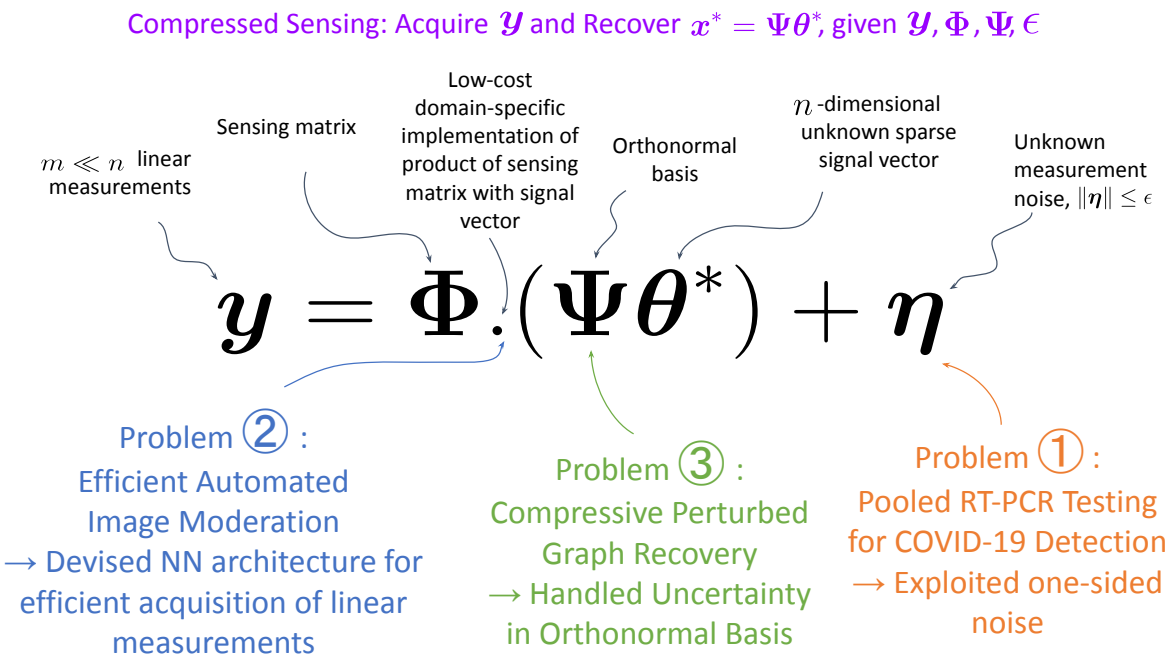


Figure 1.1: Compressed Sensing and the different aspects of it tackled in this thesis are illustrated. Compressed Sensing is the problem of efficiently acquiring $m \ll n$ linear measurements $\boldsymbol{y}$ of an n-dimensional unknown signal vector $\boldsymbol{x}^* = \boldsymbol{\Psi\theta}^*$ using a sensing matrix $\boldsymbol{\Phi}$ in a domain-specific manner, and later recovering the signal $\boldsymbol{x}^*$ from $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$, and an upper bound on the magnitude of measurement noise, $\boldsymbol{\eta}$. Each problem solved in this thesis advanced a different aspect of Compressed Sensing: (1) For Pooled RT-PCR testing for COVID-19 Detection, additional information of the noise being one-sided was exploited by a new algorithm. (2) For Efficient Automated Image Moderation, a new Neural Network architecture was developed for efficiently acquiring measurements of the signal vector. (3) For Compressive Perturbed Graph Recovery, a new algorithm was developed for handling uncertainty in the orthonormal basis.

entries of $\boldsymbol{y}$, an entry of $\boldsymbol{y}$ is equal to 0 if and only if the corresponding entry of $\boldsymbol{\Phi x}^*$ is also 0. We exploit this kind of one-sided noise by combining algorithms from the field of group testing [19] with CS decoding algorithms to create a new kind of algorithm. This is applied to the problem of pooled RT-PCR Testing for COVID-19 detection (see Sec. 1.1).

Next, in Chapter 3 we devise a new signal acquisition method for acquiring signals efficiently via deep neural networks, opening a new application area for compressed sensing. The entries of the signal vector $\boldsymbol{x}^*$ represent the classification of n images into two classes in an class-imbalanced classification setting, with a 1 indicating the rare class, and a 0 indicating the background class. Similarly, for an outlier detection setting, a 1 represents an outlier image, and

a 0 represents an in-distribution image. Without this method, the entries of the signal vector $\boldsymbol{x}^*$ are acquired using a deep neural network performing binary classification or outlier detection on n images separately. However, most images are known to belong to one particular class, or are not outliers, making $\boldsymbol{x}^*$ sparse in the canonical domain. We devise a new neural network architecture which can be trained to output the product of a binary sensing matrix and the signal vector $\boldsymbol{x}^*$. This is more efficient than acquiring each entry of $\boldsymbol{x}^*$ separately. CS decoding algorithms are then used to retrieve $\boldsymbol{x}^*$. We apply this to the problem of Automated Image Moderation, detailed in Sec. 1.2

Finally, in Chapter 4, we handle the case where there is uncertainty in the knowledge of the orthonormal basis $\boldsymbol{\Psi}$. The signal $\boldsymbol{x}^*$ is assumed to be sparse in the basis formed by the eigenvectors of the Laplacian matrix of an undirected, unweighted graph which is known upto a few edge perturbations (i.e. a few edges of which have been added or dropped). As the Laplacian matrix is symmetric, its eigenvector matrix $\boldsymbol{\Psi}$ is orthonormal. Furthermore, uncertainty in the Laplacian matrix induces uncertainty in $\boldsymbol{\Psi}$. Given compressive measurements of signals defined on the true graph, we develop a cross-validation based algorithm to disambiguate between the possible graphs from which these measurements could have originated, jointly recovering the graph as well as the the signal. We apply this technique to graph signals defined on several well known graphs, as also the problem of recovering compressively acquired images with sharp edges in them.

In the following sections, we give an overview of each of the problems considered in our work, our approaches to solve them, and some glimpses of experimental and theoretical results.

1.1 Pooled RT-PCR testing for COVID-19 Detection

The COVID-19 pandemic severely affected almost all inhabitants of the globe, either directly or via economic disruption due to lockdowns and social distancing [20]. Timely testing of symptomatic people was critical to quick diagnosis and making medical interventions at the appropriate time. However, during the peak of the pandemic, the testing infrastructure was strained, with lab technicians working round the clock, potential shortages of reagents, and people waiting many days to get test results or even being denied testing [21, 22]. Hence rapid

expansion of COVID-19 testing capacity was needed at the time. Furthermore, widespread screening of people for COVID-19 could help slow down its spread and prevent or reduce lockdowns, and help quicker re-opening of offices, campuses and public spaces. Hence to prevent the spread of the disease, rapid scaling of testing infrastructure was needed.

The gold standard test for COVID-19 is the RT-PCR (Reverse Transcription Polymerase Chain Reaction) test [23, 24], which is used to test nasal swab sample collect from a person for the presence of the SARS-nCOV-2 virus. The test proceeds with extraction of viral RNA from the sample, conversion of the RNA to complementary DNA (cDNA), and amplification of the cDNA in a sample over many heating and cooling cycles, which is detected by fluorescent markers when its amount reaches a threshold value. The cycle time at this threshold value (threshold cycle or C_t) is reported. Since the cDNA roughly doubles in each cycle, the amount of virus present originally may be determined from the C_t value. If the fluorescence does not reach the threshold value even after some maximum number of cycles, then it is inferred to not have the virus. RT-PCR is known to detect even very small amounts of virus molecules in a sample – even a single molecule may be detected, due to repeated amplification [25, 26]. However, the test usually requires 3-4 hours, is labour-intensive, and requires expensive chemical reagents. Moreover, since only a small fraction of the population had COVID-19 even at the peak of the pandemic, most of the tests are "wasted" on people who did not have the disease.

Pooled testing (or Group Testing) [19, 27] is an effective way of reducing the number of tests needed for testing a disease and increasing the testing capacity when the fraction of people having the disease is small. In the earliest form of pooled testing, called Dorfman Testing [28], the samples to be tested from n distinct individuals are divided into groups of size r each and combined to form $\frac{n}{r}$ pooled samples, which are tested for the presence of the disease. The key idea is that if a pooled sample tests negative, then all the r samples constituting it must be negative (assuming accurate test results). If a pooled sample tests positive, then at least one of the r samples constituting it must be positive – hence these samples are re-tested in a second round of testing. Dorfman testing is an adaptive group testing method since the samples to be tested in the second round depend on the results of the tests in the first round. Non-adaptive group testing methods also exist (such as Combinatorial Orthogonal Matching Pursuit (COMP) [29]), in which only a single round of testing needs to be performed. In COMP, $m < n$ pooled samples are tested, with each sample being part one or more pooled samples as given

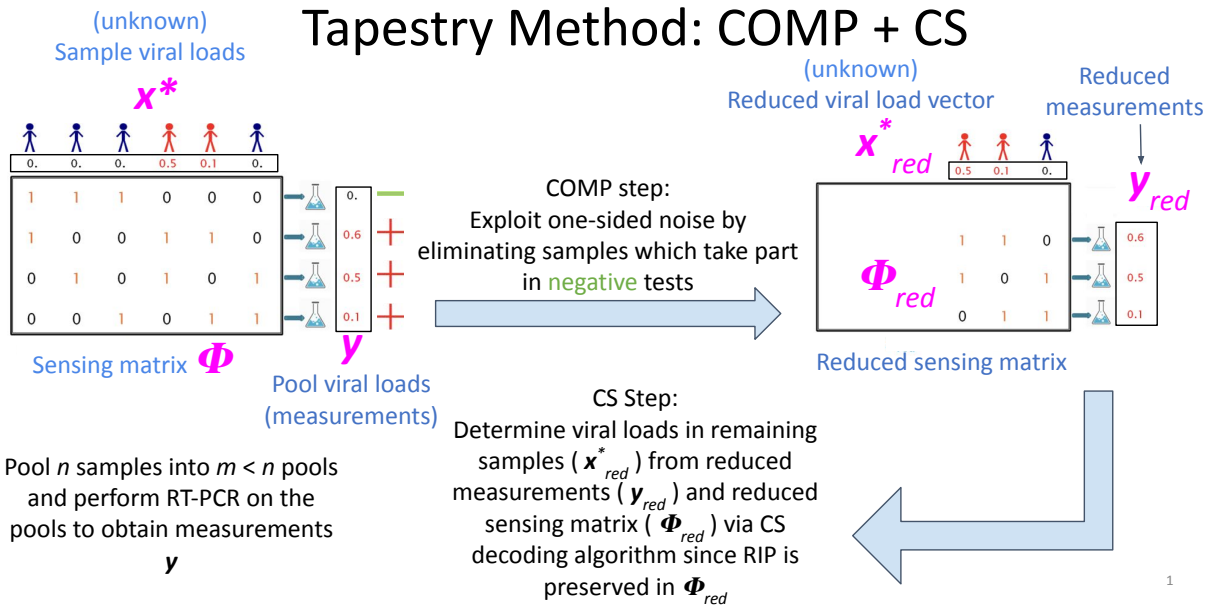


Figure 1.2: Illustration of the Tapestry method for pooled RT-PCR Testing of COVID-19

by a pooling scheme, and the samples which take part in negative tests are determined to be negative, and remaining samples are declared to be positive. If the number of positive samples s is sufficiently small and for pooling schemes with certain properties in case of COMP, Dorfman Testing and COMP are able to significantly reduce the number of tests needed for testing the n samples.

However, both of these methods have some drawbacks in the context of RT-PCR testing for COVID-19. In Dorfman Testing, one must wait for a second round of testing to determine the positivity of all samples. Since each round of RT-PCR tests takes 3-4 hours and is labour-intensive, this is undesirable. In COMP, only the positivity of the samples may be determined – not the amount of virus in each sample. However, RT-PCR tests done on individual samples return a measure of the viral load in each sample. This information is crucial to determining the severity of the disease or whether a person might be infectious [30, 31]. One way of determining the viral loads from fewer than n pooled tests might be Compressed Sensing. The amount of viral load in a pool sample is equal to the sum of viral loads in the constituent samples. If equal volume of liquid from the r samples is mixed to create a pooled sample, then that pool may be represented by a binary vector of length n, with the j^{th} entry being equal to 1 indicating that a unit volume of the j^{th} sample has been added to that pool, and 0 indicating that the pool does not contain the j^{th} sample. In the same vein, m pools may be represented by the

Matrix: 93x961 Kirkman							
	COMP	NNLASSO			COMP-NNLASSO		
s	#FP	#FN	#FP	RMSE	#FN	#FP	RMSE
5	1.6	0.1	78.2	0.254	0.0	0.8	0.047
8	7.9	0.2	82.2	0.326	0.1	4.0	0.069
10	15.3	0.5	85.4	0.418	0.2	7.8	0.100
12	25.3	0.9	83.5	0.558	0.6	13	0.149
15	46.1	1.9	95.1	0.656	1.0	28.7	0.295
17	62.3	2.3	113.2	0.723	1.0	46.5	0.404
20	91.5	2.8	141.4	0.787	1.1	77.6	0.563

Table 1.1: Performance of COMP, NNLASSO, and COMP-NNLASSO on synthetic data for 93×961 Kirkman triple matrix. For each criterion and each s mean value across 1000 signals is reported. See Chapter 2 for more details.

rows of an $m \times n$ binary 'pooling matrix' $\mathbf{\Phi}$. The viral loads in each sample may be represented by an unknown sparse real vector $\boldsymbol{x}$ of size n. The viral loads in the pooled samples may be represented by a vector $\boldsymbol{y}$ of size m, with the entries of $\boldsymbol{y}$ measured via RT-PCR of the pooled samples. The relationship $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}$ holds, with $m < n$. Hence a CS decoding algorithm may be used to recover the sparse vector $\boldsymbol{x}$ from $\boldsymbol{y}$, provided $\mathbf{\Phi}$ obeys conditions such as RIP which are sufficient for CS recovery. However, CS decoding algorithms assume noise in all entries of $\boldsymbol{y}$ and ignore the following property of RT-PCR tests – RT-PCR will returns 0 viral amount for a negative pool, and a non-zero viral amount for a positive pool. This additional information can be used to enhance the accuracy of CS decoding algorithms.

In our work, we propose Tapestry, a new kind of algorithm which combines non-adaptive group testing and compressed sensing methods, and has advantages of both the methods. The overall Tapestry method is sketched in Fig. 1.2. Tapestry decoding has two stages – in the first stage, COMP decoding is used, i.e. all samples which took part in at least one negative test are declared as negative i.e. having viral load equal to 0. A reduced matrix $\bar{\mathbf{\Phi}}$ is obtained from $\mathbf{\Phi}$ by eliminating the columns corresponding to such samples, as well as rows corresponding to negative tests. Entries of $\boldsymbol{y}$ which are 0 are removed to obtain a reduced measurement vector $\bar{\boldsymbol{y}}$. CS decoding is performed using $\bar{\boldsymbol{y}}$ and $\bar{\mathbf{\Phi}}$ to recover the viral loads in the remaining samples.

This CS step after the COMP step ensures that we have fewer false positives than COMP (at the cost of a small number of false negatives). The COMP step before performing the CS step ensures that the additional information about pool noise is utilized to reduce the problem size for CS, which greatly improves its accuracy. This is readily observed in an empirical result – Table 1.1 shows the performance of COMP, NNLASSO (a CS decoding method), and COMP-NNLASSO (i.e. the Tapestry method with COMP followed by NNLASSO) for various metrics on some synthetically generated data. We see that COMP-NNLASSO has fewer false positives than COMP, fewer false positives and false negatives than COMP-NNLASSO, and much smaller RRMSE than NNLASSO.

We summarize the main contributions made in this part of the thesis below:

1. We present Tapestry, a novel algorithm which is a combination of non-adaptive group testing and compressed sensing methods, designed to take advantage of the one-sided noise characteristic to pooled RT-PCR testing of COVID-19, which is not taken into account by standard CS recovery algorithms. Tapestry reduces the number of tests, gives results in a single round, and gives a quantitative readout of the viral loads in each sample.

2. We develop a novel noise and synthetic data generation model for readouts obtained from RT-PCR.

3. We perform extensive empirical evaluation of the Tapestry method using synthetic data generated via the above model, for a range of pooling matrix sizes and rate of infection. Tapestry can reduce the number of tests required by 2.5 to 10 times depending on the fraction of infected people, while achieving clinically acceptable sensitivity and specificity.

4. We validate the Tapestry method in wet labs using synthetic oligomers.

5. We use ultra-sparse binary pooling matrices derived from Kirkman Triple Systems [32], which require a low degree of effort for pooling, conserves samples, and allow for flexible matrix sizes while keeping the number of samples in each pool equal [33]. We prove that such matrices have the disjunctness property which is useful for group testing, are adjacency matrices of expander graphs, and consequently have the RIP in ℓ_1 norm, which is a sufficient condition for CS recovery.

6. Detection of false positive or false negative tests (e.g. due to cross-contamination of samples or pooling errors) is possible.

7. We prove that conditions sufficient for recovery via CS such as the RIP are preserved for the matrix obtained via reduction after COMP. Using this result and drawing on the literature for CS and group testing, we prove that for guaranteed recovery of s-sparse vectors, the Tapestry method requires only $m = \mathcal{O}(s \log n)$ pooled tests using a particular class of random binary matrices.

8. We also prove that RIP and other properties sufficient for CS recovery are preserved by a wider class of reductions of a matrix, which includes reductions via noisy group testing algorithms such as NCOMP [29]. Thus the Tapestry method may be enhanced to recover from false negative pooled tests by using NCOMP in the first stage instead of COMP.

9. We developed the Byom app [34] for easy manual pooling of samples, decoding test results, and keeping track of tests. This was important because many labs lacked robotic liquid handlers to perform the pooling.

Lastly, the Tapestry method is not limited to COVID-19 testing – it may be used for pooled RT-PCR testing of any kind, or in any problem where both group testing and compressed sensing might be applicable. A preprint [16] explaining usage of the Tapestry method for COVID-19 RT-PCR testing was shared on medRxiv. Technical details of the Tapestry method were published in [14]. The research in this work has been well-received by the community and has received over a hundred citations. Furthermore, [14] was one of the top 25 most downloaded papers of the IEEE Open Journal of Signal Processing in 2021. A patent for this method has been filed [35]. The work in this part of the thesis led to the founding of a company called Algorithmic Biologics [36] by Prof. Manoj Gopalkrishnan, which provides wet labs with computational and algorithmic services for pooled RT-PCR testing for COVID-19 and other kinds of RT-PCR based molecular testing.

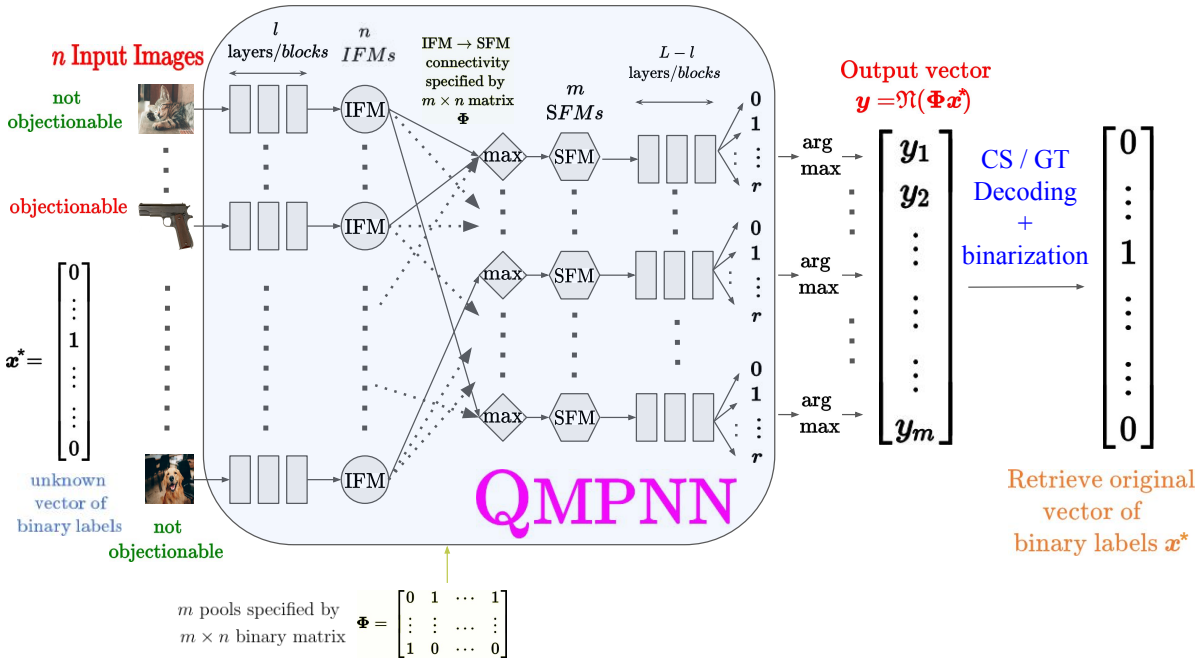


Figure 1.3: Overview of our Compressed-Sensing based Image Moderation Engine

1.2 Efficient Automated Image Moderation

Millions of images get uploaded daily on social media platforms such as Facebook, Instagram, Twitter, Imgur, etc. Care must be taken to not unwittingly expose the users of such websites to images containing objectionable content – such as images depicting weapons, illegal drugs, pornography, etc. Image Moderation is the process of determining whether an image has objectionable content. This may be performed by human moderators by inspection of each image. Automated moderation engines, which typically employ neural networks to examine the content of an image to determine whether it is objectionable (e.g. detection of firearms/guns [37], knives [38, 39] or violent scenes [40]), can reduce the labour cost and manual work involved in image moderation. However, since large neural networks require several giga floating point operations for every forward pass [41] and consume considerable amounts of power [42], such automated moderation incurs huge costs in terms of amount of computation and energy.

While the number of images to be moderated is large, the actual fraction of images which may be objectionable is small (e.g. 0.1% for Facebook [43] and upto 6% for Reddit [44]). Such inherent sparsity of objectionable images indicates that this problem may be amenable to Compressed Sensing, which can make the detection of objectionable images more efficient.

Consider n images which need to be processed and determined to be objectionable or not. An (unknown) n-dimensional binary vector $\boldsymbol{x}^*$ may be used to indicate whether each of the n images is objectionable or not, with a 1 indicating that the image is objectionable, and 0 indicating that it is not. Since the fraction of objectionable images is small, only some $s \ll n$ entries of $\boldsymbol{x}^*$ will be non-zero. Typically, classification of these n images as being objectionable or not will be performed individually for each image by a neural network – we refer to this as an 'individual neural network' or INN. Prior work in [45] uses a 'pooled neural network' to perform classification of a 'pool' of some r images as having at least one objectionable image or not. If the pooled network determines that the pool contains at least one objectionable image, their algorithm runs the INN on each of the r images to classify them as objectionable or non-objectionable; otherwise all the r images are declared to be not objectionable. This is a two-stage Dorfman Testing method, which makes detection of objectionable images more efficient if $s \ll n$. In a pooled neural network first l layers of the INN are run for all r images to obtain r Intermediate Feature-Maps (IFMs). An entry-wise max is performed over these r IFMs to obtain a single Superposed Feature-Map (SFM), which has the same dimensions as each of the IFMs. The remaining layers of the INN are run only on this SFM. Thus, running a pooled neural network on r images is more efficient than running r INNs separately on the images, since processing of $r-1$ IFMs is not performed. Note that the number of parameters of the pooled neural network is same as the INN, but the pooled neural network needs to be separately trained to output the presence or absence of an objectionable image in a pool.

In our work, we propose the 'quantitative matrix-pooled neural network' (QMPNN), which takes as input n images and a specification of m pools of images given by a binary $m \times n$ matrix $\boldsymbol{\Phi}$, computes IFMs for each of the n images, computes an SFM for each of the m pools from these IFMs, and is trained to output the *count* of objectionable images in each of the m pools in a single forward pass. The pooling scheme is specified as follows: if the j^{th} image is present in the i^{th} pool, Φ_{ij} is 1, otherwise it is 0. Let $\boldsymbol{y}$ be the vector of outputs from the QMPNN, with each entry of $\boldsymbol{y}$ indicating the count of objectionable images in the corresponding pool as predicted by the QMPNN. Hence the relationship $\boldsymbol{y} = \boldsymbol{\Phi x}^*$ holds if the prediction is perfect. In practice, since the QMPNN may not make perfect predictions, the relationship may be represented as:

$$\boldsymbol{y} = \mathfrak{N}(\boldsymbol{\Phi x}^*), \tag{1.5}$$

where the operator $\mathfrak{N}(.)$ represents noise or errors in the QMPNN predictions. The vector $\boldsymbol{x}^*$ may be recovered by a CS decoding algorithm and binarized to have only 0 or 1 entries. We use the following two variants of LASSO:

1. Constrained LASSO or CLASSO:

$$\bar{\boldsymbol{x}} \triangleq \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_1, \text{ s.t. } \forall i \in [n], x_i \in [0,1]$$
$$\hat{\boldsymbol{x}} = [\hat{x}_1 \dots \hat{x}_n]^T \text{ where } \forall i \in [n],\ \hat{x}_i = \begin{cases} 1 & \text{if } \bar{x}_i \geq \tau \\ 0 & \text{otherwise} \end{cases}, \tag{1.6}$$

   in which the LASSO objective is minimized with constraints on the entries of vector $\boldsymbol{x}$ to lie between 0 and 1. The final estimate $\hat{\boldsymbol{x}}$ is obtained via thresholding the entries of the intermediate estimate $\bar{\boldsymbol{x}}$ with the parameter τ.

2. Mixed Integer Programming (MIP):

$$\hat{\boldsymbol{x}} \triangleq \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \text{ s.t. } \forall i \in [n], x_i \in \{0,1\}, \tag{1.7}$$

   in which the LASSO objective is minimized with the entries of the vector $\boldsymbol{x}$ constrained to be 0 or 1, using a branch and bound method.

The complete architecture of our system is presented in Fig. 1.3. A sample numerical result on important measures such as sensitivity and specificity [46] of detection of objectionable content, as well as amount of computation used by various image moderation methods is presented in Fig. 1.4 – see Chapter 4 for precise definitions.

Finally, we extend our method to outlier detection, creating a novel *pooled deep outlier detection* method, which we employ to the problem of efficient automated moderation of off-topic images on topical forums on the internet such as Reddit. For example, on a forum meant for sharing images of cars, an image of an aeroplane may be off-topic and needs to be removed from the forum. Currently such moderation is done either manually or via text-based automated moderators [44, 47]. In general, the off-topic images may be from classes not known a-priori at the time of training. In our setting, training data is available for on-topic class and from some

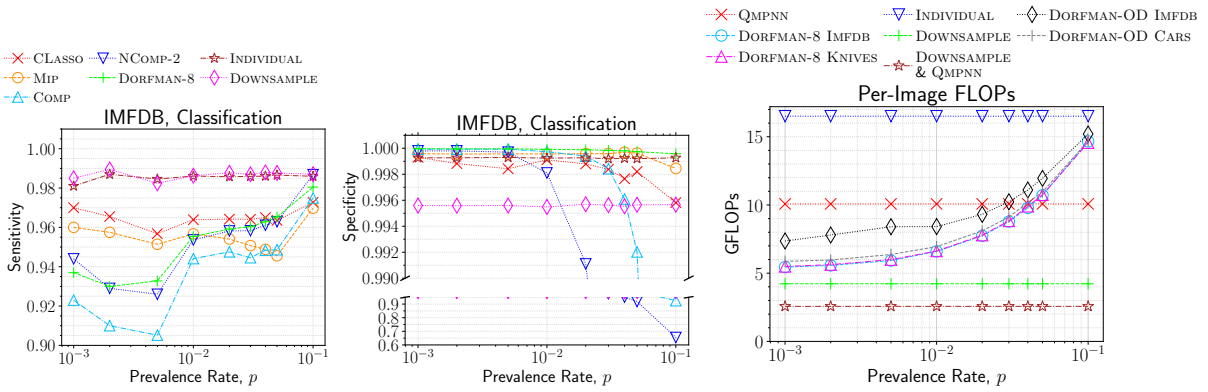


Figure 1.4: Left, Center: Sensitivity and Specificity of our methods (CLASSO and MIP) compared with various baselines on a firearms classification task. Our compressed sensing based methods exhibit significantly higher sensitivity than binary group testing methods Right: Image Moderation cost in giga floating point operations (GFLOPs) for various methods. Our method (QMPNN) has uses significantly lower amount of computation than Individual testing of each image (INDIVIDUAL), and lower than the Dorfman method used in [45] for higher prevalence rate fo objectionable images. See Chapter 3 for a detailed discussion of this and other results.

known off-topic classes, whereas test data may have off-topic images from unknown classes. In this method, the output of the last-but-one layers in a QMPNN are treated as feature vectors for the m pools of images. A distribution of the feature vectors of pools containing on-topic images is learned on the training data, modelled using a Gaussian Mixture Model (GMM). At test time, the negative log likelihood given by the GMM is used as an anomaly score for a pool, which is then used to predict the number of outlier images in the pool. This is then decoded by the CS algorithms described above to predict which images were outliers. We also use the same method with an individual neural network for outlier detection on individual images. We also propose using Dorfman Testing for outlier detection, using the outlier detection method for a single pool of images to predict if a pool contained outliers, and if so, using the outlier detection method for individual images on each image of the pool to predict which one of them were outliers.

To summarize, we make the following contributions in this part of the thesis:

1. We frame the problem of detection of objectionable images via a neural network as a compressed sensing problem. We present QMPNN, which efficiently recovers linear measurements of the vector $\boldsymbol{x}^*$ of objectionable status of each image with a binary sensing

matrix $\mathbf{\Phi}$.

2. Our innovations in the QMPNN open a new application area of compressed sensing and non-adaptive group testing. Crucially, in the QMPNN, the IFMs for each image are computed only once per image and all of the computation for the m pools is done in a single forward pass on the GPU. Without these innovations, the QMPNN would be too inefficient and may even require more time and/or computation than the INN for pooling schemes in which one image takes part in multiple pools, and compressed sensing would not have been applicable.

3. We evaluate our methods over a wide range of prevalence rates of objectionable images. Our method is much more efficient than individual testing of images for being objectionable, while maintaining reasonable accuracy. Our method is also more efficient than the Dorfman Testing method used in [45] for higher prevalence rates.

4. The CS decoding algorithms employed are noise-tolerant, as opposed to the group testing (GT) methods employed in [45], which do not handle the case of a pool falsely testing negative. As a result, our methods have higher sensitivity than those in [45] and other baseline binary GT methods.

5. To the best of our knowledge, our pooled deep outlier detection method is the first method to employ CS or GT methods to the problem of efficient deep outlier detection.

A paper based on the work done in this part of the thesis has been submitted to a reputed peer-reviewed journal. A preprint version is available [17].

1.3 Compressive Perturbed Graph Recovery

Graphs are ubiquitous data structures, and are naturally found in many application domains, such as transportation networks, biological networks, epidemiology, social networks, etc. In some other applications domains, such as image processing, computer graphics, statistics, etc, a latent graph may be present, such as a 2D lattice, a mesh representing a 3D model, a probabilistic graphical model. Typically, data is arranged on the nodes of such graphs, such as pixel

values in an image, or positivity for a virus on a contact trace graph. Such data are known as graph signals – a vector of values representing a mapping from the nodes of a graph to a real number. Typically, the values in such graph signals are intimately tied to the structure of the graph. For example, there is correlation in pixel values of the neighbouring pixels of an image. Similarly, if two people are frequently in contact with each other, they are more likely to both be positive or negative for a virus. Often these graph signals have a sparse or compressible representation in the basis formed by the eigenvectors of the Laplacian matrix of the graph, which is known as the Graph Fourier Transform (GFT) basis in the field of Graph Signal Processing (GSP) [48, 49, 50]. The Laplacian matrix of an undirected, unweighted graph is $L = D - W$, where W is the adjacency matrix, and D is the degree matrix (a diagonal matrix with the entries containing the degrees of the nodes of the graph). For example, piece-wise smooth images and natural images have sparse or compressible representation in the 2D Discrete Cosine Transform (DCT) basis – due to which they are used in image compression [51] – and the 2D DCT basis vectors are a set of eigenvectors of the 2D lattice graph of the same size [52]. The GFT may be seen as a generalization of the Discrete Fourier Transform to graphs [48]. GFT basis vectors are themselves graph signals. Similar to the Fourier Transform, GFT basis vectors have a notion of frequency, with low-frequency eigenvectors having values which vary smoothly along the edges of the associated graph, and high-frequency eigenvectors having values which vary more rapidly along the edges.

Graph signals may be compressively acquired as in Eqn. 1.1 – for example, a single-pixel camera [53] can directly acquire linear measurements of a raster image corresponding to a scene such that the number of measurements is much fewer than the number of pixels in the image; similarly, viral loads of a virus in samples collected from some people may be acquired in a pooled form, using much fewer pools than the number of samples. Recovery of the graph signal from compressive measurements may be performed by exploiting the sparsity of its GFT, for example, by using the GFT matrix as the orthonormal basis $\mathbf{\Psi}$ in the LASSO (Eqn. 1.4). In our work, we consider the case when the underlying graph – termed as the *actual* graph – is not fully known. Instead, version of it with a few *edge perturbations* is known, which we term as the *nominal* graph. That is, the nodes of the nominal graph are the same as that in the actual graph, but a few edges present in the actual graph are not present in the nominal graph, and a few edges not present in the actual graph are present in the nominal graph. For example, in a pooled testing with contact tracing application such as [54], some edges may not show

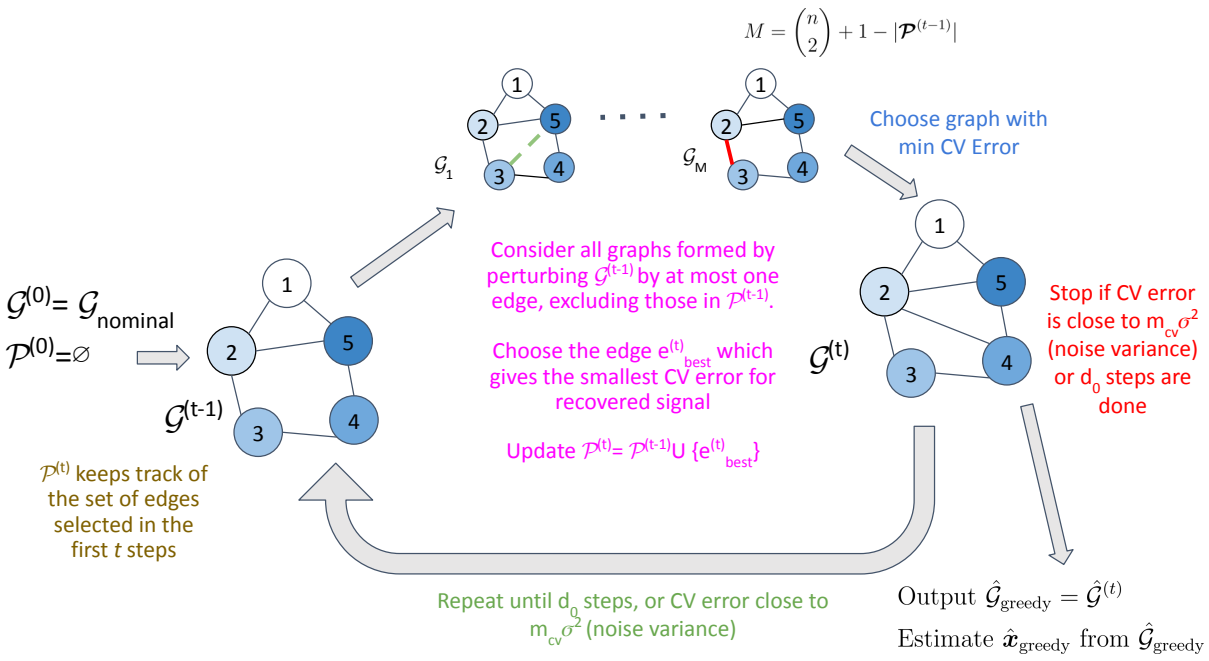


Figure 1.5: Main steps of the Greedy Edge Selection Algorithm

up due to bluetooth connectivity issues; similarly some spurious edges may show up if two people are in adjoining rooms separated by a wall. The eigenvectors of the Laplacian matrix get perturbed upon perturbation of a graph – hence if signal recovery is performed using the GFT basis of the nominal graph, it will have a larger error than if the GFT basis of the actual graph were to be used. Hence it is desirable to recover the actual graph as well. We term the problem of the recovery of the actual graph and the graph signal from the nominal graph, compressive measurements of the graph signal, and an upper bound d_0 on the number of edge perturbations between the actual and the nominal graph as the *compressive perturbed graph recovery* problem. In this work, we consider undirected and unweighted graphs only. From the perspective of compressed sensing, in this problem there is uncertainty in the basis matrix $\mathbf{\Psi}$. There are some works in the CS literature which handle uncertainty in the basis matrix, such as perturbation of the sinusoid frequencies of a Fourier matrix [55, 56, 57], or direct additive perturbation, such as in [58]. However, in our problem, the uncertainty comes in an indirect manner, from the perturbation of the eigenvectors of the Laplacian matrix of a graph, hence these related methods are not directly applicable. To the best of our knowledge, this is the first work to consider this problem.

We solve this problem by iteratively refining the nominal graph, finding a better approximation of the actual graph in each iteration, using cross-validation, which is a popular technique

typically employed in compressed sensing for the selection of the hyperparameter(s) of a CS recovery algorithm (e.g. [59, 60, 61]). The main idea is that for any two graphs, the graph which has fewer edge perturbations relative to the actual graph will typically have GFT basis vectors which are closer to the actual graph; hence the signal recovered using the GFT basis of this graph will have a smaller error, and correspondingly the cross-validation (CV) error for this signal will also be lower. First, the signal measurements $\boldsymbol{y}$ and the sensing matrix $\boldsymbol{\Phi}$ are divided into two sets – $\boldsymbol{y}_{\mathrm{r}}$ and $\boldsymbol{\Phi}_{\mathrm{r}}$ to be used for signal recovery, and $\boldsymbol{y}_{\mathrm{cv}}$ and $\boldsymbol{\Phi}_{\mathrm{cv}}$ to be used for computing cross-validation error. At each iteration, all graphs which are at most a single edge perturbation away from the current graph are considered. The GFT basis for each of these graphs is computed by performing the eigendecomposition of the corresponding Laplacian matrices. A signal is recovered for each such graph via a CS recovery algorithm such the LASSO (Eqn. 1.4), using $\boldsymbol{y}_{\mathrm{r}}$ and $\boldsymbol{\Phi}_{\mathrm{r}}$, and the corresponding GFT basis as $\boldsymbol{\Psi}$. Cross-validation errors for each of the recovered signals are computed, using the formula $\epsilon_{\mathrm{cv}}(\boldsymbol{x}) = \|\boldsymbol{y}_{\mathrm{cv}} - \boldsymbol{\Phi}_{\mathrm{cv}}\boldsymbol{x}\|_2^2$. The edge perturbation for which the recovered signal had the lowest CV error is selected. If the CV error does not decrease in an iteration, or if its value is within some factor of the variance of the noise, the algorithm stops and outputs the currently selected graph as the actual graph and the signal recovered using it as the graph signal. We call this the Greedy Edge Selection algorithm, illustrated in Fig. 1.5.

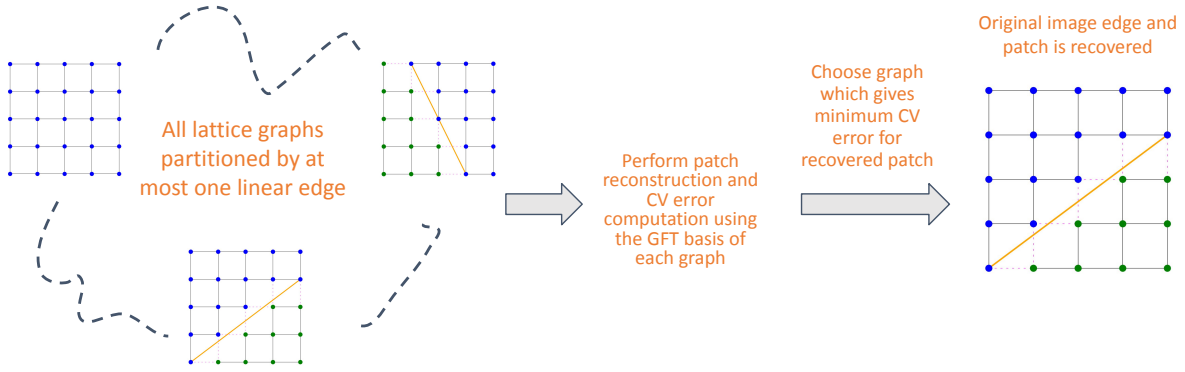
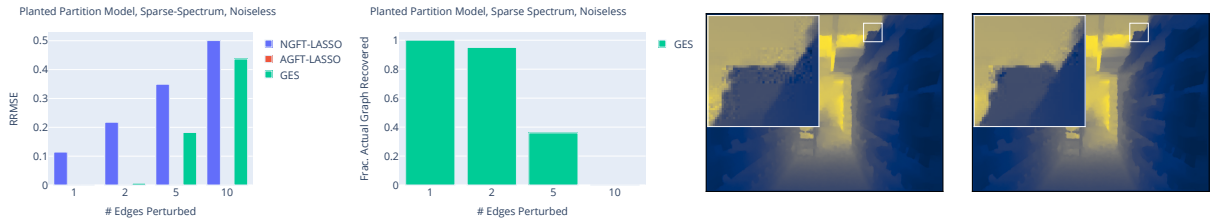


Figure 1.6: Illustration of the ILECIR algorithm

We find an application of the compressive perturbed graph recovery problem in the recovery of an image from compressive measurements taken patch-wise such as in [62, 63]. Typically, each image patch may be recovered via LASSO with the DCT basis, which is a GFT of the 2D lattice graph of same size as the patch. This works because in natural images and piece-wise smooth images, adjacent pixels have correlated values, which is modelled well by



(a) GES RMSE **(b)** GES Graph Recovery **(c)** NYU Depth, DCT **(d)** NYU Depth, ILECIR

Figure 1.7: A sample result using GES and ILECIR. (a) Mean value of the Relative Root Mean Square Error (RRMSE) of the recovered signal. (b) Fraction of cases in which actual graph was recovered. (c) Recovery of a depth-map image via DCT-based recovery. (d) Recovery of the same image via ILECIR. See Chapter 4 for a detailed discussion on these results.

the low-frequency vectors of the GFT basis. However, if an image has a sharp edge, then such correlation is broken for the pixels on the opposite sides of the edge. Hence the representation of an image patch containing a sharp edge in the DCT basis will not be very sparse, resulting in poor recovery from compressive measurements. An alternative is to use the GFT basis of the graph obtained after dropping the edges of the 2D lattice graph which go across the image edge, so that the correlation of the value of these basis vectors on nodes which are on opposing sides of the image edge gets broken. We call this the image edge partitioned graph. However, since the location of the image edge is unknown a-priori, this graph is not known. Thus we have an instance of the compressive perturbed graph recovery problem. This problem is solved in a similar manner as the GES algorithm by comparing cross-validation errors of signals recovered using the GFT of different image edge partitioned graphs, where only linear image edges are considered. Notably, the graph edges are dropped in a structured manner rather than one by one. We call this algorithm Inferred Linear-Edge Compressive Image Recovery (ILECIR), illustrated in Fig. 1.6. A sample image reconstruction result using GES and ILECIR is illustrated in Fig. 1.7.

The contributions made in this part of the thesis are summarized below:

1. We present the Compressive Perturbed Graph Recovery problem, which is a new kind of basis perturbation problem in the context of compressed sensing.

2. We solve this prolem via a novel algorithm called the Greedy Edge Selection algorithm, which iteratively refines the nominal graph one edge at a time, using cross-validation to disambiguate between candidate graphs to recover the actual graph and the signal.

3. We frame the problem of edge-aware recovery of compressively acquired images as a compressive perturbed graph recovery problem, and solve it via a new algorithm called Inferred Linear-Edge Compressive Recovery, which performs structured edge perturbations as opposed to the GES.

4. Using cross-validation theory for compressed sensing [60, 61], we prove guarantees for recovery of the signal and the actual graph from *noisy* measurements for a brute-force version of the GES algorithm, and give similar solution improvement guarantees for the GES and ILECIR algorithms.

5. We perform an extensive empirical evaluation of GES using synthetic data on commonly used graphs from the Network Science literature [64]. A sample result in Fig. 1.7 shows that our method makes signal recovery viable for upto 5 edge perturbations.

6. We evaluate ILECIR performance on image recovery from compressive measurements on a variety of images – synthetically generated piece-wise smooth images, natural images, cartoon images, and depth-maps. A sample recovered image is shown in Fig. 1.7, demonstrating superiority of ILECIR over DCT-based recovery.

7. Our cross-validation based approach is agnostic to the underlying signal model (e.g. sparsity in GFT) and recovery method (e.g. LASSO), and may be used for graph signals which obey some other signal model than sparsity of GFT. As an example, we demonstrate empirically that sparsity in Graph Total Variation [50] may be exploited in ILECIR.

A manuscript for this part of the thesis is being prepared for submission to a reputed peer-review journal.

1.4 Organization of the thesis

The remaining chapters are organized as follows. Chapter 2 contains the details for the Pooled RT-PCR Testing for COVID-19 work. Chapter 3 gives the details for the Efficient Automated Image Moderation work. Chapter 4 gives the details for the Compressive Perturbed Graph Recovery work. Each chapter is self-contained and outlines the relevant background, literature

review, detailed description of experiments, data used, training methodology, empirical results, as well as proofs of theoretical results. Some lengthy material such as elaborate proofs, side investigations, additional results, or details necessary for reproduction of experiments, are included as appendices of the relevant chapter. Concluding remarks are made in Chapter 5, with a discussion on possible extensions of the work done in this thesis.

This page was intentionally left blank.

Chapter 2

Pooled RT-PCR Testing for COVID-19 Detection

2.1 Introduction

The coronavirus disease of 2019 (COVID-19) crisis led to widespread lockdowns in several countries, and had a major negative impact on the economy. Early identification of infected individuals was desirable for quarantining of the individuals and thus controlling the spread of the disease. Such individuals could often be asymptomatic for many days. Widespread testing with the RT-PCR (reverse transcription polymerase chain reaction) method could help identify the infected individuals. However, widespread testing was not an available option in many countries due to constraints on resources such as testing time ($\sim 3-4$ hours per round), basic equipment, skilled manpower and reagents.

The low rate of COVID-19 infection in the world population [65] meant that most samples tested were not infected, so that most tests were wasted on uninfected samples. Group testing is a process of pooling together samples of n different people into multiple pools, and testing

the pools instead of each individual sample. A negative result on a pool implies that all samples participating in it were negative. This saves a huge amount of testing resources, especially with low infection rates. Group testing for medical applications has a long history dating back to the 1940s when it was proposed for testing of blood samples for syphilis [28]. Simple two-round group testing schemes had already been applied in the field by several research labs [66, 67] for COVID-19 testing. Such two-round group testing schemes require pooling of samples and a second round of sample handling for all samples in positive pools. This second round of sample handling can increase the time to result and be laborious to perform since it requires the technician to wear PPE one more time, do another round of RNA extraction, and PCR. In situations where the result needs to be delivered fast, a second round of sample handling and testing must be avoided. In such situations, these schemes are less attractive.

In this chapter, we present Tapestry, a novel combination of ideas from combinatorial group testing and compressed sensing (CS) [3] which uses the quantitative output of PCR tests to reconstruct the viral load of each sample in a single round. Tapestry has been validated with wet lab experiments with oligomers [16]. In this work, we elaborate on the results from the algorithmic perspective for the computer science and signal processing communities. We enumerate the salient features of the Tapestry method and our contributions in this chapter below:

1. Tapestry delivers results in a single round of testing, without the need for a second confirmatory round, at clinically acceptable false negative and false positive rates. The number m of required tests is only $O(k \log n)$ for random binary pooling matrix constructions, as per compressed sensing theory for random binary matrices [12]. In the targeted use cases where the number of infected samples $k \ll n$, we see that $m \ll n$. However, our deterministic pooling matrix constructions based on Kirkman Triple Systems [32, 68] require fewer tests in practice (see Sec. 2.3.7.8 for a discussion on why this may be the case). Consequently we obtain significant savings in testing time and resources such as number of tests, quantity of reagents, and manpower.

2. Tapestry reconstructs relative viral loads i.e., ratio of viral amounts in each sample to the highest viral amount across pools. It is believed that super-spreaders and people with severe symptoms have higher viral load [30, 31], so this quantitative information might have epidemiological relevance.

3. Tapestry takes advantage of quantitative information in PCR tests. Hence it returns far fewer false positives than traditional binary group testing algorithms such as COMP (Combinatorial Orthogonal Matching Pursuit)[29], while maintaining clincally acceptable false negative rates. Furthermore, it takes advantage of the fact that a pool tests negative if and only if it has viral load exactly zero. Traditional CS algorithms do not take advantage of this information. Hence, Tapestry demonstrates better sensitivity and specificity than CS algorithms.

4. The combination of COMP and CS as done in Tapestry is theoretically valid – we prove that conditions sufficient for recovery via CS such as the Restricted Isometry Property (RIP) [4] are preserved for the matrix obtained via reduction after COMP. We prove that such conditions are also preserved by a wider class of reductions of a matrix, which includes reductions via noisy group testing algorithms such as Noisy COMP (NCOMP) [29]. Thus the Tapestry method may be enhanced to recover from false negative pooled tests by using NCOMP in the first stage instead of COMP.

5. Because each sample is tested in three pools, Tapestry can detect some degree of noise in terms of cross-contamination of samples and pipetting errors.

6. Tapestry allows PCR test measurements to be noisy. We develop a novel noise model to describe noise in PCR experiments. Our algorithms are tested on this noise model in simulation.

7. All tuning parameters for execution of the algorithms are inferred on the fly in a data driven fashion.

8. Each sample contributes to exactly three pools, and each pool has the same number of samples. This simplifies the experimental design, conserves samples, keeps pipetting overhead to a minimum, and makes sure that dilution due to pool size is in a manageable regime.

9. We developed the Byom app [34] for easy manual pooling of samples, decoding test results, and keeping track of tests.

The organization of this chapter is as follows. We first present a brief overview of the RT-PCR method in Sec. 2.2. The precise mathematical definition of the computational problem

being solved in this chapter is then put forth in Sec. 2.3.1. We describe traditional and CS-based group-testing algorithms for this problem in Sec. 2.3.2, 2.3.3 and 2.3.4. The Tapestry method is described in Sec. 2.3.4. Theorems on RIP preservation are given in Sec. 2.3.5. The sensing matrix design problem, as well as theoretical guarantees using Kirkman Triple Systems or random binary matrices, are described in Sec. 2.3.7. Results on synthetic data are presented in Sec. 2.4. This is followed by results on data from lab experiments performed with oligomers to mimic the clinical situation as closely as possible. In Sec. 2.5, we compare our work to recent related approaches. We conclude in Sec. 2.6 with a glance through different scenarios where our work could be deployed. The appendices contain several additional experimental details as well as proofs of some theoretical results.

Author Contributions: The author of this thesis has worked on the Tapestry method under the guidance of Prof. Manoj Gopalkrishnan and Prof. Ajit Rajwade. This work was done in collaboration with a number of other researchers, and was published in [14, 16, 69]. The contents of this chapter first appeared in [14] and have been reproduced here with minor changes, except the proof of RIP preservation by NCOMP-reduction, which is new. The author of this thesis has made the following contributions to the work: proposal and implementation of the Tapestry method, derivation of noise and data generation model, experimentation with synthetically generated data, proof of RIP preservation, proofs of disjunctness, expansion, RIP-1, and large COMP-reduction properties of Kirkman matrices, comparison with Dorfman pooling, implementation of graceful failure mode, comparison with related work.

2.2 RT-PCR Method

We present here a brief summary of the RT-PCR process, referring to [24] for more details. In the RT-PCR method for COVID-19 testing, a sample in the form of naso- or oro-pharyngeal swabs is collected from a patient. The sample is then dispersed into a liquid medium. The RNA molecules of the virus present in this liquid medium are converted into complementary DNA (cDNA) via a process called reverse transcription. DNA fragments called primers complementary to cDNA from the viral genome are then added. They attach themselves to specific sections of the cDNA from the viral genome if the virus is present in the sample. The cDNA of these spe-

cific viral genes then undergoes a process of exponential amplification in an RT-PCR machine. Here, cDNA is put through several cycles of alternate heating and cooling in the presence of Taq polymerase and appropriate reagents. This triggers the creation of many new identical copies of specific portions of the target DNA, roughly doubling in number with every cycle of heating and cooling. The reaction volume contains sequence-specific fluorescent markers which report on the total amount of amplified DNA of the appropriate sequence. The resulting fluorescence is measured, and the increase can be observed on a computer screen in real time. The time when the amount of fluorescence exceeds the threshold level is known as the threshold cycle C_t, and is a quantitative readout from the experiment. A smaller C_t indicates greater number of copies of the virus. Usually C_t takes values anywhere between 16 to 32 cycles in real experiments. PCR can detect even single molecules. A single molecule typically would have C_t value of around 40 cycles. A typical RT-PCR setup can test 96 samples in parallel. The test takes about 3-4 hours to execute.

2.3 Testing Methods

2.3.1 Statement of the Computational Problem

Let $\boldsymbol{x}$ denote a vector of n elements where x_i is the viral load (i.e. viral amount) of the i^{th} person. Throughout this chapter we assume that only one sample per person is extracted. Hence $\boldsymbol{x}$ contains the viral loads corresponding to n different people. Note that $x_i = 0$ implies that the i^{th} person is not infected. Due to the low infection rate for COVID-19 even in severely affected countries [65], $\boldsymbol{x}$ is considered to be a sparse vector with at the most $k \ll n$ positive-valued elements. In group testing, small and equal volumes of the samples of a subset of these n people are pooled together according to a sensing or pooling matrix $\boldsymbol{A} = (A_{ji})_{m\times n}$ whose entries are either 0 or 1. The viral loads of the pools are given by:

$$z_j = \sum_{i=1}^{n} A_{ji}x_i = \boldsymbol{A^j x}, 1 \leq j \leq m, 1 \leq i \leq n, \tag{2.1}$$

where $A_{ji} = 1$ if a portion of the sample of the i^{th} person is included in the j^{th} pool, and $\boldsymbol{A^j}$ is the j^{th} row of $\boldsymbol{A}$. In all, some $m < n$ pools are created and individually tested using RT-PCR. We now have the relationship $\boldsymbol{z} = \boldsymbol{Ax}$, where $\boldsymbol{z}$ is the m-element vector of viral loads in the mixtures, and $\boldsymbol{A}$ denotes a $m \times n$ binary 'pooling matrix' (also referred to as a 'sensing matrix' in CS literature). Note that each positive RT-PCR test yields a noisy version of z_j, which we refer to as y_j. The relation between the 'clean' and noisy versions is given as follows (also see Eqn. 2.7):

$$y_j = z_j(1+q)^{e_j} = (1+q)^{e_j}\boldsymbol{A^j x}, \tag{2.2}$$

where $e_j \sim \mathcal{N}(0, \sigma^2)$ and $q \in (0, 1)$ is the fraction of viral cDNA that replicates in each cycle. The factor $(1+q)^{e_j}$ reflects the stochasticity in the growth of the numbers of DNA molecules during PCR. Here σ is known and constant. Equivalently for positive tests, we have:

$$\log y_j = \log(\boldsymbol{A^j x}) + \log(1+q)e_j. \tag{2.3}$$

In case of negative tests, y_j as well as z_j are 0-valued, and no logarithms need to be computed. In non-adaptive group testing, the core computational problem is to estimate $\boldsymbol{x}$ given $\boldsymbol{y}$ and $\boldsymbol{A}$ without requiring any further pooled measurements. It should be noted that though we have treated each element of $\boldsymbol{x}$ to be a fixed quantity, it is in reality a random variable of the form $x_i \sim$ Poisson(λ_i) where $\lambda_i \geq 0$. If matrix $\boldsymbol{A}$ contains only ones and zeros, this implies that $z_j \sim$ Poisson$(\boldsymbol{A^j x})$ because the sum of Poisson random variables is also a Poisson random variable.

2.3.1.1 Derivation of Noise Model

For a positive pool j, the quantitative readout from RT-PCR is not its viral load but the observed cycle time t_j when its fluorescence reaches a given threshold F (see Sec. 2.2). In order to be able to apply CS techniques (see Sec. 2.3.3), we derive a relationship between the cycle time of a sample and its viral load. Because of exponential growth (see [70]), the number of molecules

of viral cDNA in pool j at cycle time t, denoted by $v_j(t)$ is given by:

$$v_j(t) = z_j(1+q)^t. \tag{2.4}$$

Also, t is a real number, with $\lfloor t \rfloor$ indicating the number of PCR cycles that have passed, and $t - \lfloor t \rfloor$ indicating the fraction of wall-clock time within the current cycle. The fluorescence of the pool, $f_j(t)$, is directly proportional to the number of virus molecules $v_j(t)$. That is,

$$f_j(t) = Kv_j(t) = Kz_j(1+q)^t, \tag{2.5}$$

where K is a constant of proportionality. Suppose the fluorescence of pool j should reach the threshold value F at cycle time τ_j, according to Eqn. 2.5. Due to the stochastic nature of the reaction, as well as measurement error in the PCR machine, the threshold cycle output by the machine will not reflect this true cycle time. We model this discrepancy as Gaussian noise. Hence, the true cycle time τ_j and the observed cycle time t_j are related as $\tau_j = t_j + e_j$, where $e_j \sim \mathcal{N}(0, \sigma^2)$ as before. Now, since $f_j(\tau_j) = F$, using Eqn. 2.5, we have

$$F = Kz_j(1+q)^{\tau_j} = Ky_j(1+q)^{t_j}. \tag{2.6}$$

The latter equality is since we use the noisy cycle threshold t_j to compute viral load, where y_j is defined to be the noisy viral load of pool j. Hence we find

$$y_j = z_j(1+q)^{\tau_j - t_j} = z_j(1+q)^{e_j} = (1+q)^{e_j}\boldsymbol{A^j x}, \tag{2.7}$$

obtaining the relationship from Eqn. 2.2.

Constants F and K are unknown. Hence it is not possible to directly obtain y_j from t_j without additional machine-specific calibration. However, we can find the ratio between the noisy viral loads of two pools using Eqn. 2.6. Let y_{min} be the noisy viral load of the pool with the minimum observed threshold cycle (t_{min}) among all pools. Then we define relative viral

loads as:

$$\widetilde{y}_j = \frac{y_j}{y_{min}} = (1+q)^{t_{min}-t_j}, \widetilde{z}_j = \frac{z_j}{y_{min}}, \widetilde{\boldsymbol{x}} = \frac{\boldsymbol{x}}{y_{min}} \tag{2.8}$$

where $\widetilde{z}_j$ is the relative viral load of a pool, $\widetilde{y}_j$ is its noisy version, and $\widetilde{\boldsymbol{x}}$ is the vector of relative viral loads of each sample. We note that due to Eqn. 2.7, the following relation holds:

$$\widetilde{y}_j = \widetilde{z}_j(1+q)^{e_j} = (1+q)^{e_j}\boldsymbol{A^j}\widetilde{\boldsymbol{x}}, \tag{2.9}$$

Hence we can apply CS techniques from Sec. 2.3.3 to determine the relative magnitudes of viral loads without knowing F and K. We provide more comments about the settings of various noise model parameters for our experiments, in Sec. 2.4, particularly in Sec. 2.4.1.6.

2.3.2 Combinatorial Group-Testing

Combinatorial Orthogonal Matching Pursuit (COMP) is a binary nonadaptive group testing method [71, Sec. 2.3]. Here one uses the simple idea that if a mixture $\widetilde{y}_j$ tests negative then any sample $\widetilde{x}_i$ for which $A_{ji} = 1$ must be negative. Note that pools which test negative are regarded as noiseless observations, as argued in Sec. 2.3.1.1. The other samples are all considered to be positive. This algorithm guarantees that there are no 'false negatives'. However it can produce a very large number of 'false positives'. For example, a sample $\widetilde{x}_k$ will be falsely reported to be positive if every mixture $\widetilde{y}_j$ it is part of, also contains at least one other genuinely positive sample. The COMP algorithm is largely insensitive to noise. Moreover a small variant of it can also produce a list of 'high confidence positives', after identifying the (sure) negatives. This happens when a positive mixture $\widetilde{y}_j$ contains only one sample $\widetilde{x}_i$, not counting the other samples which were declared sure negatives in the earlier step. Such a step of identifying 'high confidence positives' is included in the so-called **Definite Defectives** (DD) Algorithm [71, Sec. 2.4]. However DD labels all remaining items to be negative, potentially leading to a large number of false-negatives. The performance guarantees for COMP have been analyzed in [29] and show that COMP requires $ek(1+\delta)\log n$ tests for an error probability less than $n^{-\delta}$ (see Sec. 2.3.7.8). This analysis has been extended to include the case of noisy test results as well [29]. However

COMP can result in a large number of false positives if not enough tests are used, and it also does not predict viral loads.

2.3.3 Compressed Sensing for Pooled Testing

Group testing is intimately related to the field of compressed sensing (CS) [72], which has emerged as a significant sub-area of signal and image processing [3], with many applications in biomedical engineering [73, 74, 75]. In CS, an image or a signal $\boldsymbol{x}$ with n elements, is directly acquired in compressed format via m linear measurements of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\eta}$. Here, the measurement vector $\boldsymbol{y}$ has m elements, and $\boldsymbol{A}$ is a matrix of size $m \times n$, and $\boldsymbol{\eta}$ is a vector of noise values. If $\boldsymbol{x}$ is a sparse vector with $k \ll n$ non-zero entries, and $\boldsymbol{A}$ obeys the so-called restricted isometry property (RIP), then *exact* recovery of $\boldsymbol{x}$ from $\boldsymbol{y}, \boldsymbol{A}$ is possible [4] if $\boldsymbol{\eta} = \boldsymbol{0}$. In the case of measurement noise, the recovery of $\boldsymbol{x}$ produces a solution that is provably close to the original $\boldsymbol{x}$. A typical recovery problem P0 consists of optimizing the following cost function:

$$\min \|\boldsymbol{x}\|_0 \text{ s.t. } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \varepsilon, \tag{2.10}$$

where ε is an upper bound (possibly a high probability upper bound) on $\|\boldsymbol{\eta}\|_2$, and $\|\boldsymbol{x}\|_0$ is the number of non-zero elements in $\boldsymbol{x}$. In the absence of noise, a unique and exact solution to this problem is possible with as few as $2k$ measurements in $\boldsymbol{y}$ if $\boldsymbol{x}$ has k non-zero elements [4]. Unfortunately, this optimization problem P0 is NP-Hard and the algorithm requires brute-force subset enumeration. Instead, the following problem P1 (often termed 'Basis Pursuit Denoising' or BPDN) is solved in practice:

$$\min \|\boldsymbol{x}\|_1 \text{ s.t. } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \varepsilon. \tag{2.11}$$

P1 is a convex optimization problem which yields the same solution as the earlier problem (with similar conditions on $\boldsymbol{x}, \boldsymbol{A}$) at significantly lower computational cost, albeit with $O(k \log n)$ measurements (i.e. typically greater than $2k$) [3, 4].

The order k restricted isometry constant (RIC) of a matrix $\boldsymbol{A}$ is defined as the smallest constant δ_k, for which the following relationship holds for all k-sparse vectors $\boldsymbol{x}$ (i.e. all vectors with at the most k non-zero entries): $(1-\delta_k)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{x}\|_2^2$. The matrix $\boldsymbol{A}$ is said to obey the order k restricted isometry property (RIP) if δ_k is close to 0. This property essentially implies that no k-sparse vector (other than the zero vector) can lie in the null-space of $\boldsymbol{A}$. Unique recovery of k-sparse signals requires that no $2k$-sparse vector lies in the nullspace of $\boldsymbol{A}$ [4]. A matrix $\boldsymbol{A}$ which obeys RIP of order $2k$ satisfies this property. It has been proved that matrices with entries randomly and independently drawn from distributions such as Rademacher or Gaussian, obey the RIP of order k with high probability [11], provided they have at least $O(k \log n)$ rows. There also exist **deterministic binary sensing matrix designs** (e.g. [76]) which require $O(\max(k^2, \sqrt{n}))$ measurements. However it has been shown recently [10] that the constant factors in the deterministic case are *significantly smaller* than those in the former random case when $n < 10^5$, making the deterministic designs more practical for typically encountered problem sizes. The solution to the optimization problems P0 and P1 in Eqns. 2.10 and 2.11 respectively, are provably robust to noise [3], and the recovery error decreases with decrease in noise magnitude. The error bounds for P0 in Eqn. 2.10 are of the form, for solution $\hat{\boldsymbol{x}}$ [77]:

$$\frac{\varepsilon}{\sqrt{1+\delta_{2k}}} \leq \|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2 \leq \frac{\varepsilon}{\sqrt{1-\delta_{2k}}}, \tag{2.12}$$

whereas those for P1 in Eqn. 2.11 have the form [77]:

$$\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2 \leq \varepsilon\zeta(\delta_{2k}). \tag{2.13}$$

Here $\zeta(\delta_{2k})$ is a monotonically increasing function of $\delta_{2k} \in (0,1)$ and has a small value in practice.

The Restricted Isometry Property as defined above is also known as RIP-2, because it uses the ℓ_2-norm. Many other sufficient conditions for recovery of k-sparse vectors exist. We define the following which we use later in Sec. 2.3.7 and Appendix 2.D to prove theoretical guarantees of our method.

Definition 2.1. ***RIP-1:*** *[78, Defn. 8] A $m \times n$ matrix $\boldsymbol{A}$ is said to obey RIP-1 of order k if* $\exists$

$\delta_k \in (0,1)$ *such that for all* k*-sparse vectors* $\boldsymbol{x} \in \mathbb{R}^n$*,*

$$\|\boldsymbol{x}\|_1 \leq \|\boldsymbol{A}\boldsymbol{x}\|_1 \leq (1+\delta_k)\|\boldsymbol{x}\|_1$$

.

Definition 2.2. ***RNSP:*** *[10, Eqn. 12] A* $m \times n$ *matrix* $\boldsymbol{A}$ *is said to obey the Robust Nullspace Property (RNSP) of order* k *if* $\exists\ \rho < 1$ *and* $\tau > 0$ *such that for all* $\boldsymbol{x} \in \mathbb{R}^n$ *it holds that*

$$||\boldsymbol{x}_S||_2 \leq \rho||\boldsymbol{x}_{\bar{S}}||_1 + \tau||\boldsymbol{A}\boldsymbol{x}||_2$$

for all $S \subset \{1 \ldots n\}$ *with* $|S| \leq k$*.*

Definition 2.3. ℓ_2***-RNSP:*** *[12, Defn. 1] A* $m \times n$ *matrix* $\boldsymbol{A}$ *is said to obey the* ℓ_2*-robust Nullspace Property (*ℓ_2*-RNSP) of order* k *if* $\exists\ \rho \in (0,1)$ *and* $\tau > 0$ *such that for all* $\boldsymbol{x} \in \mathbb{R}^n$ *it holds that*

$$||\boldsymbol{x}_S||_2 \leq \frac{\rho}{\sqrt{k}}||\boldsymbol{x}_{\bar{S}}||_1 + \tau||\boldsymbol{A}\boldsymbol{x}||_2$$

for all $S \subset \{1 \ldots n\}$ *with* $|S| \leq k$*.*

Over the years, a variety of different techniques for compressive recovery have been proposed. We use some of these for our experiments in Sec. 2.3.4. These algorithms use different forms of sparsity and incorporate different types of constraints on the solution.

2.3.4 CS and Traditional GT Combined

The complete pipeline of the Tapestry method is presented in Algorithm 2.1. First, a wet lab technician performs pooling of n samples into m pools according to a $m \times n$ pooling matrix $\boldsymbol{A}$. Then they run the RT-PCR test on these m pools (in parallel). The output of the RT-PCR tests – the threshold cycle (C_t) values of each pool – is processed to find the relative viral load vector $\widetilde{\boldsymbol{y}}$ of the m pools (as shown in Eqn. 2.8). This is given as input to the Tapestry decoding algorithm, which outputs a sparse relative viral load vector $\widetilde{\boldsymbol{x}}$.

Algorithm 2.1 Tapestry Method

Input: n samples, $m \times n$ pooling matrix $\boldsymbol{A}$

1: Perform pooling according to pooling matrix $\boldsymbol{A}$ and create m pooled samples

2: Run RT-PCR test on these m pooled samples and receive $m \times 1$ vector of cycle threshold values $\boldsymbol{t}$

3: Compute $m \times 1$ vector of relative viral loads $\widetilde{\boldsymbol{y}}$ from $\boldsymbol{t}$

4: Use COMP to filter out negative tests and sure negative samples. Compute submatrix $\boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}$, $\widetilde{\boldsymbol{y}}_{\bar{\mathcal{Y}}}$ and list $\mathcal{HCP}$ of 'high-confidence positives' along with their viral loads (see Sec. 2.3.2).

5: Use a CS decoder to recover relative viral loads $\widetilde{\boldsymbol{x}}_{\bar{\mathcal{X}}}$ from $\widetilde{\boldsymbol{y}}_{\bar{\mathcal{Y}}}$, $\boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}$

6: Compute $n \times 1$ relative viral load vector $\widetilde{\boldsymbol{x}}$ by setting its entries from $\widetilde{\boldsymbol{x}}_{\bar{\mathcal{X}}}$, and setting remaining entries to 0.

7: **return** $\widetilde{\boldsymbol{x}}$, $\mathcal{HCP}$.

The Tapestry decoding algorithm, our approach toward group-testing for COVID-19, involves a two-stage procedure[1]. In the first stage, we apply the COMP algorithm described in Sec. 2.3.2, to identify the sure negatives (if any) in $\widetilde{\boldsymbol{x}}$ to form a set $\mathcal{X}$. Let $\mathcal{Y}$ be the set of zero-valued measurements in $\widetilde{\boldsymbol{y}}$ (i.e. negative tests). Please refer to Sec. 2.3.1.1 for the definition of $\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}$. Moreover, we define $\bar{\mathcal{X}}, \bar{\mathcal{Y}}$ as the complement-sets of $\mathcal{X}, \mathcal{Y}$ respectively. Also, let $\boldsymbol{y}_{\bar{\mathcal{Y}}}$ be the vector of $m - |\mathcal{Y}|$ measurements which yielded a positive result. Let $\boldsymbol{x}_{\bar{\mathcal{X}}}$ be the vector of $n - |\mathcal{X}|$ samples, which does not include the $|\mathcal{X}|$ surely negative samples. Let $\boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}$ be the submatrix of $\boldsymbol{A}$, having size $(m - |\mathcal{Y}|) \times (n - |\mathcal{X}|)$, which excludes rows corresponding to zero-valued measurements in $\boldsymbol{y}$ and columns corresponding to negative elements in $\boldsymbol{x}$. In the second stage, we apply a CS algorithm to recover $\widetilde{\boldsymbol{x}}_{\bar{\mathcal{X}}}$ from $\widetilde{\boldsymbol{y}}_{\bar{\mathcal{Y}}}$, $\boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}$. *To avoid symbol clutter, we henceforth just stick to the notation* $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{A}, m, n$*, even though they respectively refer to* $\widetilde{\boldsymbol{y}}_{\bar{\mathcal{Y}}}, \widetilde{\boldsymbol{x}}_{\bar{\mathcal{X}}}, \boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}, m - |\mathcal{Y}|, n - |\mathcal{X}|$.

Note that the CS stage following COMP is very important for the following reasons:

1. COMP typically produces a large number of false positives. The CS algorithms help reduce the number of false positives as we shall see in later sections.

2. COMP does not estimate viral loads, unlike CS algorithms.

3. In fact, unlike CS algorithms, COMP treats the measurements in $\boldsymbol{y}$ as also being binary, thus

[1]The two-stage procedure is purely algorithmic. It does not require two consecutive rounds of testing in a lab.

discarding a lot of useful information.

4. COMP preserves the RIP-1, RIP-2, RNSP, and ℓ_2-RNSP of the pooling matrix, i.e. if $\boldsymbol{A}$ obeys any of RIP-1, RIP-2, RNSP or ℓ_2-RNSP of order k, then $\boldsymbol{A}_{\bar{\mathcal{X}},\bar{\mathcal{Y}}}$ also obeys the same property of the same order k with the same parameters. We formalize these claims in Sec. 2.3.5 and prove them in Appendix 2.D.

However, the COMP algorithm prior to applying the CS algorithm is also very important for the following reasons:

1. Viral load in negative pools is exactly 0. COMP identifies the sure negatives in $\boldsymbol{x}$ from the negative measurements in $\boldsymbol{y}$. Traditional CS algorithms do not take advantage of this information, since they assume all tests to be noisy (Eqns. 2.10 and 2.11). It is instead easier to discard the obvious negatives before applying the CS step.

2. Since COMP identifies the sure negatives, therefore, it effectively reduces the size of the problem to be solved by the CS step from (m, n) to $(m - |\mathcal{Y}|, n - |\mathcal{X}|)$.

3. In a few cases, a (positive) pool in $\bar{\mathcal{Y}}$ may contain only one contributing sample in $\bar{\mathcal{X}}$, after negatives have been eliminated by COMP. Such a sample is called a 'high-confidence positive', and we denote the list of high-confidence positives as $\mathcal{HCP}$. In rare cases, the CS decoding algorithms we employed (see further in this section) did not recognize such a positive. However, such samples will still be returned by our algorithm as positives, in the set $\mathcal{HCP}$ (see last step of Alg. 2.1, and 'definite defectives' in Sec. 2.3.2).

For CS recovery, we employ one of the following algorithms after COMP: the non-negative LASSO (NNLASSO), non-negative orthogonal matching pursuit (NNOMP), Sparse Bayesian Learning (SBL), and non-negative absolute deviation regression (NNLAD). For problems of small size, we also apply a brute force (BF) search algorithm to solve a problem similar to P0 from Eqn. 2.10 combinatorially.

2.3.4.1 The Non-negative LASSO (NNLASSO)

The LASSO (least absolute shrinkage and selection operator) is a penalized version of the constrained problem P1 in Eqn. 2.11, and seeks to minimize the following cost function:

$$J_{lasso}(\boldsymbol{x}; \boldsymbol{y}, \boldsymbol{A}) := \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1. \tag{2.14}$$

Here λ is a regularization parameter which imposes sparsity in $\boldsymbol{x}$. The LASSO has rigorous theoretical guarantees [6, Chapter 11] for recovery of $\boldsymbol{x}$ as well as recovery of the support of $\boldsymbol{x}$ (i.e. recovery of the set of non-zero indices of $\boldsymbol{x}$). Given the non-negative nature of $\boldsymbol{x}$, we implement a variant of LASSO with a non-negativity constraint, leading to the following optimization problem:

$$J_{nnlasso}(\boldsymbol{x}; \boldsymbol{y}, \boldsymbol{A}) := \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \text{ s.t. } \boldsymbol{x} \geq \boldsymbol{0}. \tag{2.15}$$

Selection of λ: There are criteria defined in [6] for selection of λ under i.i.d. Gaussian noise, so as to guarantee statistical consistency. However, in practice, **cross-validation** (CV) can be used for optimal choice of λ in a purely data-driven fashion from the available measurements. The details of this are provided in Appendix 2.B.

2.3.4.2 Non-negative Orthogonal Matching Pursuit (NNOMP)

Orthogonal Matching Pursuit (OMP) [79] is a greedy approximation algorithm to solve the optimization problem in Eqn. 2.10. Rigorous theoretical guarantees for OMP have been established in [80]. OMP proceeds by maintaining a set $\mathcal{H}$ of 'selected coefficients' in $\boldsymbol{x}$ corresponding to columns of $\boldsymbol{A}$. In each round a column of $\boldsymbol{A}$ is picked greedily, based on the criterion of maximum absolute correlation with a residual vector $\boldsymbol{r} := \boldsymbol{y} - \sum_{k\in\mathcal{H}} \boldsymbol{A_k}\hat{x}_k$. Each time a column is picked, *all* the coefficients extracted so far (i.e. in set $\mathcal{H}$) are updated. This is done by computing the orthogonal projection of $\boldsymbol{y}$ onto the subspace spanned by the columns in $\mathcal{H}$. The OMP algorithm can be quite expensive computationally. Moreover, in order to maintain non-negativity of $\boldsymbol{x}$, the orthogonal projection step would require the solution of a non-negative least squares problem, further adding to computational costs. However, a fast implementation

of a non-negative version of OMP (NNOMP) has been developed in [81], which is the implementation we adopt here. For the choice of ε in Eqn. 2.10, we can use CV as described in Sec. 2.3.4.1.

2.3.4.3 Sparse Bayesian Learning (SBL)

Sparse Bayesian Learning (SBL) [7, 8] is a non-convex optimization algorithm based on Expectation-Maximization (EM) that has empirically shown superior reconstruction performance to most other CS algorithms with manageable computation cost [82]. In SBL, we consider the case of Gaussian noise in $\boldsymbol{y}$ and a Gaussian prior on elements of $\boldsymbol{x}$, leading to:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(-\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2/(2\sigma^2))}{(2\pi\sigma^2)^{n/2}} \tag{2.16}$$

$$p(x_i; \varphi_i) = \frac{\exp(-x_i^2/(2\varphi_i))}{\sqrt{2\pi\varphi_i}}; \varphi_i \geq 0. \tag{2.17}$$

Since both $\boldsymbol{x}$ and $\boldsymbol{\varphi}$ (the vector of the $\{\varphi_i\}_{i=1}^n$ values) are unknown, the optimization for these quantities can be performed using an EM algorithm. In the following, we shall denote $\boldsymbol{\Phi} := \text{diag}(\boldsymbol{\varphi})$. Moreover, we shall use the notation $\boldsymbol{\Phi}^{(l)}$ for the estimate of $\boldsymbol{\Phi}$ in the l^{th} iteration. The E-step of the EM algorithm here involves computing $Q(\boldsymbol{\Phi}|\boldsymbol{\Phi}^{(l)}) := E_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\Phi}^{(l)}} \log p(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{\Phi})$. It is to be noted that the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\Phi}^{(l)})$ has the form $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu} := \boldsymbol{\Sigma}\boldsymbol{A}^T\boldsymbol{y}/\sigma^2$ and $\boldsymbol{\Sigma} := (\boldsymbol{A}^T\boldsymbol{A}/\sigma^2 + (\boldsymbol{\Phi}^{(l)})^{-1})^{-1}$. The M-step involves maximization of $Q(\boldsymbol{\Phi}|\boldsymbol{\Phi}^{(l)})$, leading to the update $\boldsymbol{\Phi}^{(l+1)} = \text{diag}(\mu_i^2 + \Sigma_{ii})$. The E-step and M-step are executed alternately until convergence. Convergence to a fixed-point is guaranteed, though the fixed point may or may not be a local minimum. However, all local minima are guaranteed to produce sparse solutions for $\boldsymbol{x}$ (even in the presence of noise) because most of the φ_i values shrink towards 0. The SBL procedure can also be modified to dynamically update the noise variance σ^2 (as followed in this work), if it is unknown. All these results can be found in [8]. Unlike NNLASSO or NNOMP, the SBL algorithm from [8] expressly requires Gaussian noise. However we use it as is in this work for the simplicity it affords. Unlike NNOMP or NNLASSO, there is no explicit non-negativity constraint imposed in the basic SBL algorithm. In our implementation, the non-negativity is simply imposed at the end of the optimization by setting to 0 any negative-valued elements in $\boldsymbol{\mu}$, though more principled, albeit more computationally heavy, approaches such as [83] can be adopted.

2.3.4.4 Non-negative Absolute Deviation Regression (NNLAD)

The Non-Negative Absolute Deviation Regression (NNLAD) [84] and Non-negative Least squares (NNLS) [12] seek to respectively minimize

$$J_{nnlad}(\boldsymbol{x}; \boldsymbol{y}, A) := \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_1 \text{ s.t. } \boldsymbol{x} \geq \boldsymbol{0}, \tag{2.18}$$

$$J_{nnls}(\boldsymbol{x}; \boldsymbol{y}, A) := \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \text{ s.t. } \boldsymbol{x} \geq \boldsymbol{0}. \tag{2.19}$$

It has been shown in [84] that NNLAD is sparsity promoting for certain conditions on the sensing matrix $\boldsymbol{A}$, and that its minimizer $\boldsymbol{x}^*$ obeys bounds of the form $||\boldsymbol{x} - \boldsymbol{x}^*||_1 \leq C||\boldsymbol{\eta}||_1$, where C is a constant independent of $\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{\eta}, \boldsymbol{y}$. A salient feature of NNLAD/NNLS is that they do not require any parameter tuning. This property makes them useful for matrices of smaller size where cross-validation may be unreliable.

2.3.5 Preservation of conditions sufficient for CS recovery

Consider the following class of reductions of a $0/1$ binary matrix $\boldsymbol{A}$ of size $m \times n$, which we term "consistent reduction" or `C-reduction`:

1. Remove arbitrary columns.
2. Remove rows which have only 0 entries in the remaining columns. No other rows are removed.

Let the indices of the removed columns be denoted by a set $\boldsymbol{\mathcal{X}}$, those of the removed rows be denoted by a set $\boldsymbol{\mathcal{Y}}$, those of the remaining columns be denoted by the complement set $\bar{\boldsymbol{\mathcal{X}}} = \{1 \dots n\} - \boldsymbol{\mathcal{X}}$, and those of the remaining rows be denoted by the complement set $\bar{\boldsymbol{\mathcal{Y}}} = \{1 \dots n\} - \boldsymbol{\mathcal{Y}}$. The reduced matrix is denoted by $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}}, \bar{\boldsymbol{\mathcal{Y}}}}$.

First, we note that the reduction of a matrix via COMP is a `C-reduction`, since the removed rows – which corresponded to negative tests – could not have had a 1 entry in the remaining columns. Otherwise those columns would have been removed since it would mean that the corresponding samples took part in a negative test, leading to a contradiction. Moreover,

any such row which had a 1 entry in only those columns which were removed by COMP must have corresponded to a negative test (since all samples in that pool are surely negative), and it would have been removed by COMP. Hence COMP-reduction is an instance of `C-reduction`.

We state the following lemma which is important for proving preservation of RIP, RNSP, etc:

Lemma 2.4. *Let $\bar{\boldsymbol{x}}$ denote a $|\bar{\boldsymbol{\mathcal{X}}}|$ dimensional vector. Consider a n-dimensional vector $\boldsymbol{x}$ such that $\boldsymbol{x}_{\bar{\boldsymbol{\mathcal{X}}}} = \bar{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$. Then, $\|\boldsymbol{A}\boldsymbol{x}\|_p = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_p$ for any $p \in \mathbb{Z}_+$.*

Essentially, the product of the reduced matrix with any vector $\bar{\boldsymbol{x}}$ in the reduced space has the same ℓ_p-norm as the product of the original matrix with a vector $\boldsymbol{x}$ which contains the same entries as $\bar{\boldsymbol{x}}$ in the indices in $\bar{\boldsymbol{\mathcal{X}}}$ and 0 entries in the remaining indices. Next, the following theorem states that RIP-1 is preserved by `C-reduction`, with the same RIC for the reduced matrix:

Theorem 2.5. *Let d be a constant. If the scaled matrix $\frac{1}{d}\boldsymbol{A}$ satisfies RIP-1 of order k with k-order RIC $\delta_k \in (0,1)$, and $k \leq |\bar{\boldsymbol{\mathcal{X}}}|$, then the scaled `C-reduced` matrix $\frac{1}{d}\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ also satisfies RIP-1 of order k with RIC δ_k. That is, if*

$$\|\boldsymbol{x}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}\boldsymbol{x}\|_1 \leq (1+\delta_k)\|\boldsymbol{x}\|_1,$$

for all n-dimensional k-sparse vectors $\boldsymbol{x}$ then,

$$\|\bar{\boldsymbol{x}}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_1 \leq (1+\delta_k)\|\bar{\boldsymbol{x}}\|_1$$

for all $|\bar{\boldsymbol{\mathcal{X}}}|$-dimensional k-sparse vectors $\bar{\boldsymbol{x}}$.

Similar theorems for the preservation of RIP-2 (Theorem 2.6), ℓ_2-RNSP (Theorem 2.7) and RNSP (Theorem 2.8) by `C-reduction` are stated in Appendix 2.D to avoid repetition. The proofs of Lemma 2.4 and the above mentioned theorems are also provided in Appendix 2.D. Since COMP-reduction is an instance of `C-reduction`, the above theorems hold for COMP-reduction, due to which COMP may be used in the first stage of Tapestry while preserving CS recovery guarantees for the second stage of Tapestry.

In case of false negative tests arising from pooling errors, reduction via COMP may erroneously eliminate some positive samples, leading to error in the final output by Tapestry. To handle such cases, one may use a noise-resilient version of COMP instead, called Noisy Combinatorial Orthogonal Matching Pursuit or NCOMP [29]. In NCOMP, a sample is declared negative only if at least a fraction ϕ of the tests it take part in are negative. The work in [29] gives high-probability guarantees for recovery with NCOMP. Intuitively, one may see that if less than a fraction ϕ of the tests for a positive sample are falsely negative, then NCOMP will not declare it as negative, and thus can recover from such erroneous tests.

We define NCOMP-reduction as follows: (1) remove all columns corresponding to samples which have been declared negative by NCOMP, and (2) remove all rows which have only 0 entries in the remaining columns. We see that such a reduction is also a `C-reduction` – hence the theorems regarding preservation of conditions sufficient for CS recovery are applicable to NCOMP-reduction as well. Thus NCOMP may be used in the first stage of the Tapestry method in order to recover from pooling errors leading to false negative tests, with the guarantees as given in [29]. The samples which are not eliminated by NCOMP due to false positive tests may be handled in the CS stage of Tapestry.

2.3.6 Generalized Binary Search Techniques

There exist *adaptive group testing* techniques which can determine k infected samples in $O(k \log n)$ tests via repeated binary search. These techniques are impractical in our setting due to their sequential nature and large pool sizes. We provide details of these techniques in Appendix 2.A. We also compare with a two-stage approach called Dorfman Testing [28] in Sec. 2.4.1.7.

2.3.7 Sensing Matrix Design

2.3.7.1 Physical Requirements of the Sensing Matrix

The sensing matrix $\boldsymbol{A}$ must obey some properties specific to this application such as being non-negative. For ease and speed of pipetting, it is desirable that the entries of $\boldsymbol{A}$ be (1) binary (where $A_{ji} = 0$ indicates that sample i did not contribute to pool j, and $A_{ji} = 1$ indicates that a fixed volume of sample i was pipetted into pool j), and (2) sparse. Sparsity ensures that not too many samples contribute to a pool, and that a single sample does not contribute to too many pools. The former is important because typically the volume of sample that is added in a PCR reaction is fixed. Increasing pool size means each sample contributes a smaller fraction of that volume. This leads to dilution which manifests as a shift of the C_t value towards larger numbers. If care is not taken in this regard, this can affect the power of PCR to discriminate between positive and negative samples. The latter is important because contribution of one sample to a large number of pools could lead to depletion of sample.

2.3.7.2 RIP-1 of Expander Graph Adjacency Matrices

The Restricted Isometry Property (RIP-2) of sensing matrices is a sufficient condition for good CS recovery as described in Sec. 2.3.3. However the matrices which obey the aforementioned physical constraints are not guaranteed to obey RIP-2. Instead, we consider sensing matrices which are adjacency matrices of **expander graphs**. A left-regular bipartite graph $G((\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}), \mathcal{E} \subseteq \mathcal{V}_\mathcal{I} \times \mathcal{V}_\mathcal{O})$ with degree of each vertex in $\mathcal{V}_\mathcal{I}$ being d, is said to be a (k, ϵ)-**unbalanced expander graph** for some integer $k > 0$ and some real-valued $\epsilon \in (0, 1)$, if for every subset $\mathcal{S} \subseteq \mathcal{V}_\mathcal{I}$ with $|\mathcal{S}| \leq k$, we have $|N(\mathcal{S})| \geq (1 - \epsilon)d|\mathcal{S}|$. Here $N(\mathcal{S})$ denotes the union set of neighbors of all nodes in $\mathcal{S}$. Intuitively a bipartite graph is an expander if every 'not too large' subset has a 'large' boundary. It can be proved that a randomly generated left-regular bipartite graph with $|\mathcal{V}_\mathcal{O}| \geq O(k \log n)$, $n = |\mathcal{V}_\mathcal{I}|$ is an expander, with high probability [85, 86]. Moreover, it has been shown in [78, Thm. 1] that the scaled adjacency matrix $\boldsymbol{A}/d$ of a (k, ϵ)-unbalanced expander graph obeys RIP-1 (Definition 2.1) of order k. Here columns of $\boldsymbol{A}$ correspond to vertices in $\mathcal{V}_\mathcal{I}$, and rows correspond to vertices in $\mathcal{V}_\mathcal{O}$. That is, for any k-sparse vector $\boldsymbol{x}$, the following relationship holds: $\|\boldsymbol{x}\|_1 \leq \|\boldsymbol{A}\boldsymbol{x}\|_1/d \leq (1 + C\epsilon)\|\boldsymbol{x}\|_1$ for some abso-

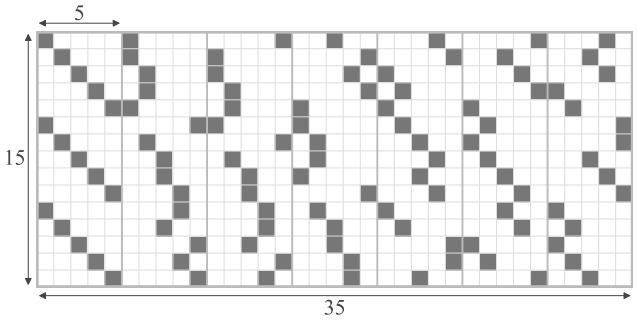


Figure 2.1: A full Kirkman matrix with $m = 15$ rows and $n = \binom{m}{2}/3 = 35$ columns. Each cell denotes an entry of the matrix, with white cells denoting the location of a 0 entry and the greyed out cells indicating the location of a 1 entry. Each column has exactly 3 entries with value 1. Each row has 7 entries with value 1. There are $(m-1)/2 = 7$ groups of columns, each consisting of $m/3 = 5$ columns. Each row in a column group has exactly one 1 entry. Matrices of size 15×20, 15×25, 15×30 or 15×35 may be served by choosing the first 4, 5, 6, or 7 column groups, while keeping the number of 1 entries in each row equal.

lute constant $C > 1$. This property again implies that the null-space of $\boldsymbol{A}$ cannot contain vectors that are 'too sparse' (apart from the zero-vector). This summarizes the motivation behind the use of expanders in compressive recovery of sparse vectors, and also in group testing [78].

2.3.7.3 Matrices derived from Kirkman Triple Systems

Although randomly generated left-regular bipartite graphs are expanders, we would need to verify whether a *particular* such graph is a good expander, which may take prohibitively long in practice [85]. In the application at hand, this can prove to be a critical limitation since matrices of various sizes may have to be served, depending on the number of samples arriving in that batch at the testing centre, and the number of tests available to be performed. Hence, we have chosen to employ deterministic procedures to design such matrices, based on objects from combinatorial design theory known as **Kirkman triples** (see [32, 68]).

We first recall Kirkman Triple Systems (an example of which is illustrated in Fig. 2.1) which are Steiner Triple Systems with an extra property. Steiner Triple Systems consist of $n = \binom{m}{2}/3$ column vectors with m elements each, with each entry being either 0 or 1 such that each column has exactly three 1s, every pair of rows has dot product equal to 1 and every pair of columns has dot product at most 1 [13]. This means that each column of a Steiner Triple

System corresponds to a triplet of rows (i.e. contains exactly three 1s), and every pair of rows occurs together in exactly one such triplet (i.e. for every pair of rows indexed by i, j, there exists exactly one column index k for which $A_{ik} = A_{jk} = 1$). If the columns of a Steiner Triple System can be arranged such that the sum of columns from i to $i + m/3 - 1$ equals $\mathbf{1} \in \mathbb{R}^m$ for every $i \equiv 1$ modulo $m/3$ then the Steiner Triple System is said to be **resolvable**, and is known as a **Kirkman Triple System** [68]. That is, the set of columns of a Kirkman Triple System can be partitioned into $(m-1)/2$ disjoint groups, each consisting of $m/3$ columns, such that each row has exactly one 1 entry in a given such group of columns. Because of this property, we may choose any l such groups of columns of a Kirkman Triple System to form a $m \times n$ matrix, $n > m$, with $n = lm/3$, and $3 < l \leq (m-1)/2$, while keeping the number of 1 entries in each row the same. From here on, we refer to such matrices as **Kirkman matrices**. If $l = (m-1)/2$, then we refer to it as a **full** Kirkman matrix, else it is referred to as a **partial** Kirkman matrix. Note that in a partial Kirkman matrix, the dot product of any two rows may be at most 1, whereas in a full Kirkman matrix, it must be equal to 1.

Notice that $m = 6t + 3$ for some $t \in Z_{\geq 0}$ for a Kirkman Triple System to exist, since $m - 1$ must be divisible by 2, and m must be divisible by 3. This, and the existence of Kirkman Triple Systems for all $t \in Z_{\geq 0}$ have been proven in [32]. Explicit constructions of Kirkman Triple Systems for $m \leq 99$ exist [68]. Generalizations of Kirkman Triple Systems under the name of the Social Golfer Problem is an active area of research (see [87, 88]). The Social Golfer Problem asks if it is possible for $g \times p$ golfers to play in g groups of p players each for w weeks, such that no two golfers play in the same group more than once [89, Sec. 1.1]. Kirkman Triple Systems with m rows and $\binom{m}{2}/3$ columns are a solution to the Social Golfer Problem for the case when $p = 3$, $g = m/3$ and $w = (m-1)/2$. Full or partial Kirkman matrices may be constructed via greedy search techniques used for solving the Social Golfer Problem (such as in [90]). Previously, Kirkman matrices have been proposed for use as Low-Density Parity Check codes in [33], due to high girth[2] of Kirkman matrix bipartite graphs and the ability to serve only part of the matrix while keeping the row weights[3] equal. Matrices derived from Steiner Triple Systems have previously been used for pooled testing for transcription regulatory network mapping in [91]. Further, matrices derived from Steiner Systems [92], a generalization of Steiner Triple Systems, have been proposed for optimizing 2-stage binary group testing in

[2]The girth of a graph is equal to the length of the shortest cycle in it.

[3]defined as the number of 1 entries in a row

[93].

2.3.7.4 RIP-1 and Expansion Properties of Kirkman Matrices

We show that Kirkman matrix bipartite graphs are (k, ϵ)-unbalanced expanders, with $\epsilon = (k-1)/2d$, where d is the left-degree of the graph and is 3 for Kirkman matrices. Given a set S of column vertices such that $|S| \leq k$, we note that the size of the union set of neighbours of S, $|N(S)|$, is at least $|S|d - pr$, where $p = \binom{|S|}{2}$ is the number of (unordered) pairs of columns in S, and r is the maximum number of row vertices in common between any two column vertices. For a Kirkman matrix, since any two columns have dot product at most 1, hence $r = 1$. Therefore, $|N(S)| \geq d|S|(1 - (|S| - 1)/2d)$. Since $|S| \leq k$, therefore $|N(S)| \geq d|S|(1 - (k-1)/2d)$. This implies that Kirkman matrix biparite graphs are (k, ϵ)-unbalanced expanders, with $\epsilon = (k-1)/2d$. If we put in the requirement that $d = 3$ for Kirkman matrices and $\epsilon < 1$, we find that $k < 7$. Hence it follows from [78, Thm. 1] that the scaled Kirkman matrix has RIP-1 of order k for $k < 7$ and $\epsilon = (k-1)/6$. This suggests exact recovery for upto 3 infected samples using CS. However, in practice, we observe that using our method we are able to recover much higher number of positives, at the cost of an acceptable number of false positives and rare false negatives (Sec. 2.4).

2.3.7.5 Optimality of Girth 6 Matrices

A Steiner Triple System bipartite graph does not have a cycle of length 4. If it did, then there would exist two rows a and b, and two columns u and v of the Steiner Triple System matrix $\boldsymbol{A}$ such that $A_{au} = A_{bu} = 1$ and $A_{av} = A_{bv} = 1$. This would violate the property that dot product of any two rows of the Steiner Triple System must be equal to 1. Furthermore, [33, Lemma 1] show that Steiner Triple System bipartite graphs have girth equal to 6. Since Kirkman Triple Systems are resolvable Steiner Triple Systems (see definitions earlier in this section), their bipartite graphs also have girth equal to 6. For a bipartite graph constructed from a partial Kirkman matrix, the girth is at least 6, since dropping some column vertices will not introduce new cycles in the graph. Furthermore, it is shown in [10, Thm. 10] that adjacency matrices of left-regular graphs with girth at least 6 satisfy RNSP (Definition 2.2) of order k (for suitable k). Consequently, they may be used for CS decoding [10, Thm. 5]. They also give

lower bounds on the number of rows m of left-regular bipartite graph matrices whose column weight[4] is more than 2, for them to have high girth and consequently satisfy RNSP of order k, given k and n [10, Eqn. 32, 33]. Given k and n, these lower bounds are minimized for graphs of girth 6 and 8, and the bounds are, respectively, $m \geq k\sqrt{n}$ and $m \geq k^{3/2}\sqrt{n}$ ([10, Eqn. 37]). However, with the additional requirement that $m < n$ for CS, it is found that girth 6 matrices can recover $k < \sqrt{n}$ defects, while girth 8 matrices can only recover $k < \sqrt[3]{n}$ defects. Hence, matrices whose bipartite graphs have girth equal to 6 are optimal in this sense. Full Kirkman matrix bipartite graphs are left-regular and have girth 6, as argued earlier, and hence they satisfy RNSP, may be used for compressive sensing, and are optimal in the sense of being able to handle most number of defects while minimizing the number of measurements. We note that since we employ Kirkman triples, each column has only three 1s. The theoretical guarantees for such matrices hold for signals with ℓ_0 norm less than or equal to 2. However, we have obtained acceptable false positive and false negative rates in practice for much larger sparsity levels, as will be seen in Sec. 2.4.

2.3.7.6 Disjunctness Property of Kirkman Matrices

In order for a matrix to be suitable for our method, it should not only be good for CS decoding algorithms, but also for COMP. Kirkman matrices are 2-disjunct, and can recover up to 2 defects exactly using COMP. In a k-disjunct matrix, there does not exist any column such that its support is a subset of the union of the support of k other columns [71]. Matrices which are k-disjunct have exact support recovery guarantee for k-sparse vectors, using COMP (see [71]). Disjunctness follows from the following properties of Kirkman matrices – that two columns in a Kirkman matrix have at most one row in common with an entry of 1, and that each column has exactly three 1 entries. Consider R_a, R_b, and R_c, the sets of rows for which the three columns a, b and c respectively have a 1 entry. Note that $|R_a| = |R_b| = |R_c| = 3$, and $|R_p \cap R_q| \leq 1$ for $p, q \in \{a, b, c\}, p \neq q$. If $R_c \subseteq R_a \cup R_b$, then either $|R_c \cap R_a| > 1$ or $|R_c \cap R_b| > 1$, which presents a contradiction.

Empirically we find that even for $k > 2$, COMP reports only a small fraction of the total number of samples as positives when using Kirkman matrices (Table 2.1). In Appendix 2.K (Proposition 6), we prove that *if a fraction $f \in (0, 1)$ of the tests come out to be positive, then*

[4] defined as the number of 1 entries in a column

COMP *reports strictly less than fraction* f^2 *of the samples as positive for a full Kirkman matrix*. This provides intuition behind why Kirkman matrices may be well-suited for our combined COMP + CS method, since most samples are already eliminated by COMP. On the other hand, CS decoding (without the earlier COMP step) on the full Kirkman matrix does not perform as well, as shown in Appendix 2.M.

2.3.7.7 Advantages of using Kirkman Matrices

As we have seen in earlier sections, Kirkman matrices are suitable for use in compressed sensing due to their expansion, RIP-1 and high girth properties, as well as for binary group testing due to disjunctness. Furthermore, the dot product between two columns of a Kirkman matrix being at most 1 ensures that no two samples participate in more than one test together. This has favourable consequences in terms of placing an upper bound on the **mutual coherence** of the matrix, defined as:

$$\mu(\boldsymbol{A}) := \max_{i \neq j} \frac{|\boldsymbol{A_i^t A_j}|}{\|\boldsymbol{A_i}\|_2 \|\boldsymbol{A_j}\|_2}, \tag{2.20}$$

where $\boldsymbol{A_i}$ refers to the i^{th} column of $\boldsymbol{A}$. Matrices with lower $\mu(\boldsymbol{A})$ values have lower values of worst case upper bounds on the reconstruction error [94]. These bounds are looser than those based on the RIC that we saw in previous sections. However, unlike the RIC, the mutual coherence is efficiently computable.

A practical benefit of Kirkman triples that is not shared by Steiner triples is that the former can be served for number of samples far less than $n = \binom{m}{2}/3$ while keeping pools balanced (i.e. ensuring that each pool is created from the same number of samples). In fact, we can choose n to be any integer multiple of $m/3$, and ensure that every pool gets the same number of samples, as discussed in section 2.3.7.3. Notice that the expansion, RIP-1, high girth and disjunctness properties hold for full as well as partial Kirkman matrices, as proven in previous sections. This allows us to characterize the properties of the full Kirkman matrix, and use that analysis to predict how it will behave in the clinical situation where the pooling matrix to be served may require very specific values of m, n depending on the prevalence rate.

Column weight: Kirkman matrices have column weight equal to 3 - that is, each sample

goes to 3 pools. It is possible to construct matrices with higher number of pools per sample (such as those derived from the Social Golfer Problem [87], which will retain several benefits of the Kirman matrices: (1) They would have the ability to serve only part of the matrix; (2) They would retain the the expander and RIP-1 properties, following a proof similar to the one in Sec. 2.3.7.4; (3) They would not have any 4-cycles in the corresponding bipartite graph, following a similar argument as in Sec. 2.3.7.5; and (4) They would possess the disjunctness property following a proof similar to the one in Sec. 2.3.7.6). Nevertheless, the time and effort needed for pooling increases with more pools per sample. Further, higher pools per sample will come at the cost of a larger number of tests (if pool size is kept constant), or larger pool size (if number of tests is kept constant). Higher number of tests is undesirable for obvious reasons, while larger pool size may lead to dilution of the sample within a pool, leading to individual RT-PCR tests failing.

2.3.7.8 Optimal Binary Sensing Matrices with Random Construction

While Kirkman matrices which satisfy RNSP of order k must have at least $k\sqrt{n}$ measurements, we can get much better bounds in theory if we use random constructions. From [12, Prop. 10] we see that with high probability, $0/1$ Bernoulli(p) matrices need only $O(k \log n)$ measurements in order to satisfy ℓ_2-RNSP (Definition 2.3) of order k, with $p \in (0, 1)$ being the probability with which each entry of the matrix is independently 1.

In Appendix 2.D, we prove that ℓ_2-RNSP is preserved by COMP. That is, the reduced matrix $\boldsymbol{A}_{\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}}$ obeys ℓ_2-RNSP of order k with the same parameters as the original matrix $\boldsymbol{A}$. Hence our method only needs $O(k \log n)$ measurements for robust recovery of k-sparse vectors with such random matrix constructions. Bernoulli(p) matrices are also good for COMP – [29, Thm. 4] shows that Bernoulli(p) matrices with $p = 1/k$ need only $O(k \log n)$ measurements for exact support recovery of k-sparse vectors with COMP with vanishingly small probability of error.

In practice, we observe that Kirkman matrices perform better than Bernoulli(p) matrices using our method in the regime of our problem size. This gap between theory and practice may be arising due to the following reasons: (1) The $k\sqrt{n}$ lower bound for Kirkman triples is for a sufficient but not necessary condition for sparse recovery; (2) The $O(k \log n)$ may be ignoring a very large constant factor which affects the performance of moderately-sized problems such

as the ones reported in this work; and (3) The theoretical bounds are for exact recovery with vanishingly small error, whereas we allow some false positives and rare false negatives in our experiments. Similar comparisons between binary and Gaussian random matrices have been recently put forth in [10]. Moreover, the average column weight of Bernoulli(p) matrices is pm, where m is the number of measurements. This is typically much higher than column weight 3 of Kirkman matrices and hence undesirable (see Sec. 2.3.7.7). In Appendix 2.E, we compare the performance of Kirkman matrices with Bernoulli(0.1) and Bernoulli(0.5) matrices.

2.3.7.9 Mutual Coherence optimized Sensing Matrices

As mentioned earlier, the mutual coherence from Eqn. 2.20 is efficient to compute and optimize over. Hence, there is a large body of literature on designing CS matrices by minimizing $\mu(\boldsymbol{A})$ w.r.t. $\boldsymbol{A}$, for example [95]. We followed such a procedure for designing sensing matrices for some of our experimental results in Sec. 2.4.2. For this, we follow simulated annealing to update the entries of $\boldsymbol{A}$, starting with an initial condition where $\boldsymbol{A}$ is a random binary matrix. For synthetic experiments, we compared such matrices with Bernoulli(p) random matrices, adjacency matrices of biregular random sparse graphs (i.e. matrices in which each column has the same weight, and each row has the same weight - which may be different than the column weight), and Kirkman matrices. We found that matrices of Kirkman triples perform very well empirically in the regime of sizes we are interested in, besides facilitating easy pipetting, and hence the results are reported using only Kirkman matrices.

2.4 Experimental Results

In this section, we show a suite of experimental results on synthetic data as well as on real data.

2.4.1 Results on Synthetic Data

2.4.1.1 Choice of Sensing Matrix

Recall from section 2.2 that a typical RT-PCR setup can test 96 samples in parallel. Three of these tests are used as control by the RT-PCR technician in order to have confidence that the RT-PCR process has worked. Hence, in order to optimize the available test bandwidth of the RT-PCR setup, the number of tests we perform in parallel should be ≤ 93, and as close to 93 as possible. Since in Kirkman matrices, the number of rows must be $6t + 3$ for some $t \in \mathbb{Z}_{\geq 0}$, hence we choose 93. With this choice, the number of samples tested n has to be a multiple of $93/3 = 31$, hence we chose $n = 961$. This matrix is not a full Kirkman matrix – a full matrix with 93 rows will have 1426 columns. However, we keep the number of columns of the matrix under 1000 due to challenges in pooling large number of samples. Furthermore, $n = 961$, $m = 93$ satisfies more than 10x factor improvement in testing while detecting 1% infected samples with reasonable sensitivity and specificity and is in a regime of interest for widespread screening or repeated testing.

We also present results with a 45×105 partial Kirkman matrix in Appendix 2.L. This matrix gives 2.3x improvement in testing while detecting 9.5% infected samples with reasonable sensitivity and specificity. Further, two such batches of 105 tests in 45 pools may be run in parallel in a single RT-PCR setup.

2.4.1.2 Signal/Measurement Generation

For the case of synthetic data, we generated k-sparse signal vectors $\boldsymbol{x}$ of dimension $n = 961$, for each k in $\{5, 8, 10, 12, 15, 17, 20\}$. We choose a wide range of k in order to demonstrate that not only do our algorithms have high sensitivity and specificity for large values of k, they also keep performing reasonably, well beyond the typical operating regime. The support of each signal vector $\boldsymbol{x}$ – given k – was chosen by sampling a k-sparse binary vector uniformly at random from the set of all k-sparse binary vectors. The magnitudes of the non-zero elements of $\boldsymbol{x}$ were picked uniformly at random from the range $[1, 32768]$. This high dynamic range in the value of $\boldsymbol{x}$ was chosen to reflect a variance in the typical threshold cycle values (C_t) of real PCR

experiments, which can be between 16 and 32. From Eqn. 2.6, we can infer that viral loads vary roughly as 2^{-C_t} (setting $q = 1$), up to constant multiplicative terms. In all cases, $m = 93$ noisy measurements in $\boldsymbol{y}$ were simulated following the noise model in Eqn. 2.3 with $\sigma = 0.1$ and $q = 0.95$. A 93×961 Kirkman sensing matrix was used for generating the measurements. The Poisson nature of the elements of $\boldsymbol{x}$ in Eqn. 2.3 was ignored. This approximation was based on the principle that if $X \sim \text{Poisson}(\lambda)$, then Std. Dev.$(X)/E(X) = \sqrt{\lambda}/\lambda = 1/\sqrt{\lambda}$ which becomes smaller and smaller as λ increases. The recovery algorithms were tested on $Q = 1000$ randomly generated signals for each value of k.

2.4.1.3 Algorithms tested

The following algorithms were compared:

1. COMP (see Table 2.1)
2. COMP followed by NNLASSO (see Table 2.2)
3. COMP followed by SBL (see Table 2.3)
4. COMP followed by NNOMP (see Table 2.4)
5. COMP followed by NNLAD (see Table 2.5)
6. COMP followed by NNLS (see Table 2.16 in the Appendix)

For each algorithm any positives missed during the CS stage but caught by DD were declared as positives, as mentioned in Sec. 2.3.4. For small sample sizes we also tested COMP-BF, i.e. COMP followed by brute-force search for samples in $\boldsymbol{x}$ with non-zero values. Details of this algorithm and experimental results with it are presented in Appendix 2.C.

2.4.1.4 Comparison Criteria

In the following, $\hat{\boldsymbol{x}}$ denotes the estimate of $\boldsymbol{x}$. Most numerical algorithms do not produce vectors that are exactly sparse and have many entries with very tiny magnitude, due to issues such as choice of convergence criterion. Since in this application, support recovery is of paramount

importance to identify which samples in $\boldsymbol{x}$ were infected, we employed the following post-processing step: All entries in $\hat{\boldsymbol{x}}$ whose magnitude fell below a threshold $\tau := 0.2 \times x_{\min}$ were set to zero, yielding a vector $\bar{\boldsymbol{x}}$. Here $x_{\min}$ refers to the least possible value of the viral load, and this can be obtained offline from practical experiments on individual samples. In these synthetic experiments, we simply set $x_{\min} := 1$. We observed that varying the value of τ over a fairly wide range had negligible impact on the results, as can be observed in Tables 2.20, 2.21 and 2.22 in Appendix 2.I. For SBL, we set τ to 0 and also set the negative entries in the estimate to 0. For NNOMP, such thresholding was inherently not needed. The various algorithms were compared with respect to the following criteria:

1. RMSE := $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_2 / \|\boldsymbol{x}\|_2$
2. Number of false positives (FP) := $|\{i : x_i = 0, \hat{x}_i > 0\}|$
3. Number of false negatives (FN) := $|\{i : x_i > 0, \hat{x}_i = 0\}|$
4. Sensitivity (also called Recall or True Positive rate) := #correctly detected positives/#actual positives
5. Specificity (also called True Negative Rate) := #correctly detected negatives/#actual negatives.

2.4.1.5 Main Results

It should be noted that all algorithms were evaluated on 1000 randomly generated sparse signals, given the same sensing matrix. The average value as well as standard deviation of all quality measures (over the 1000 signals) are reported in the Tables 2.1, 2.2, 2.3, 2.4, 2.5 and 2.16. A comparison of Table 2.1 to Tables 2.2, 2.3, 2.4, 2.5 and 2.16 indicates that COMP followed by NNLASSO/SBL/NNOMP/NNLAD/NNLS significantly reduces the false positives at the cost of a rare false negative. The RMSE is also significantly improved, since COMP does not estimate viral loads. At the same time, COMP significantly reduces the size of the problem for the CS stage. For example, for the 93×961 Kirkman matrix, when number of infected samples k is 12, the average size of the matrix after COMP filtering is $\sim 30 \times 37$. From Table 2.1 we see that Definite Defectives classifies many positives as high-confidence positives, for k upto 8. We note that the experimental results reported in these tables are quite encouraging, since these

Table 2.1: Performance of COMP and DD (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**	$\#\mathcal{HCP}$
5	1.000 ± 0.000	0.0 ± 0.0	1.6 ± 1.2	1.0000 ± 0.0000	0.9983 ± 0.0013	4.7
8	1.000 ± 0.000	0.0 ± 0.0	7.9 ± 3.0	1.0000 ± 0.0000	0.9918 ± 0.0031	4.3
10	1.000 ± 0.000	0.0 ± 0.0	15.3 ± 4.5	1.0000 ± 0.0000	0.9839 ± 0.0048	2.5
12	1.000 ± 0.000	0.0 ± 0.0	25.3 ± 6.7	1.0000 ± 0.0000	0.9733 ± 0.0070	1.1
15	1.000 ± 0.000	0.0 ± 0.0	46.1 ± 10.3	1.0000 ± 0.0000	0.9512 ± 0.0109	0.2
17	1.000 ± 0.000	0.0 ± 0.0	62.3 ± 13.7	1.0000 ± 0.0000	0.9340 ± 0.0146	0.1
20	1.000 ± 0.000	0.0 ± 0.0	91.5 ± 18.1	1.0000 ± 0.0000	0.9028 ± 0.0192	0.0

Table 2.2: Performance of Comp followed by NNLASSO (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.047 ± 0.020	0.0 ± 0.1	0.8 ± 0.9	0.9990 ± 0.0141	0.9991 ± 0.0009
8	0.069 ± 0.028	0.1 ± 0.2	4.0 ± 2.1	0.9925 ± 0.0307	0.9958 ± 0.0022
10	0.100 ± 0.049	0.2 ± 0.5	7.9 ± 3.4	0.9780 ± 0.0454	0.9917 ± 0.0035
12	0.149 ± 0.092	0.6 ± 0.7	12.9 ± 5.0	0.9538 ± 0.0591	0.9864 ± 0.0052
15	0.295 ± 0.166	1.0 ± 1.1	28.5 ± 16.5	0.9316 ± 0.0722	0.9699 ± 0.0175
17	0.404 ± 0.184	1.1 ± 1.4	46.3 ± 23.5	0.9355 ± 0.0817	0.9509 ± 0.0249
20	0.563 ± 0.172	1.1 ± 1.8	78.4 ± 26.7	0.9452 ± 0.0923	0.9167 ± 0.0284

Table 2.3: Performance of COMP followed by SBL (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.043 ± 0.017	0.0 ± 0.0	0.9 ± 0.9	0.9998 ± 0.0063	0.9991 ± 0.0010
8	0.058 ± 0.021	0.0 ± 0.2	4.3 ± 2.1	0.9958 ± 0.0227	0.9955 ± 0.0023
10	0.071 ± 0.025	0.1 ± 0.2	8.2 ± 3.1	0.9937 ± 0.0247	0.9913 ± 0.0033
12	0.094 ± 0.035	0.1 ± 0.4	13.6 ± 4.4	0.9886 ± 0.0310	0.9856 ± 0.0046
15	0.123 ± 0.108	0.3 ± 0.6	25.1 ± 6.9	0.9804 ± 0.0396	0.9735 ± 0.0073
17	0.165 ± 0.179	0.5 ± 0.8	35.1 ± 9.9	0.9713 ± 0.0491	0.9628 ± 0.0105
20	0.318 ± 0.305	1.3 ± 1.6	54.5 ± 13.2	0.9349 ± 0.0803	0.9420 ± 0.0140

Table 2.4: Performance of COMP followed by NNOMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.043 ± 0.019	0.0 ± 0.1	0.3 ± 0.6	0.9982 ± 0.0209	0.9997 ± 0.0006
8	0.060 ± 0.025	0.1 ± 0.4	1.8 ± 2.0	0.9831 ± 0.0472	0.9981 ± 0.0021
10	0.077 ± 0.035	0.3 ± 0.5	3.7 ± 3.2	0.9739 ± 0.0541	0.9961 ± 0.0034
12	0.115 ± 0.067	0.5 ± 0.7	7.8 ± 4.9	0.9565 ± 0.0560	0.9918 ± 0.0051
15	0.242 ± 0.190	1.5 ± 1.4	15.6 ± 6.0	0.9013 ± 0.0951	0.9835 ± 0.0064
17	0.361 ± 0.243	2.8 ± 2.2	20.8 ± 5.6	0.8329 ± 0.1268	0.9780 ± 0.0059
20	0.589 ± 0.282	6.1 ± 3.0	27.0 ± 5.2	0.6941 ± 0.1520	0.9713 ± 0.0055

Table 2.5: Performance of COMP followed by NNLAD (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	RMSE	#FN	#FP	Sens.	Spec.
5	0.050 ± 0.021	0.0 ± 0.0	1.0 ± 1.0	0.9996 ± 0.0089	0.9990 ± 0.0010
8	0.077 ± 0.034	0.0 ± 0.2	4.9 ± 2.3	0.9939 ± 0.0270	0.9949 ± 0.0024
10	0.107 ± 0.050	0.2 ± 0.4	9.3 ± 3.0	0.9809 ± 0.0441	0.9903 ± 0.0032
12	0.167 ± 0.095	0.5 ± 0.7	14.4 ± 5.1	0.9574 ± 0.0586	0.9848 ± 0.0054
15	0.296 ± 0.160	0.9 ± 1.0	29.7 ± 16.7	0.9393 ± 0.0687	0.9686 ± 0.0176
17	0.401 ± 0.171	0.8 ± 1.2	50.2 ± 23.9	0.9549 ± 0.0717	0.9468 ± 0.0253
20	0.530 ± 0.166	0.2 ± 0.9	87.9 ± 25.1	0.9884 ± 0.0427	0.9066 ± 0.0267

experiments are challenging due to small m and fairly large k, n, albeit with testing on synthetic data. We noticed that running the CS algorithms without the COMP step did not perform as well, results for which are presented in Appendix 2.M. We observed that the advantages of our combined group testing and compressed sensing approach hold regardless of the sensing matrix size. For comparison, results of running our algorithms using a 45×105 Kirkman matrix instead of the 93×961 Kirkman matrix are presented in Appendix 2.L.

2.4.1.6 Parameter Selection

As mention earlier, the regularization parameters in various estimators such as COMP-NNLASSO, COMP-NNLAD, COMP-NNOMP, etc. are estimated via cross-validation. For these estimators, we therefore do not require knowledge of the σ parameter in the noise model from Eqn. 2.3. The q parameter in the noise model is set to 0.95 in all our experiments. It is a reasonable choice as the molecule count is known to roughly double in each cycle of RT-PCR [70]. Moreover, variation of q in the range from 0.7 to 1 showed negligible variation in the results of our wet-lab experiments as can be seen in Appendix 2.H and Tables 2.18 and 2.19. Also note that we only report viral loads relative to $y_{\min}$ (see Eqn. 2.8) - we do not attempt to estimate $y_{\min}$. These relative viral loads are interpretable by the RT-PCR technicians since they know $t_{\min}$, the minimum C_t (threshold cycle) value observed in that experiment. Note as well that since $y_{\min}$ is the viral load of the pool with the minimum C_t value – it corresponds to the

pool with the maximum viral load in that experiment.

2.4.1.7 Comparison with Dorfman Pooling

Table 2.6: Expected number of tests needed by optimal Dorfman Testing for number of samples (n) 105 and 961 for various k. Note that our proposed methods based on CS require much fewer tests (45 and 93) typically, and do not require two rounds of testing.

$N = 105$			$N = 961$		
k	**# Tests**	**Pool Size**	k	**# Tests**	**Pool Size**
5	43.7	5	5	136.5	14
8	55.3	4	8	172.2	11
10	61.3	4	10	192.2	11
12	67.0	4	12	209.6	9
15	73.9	3	15	233.7	9
17	78.2	3	17	248.7	8
20	84.3	3	20	269.4	7

We also performed a comparison of our algorithms with the popular two-stage Dorfman pooling method (an adaptive method), with regard to the number of tests required. In the first stage of the Dorfman pooling technique, the n samples are divided into n/g pools, each of size g. Each of these n/g pools are tested, and a negative result leads to all members of that pool being considered negative (i.e. non-infected). However, the pools that are tested positive are passed onto a second stage, where all members of those pools are individually tested. The optimal pool size g^* will minimize the expected number of tests taken by this process (given that the membership in each pool is decided randomly). A formula for the expected number of tests taken by Dorfman testing is derived in [28]. The derivation in [28] assumes the following: (1) Any given sample may be positive with probability p, independently of the other samples; (2) The number of samples n is divisible by the pool size g. We modify the formula from [28] for the case that n is not divisible by g (Appendix 2.J), and find g^* by choosing the value of g which minimizes this number. We set $p = k/n$, so that out of n samples, the number of infected samples is k in expectation. Table 2.6 shows the expected number of tests computed from the formula in Appendix 2.J, assuming that the expected number of infected samples k (and thus the

Table 2.7: Estimated sparsity k_{est} versus true sparsity k (on synthetic data) for 93×961 Kirkman matrix. Mean and standard deviation of estimated sparsity is computed over 1000 signals for each k.

k	k_{est}
5	5.01 ± 0.33
10	10.07 ± 0.69
15	15.13 ± 1.17
20	20.26 ± 1.63
25	25.54 ± 2.03
30	30.53 ± 2.61

Table 2.8: Comparison of mean number of false negative and false positives for COMP, COMP-SBL and COMP-SBL with graceful failure mode for high values of k for the 93×961 Kirkman matrix. The algorithm goes into graceful failure mode when estimated sparsity is greater than or equal to 20

	COMP		**COMP-SBL**		**COMP-SBL-graceful**	
k	**#FN**	**#FP**	**#FN**	**#FP**	**#FN**	**#FP**
15	0	45.3	0.3	24.9	0.3	24.8
20	0	92.7	1.3	55.4	0.4	80.2
25	0	151.2	4	97.5	0	151.2
30	0	212.1	6.9	140.6	0	212.1

optimal pool size g^*) is known in advance. We also empirically verified the expected number of tests by performing 1000 Monte Carlo simulations of Dorfman testing with the optimal pool size g^* for each case, and did not observe much deviation from the numbers reported in Table 2.6. Comparisons of Tables 2.1, 2.2, 2.3, 2.4 with the two-stage Dorfman pooling method in 2.6 show that our methods require much fewer tests, albeit with a slight increase in number of false negatives. Moreover, all our methods are single-stage methods and therefore require less time for testing, unlike the Dorfman method which requires two stages of testing.

2.4.1.8 Estimation of number of infected samples

The number of CS measurements for successful recovery depends on the number of non-zero elements (ℓ_0 norm) of the underlying signal. For example, this varies as $O(k \log n)$ for random-

ized sensing matrices [3] or as $O(\max(k^2, \sqrt{n})$ for deterministic designs [76]. There is a lower bound of $k\sqrt{n}$ measurements for certain types of expander matrices to satisfy a sufficient (but not necessary) condition for recovery [10]. However, in practice k is always unknown, which leads to the question as to how many measurements are needed as a minimum for a *particular* problem instance. To address this, we adopt the technique from [96] to estimate k on the fly from the compressive measurements. This technique does not require signal recovery for estimating k. The relative error in the estimate of k is shown to be $O(\sqrt{\log m/m})$ [97], which diminishes as m increases (irrespective of the true k). Table 2.7 shows the accuracy of our sparsity estimate on synthetic data.

The advantage of this estimate of k is that it can drive the COMP-BF algorithm, as well as act as an indicator of whether there exist any false negatives. We can use this knowledge to enable a *graceful failure mode*. In this mode, if our estimate of k is larger than what the CS algorithms can handle, we return only the output of the COMP stage. Hence in such rare cases, it minimizes the number of false negatives, at the cost of many false positives. In these cases a second stage of individual testing must be done on the samples which were declared positive. Table 2.8 shows the effect of using graceful failure mode with COMP followed by SBL for large values of k. In these experiments, output of COMP is returned if the estimated sparsity, k_{est}, is greater than or equal to 20. We see that COMP-SBL with graceful failure mode matches the behaviour of COMP-SBL at sparsity value lower than 20, and that of COMP at sparsity value greater than 20. At sparsity equal to 20, it compromises between the high false positives of COMP, and the high false negatives of COMP-SBL. This is because of the variability in k_{est}, which can occasionally be less than 20 even if k is equal to 20.

2.4.2 Results on Real Data

We acquired real data in the form of test results on pooled samples from two labs: one at the National Center of Biological Sciences (NCBS) in India, and the other at the Wyss Institute at the Harvard Medical School, USA. In both cases, viral RNA was artificially injected into k of the n samples where $k \ll n$. From these n samples, a total of m mixtures were created. For the datasets obtained from NCBS that we experimented with, we had $m = 16$, $n = 40$, $k \in \{1, 2, 3, 4\}$. For the data from the Wyss Institute, we had $m = 24$, $n = 60$, $k = 2$;

$m = 30$, $n = 120$, $k = 2$; and $m = 90$, $n = 1140$, $k = 11$. The results for all these datasets are presented in Table 2.9 and Table 2.10. The 16×40 and 24×60 pooling matrices were obtained by performing a simulated annealing procedure to minimize the mutual coherence (see Sec. 2.3.7.9), starting with a random sparse binary matrix as initial condition. The 30×120 and the 90×1140 pooling matrices were derived from social golfer problems and had column weight 3. We used $q = 0.95$ in all cases to obtain relative viral loads from C_t values, using Eqn. 2.8. While q may be estimated from raw RT-PCR data (Appendix 2.H), we found $q = 0.95$ to be a reasonable choice, and did not observe any variation in the number of reported positives when this parameter was changed between 0.7 to 1. For NNLASSO, NNLS and NNLAD, we use $\tau = 0.2 \times \widetilde{y}_{\max}$ as the threshold below which an estimated relative viral load is set to 0, since value of $x_{\min}$ may not always be available for real experiments. Here $\widetilde{y}_{\max}$ is the relative viral load of the pool with the largest C_t value, and consequently the smallest viral amount. We see that the CS algorithms reduce the false positives, albeit with an introduction of occasional false negatives for higher values of k. We also refer the reader to our work in [16] for a more in-depth description of results on real experimental data.

2.4.3 Discussion

Each algorithm we ran presented a different set of tradeoffs between sensitivity and specificity. While COMP provides us with sensitivity equal to 1, it suffers many false positives, especially for higher k. For other algorithms, in general both the sensitivity and the specificity decrease as k is increased. COMP-NNOMP (Table 2.4) has the highest specificity, but it comes at the cost of sensitivity. COMP-SBL (Table 2.3) has the best sensitivity for most values of k amongst the CS algorithms. COMP-NNLASSO (Table 2.2) has better specificity than COMP-SBL for small values of k, but loses out for $k \geq 15$. COMP-NNLAD and COMP-NNLS (Tables 2.5 and 2.16) start behaving like COMP for higher values of k, effectively bounding the number of false negatives. However, their number of false positives is almost as much as those with COMP.

Ideally, we want both high sensitivity and high specificity while catching a large number of infected samples. Hence, we look at k^*, which is the maximum number of infected samples k for which the sensitivity and specificity of the algorithm are greater than or equal to some threshold values. For the 45×105 Kirkman matrix, we chose the sensitivity threshold as 0.99

Table 2.9: Results of wet lab experiments with each algorithm (continued in Table 2.10)

Dataset	Algorithm	# true pos	# false neg	#false pos
Harvard $24 \times 60, k = 2$	COMP	2	0	1
	COMP-SBL	2	0	1
	COMP-NNOMP	2	0	0
	COMP-NNLASSO	2	0	1
	COMP-NNLAD	2	0	1
	COMP-NNLS	2	0	1
Harvard $30 \times 120, k = 2$	COMP	2	0	1
	COMP-SBL	2	0	1
	COMP-NNOMP	2	0	1
	COMP-NNLASSO	2	0	1
	COMP-NNLAD	2	0	1
	COMP-NNLS	2	0	1
Harvard $90 \times 1140, k = 11$	COMP-SBL	11	0	6
	COMP-NNOMP	11	0	3
	COMP-NNLASSO	11	0	13
NCBS-0 $16 \times 40, k = 0$	COMP	0	0	0
	COMP-SBL	0	0	0
	COMP-NNOMP	0	0	0
	COMP-NNLASSO	0	0	0
	COMP-NNLAD	0	0	0
	COMP-NNLS	0	0	0
NCBS-1 $16 \times 40, k = 1$	COMP	1	0	0
	COMP-SBL	1	0	0
	COMP-NNOMP	1	0	0
	COMP-NNLASSO	1	0	0
	COMP-NNLAD	1	0	0
	COMP-NNLS	1	0	0

Table 2.10: Continuation of Table 2.9

Dataset	Algorithm	# true pos	# false neg	#false pos
NCBS-2 $16 \times 40, k = 2$	COMP	2	0	0
	COMP-SBL	2	0	0
	COMP-NNOMP	2	0	0
	COMP-NNLASSO	2	0	0
	COMP-NNLAD	2	0	0
	COMP-NNLS	2	0	0
NCBS-3 $16 \times 40, k = 3$	COMP	3	0	1
	COMP-SBL	2	1	1
	COMP-NNOMP	2	1	0
	COMP-NNLASSO	2	1	1
	COMP-NNLAD	3	0	1
	COMP-NNLS	2	1	1
	COMP-BF	2	1	1
NCBS-4 $16 \times 40, k = 4$	COMP	4	0	3
	COMP-SBL	3	1	2
	COMP-NNOMP	2	2	2
	COMP-NNLASSO	3	1	2
	COMP-NNLAD	2	2	2
	COMP-NNLS	3	1	2
	COMP-BF	2	2	2

and the specificity threshold as 0.95. For the 93×961 Kirkman matrix, we chose both thresholds to be 0.99, since a specificity threshold of 0.95 gives too many false positives for 961 samples. We observed that COMP-SBL has $k^* = 10$ for both matrices, which is the highest amongst all algorithms tested. Typically we do not know the number of infections, but a prevalence rate of infection. The number of infected samples out of a given set of n samples may be treated as a Binomial random variable with probability of success equal to the prevalence rate. Under this assumption, using COMP-SBL with the 93×961 Kirkman matrix, we observed that the maximum prevalence rate for which sensitivity and specificity are both above 0.99 is 1%. Similarly, using COMP-SBL with the 45×105 Kirkman matrix, we observed that the maximum prevalence rate for which sensitivity is above 0.99 and specificity is above 0.95 is 9.5%. Thus, Tapestry is viable at prevalence rates as high as 9.5%, while reducing testing cost by a factor of 2.3. On the other hand, if the prevalence rate is only 1% or less, it can reduce testing cost by a factor of 10.3.

Comments about sensitivity and specificity: We observe that the sensitivity and specificity of our method on synthetic data is *within the recommendations of the U.S. Food and Drugs Administration (FDA)*, as provided in this document [98]. The document provides recommendations for percent positive agreement (PPA) and percent negative agreement (PNA) of a COVID-19 test with a gold standard test (such as RT-PCR done on individual samples). PPA and PNA are used instead of sensitivity and specificity when ground-truth positives are not known. Since for synthetic data we know the ground truth positives, we compare their PPA and PNA recommendations with the sensitivity and specificity observed by us. We use COMP-SBL for comparison, since we consider it to be our best method.

For 'Testing patients suspected of COVID-19 by their healthcare provider' (point G.4.a, page 7 of [98]), the document considers positive and negative agreement of $\geq 95\%$ as acceptable clinical performance (page 9, row 2 of table in [98]). The sensitivity and specificity of our method on the 93×961 Kirkman matrix is within this range for $k \leq 17$ infected samples (Table 2.3). For the 45×105 matrix, it is within this range for $k \leq 10$ infected samples (Table 2.25).

For 'Screening individuals without symptoms or other reasons to suspect COVID-19 with a previously unauthorized test' (point G.4.c, page 10 of [98]), the document considers positive agreement of $\geq 95\%$ and negative agreement of $\geq 98\%$ as acceptable (along with the lower

bounds of two-sided 95% confidence interval to be $> 76\%$ and $> 95\%$ respectively). Similarly, for 'Adding population screening of individuals without symptoms or other reasons to suspect COVID-19 to an authorized test' (point G.4.d, page 12 of [98]) the document has the same criterion as for point G.4.c. Our sensitivity and specificity are within the ranges specified for the 93×961 Kirkman matrix for $k \leq 12$ (Table 2.3). While we do not report confidence intervals (as suggested for point G.4.c and G.4.d of [98]), the standard deviation of sensitivity and specificity reported by us are fairly low, and we believe the performance of our method is within the recommendations of [98]. Since our numbers are on synthetic data - these numbers may vary upon full clinical validation, especially considering that there may be more sources of error in a real test. Nonetheless, we find these numbers to be encouraging.

Further, we note that while our method incurs an occasional false negative, the viral loads of these false negative values are fairly small. This means that super-spreaders (who are believed to have high viral load [30]) will almost always be caught by our method. We discuss this in more detail in Appendix 2.G, and provide a table of mean and standard deviations of viral loads of false negatives (Table 2.17) for all our methods on synthetic data.

Tapestry can detect certain errors caused by incorrect pipetting, pool contamination, or failed RT-PCR amplification of some pools. This is done by performing a consistency check after the COMP stage. If there is a pool which is positive, but all of the samples involved in it have been declared negative by COMP, this is indicative of error. In case of error, we list all samples categorized by the number of tests that they are positive in. However, the COMP consistency check will not catch all errors. Alternately, the NCOMP [29] algorithm may be used to correct for errors in the first stage of Tapestry since sufficient conditions for CS recovery are preserved by reduction via NCOMP, as mentioned in Sec. 2.3.5. A full exposition on detection and correction of errors is left as future work.

Although Tapestry can work with a variety of sensing matrix designs, we found Kirkman matrices to be most suitable for our purposes. This is due to lower column weight and smaller pool sizes presented by Kirkman matrices. Our algorithms also exhibit a more stable behaviour over a wide range of the number of infected samples k when using Kirkman matrices. We compare some alternative matrix designs in Appendix 2.E.

2.5 Relation to Previous Work

We review some recent work which apply CS or combinatorial group testing for COVID-19 testing. The works in [99, 100, 101] adopt a nonadaptive CS based approach. The works in [102, 103, 104] use combinatorial group testing. Compared to these methods, our work is different in the following ways (also see [16]):

1. *Real/Synthetic data*: Our work as well as that in [100] have tested results on real data, while the rest present only numerical or theoretical results.

2. *Quantitative Noise model*: Our work uses the physically-derived noise model in Eqn. 2.3 (as opposed to only Gaussian noise). This noise model is not considered in [99]. The work in [101] considers unknown noise. Combinatorial group testing methods [102, 103, 104] do not make use of quantitative information. The work in [100] uses only binary test information, even though the decoding algorithm is based on CS.

3. *Algorithms*: The work in [99] adopts the BPDN technique (i.e P1 from Eqn. 2.11) as well as the brute-force search method for reconstruction. The work in [100, 105] uses the LASSO, albeit with a ternary representation for the viral loads. The work in [101] uses NNLAD. We use the LASSO with a non-negative constraint, the brute-force method, NNLAD, as well as other techniques such as SBL and NNOMP, all in combination with COMP. The work in [99] assumes knowledge of the (Gaussian) noise variance for selection of ε in the estimator in Eqn. 2.11, whereas we use cross-validation for all our estimators. The technique in [100] uses a slightly different form of cross-validation for selection of the regularization parameter in LASSO. Amongst combinatorial algorithms, [104] uses COMP, while [102] and [103] use message passing.

4. *Sensing matrix design*: The work in [99] uses randomly generated expander graphs, whereas we use Kirkman matrices. The work in [100] uses randomly generated sparse Bernoulli matrices or Reed-Solomon codes, while [103] uses Low-Density Parity Check (LDPC) codes [106]. The work in [101] uses Euler square matrices [107], and the work in [104] uses the Shifted Transversal Design [108]. Both are deterministic disjunct matrices like Kirkman matrices. Each sample in our matrix participates in 3 pools as opposed to 5 pools as used in

[103], 6 pools as used in [100] and [104], and 8 pools as used in [101], which is advantageous from the point of view of pipetting time.

5. *Sparsity estimation:* Our work uses an explicit sparsity estimator and does not rely on any assumption regarding the prevalence rate.

6. *Numerical comparisons:* We found that COMP-NNLAD works better than the NNLAD method used in [101] on our matrices (see Tables 2.5 and 2.32). We also found that COMP-NNLASSO and COMP-SBL have better sensitivity and specificity than COMP-NNLAD (see Tables 2.2, 2.3, and 2.5). The method in [100] can correctly identify up to 5/384 (1.3%) of samples with 48 tests, with an average number of false positives that was less than 2.75, and an average number of false negatives that was less than 0.33. On synthetic simulations with their 48×384 Reed-Solomon code based matrix (released by the authors) for a total of 100 $\boldsymbol{x}$ vectors with ℓ_0 norm of 5 using COMP-NNLASSO, we obtained 1.51 false positives and 0.02 false negatives on an average with a standard deviation of 1.439 and 0.14 respectively. Using COMP-SBL instead of COMP-NNLASSO with all other settings remaining the same, we obtained 1.4 false positives and 0.0 false negatives on an average with a standard deviation of 1.6 and 0.1 respectively. As such, a direct numerical comparison between our work and that in [100] is not possible, due to lack of available real data, however these numbers yield some indicator of performance.

7. *Number of Tests:* We use 93 tests for 961 samples while achieving more than 0.99 sensitivity and specificity for $k = 10$ infections using COMP-SBL. In a similar setting, [103] use 108 tests for $Q = 1000$ samples under prevalence rate 0.01 for exact 2-stage recovery. The work in [104] uses 186 tests for 961 samples under the same prevalence rate, albeit for sensitivity equal to 1 and very high specificity. Matrix sizes studied in other work are very different than ours. The work in [109] builds on top of our Tapestry scheme to reduce the number of tests, but it is a two-stage adaptive technique and hence will require much more testing time.

2.6 Conclusion

We have presented a single-round technique for prediction of infected samples as well as the viral loads, from an array of n samples, using a compressed sensing approach. We have advanced

the field of compressed sensing via our method which takes advantage of the heteroschedasticity inherent in RT-PCR testing by combining group testing with compressed sensing. We have proven theoretical guarantees for our method. We have empirically shown on synthetic data as well as on some real lab acquisitions that our technique can correctly predict the positive samples with a very small number of false positives and false negatives. Moreover, we have presented techniques for appropriate design of the mixing matrix. As of December 2023, the main thrust of the COVID-19 pandemic is over – however, our single-round testing method may be quickly deployed in many different scenarios in the future if such a need arises:

1. Testing of 105 symptomatic individuals in 45 tests.
2. Testing of 195 asymptomatic individuals in 45 tests assuming a low rate of infection. A good use case for this is airport security personnel, delivery personnel, or hospital staff.
3. Testing of 399 individuals in 63 tests. This can be used to test students coming back to campuses, or police force, or asymptomatic people in housing blocks and localities currently under quarantine.
4. Testing of 961 people in 93 tests, assuming low infection rate. This might be suitable for airports and other places where samples can be collected and tested immediately, and it might be possible to obtain liquid handling robots.

Outputs: We have designed an Android app named Byom Smart Testing to make our Tapestry protocol easy to deploy in the future. The app can be accessed at [34]. We also share our code and some amount of data at [110]. More information is also available at our website [111].
Future work: Future work will involve extensive testing on real COVID-19 data, and extensive implementation of a variety of algorithms for sensing matrix design as well as signal recovery, keeping in mind the accurate statistical noise model and accounting for occasional pipetting errors.

Appendices

Appendix 2.A Generalized Binary Search Techniques

In the class of 'adaptive group testing' techniques, the n samples are distributed into two or more groups, each of smaller size, and the smaller groups are then individually tested. In one particular adaptive method called generalized binary splitting (GBS) [27], this procedure is repeated (in a binary search fashion) until a single infected sample is identified. This requires $O(\log n)$ sequential tests, where each test requires mixing up to $n/2$ samples. This sample is then discarded, and the entire procedure is performed on the remaining $n-1$ samples. Such a procedure does not introduce any false negatives, and does not require prior knowledge of the number of infected samples k. It requires a total of only $O(k \log n)$ tests, if k is the number of infected samples. However such a multi-stage method is impractical to be deployed due to its sequential nature, since each RT-PCR stage requires nearly 3-4 hours. Moreover, each mixture that is tested contains contributions from as many as $O(n)$ samples, which can lead to significant dilution or may be difficult to implement in the lab. Hence in this work, we do not pursue this particular approach. Such an approach may be very useful if each individual test had a quick turn-around time.

Appendix 2.B Details of Cross-Validation for Compressed Sensing

For this, the measurements in $\boldsymbol{y}$ are divided into two randomly chosen disjoint sets: one for reconstruction ($\mathcal{R}$) and the other for validation ($\mathcal{V}$). A decoding algorithm such as NNLASSO is executed independently on multiple values of the regularization parameter λ from a candidate set Λ. (Other decoding algorithms will have their own parameters. For example, NNOMP or BPDN will use the parameter ε, i.e. the bound on the noise magnitude.) For each λ value, an estimate $\hat{\boldsymbol{x}}_{\boldsymbol{\lambda}}$ is produced using measurements only from $\mathcal{R}$, and the CV error $v_e(\lambda) := \sum_{i \in \mathcal{V}} (y_i - \boldsymbol{A^i} \hat{\boldsymbol{x}}_{\boldsymbol{\lambda}})^2$ is computed. The value of λ which yields the least value of $v_e(\lambda)$ is chosen, and a final estimate

of $\boldsymbol{x}$ is obtained by executing the algorithm again, but now using all measurements from $\mathcal{R} \cup \mathcal{V}$. If $\mathcal{V}$ is large enough, then $v_e(\lambda)$ is shown to be a good estimate of the actual error $\|\boldsymbol{x} - \hat{\boldsymbol{x}}_{\boldsymbol{\lambda}}\|^2$, as has been shown for Gaussian noise [60]. Nonetheless, it should be noted that CV is a method of choice for parameter selection in CS even under a variety of other noise models such as Poisson [112], etc, and we have experimentally observed that it works well even in the case of our noise model in Eqn. 2.3.

Appendix 2.C Brute-force search method

We refer to COMP-BF as a method where we apply COMP followed by a brute-force search to minimize the cost function in Eqn. 2.21 below. The brute-force search is computationally feasible only when $C(n, k)$ is 'reasonable' in value (note that the effective n is often reduced after application of COMP), and so we employ it only for small-sized matrices. The method essentially enumerates all possible supports of $\boldsymbol{x}$ which have size k. For each such candidate support set $\mathcal{Z}$, the following cost function is minimized using the `fmincon` routine of MATLAB which implements an interior-point optimizer [5]:

$$J(\boldsymbol{x}_{\boldsymbol{\mathcal{Z}}}) := \| \log \boldsymbol{y} - \log \boldsymbol{A}_{\boldsymbol{\mathcal{Z}}} \boldsymbol{x}_{\boldsymbol{\mathcal{Z}}} \|_2 \text{ such that } \boldsymbol{x}_{\boldsymbol{\mathcal{Z}}} \geq \boldsymbol{0}. \tag{2.21}$$

Results with the COMP-BF method are shown in Table 2.11. The special advantage of the brute-force method is that it requires only $m = 2k$ pools, which is less than $O(k \log n)$. However, such a method requires prior knowledge of k, or an estimate thereof. We employ a method to estimate k directly from $\boldsymbol{y}, \boldsymbol{A}$. This is described in Sec. 2.4.1.8. The results in Table 2.11 assume that the exact k was known, or that the estimator predicted the exact k. However, we observed that the estimator from Sec. 2.4.1.8 can sometimes over-estimate k. Hence, we also present results with COMP-BF where the brute-force search assumed that the sparsity was (over-estimated to be) $k + 1$ instead of k. These are shown in Table 2.12. A comparison of Tables 2.11 and 2.12 shows that RMSE deteriorates if k is incorrectly estimated. However there is no adverse effect on the number of false negatives, and only a small adverse effect on the number

[5] `https://in.mathworks.com/help/optim/ug/choosing-the-algorithm.html#bsbwxm7`

of false positives.

Table 2.11: Performance of COMP followed by brute-force search (COMP-BF) to minimize the function in Eqn. 2.21 for a 16×40 matrix optimized on mutual coherence, with different values of k, assuming that the true value of k was known. Results for both methods are reported on synthetic data.

Method	k	RMSE	#false neg.	#false pos.	sens.	spec.
COMP	2	1.00	0.00	0.50	1.00	0.99
COMP-BF	2	0.03	0.00	0.00	1.00	1.00
COMP	3	1.00	0.00	1.85	1.00	0.95
COMP-BF	3	0.05	0.00	0.00	1.00	1.00
COMP	4	1.00	0.00	3.35	1.00	0.91
COMP-BF	4	0.05	0.00	0.00	1.00	1.00

Table 2.12: Performance of COMP followed by brute-force search (COMP-BF) to minimize the cost function in Eqn. 2.21 for 16×40 matrix optimized on mutual coherence, with different values of k, assuming that the sparsity value was estimated to be $k + 1$. Results for both methods are reported on synthetic data.

Method	k	RMSE	#false neg.	#false pos.	sens.	spec.
COMP	2	1.00	0.00	0.75	1.00	0.98
COMP-BF	2	0.47	0.00	0.55	1.00	0.99
COMP	3	1.00	0.00	1.70	1.00	0.95
COMP-BF	3	0.24	0.00	0.80	1.00	0.98
COMP	4	1.00	0.00	3.00	1.00	0.92
COMP-BF	4	0.15	0.00	0.90	1.00	0.97

Appendix 2.D RIP and RNSP Preservation after COMP and NCOMP

We prove that the reduced matrix $\boldsymbol{A}_{\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}}$ obtained after `C-reduction` (Sec. 2.3.5) preserves RIP-1, RIP-2, RNSP, and ℓ_2-RNSP of the full matrix $\boldsymbol{A}$, of same order and with the same

parameters. Since reductions via COMP and NCOMP are instances of `C-reduction` as shown in Sec. 2.3.5, the results proven here hold for them. Recall definition of $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$, $\bar{\boldsymbol{\mathcal{X}}}$ and $\bar{\boldsymbol{\mathcal{Y}}}$ from Sec. 2.3.5 – $\boldsymbol{\mathcal{X}}$ is the set of indices of the columns removed by `C-reduction`, $\boldsymbol{\mathcal{Y}}$ is the set of indices of the rows removed by it, and $\bar{\boldsymbol{\mathcal{X}}}$, and $\bar{\boldsymbol{\mathcal{Y}}}$ are the corresponding complement sets.

First, from the definition of `C-reduction`, we have that

$$\boldsymbol{A}_{ij} = 0 \; \forall i \in \boldsymbol{\mathcal{Y}}, j \in \bar{\boldsymbol{\mathcal{X}}}, \tag{2.22}$$

since rows were removed by `C-reduction` if and only if they contained only 0 entries in the remaining columns. Second, we restate and prove Lemma 2.4:

Lemma 2.4. *Let $\bar{\boldsymbol{x}}$ denote a $|\bar{\boldsymbol{\mathcal{X}}}|$ dimensional vector. Consider a n-dimensional vector $\boldsymbol{x}$ such that $\boldsymbol{x}_{\bar{\boldsymbol{\mathcal{X}}}} = \bar{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$. Then, $\|\boldsymbol{A}\boldsymbol{x}\|_p = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_p$ for any $p \in \mathbb{Z}_+$.*

Proof. Let $\boldsymbol{A}^i$ denote the $1 \times n$ i^{th} row vector of the matrix $\boldsymbol{A}$.

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x}\|_p &= \Big(\sum_i |\boldsymbol{A}^i\boldsymbol{x}|^p\Big)^{1/p} \\
&= \Big(\sum_{i\in\bar{\boldsymbol{\mathcal{Y}}}} |\boldsymbol{A}^i\boldsymbol{x}|^p + \sum_{i\in\boldsymbol{\mathcal{Y}}} |\boldsymbol{A}^i\boldsymbol{x}|^p\Big)^{1/p} \\
&= \Big(\sum_{i\in\bar{\boldsymbol{\mathcal{Y}}}} |\boldsymbol{A}^i\boldsymbol{x}|^p + \sum_{i\in\boldsymbol{\mathcal{Y}}} \Big|\sum_{j\in\bar{\boldsymbol{\mathcal{X}}}} \boldsymbol{A}_{ij}\boldsymbol{x}_j + \sum_{j\in\boldsymbol{\mathcal{X}}} \boldsymbol{A}_{ij}\boldsymbol{x}_j\Big|^p\Big)^{1/p}.
\end{aligned} \tag{2.23}$$

Using Eq. (2.22), the second term in Eq. (2.23) is 0. The third term is 0 as well, since $\boldsymbol{x}_j = 0$ $\forall j \in \boldsymbol{\mathcal{X}}$. Hence,

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x}\|_p &= \Big(\sum_{i\in\bar{\boldsymbol{\mathcal{Y}}}} |\boldsymbol{A}^i\boldsymbol{x}|^p\Big)^{1/p} \\
&= \Big(\sum_{i\in\bar{\boldsymbol{\mathcal{Y}}}} \Big|\sum_{j\in\bar{\boldsymbol{\mathcal{X}}}} \boldsymbol{A}_{ij}\boldsymbol{x}_j + \sum_{j\in\boldsymbol{\mathcal{X}}} \boldsymbol{A}_{ij}\boldsymbol{x}_j\Big|^p\Big)^{1/p}.
\end{aligned} \tag{2.24}$$

Again, since $\boldsymbol{x}_j = 0 \; \forall j \in \boldsymbol{\mathcal{X}}$, the second term in Eq. (2.24) is 0. Hence,

$$\|\boldsymbol{A}\boldsymbol{x}\|_p = \Big(\sum_{i\in\bar{\boldsymbol{\mathcal{Y}}}} \Big|\sum_{j\in\bar{\boldsymbol{\mathcal{X}}}} \boldsymbol{A}_{ij}\boldsymbol{x}_j\Big|^p\Big)^{1/p}. \tag{2.25}$$

Notice that the RHS in Eq. (2.25) is simply $\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_p$ by definition of $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ and $\bar{\boldsymbol{x}}$. Hence, $\|\boldsymbol{A}\boldsymbol{x}\|_p = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_p$. □

Next, we prove Theorem 2.5 (restated below) and other theorems regarding preservation of sufficient conditions for CS recovery by `C-reduction` using Lemma 2.4:

Theorem 2.5. *Let d be a constant. If the scaled matrix $\frac{1}{d}\boldsymbol{A}$ satisfies RIP-1 of order k with k-order RIC $\delta_k \in (0,1)$, and $k \leq |\bar{\boldsymbol{\mathcal{X}}}|$, then the scaled `C-reduced` matrix $\frac{1}{d}\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ also satisfies RIP-1 of order k with RIC δ_k. That is, if*

$$\|\boldsymbol{x}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}\boldsymbol{x}\|_1 \leq (1+\delta_k)\|\boldsymbol{x}\|_1,$$

for all n-dimensional k-sparse vectors $\boldsymbol{x}$ then,

$$\|\bar{\boldsymbol{x}}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_1 \leq (1+\delta_k)\|\bar{\boldsymbol{x}}\|_1$$

for all $|\bar{\boldsymbol{\mathcal{X}}}|$-dimensional k-sparse vectors $\bar{\boldsymbol{x}}$.

Proof. Let $\bar{\boldsymbol{x}}$ be a $|\bar{\boldsymbol{\mathcal{X}}}|$ dimensional k-sparse vector. Consider a n-dimensional vector $\boldsymbol{x}$ such that $\boldsymbol{x}_{\bar{\boldsymbol{\mathcal{X}}}} = \bar{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$. $\boldsymbol{x}$ is also a k-sparse vector. By definition of RIP-1,

$$\|\boldsymbol{x}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}\boldsymbol{x}\|_1 \leq (1+\delta_k)\|\boldsymbol{x}\|_1.$$

Using Lemma 2.4, $\|\boldsymbol{A}\boldsymbol{x}\|_1 = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_1$. Hence,

$$\|\boldsymbol{x}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_1 \leq (1+\delta_k)\|\boldsymbol{x}\|_1.$$

Also note that $\|\boldsymbol{x}\|_1 = \|\bar{\boldsymbol{x}}\|_1$. Hence,

$$\|\bar{\boldsymbol{x}}\|_1 \leq \frac{1}{d}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_1 \leq (1+\delta_k)\|\bar{\boldsymbol{x}}\|_1$$

. □

Theorem 2.6. *Let d be a constant. If the scaled matrix $\boldsymbol{A}/d$ satisfies RIP-2 of order k with RIC $\delta_k \in (0,1)$ and $k \leq |\bar{\boldsymbol{\mathcal{X}}}|$, then the scaled* `C-reduced` *matrix $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}/d$ also satisfies RIP-2 of order k with RIC δ_k. That is, if*

$$(1-\delta_k)\|\boldsymbol{x}\|_2^2 \leq \frac{1}{d^2}\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{x}\|_2^2.$$

for all n-dimensional k-sparse vectors $\boldsymbol{x}$ then,

$$(1-\delta_k)\|\bar{\boldsymbol{x}}\|_2^2 \leq \frac{1}{d^2}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2^2 \leq (1+\delta_k)\|\bar{\boldsymbol{x}}\|_2^2$$

for all $|\bar{\boldsymbol{\mathcal{X}}}|$-dimensional k-sparse vectors $\bar{\boldsymbol{x}}$.

Proof. Let $\bar{\boldsymbol{x}}$ be a $|\bar{\boldsymbol{\mathcal{X}}}|$ dimensional k-sparse vector. Consider a n-dimensional vector $\boldsymbol{x}$ such that $\boldsymbol{x}_{\bar{\boldsymbol{\mathcal{X}}}} = \bar{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$. Hence $\boldsymbol{x}$ is also a k-sparse vector. By definition of RIP-2,

$$(1-\delta_k)\|\boldsymbol{x}\|_2^2 \leq \frac{1}{d^2}\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{x}\|_2^2.$$

Using Lemma 2.4, $\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2$. Hence,

$$(1-\delta_k)\|\boldsymbol{x}\|_2^2 \leq \frac{1}{d^2}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2^2 \leq (1+\delta_k)\|\boldsymbol{x}\|_2^2.$$

Also note that $\|\boldsymbol{x}\|_2 = \|\bar{\boldsymbol{x}}\|_2$. Hence,

$$(1-\delta_k)\|\bar{\boldsymbol{x}}\|_2^2 \leq \frac{1}{d^2}\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2^2 \leq (1+\delta_k)\|\bar{\boldsymbol{x}}\|_2^2$$

. □

Theorem 2.7. *If the matrix $\boldsymbol{A}$ satisfies ℓ_2-RNSP of order k with parameters ρ and τ, then the* `C-reduced` *matrix $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ also satisfies ℓ_2-RNSP of order k with the same parameters ρ and τ. That is, if*

$$\|\boldsymbol{x}_S\|_2 \leq \frac{\rho}{\sqrt{k}}\|\boldsymbol{x}_{\bar{S}}\|_1 + \tau\|\boldsymbol{A}\boldsymbol{x}\|_2 \;\forall \boldsymbol{x} \in \mathbb{R}^n$$

holds for all $S \subset \{1 \dots n\}$ *with* $|S| \leq k$ *then,*

$$\|\bar{\boldsymbol{x}}_B\|_2 \leq \frac{\rho}{\sqrt{k}}\|\bar{\boldsymbol{x}}_{\bar{B}}\|_1 + \tau\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2 \; \forall \bar{\boldsymbol{x}} \in \mathbb{R}^{|\bar{\boldsymbol{\mathcal{X}}}|}$$

holds for all $B \subset \{1 \dots |\bar{\boldsymbol{\mathcal{X}}}|\}$ *with* $|B| \leq k$.

Proof. Let $\bar{\boldsymbol{x}}$ be a $|\bar{\boldsymbol{\mathcal{X}}}|$-dimensional vector. Consider a n-dimensional vector $\boldsymbol{x}$ such that $\boldsymbol{x}_{\bar{\boldsymbol{\mathcal{X}}}} = \bar{\boldsymbol{x}}$ and $\boldsymbol{x}_{\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$. For any set $B \subset \{1, \dots, |\bar{\boldsymbol{\mathcal{X}}}|\}$, consider the set $S \subset \bar{\boldsymbol{\mathcal{X}}}$, such that $\boldsymbol{x}_S = \bar{\boldsymbol{x}}_B$, and the set $\bar{S} = \boldsymbol{\mathcal{X}} \cup (\bar{\boldsymbol{\mathcal{X}}} - S)$. By definition of ℓ_2-RNSP,

$$\|\boldsymbol{x}_S\|_2 \leq \frac{\rho}{\sqrt{k}}\|\boldsymbol{x}_{\bar{S}}\|_1 + \tau\|\boldsymbol{A}\boldsymbol{x}\|_2$$

Using Lemma 2.4, $\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2$. Also note that $\|\boldsymbol{x}_S\|_2 = \|\bar{\boldsymbol{x}}_B\|_2$, and $\|\boldsymbol{x}_{\bar{S}}\|_1 = \|\bar{\boldsymbol{x}}_{\bar{B}}\|_1$. Hence,

$$\|\bar{\boldsymbol{x}}_B\|_2 \leq \frac{\rho}{\sqrt{k}}\|\bar{\boldsymbol{x}}_{\bar{B}}\|_1 + \tau\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2.$$

□

Theorem 2.8. *If the matrix* $\boldsymbol{A}$ *satisfies RNSP of order* k *with parameters* ρ *and* τ*, then the* `C-reduced` *matrix* $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ *also satisfies RNSP of order* k *with the same parameters* ρ *and* τ*. That is, if*

$$\|\boldsymbol{x}_S\|_2 \leq \rho\|\boldsymbol{x}_{\bar{S}}\|_1 + \tau\|\boldsymbol{A}\boldsymbol{x}\|_2 \; \forall \boldsymbol{x} \in \mathbb{R}^n$$

holds for all $S \subset \{1 \dots n\}$ *with* $|S| \leq k$ *then,*

$$\|\bar{\boldsymbol{x}}_B\|_2 \leq \rho\|\bar{\boldsymbol{x}}_{\bar{B}}\|_1 + \tau\|\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}\bar{\boldsymbol{x}}\|_2 \; \forall \bar{\boldsymbol{x}} \in \mathbb{R}^{|\bar{\boldsymbol{\mathcal{X}}}|}$$

holds for all $B \subset \{1 \dots |\bar{\boldsymbol{\mathcal{X}}}|\}$ *with* $|B| \leq k$.

Proof. The proof follows the same argument as for Theorem 2.7, except the coefficient $\frac{\rho}{\sqrt{k}}$ is replaced by ρ.

□

Appendix 2.E Sensing Matrix Comparison

We generated sensing matrices optimized to have low mutual coherence, as described in Sec. 2.3.7. We have observed that these matrices, besides originating from randomly generated matrices, have a much higher number of non-zero elements and higher pool sizes compared to Kirkman or STS matrices. This leads to difficulty in pooling, increased pooling time, wastage of sample, and sample dilution. A well-tuned 93×961 matrix that we generated using this method had 6517 non-zero elements, with its smallest pool being of size 66. In comparison, the Kirkman triple matrix of the same dimensions had 2883 non-zero elements and a pool size equal to 31. Table 2.13 shows the performance of COMP followed by SBL on this matrix designed to minimize coherence, using synthetic data. For smaller values of k (upto 12), it has lower false negatives and false positives than the same algorithm on the 93×961 Kirkman matrix (Table 2.3). However, the number of false positives increases significantly for higher number of infections.

We also considered Bernoulli(p) random matrices, whose entries are 1 with probability p, and 0 otherwise. Here p is the Bernoulli parameter and lies in the range $(0, 1)$. It is shown in [12] that Bernoulli(p) matrices are good for compressed sensing. That is, they satisfy a robust nullspace property of order k if the number of rows $m = O(k \log n/k)$, with very high probability. On the other hand, Bernoulli($1/k$) sensing matrices with $O(k \log n)$ rows incur low probability of error when COMP is used to determine the support of k-sparse vectors [29]. Table 2.14 shows the performance of COMP followed by SBL on a 93×961 Bernoulli(0.5) matrix. The COMP step does not help that much in this case, leading to a large number of false positives in the subsequent SBL step. Table 2.15 shows the performance of COMP followed by SBL on a 93×961 Bernoulli(0.1) matrix. Results on this matrix are better than the Bernoulli(0.5) matrix. However, neither of these matrices come close to the performance exhibited by Kirkman triple matrices (e.g. Table 2.3).

Table 2.13: Performance of COMP followed by SBL (on synthetic data) for 93×961 matrix optimized for low mutual coherence. For each criterion, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.031 ± 0.012	0.0 ± 0.0	0.1 ± 0.3	1.0000 ± 0.0000	0.9999 ± 0.0003
8	0.041 ± 0.013	0.0 ± 0.1	1.8 ± 1.5	0.9996 ± 0.0068	0.9982 ± 0.0015
10	0.049 ± 0.013	0.0 ± 0.2	5.7 ± 3.3	0.9973 ± 0.0168	0.9940 ± 0.0034
12	0.062 ± 0.018	0.1 ± 0.3	13.0 ± 6.3	0.9920 ± 0.0262	0.9863 ± 0.0066
15	0.098 ± 0.030	0.2 ± 0.5	32.0 ± 12.8	0.9843 ± 0.0320	0.9662 ± 0.0135
17	0.120 ± 0.029	0.4 ± 0.6	49.8 ± 18.4	0.9777 ± 0.0356	0.9473 ± 0.0195
20	0.126 ± 0.029	0.6 ± 0.7	86.2 ± 28.6	0.9718 ± 0.0368	0.9084 ± 0.0304

Table 2.14: Performance of COMP followed by SBL (on synthetic data) for 93×961 Bernoulli(0.5) matrix. For each criterion, mean and standard deviation values are reported, across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.123 ± 0.051	0.1 ± 0.3	127.9 ± 127.8	0.9840 ± 0.0543	0.8662 ± 0.1337
8	0.147 ± 0.027	0.3 ± 0.5	405.9 ± 132.3	0.9675 ± 0.0652	0.5741 ± 0.1388
10	0.167 ± 0.026	0.4 ± 0.6	466.4 ± 65.7	0.9580 ± 0.0603	0.5096 ± 0.0690
12	0.190 ± 0.033	0.5 ± 0.6	475.1 ± 49.5	0.9558 ± 0.0533	0.4994 ± 0.0521
15	0.254 ± 0.050	1.0 ± 0.9	479.5 ± 37.0	0.9353 ± 0.0607	0.4932 ± 0.0391
17	0.288 ± 0.064	1.3 ± 0.9	484.6 ± 37.1	0.9241 ± 0.0554	0.4867 ± 0.0393
20	0.365 ± 0.096	1.9 ± 1.3	488.7 ± 44.9	0.9040 ± 0.0651	0.4806 ± 0.0477

Table 2.15: Performance of COMP followed by SBL (on synthetic data) for 93×961 Bernoulli(0.1) matrix. For each criterion, mean and standard deviation values are reported, across 1000 signals.

k	RMSE	#FN	#FP	Sens.	Spec.
5	0.035 ± 0.015	0.0 ± 0.0	1.5 ± 1.6	0.9998 ± 0.0063	0.9985 ± 0.0017
8	0.052 ± 0.018	0.0 ± 0.1	8.0 ± 5.6	0.9974 ± 0.0179	0.9916 ± 0.0058
10	0.073 ± 0.028	0.1 ± 0.2	17.9 ± 9.7	0.9940 ± 0.0242	0.9811 ± 0.0102
12	0.106 ± 0.039	0.2 ± 0.4	33.7 ± 18.3	0.9863 ± 0.0334	0.9645 ± 0.0193
15	0.137 ± 0.034	0.4 ± 0.6	68.8 ± 30.9	0.9757 ± 0.0386	0.9273 ± 0.0327
17	0.138 ± 0.031	0.5 ± 0.6	98.6 ± 38.5	0.9720 ± 0.0382	0.8955 ± 0.0408
20	0.141 ± 0.031	0.6 ± 0.8	155.8 ± 51.7	0.9679 ± 0.0394	0.8345 ± 0.0549

Appendix 2.F Non-negative Least Squares (NNLS)

We present the results of running COMP followed by Non-negative least squares (NNLS, section 2.3.4.4) on synthetic data for the 93×961 Kirkman matrix in Table 2.16.

Table 2.16: Performance of COMP followed by NNLS (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported, across 1000 signals.

k	RMSE	#FN	#FP	Sens.	Spec.
5	0.046 ± 0.018	0.0 ± 0.1	0.8 ± 0.9	0.9992 ± 0.0126	0.9992 ± 0.0009
8	0.070 ± 0.031	0.1 ± 0.3	4.0 ± 2.2	0.9912 ± 0.0324	0.9958 ± 0.0023
10	0.097 ± 0.044	0.2 ± 0.5	7.9 ± 3.1	0.9777 ± 0.0468	0.9917 ± 0.0032
12	0.154 ± 0.093	0.5 ± 0.7	13.1 ± 5.5	0.9566 ± 0.0586	0.9862 ± 0.0058
15	0.288 ± 0.164	0.9 ± 1.0	29.1 ± 17.0	0.9397 ± 0.0685	0.9692 ± 0.0180
17	0.397 ± 0.174	0.8 ± 1.2	50.0 ± 24.1	0.9547 ± 0.0715	0.9470 ± 0.0256
20	0.528 ± 0.167	0.2 ± 0.9	87.8 ± 25.3	0.9883 ± 0.0428	0.9067 ± 0.0269

Appendix 2.G Viral Loads of False Negatives

Table 2.17: Viral loads of false negative samples for the COMP followed by CS algorithms using a 93×961 Kirkman matrix on synthetic data, normalized by the maximum possible viral load. Mean and standard deviation reported over 1000 randomly generated signals.

k	**COMP-SBL**	**COMP-NNOMP**	**COMP-NNLASSO**	**COMP-NNLAD**	**COMP-NNLS**
5	0.013 ± 0.007	0.028 ± 0.031	0.021 ± 0.011	0.012 ± 0.014	0.011 ± 0.008
8	0.015 ± 0.017	0.023 ± 0.024	0.024 ± 0.022	0.029 ± 0.032	0.029 ± 0.030
10	0.023 ± 0.026	0.032 ± 0.052	0.035 ± 0.040	0.034 ± 0.036	0.030 ± 0.031
12	0.047 ± 0.080	0.066 ± 0.099	0.057 ± 0.066	0.067 ± 0.075	0.059 ± 0.067
15	0.093 ± 0.134	0.151 ± 0.181	0.098 ± 0.107	0.084 ± 0.083	0.081 ± 0.082
17	0.192 ± 0.227	0.224 ± 0.217	0.110 ± 0.113	0.085 ± 0.083	0.086 ± 0.088
20	0.306 ± 0.250	0.299 ± 0.240	0.192 ± 0.164	0.104 ± 0.120	0.105 ± 0.122

As seen in Tables 2.1 through 2.5, our method gives far fewer false positives when compared to COMP, at the cost of a rare false negative. Furthermore, due to the numerical nature of our method, there is asymmetry in our mode of failure, and we fail less on samples with high viral load. That is, even the rare false negatives that we fail to detect have very small viral loads. Table 2.17 shows the mean and standard deviation of viral loads of false negative samples for obtained by our method on synthetic data for a range of values of k. We see that for k upto 10, these false negative viral load values are very small. For example, if we use COMP followed by SBL, the average viral load value of false negative samples is 0.023 ± 0.026, around 40 times lower than the maximum viral load (1.0). This means that our method almost never misses a sample with high viral load. This matters particularly because COVID-19 super-spreaders are believed to have a high viral load [30].

Appendix 2.H Dependence of relative viral loads on parameter q

Typical RT-PCR output will not give us viral loads of the pools, rather, only the threshold cycle (C_t) values (see Sec. 2.2). In order to convert from the C_t values to the relative viral load vector, we use Eqn. 2.8, as explained in Sec. 2.3.1.1. For this, we need the value of the parameter q, which we set to 0.95, since we expect the viral amount in each pool to roughly double in each RT-PCR cycle.

The parameter q may be estimated from raw RT-PCR data, which contains the fluorescence values found at the end of each RT-PCR thermal cycle for each pool. If we take the natural logarithm on both sides of Eqn. 2.5, we can see that the logarithm of the fluorescence of a pool has a linear dependence on cycle time. Hence, q may be obtained by performing linear regression on log fluorescence values for any (positive) pool [70].

The value of q thus obtained will have some dependence on the pool so chosen. However, the relative viral loads of declared positives obtained by our algorithm show negligible variance in most cases, especially for samples with high viral load, over a large range of q (see Table 2.18). In some cases, especially for samples with low viral loads, we do observe some variance. However, it never happens that a sample with high relative viral load is reported to have low relative viral load, or vice-versa. Moreover, the declared positives do not change for any experiment.

Table 2.18: Effect of parameter q on the reported relative relative viral loads of real data, using COMP-SBL. Viral loads reported are relative to the viral load content of the pool with the smallest threshold cycle (C_t) value in that particular experiment. Sample IDs are the indices of the estimated viral load vector $\widetilde{\boldsymbol{x}}$ which have value more than 0 in that experiment, for any value of q. In some cases, there may be more sample ID entries for an experiment than the number of ground truth positive samples k in that experiment, due to false positives. Reported viral load of remaining samples is 0. (continued in Table 2.19)

Harvard 24×60, $k = 2$	**q** $\rightarrow$		**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**1**
	Sample IDs	**10**	0.13	0.11	0.09	0.07	0.06	0.04	0.03
		28	0.86	0.85	0.84	0.84	0.83	0.82	0.82
		54	0.08	0.09	0.09	0.10	0.10	0.11	0.11

Harvard 30×120, $k = 2$	**q** $\rightarrow$		**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**1**
	Sample IDs	**20**	0.86	0.85	0.85	0.84	0.83	0.83	0.82
		114	0.90	0.89	0.89	0.88	0.88	0.87	0.87

NCBS-1 16×40, $k = 1$	**q** $\rightarrow$		**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**1**
	Sample ID	**14**	0.78	0.77	0.77	0.76	0.75	0.74	0.73

NCBS-2 16×40, $k = 2$	**q** $\rightarrow$		**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**1**
	Sample IDs	**9**	0.59	0.58	0.58	0.57	0.57	0.56	0.56
		22	0.60	0.59	0.59	0.59	0.58	0.58	0.57

NCBS-3 16×40, $k = 3$	**q** $\rightarrow$		**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**	**1**
	Sample IDs	**4**	0.011	0.008	0.005	0.004	0.003	0.003	0.002
		6	0.07	0.06	0.05	0.05	0.04	0.04	0.03
		23	0.77	0.76	0.76	0.75	0.75	0.74	0.74

Table 2.19: Continuation of Table 2.18.

	q →		0.7	0.75	0.8	0.85	0.9	0.95	1
NCBS-4 16×40, $k = 4$	**Sample IDs**	**11**	0.014	0.009	0.006	0.004	0.003	0.003	0.003
		17	0.04	0.04	0.04	0.04	0.03	0.03	0.03
		18	0.001	0.002	0.002	0.003	0.003	0.003	0.004
		33	0.94	0.94	0.94	0.94	0.94	0.93	0.93
		36	0.11	0.10	0.09	0.08	0.07	0.06	0.05

Appendix 2.I Sensitivity of results to choice of threshold τ

Table 2.20: Sensitivity of COMP-NNLASSO to the choice of threshold τ. Reported numbers are for 1000 signals with $k = 10$ on the 93×961 Kirkman matrix.

COMP-NNLASSO					
τ	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
0	0.0989±0.0538	0.0000±0.0000	15.1620±4.4452	1.0000±0.0000	0.9841±0.0047
0.1	0.0989±0.0538	0.2570±0.4870	7.7850±3.3660	0.9743±0.0487	0.9918±0.0035
0.2	0.0989±0.0538	0.2570±0.4870	7.7760±3.3540	0.9743±0.0487	0.9918±0.0035
0.3	0.0989±0.0538	0.2590±0.4880	7.7730±3.3530	0.9741±0.0488	0.9918±0.0035
0.4	0.0989±0.0538	0.2590±0.4880	7.7670±3.3520	0.9741±0.0488	0.9918±0.0035
0.5	0.0989±0.0538	0.2600±0.4885	7.7660±3.3527	0.9740±0.0489	0.9918±0.0035
0.6	0.0989±0.0538	0.2600±0.4885	7.7640±3.3516	0.9740±0.0489	0.9918±0.0035
0.7	0.0989±0.0538	0.2600±0.4885	7.7640±3.3516	0.9740±0.0489	0.9918±0.0035
0.8	0.0989±0.0538	0.2600±0.4885	7.7620±3.3497	0.9740±0.0489	0.9918±0.0035
0.9	0.0989±0.0538	0.2610±0.4890	7.7620±3.3497	0.9739±0.0489	0.9918±0.0035
1	0.0989±0.0538	0.2610±0.4890	7.7610±3.3513	0.9739±0.0489	0.9918±0.0035

As discussed in section 2.4.1.4, we use a threshold of $\tau = 0.2 \times x_{\min}$, below which entries of the estimated viral load vector $\hat{\boldsymbol{x}}$ are set to 0. Here $x_{\min} = 1.0$ for our synthetic data. This is needed for the algorithms COMP-NNLASSO, COMP-NNLAD and COMP-NNLS. Tables 2.20, 2.21 and 2.22 show the variance in performance of these algorithms for different choices of τ, for the case when the number of infected samples $k = 10$. We see that $\tau = 0$ gives many false

Table 2.21: Sensitivity of COMP-NNLAD to the choice of threshold τ. Reported numbers are for 1000 signals with $k = 10$ on the 93×961 Kirkman matrix.

COMP-NNLAD					
τ	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
0	0.1060±0.0527	0.0000±0.0000	15.4640±4.6299	1.0000±0.0000	0.9837±0.0049
0.1	0.1060±0.0527	0.2030±0.4242	9.3480±3.1325	0.9797±0.0424	0.9902±0.0033
0.2	0.1060±0.0527	0.2030±0.4242	9.3450±3.1308	0.9797±0.0424	0.9902±0.0033
0.3	0.1060±0.0527	0.2030±0.4242	9.3440±3.1298	0.9797±0.0424	0.9902±0.0033
0.4	0.1060±0.0527	0.2030±0.4242	9.3420±3.1309	0.9797±0.0424	0.9902±0.0033
0.5	0.1060±0.0527	0.2030±0.4242	9.3420±3.1309	0.9797±0.0424	0.9902±0.0033
0.6	0.1060±0.0527	0.2030±0.4242	9.3400±3.1315	0.9797±0.0424	0.9902±0.0033
0.7	0.1060±0.0527	0.2030±0.4242	9.3380±3.1314	0.9797±0.0424	0.9902±0.0033
0.8	0.1060±0.0527	0.2030±0.4242	9.3360±3.1319	0.9797±0.0424	0.9902±0.0033
0.9	0.1060±0.0527	0.2030±0.4242	9.3340±3.1315	0.9797±0.0424	0.9902±0.0033
1	0.1060±0.0527	0.2030±0.4242	9.3310±3.1317	0.9797±0.0424	0.9902±0.0033

Table 2.22: Sensitivity of COMP-NNLS to the choice of threshold τ. Reported numbers are for 1000 signals with $k = 10$ on the 93×961 Kirkman matrix.

COMP-NNLS					
τ	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
0	0.0981±0.0520	0.0000±0.0000	15.4250±4.4020	1.0000±0.0000	0.9838±0.0046
0.1	0.0981±0.0520	0.2540±0.4896	7.9430±3.2712	0.9746±0.0490	0.9916±0.0034
0.2	0.0981±0.0520	0.2540±0.4896	7.9430±3.2712	0.9746±0.0490	0.9916±0.0034
0.3	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.4	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.5	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.6	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.7	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.8	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
0.9	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034
1	0.0981±0.0520	0.2540±0.4896	7.9420±3.2701	0.9746±0.0490	0.9916±0.0034

positives as compared to the other choices. For other values of τ, only COMP-NNLASSO shows some variance. We chose $\tau = 0.2$ to provide good tradeoff between sensitivity and specificity.

Appendix 2.J Optimal Expected Number of Dorfman Tests

A formula for the expected number of tests using Dorfman Testing is presented in [28]. However, it is valid only for the case when the number of samples is a multiple of the pool size. We slightly modify their derivation to handle the case when it is not so. Let there be n samples $\{1 \dots n\}$, with disease prevalence rate $p = k/n$. That is, any given sample is positive with probability $p = k/n$, independently of other samples. Thus, out of n samples, the expected number of samples which are infected is k. Let the samples be divided into $\lfloor n/g \rfloor$ pools of size $g \geq 2$, and one pool of size $r = n - \lfloor n/g \rfloor\, g$. Since p is the probability that a given sample is infected, hence $1 - p$ is the probability that a given sample is not infected. Then $p'(l) = 1 - (1-p)^l$ is the probability of a pool of size l being infected. Let T_g be the total number of first stage and second stage tests taken by the Dorfman testing method, given the pool size g. Let the pools thus formed be numbered $\{1 \dots \lfloor n/g \rfloor + 1\}$, with a 0-sized last pool if $r = 0$ for the purposes of our computation. Also, let $t_j = 1$ if the j^{th} pool tested positive, 0 otherwise. The 0-sized pool is considered to be tested as negative. Also note that if $r = 1$, the last pool need not be retested even if it tested as positive. Let $\bar{r}$ denote the number of tests done for the last pool if it tested positive. Hence $\bar{r} = 0$ if $r \leq 1$, else $\bar{r} = r$. Then,

$$T_g = \sum_{j=1}^{\lfloor \frac{n}{g} \rfloor} t_j g + \bar{r} t_{\lfloor \frac{n}{g} \rfloor + 1} + \left\lceil \frac{n}{g} \right\rceil . \tag{2.26}$$

By linearity of expectations, we get

$$E[T_g] = g \left\lfloor \frac{n}{g} \right\rfloor p'(g) + \bar{r} p'(r) + \left\lceil \frac{n}{g} \right\rceil . \tag{2.27}$$

The optimal expected number of tests is $\min_g E[T_g]$, and the optimal pool size is

$g^* = \arg\min_g E[T_g]$. Since there is no closed form for this, we implemented this numerically in practice – see Table 2.6 and Sec. 2.4.1.7.

Appendix 2.K Fraction of samples remaining after COMP using Kirkman matrices

Proposition 2.9. *Let $\boldsymbol{A}$ be an $m \times n$ full Kirkman matrix. Let $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ be a measurement vector for some $\boldsymbol{x} \in \mathbb{R}^n$ such that $\|\boldsymbol{x}\|_0 = k$. Let $f := \|\boldsymbol{y}\|_0/m \in (0,1)$ be the fraction of pools that are tested positive. Then the fraction of samples declared positive by* COMP *is strictly less than f^2.*

Proof. We will prove the proposition for the case when $\boldsymbol{A}$ is a Steiner Triple System matrix. Since every full Kirkman matrix is also a Steiner Triple System matrix, the same proof holds.

Let $\bar{\boldsymbol{\mathcal{Y}}}$ be the set of pools that tested positive and $\bar{\boldsymbol{\mathcal{X}}}$ be the set of samples declared positive by COMP. Then, we have $|\bar{\boldsymbol{\mathcal{Y}}}| = fm$. Recall from Sec. 2.3.7.3 that since $\boldsymbol{A}$ is a Steiner Triple System matrix each column consists of 3 entries with value 1. Notice that any column which does not have all three of its 1 entries in $\bar{\boldsymbol{\mathcal{Y}}}$ would have gotten eliminated by COMP. Hence the reduced matrix $A_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$ also has the property that each column consists of 3 entries with value 1.

Recall from Sec. 2.3.7.3 that in a Steiner Triple System matrix, each column corresponds to a triplet of rows, and each (unordered) pair of rows occurs together in exactly 1 such triplet. Hence each column of $\boldsymbol{A}$ corresponds to $\binom{3}{2} = 3$ unique pairs of rows of $\boldsymbol{A}$. Consequently, in the reduced matrix $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$, each column corresponds to 3 unique pairs of rows of $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$, since for each column in $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$, all the rows with 1 entries are in $\bar{\boldsymbol{\mathcal{Y}}}$.

Hence, for $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$, total number of such unique pairs of rows as enumerated by its columns is $3|\bar{\boldsymbol{\mathcal{X}}}|$. This number must be less than or equal to the maximum number of unique pairs of rows of $\boldsymbol{A}_{\bar{\boldsymbol{\mathcal{X}}},\bar{\boldsymbol{\mathcal{Y}}}}$, which is $\binom{|\bar{\boldsymbol{\mathcal{Y}}}|}{2} = fm(fm-1)/2$. Therefore, we have

$$|\bar{\boldsymbol{\mathcal{X}}}| \leq \frac{fm(fm-1)}{6} < \frac{fm(fm-f)}{6} = f^2\frac{m(m-1)}{6}. \tag{2.28}$$

Recall from Sec. 2.3.7.3 that the number of columns n of a Steiner Triple System matrix is equal to $\binom{m}{2}/3 = m(m-1)/6$. Hence,

$$\frac{|\bar{\boldsymbol{\mathcal{X}}}|}{n} < f^2. \tag{2.29}$$

□

This gives some intuition as to why COMP eliminates so many samples when using Kirkman matrices, as observed from Table 2.1, especially in the regime $k \ll m$. Since each sample can only make 3 tests positive, number of positive tests is at most $3k$. Hence $f \leq 3k/m$ for Kirkman matrices. For example, when using a full Kirkman matrix, if the fractions of positive tests f is 0.1, fraction of samples that remain after COMP is at most $f^2 = 0.01$. Similarly, if $f = 0.3$, then $f^2 = 0.09$, and so on.

Appendix 2.L Synthetic data results on 45×105 Kirkman triple matrix

We present performance of COMP followed by CS algorithms on synthetic data for the Kirkman triple matrix of size 45×105 in Tables 2.23, 2.24, 2.25, 2.26, 2.27, and 2.28. We followed the same methodology as in Sec. 2.4.1.1 for these experiments. Using COMP without the CS step gives acceptable performance up to $k = 8$. For larger values of k, significant improvements are seen by performing the additional CS step after COMP. COMP-SBL is the best algorithm, achieving high sensitivity and specificity for a wide range of k. It is followed closely by COMP-NNLAD, COMP-NNLASSO, and COMP-NNLS, all of which give higher false negatives than COMP-SBL for high values of k. COMP-NNOMP provides the highest specificity, albeit at the cost of lower sensitivity than the other algorithms, especially for high values of k. We note that COMP-SBL and COMP-NNLAD achieve greater than 0.99 sensitivity and 0.95 specificity for k upto 10. COMP-SBL's performance degrades gracefully for higher values of k.

Table 2.23: Performance of COMP and DD (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**	$\#\mathcal{HCP}$
5	1.000 ± 0.000	0.0 ± 0.0	1.0 ± 1.0	1.0000 ± 0.0000	0.9899 ± 0.0099	4.8
8	1.000 ± 0.000	0.0 ± 0.0	4.4 ± 2.2	1.0000 ± 0.0000	0.9541 ± 0.0223	5.2
10	1.000 ± 0.000	0.0 ± 0.0	8.0 ± 3.2	1.0000 ± 0.0000	0.9163 ± 0.0338	4.0
12	1.000 ± 0.000	0.0 ± 0.0	12.2 ± 4.1	1.0000 ± 0.0000	0.8689 ± 0.0446	2.5
15	1.000 ± 0.000	0.0 ± 0.0	19.9 ± 5.8	1.0000 ± 0.0000	0.7791 ± 0.0647	0.9
17	1.000 ± 0.000	0.0 ± 0.0	24.9 ± 6.6	1.0000 ± 0.0000	0.7174 ± 0.0747	0.5
20	1.000 ± 0.000	0.0 ± 0.0	32.0 ± 8.1	1.0000 ± 0.0000	0.6233 ± 0.0955	0.1

Table 2.24: Performance of COMP followed by NNLASSO (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.047 ± 0.020	0.0 ± 0.1	0.5 ± 0.7	0.9994 ± 0.0109	0.9949 ± 0.0072
8	0.064 ± 0.024	0.0 ± 0.2	2.3 ± 1.6	0.9941 ± 0.0265	0.9761 ± 0.0165
10	0.079 ± 0.031	0.1 ± 0.3	3.9 ± 2.2	0.9892 ± 0.0317	0.9585 ± 0.0230
12	0.100 ± 0.041	0.3 ± 0.5	6.1 ± 2.9	0.9783 ± 0.0409	0.9340 ± 0.0313
15	0.143 ± 0.076	0.5 ± 0.7	9.4 ± 3.9	0.9654 ± 0.0465	0.8956 ± 0.0432
17	0.178 ± 0.097	0.9 ± 0.9	12.0 ± 5.3	0.9493 ± 0.0521	0.8639 ± 0.0605
20	0.256 ± 0.131	1.2 ± 1.1	17.1 ± 9.5	0.9380 ± 0.0557	0.7993 ± 0.1123

Table 2.25: Performance of COMP followed by SBL (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.046 ± 0.019	0.0 ± 0.1	0.5 ± 0.7	0.9994 ± 0.0109	0.9945 ± 0.0075
8	0.058 ± 0.020	0.0 ± 0.1	2.4 ± 1.6	0.9974 ± 0.0179	0.9749 ± 0.0169
10	0.070 ± 0.023	0.1 ± 0.3	4.3 ± 2.3	0.9930 ± 0.0255	0.9550 ± 0.0241
12	0.085 ± 0.035	0.1 ± 0.3	6.7 ± 2.9	0.9911 ± 0.0271	0.9281 ± 0.0312
15	0.112 ± 0.041	0.2 ± 0.5	10.8 ± 4.0	0.9846 ± 0.0319	0.8799 ± 0.0445
17	0.142 ± 0.081	0.4 ± 0.6	13.7 ± 4.5	0.9754 ± 0.0376	0.8445 ± 0.0517
20	0.195 ± 0.145	0.7 ± 0.9	18.0 ± 5.8	0.9628 ± 0.0443	0.7885 ± 0.0685

Table 2.26: Performance of COMP followed by NNOMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.046 ± 0.020	0.0 ± 0.1	0.2 ± 0.5	0.9984 ± 0.0178	0.9980 ± 0.0047
8	0.060 ± 0.024	0.1 ± 0.3	1.0 ± 1.3	0.9882 ± 0.0377	0.9895 ± 0.0134
10	0.073 ± 0.029	0.2 ± 0.5	2.2 ± 2.1	0.9774 ± 0.0478	0.9769 ± 0.0217
12	0.090 ± 0.036	0.3 ± 0.6	3.9 ± 2.8	0.9713 ± 0.0503	0.9582 ± 0.0300
15	0.135 ± 0.069	0.6 ± 0.8	7.7 ± 3.7	0.9573 ± 0.0528	0.9139 ± 0.0414
17	0.175 ± 0.103	1.0 ± 1.0	10.2 ± 3.8	0.9406 ± 0.0590	0.8842 ± 0.0426
20	0.266 ± 0.169	2.0 ± 1.6	12.7 ± 3.8	0.8998 ± 0.0789	0.8502 ± 0.0448

Table 2.27: Performance of COMP followed by NNLAD (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	RMSE	#FN	#FP	Sens.	Spec.
5	0.050 ± 0.022	0.0 ± 0.0	0.6 ± 0.8	0.9998 ± 0.0063	0.9936 ± 0.0079
8	0.068 ± 0.028	0.0 ± 0.2	2.7 ± 1.7	0.9968 ± 0.0199	0.9721 ± 0.0174
10	0.087 ± 0.037	0.1 ± 0.3	5.0 ± 2.3	0.9903 ± 0.0309	0.9479 ± 0.0246
12	0.105 ± 0.041	0.2 ± 0.4	7.3 ± 3.0	0.9841 ± 0.0356	0.9215 ± 0.0318
15	0.150 ± 0.070	0.5 ± 0.7	10.5 ± 3.9	0.9693 ± 0.0441	0.8837 ± 0.0431
17	0.195 ± 0.100	0.8 ± 0.8	13.0 ± 5.7	0.9554 ± 0.0491	0.8527 ± 0.0644
20	0.261 ± 0.126	1.1 ± 1.1	18.1 ± 10.2	0.9438 ± 0.0548	0.7870 ± 0.1203

Table 2.28: Performance of COMP followed by NNLS (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion and each k value, mean and standard deviation values are reported across 1000 signals.

k	RMSE	#FN	#FP	Sens.	Spec.
5	0.046 ± 0.019	0.0 ± 0.0	0.5 ± 0.7	0.9998 ± 0.0063	0.9947 ± 0.0071
8	0.063 ± 0.023	0.0 ± 0.2	2.2 ± 1.5	0.9944 ± 0.0265	0.9772 ± 0.0159
10	0.079 ± 0.033	0.1 ± 0.4	4.1 ± 2.2	0.9863 ± 0.0358	0.9565 ± 0.0234
12	0.097 ± 0.036	0.2 ± 0.5	6.2 ± 2.9	0.9810 ± 0.0382	0.9337 ± 0.0313
15	0.138 ± 0.068	0.5 ± 0.7	9.3 ± 4.0	0.9660 ± 0.0469	0.8966 ± 0.0444
17	0.183 ± 0.099	0.8 ± 0.8	12.1 ± 5.9	0.9525 ± 0.0500	0.8629 ± 0.0667
20	0.251 ± 0.129	1.2 ± 1.1	17.6 ± 10.5	0.9425 ± 0.0553	0.7935 ± 0.1231

Appendix 2.M Synthetic data results using CS algorithms only

We provide results of running CS algorithms on synthetic data for the 93×961 and 45×105 Kirkman triple matrices, without doing the COMP preprocessing step, in Tables 2.29, 2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37 and 2.38. We find that running the CS algorithms without the COMP step increases false negatives or false positives by a large factor for each algorithm. We observed that this method was computationally more expensive due to the algorithms running on the full matrix. We used the same methodology as in Sec. 2.4.1.1. Results for all algorithms are over 1000 randomly generated signals for each k, except for NNLASSO on 93×961 Kirkman triple matrix, for which only 100 signals were used.

Table 2.29: Performance of NNLASSO without COMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion, mean and standard deviation are reported over 100 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.254 ± 0.368	0.1 ± 0.3	78.2 ± 28.2	0.9780 ± 0.0690	0.9182 ± 0.0295
8	0.326 ± 0.360	0.2 ± 0.5	82.2 ± 33.3	0.9738 ± 0.0672	0.9137 ± 0.0349
10	0.418 ± 0.346	0.5 ± 0.7	85.4 ± 36.3	0.9470 ± 0.0717	0.9102 ± 0.0382
12	0.558 ± 0.295	0.9 ± 0.9	83.5 ± 30.9	0.9283 ± 0.0759	0.9120 ± 0.0326
15	0.656 ± 0.196	1.9 ± 1.4	95.1 ± 34.2	0.8713 ± 0.0915	0.8995 ± 0.0362
17	0.723 ± 0.161	2.3 ± 1.4	113.2 ± 43.2	0.8635 ± 0.0823	0.8801 ± 0.0457
20	0.787 ± 0.099	2.8 ± 1.8	141.4 ± 43.2	0.8580 ± 0.0887	0.8497 ± 0.0459

Table 2.30: Performance of SBL without COMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion, mean and standard deviation are reported over 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.105 ± 0.213	0.0 ± 0.0	388.4 ± 63.2	1.0000 ± 0.0000	0.5937 ± 0.0661
8	0.113 ± 0.211	0.1 ± 0.2	420.2 ± 55.5	0.9925 ± 0.0297	0.5591 ± 0.0582
10	0.065 ± 0.024	0.0 ± 0.1	443.9 ± 42.9	0.9990 ± 0.0099	0.5332 ± 0.0452
12	0.070 ± 0.024	0.1 ± 0.4	449.9 ± 38.1	0.9942 ± 0.0295	0.5259 ± 0.0401
15	0.108 ± 0.139	0.1 ± 0.4	450.0 ± 29.1	0.9913 ± 0.0277	0.5243 ± 0.0308
17	0.179 ± 0.236	0.6 ± 1.2	456.5 ± 30.0	0.9659 ± 0.0702	0.5165 ± 0.0318
20	0.317 ± 0.351	1.4 ± 2.2	467.4 ± 30.5	0.9290 ± 0.1098	0.5033 ± 0.0324

Table 2.31: Performance of NNOMP without COMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion, mean and standard deviation are reported over 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.048 ± 0.022	0.0 ± 0.2	2.6 ± 8.5	0.9940 ± 0.0341	0.9973 ± 0.0089
8	0.062 ± 0.025	0.2 ± 0.4	9.5 ± 17.7	0.9750 ± 0.0559	0.9900 ± 0.0186
10	0.168 ± 0.280	0.8 ± 1.7	16.7 ± 22.1	0.9180 ± 0.1740	0.9825 ± 0.0232
12	0.242 ± 0.331	2.0 ± 3.3	19.4 ± 21.7	0.8292 ± 0.2745	0.9795 ± 0.0229
15	0.526 ± 0.416	6.4 ± 5.3	11.9 ± 16.9	0.5747 ± 0.3565	0.9874 ± 0.0178
17	0.734 ± 0.410	10.3 ± 6.2	9.8 ± 15.5	0.3918 ± 0.3632	0.9897 ± 0.0164
20	0.902 ± 0.306	15.4 ± 5.4	6.2 ± 10.1	0.2290 ± 0.2697	0.9934 ± 0.0107

Table 2.32: Performance of NNLAD without COMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion, mean and standard deviation are reported over 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.080 ± 0.028	0.0 ± 0.1	19.9 ± 6.3	0.9972 ± 0.0235	0.9791 ± 0.0066
8	0.103 ± 0.036	0.1 ± 0.3	43.8 ± 16.2	0.9914 ± 0.0322	0.9541 ± 0.0170
10	0.130 ± 0.050	0.2 ± 0.5	45.9 ± 23.6	0.9750 ± 0.0479	0.9517 ± 0.0249
12	0.178 ± 0.086	0.6 ± 0.7	38.3 ± 24.1	0.9539 ± 0.0596	0.9597 ± 0.0254
15	0.293 ± 0.150	0.9 ± 1.0	37.7 ± 18.3	0.9390 ± 0.0672	0.9602 ± 0.0193
17	0.404 ± 0.177	0.8 ± 1.2	53.9 ± 22.9	0.9558 ± 0.0721	0.9429 ± 0.0243
20	0.542 ± 0.163	0.3 ± 1.0	88.1 ± 24.8	0.9855 ± 0.0489	0.9063 ± 0.0263

Table 2.33: Performance of NNLS without COMP (on synthetic data) for 93×961 Kirkman triple matrix. For each criterion, mean and standard deviation are reported over 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.072 ± 0.024	0.1 ± 0.2	64.0 ± 5.6	0.9880 ± 0.0492	0.9330 ± 0.0058
8	0.091 ± 0.031	0.2 ± 0.5	66.7 ± 4.2	0.9739 ± 0.0564	0.9300 ± 0.0044
10	0.120 ± 0.052	0.4 ± 0.6	69.0 ± 4.1	0.9634 ± 0.0564	0.9275 ± 0.0043
12	0.164 ± 0.087	0.6 ± 0.8	70.1 ± 7.8	0.9472 ± 0.0634	0.9261 ± 0.0082
15	0.301 ± 0.160	1.0 ± 1.1	67.2 ± 12.0	0.9359 ± 0.0706	0.9289 ± 0.0127
17	0.412 ± 0.170	0.7 ± 1.2	69.3 ± 10.9	0.9561 ± 0.0713	0.9266 ± 0.0115
20	0.538 ± 0.155	0.3 ± 1.0	92.1 ± 18.0	0.9855 ± 0.0499	0.9021 ± 0.0192

Table 2.34: Performance of NNLASSO without COMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.066 ± 0.022	0.0 ± 0.2	19.6 ± 2.8	0.9914 ± 0.0416	0.8037 ± 0.0283
8	0.082 ± 0.024	0.1 ± 0.3	21.5 ± 2.7	0.9840 ± 0.0433	0.7788 ± 0.0277
10	0.096 ± 0.030	0.2 ± 0.4	22.2 ± 2.7	0.9785 ± 0.0442	0.7666 ± 0.0283
12	0.111 ± 0.040	0.4 ± 0.6	22.6 ± 2.7	0.9699 ± 0.0488	0.7573 ± 0.0289
15	0.150 ± 0.075	0.7 ± 0.8	23.3 ± 5.1	0.9561 ± 0.0526	0.7414 ± 0.0562
17	0.182 ± 0.089	0.9 ± 0.9	24.3 ± 7.4	0.9467 ± 0.0537	0.7239 ± 0.0845
20	0.258 ± 0.136	1.2 ± 1.1	29.7 ± 16.9	0.9379 ± 0.0567	0.6504 ± 0.1983

Table 2.35: Performance of SBL without COMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.078 ± 0.026	0.0 ± 0.0	49.9 ± 3.6	0.9996 ± 0.0089	0.5012 ± 0.0364
8	0.088 ± 0.029	0.0 ± 0.2	48.9 ± 3.8	0.9959 ± 0.0230	0.4962 ± 0.0395
10	0.095 ± 0.029	0.1 ± 0.3	47.8 ± 4.0	0.9898 ± 0.0316	0.4968 ± 0.0418
12	0.105 ± 0.034	0.2 ± 0.4	47.2 ± 3.9	0.9865 ± 0.0331	0.4924 ± 0.0420
15	0.122 ± 0.051	0.3 ± 0.6	46.3 ± 4.1	0.9789 ± 0.0377	0.4855 ± 0.0457
17	0.138 ± 0.081	0.4 ± 0.7	45.6 ± 4.2	0.9736 ± 0.0383	0.4815 ± 0.0474
20	0.202 ± 0.154	0.8 ± 0.9	45.4 ± 4.6	0.9595 ± 0.0470	0.4663 ± 0.0542

Table 2.36: Performance of NNOMP without COMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.050 ± 0.023	0.0 ± 0.1	1.6 ± 3.7	0.9986 ± 0.0167	0.9836 ± 0.0365
8	0.070 ± 0.027	0.1 ± 0.4	5.7 ± 6.3	0.9819 ± 0.0515	0.9408 ± 0.0646
10	0.085 ± 0.032	0.2 ± 0.5	9.9 ± 6.8	0.9759 ± 0.0520	0.8960 ± 0.0720
12	0.103 ± 0.036	0.4 ± 0.6	14.4 ± 6.5	0.9667 ± 0.0509	0.8451 ± 0.0700
15	0.141 ± 0.066	0.6 ± 0.8	18.8 ± 5.0	0.9575 ± 0.0509	0.7912 ± 0.0555
17	0.182 ± 0.108	1.1 ± 1.1	20.0 ± 3.7	0.9368 ± 0.0639	0.7724 ± 0.0421
20	0.278 ± 0.177	2.1 ± 1.7	20.1 ± 2.8	0.8944 ± 0.0828	0.7634 ± 0.0332

Table 2.37: Performance of NNLAD without COMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.067 ± 0.026	0.0 ± 0.1	5.6 ± 2.4	0.9986 ± 0.0167	0.9441 ± 0.0242
8	0.089 ± 0.031	0.1 ± 0.2	11.4 ± 4.1	0.9926 ± 0.0305	0.8822 ± 0.0419
10	0.106 ± 0.040	0.1 ± 0.4	14.5 ± 5.0	0.9865 ± 0.0362	0.8475 ± 0.0529
12	0.128 ± 0.047	0.3 ± 0.5	15.8 ± 5.3	0.9778 ± 0.0404	0.8301 ± 0.0565
15	0.166 ± 0.076	0.6 ± 0.7	16.7 ± 5.0	0.9615 ± 0.0478	0.8141 ± 0.0560
17	0.201 ± 0.094	0.9 ± 0.9	17.1 ± 5.5	0.9494 ± 0.0513	0.8052 ± 0.0629
20	0.272 ± 0.125	1.2 ± 1.2	21.3 ± 9.9	0.9404 ± 0.0578	0.7500 ± 0.1164

Table 2.38: Performance of NNLS without COMP (on synthetic data) for 45×105 Kirkman triple matrix. For each criterion, mean and standard deviation values are reported across 1000 signals.

k	**RMSE**	**#FN**	**#FP**	**Sens.**	**Spec.**
5	0.066 ± 0.021	0.1 ± 0.2	19.3 ± 2.8	0.9882 ± 0.0471	0.8067 ± 0.0285
8	0.082 ± 0.025	0.1 ± 0.4	21.4 ± 2.5	0.9831 ± 0.0452	0.7793 ± 0.0261
10	0.094 ± 0.029	0.3 ± 0.5	22.0 ± 2.7	0.9745 ± 0.0492	0.7684 ± 0.0284
12	0.110 ± 0.037	0.3 ± 0.5	22.4 ± 2.8	0.9722 ± 0.0446	0.7596 ± 0.0304
15	0.147 ± 0.068	0.6 ± 0.7	23.1 ± 3.0	0.9571 ± 0.0493	0.7435 ± 0.0332
17	0.183 ± 0.089	0.9 ± 0.9	23.6 ± 3.8	0.9474 ± 0.0537	0.7324 ± 0.0429
20	0.262 ± 0.131	1.3 ± 1.2	25.3 ± 7.2	0.9368 ± 0.0599	0.7026 ± 0.0843

Chapter 3

Image Moderation

3.1 Introduction

Compressed sensing (CS) has been a very extensively studied branch of signal/image processing, which involves acquiring signals/images directly in compressed form as opposed to performing compression post acquisition. Consider a possibly noisy vector $\boldsymbol{y} \in \mathbb{R}^m$ of measurements of the signal $\boldsymbol{x} \in \mathbb{R}^n$ with $m \ll n$, where the measurements are acquired via a sensing matrix $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$ (implemented in hardware). Then we have the relationship:

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{\eta}, \tag{3.1}$$

where $\boldsymbol{\eta} \in \mathbb{R}^m$ is a vector of i.i.d. noise values. CS theory [3, 6] states that under two conditions, $\boldsymbol{x}$ can be stably and robustly recovered from $\boldsymbol{y}, \boldsymbol{\Phi}$, with rigorous theoretical guarantees, by solving convex optimization problems such as the LASSO [6], given as follows:

$$\hat{\boldsymbol{x}} \triangleq \text{argmin}_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|^2 + \lambda \|\boldsymbol{x}\|_1, \tag{3.2}$$

where λ is a carefully chosen regularization parameter. The two conditions are: ($\mathcal{C}1$) $\boldsymbol{x}$ should be a sufficiently sparse vector, and ($\mathcal{C}2$) no sparse vector, except for a vector of all zeroes, should lie in the null-space of $\boldsymbol{\Phi}$. $\mathcal{C}1$ and $\mathcal{C}2$ ensure that $\boldsymbol{y}$ and $\boldsymbol{x}$ are uniquely mapped to each other via $\boldsymbol{\Phi}$ even though $m \ll n$. $\mathcal{C}2$ is typically satisfied when the entries of $\boldsymbol{\Phi}$ belong to sub-Gaussian distributions and when $m \geq O(k \log n/k)$. If these conditions are met, then successful recovery of a vector $\boldsymbol{x}$ with at the most k non-zero elements is ensured [3, 6].

Group testing (GT), also called pooled testing, is an area of information theory which is closely related to CS [72]. Given a set of n samples (for example, blood/urine samples) that need to be tested for a *rare* disease, GT replaces tests on the individual samples by tests on *pools* (also called *groups*), where each pool is created using a subset of the n samples. Given the test results on each pool (as to whether or not the pool contains one or more diseased samples), the aim is to infer the status of the individual n samples. This approach dates back to the classical work of Dorfman [28] and has been recently very successful in saving resources in COVID-19 RT-PCR testing [14, 100]. In some cases, side information in the form of contact tracing matrices has also been used to further enhance the results, using approximate message passing and the concept of group sparsity [54]. If the test results report the *number* of defective/diseased samples in a pool instead of just a binary result, it is referred to as quantitative group testing (QGT) [113].

Image moderation is the process of examining image content to determine whether it depicts any objectionable content or violates copyright issues. In this work, we are concerned with semantic image moderation to determine whether the image *content* is objectionable, for example depiction of violence (such as images of weapons), or whether it is off-topic for the chosen forum (such as images of a tennis game being shared on an online forum for baseball).

Manually screening the millions of images shared on online forums like Facebook, Instagram, etc. is very tedious. There exist many commercial solutions for automated image moderation (Amazon, Azure, Picpurify, Webpurify). However, there exists only a small-sized body of academic research work in this area, mostly focused on specific categories of objectionable content, such as firearms/guns [37], knives [38, 39] or detection of violent scenes in images/videos [40]. Most of these papers employ NNs for classification, given their success in image classification tasks since [114]. However large-scale NNs require several Giga-flops of operations for a single forward pass [41] and consume considerable amounts of power as shown

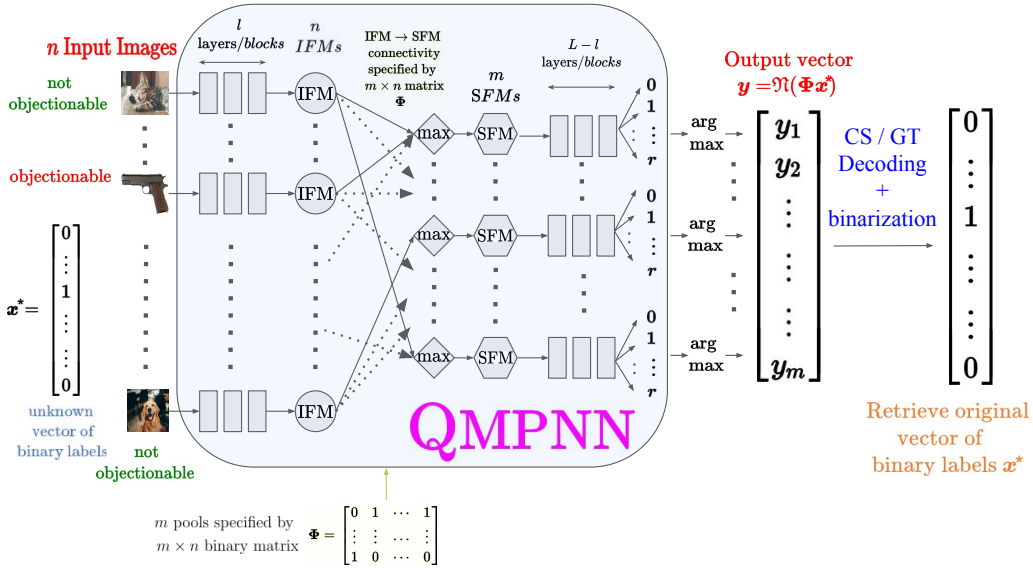


Figure 3.1: Main Components of our Image Moderation Engine for Classification (see also Alg. 3.1), which takes n input images and combines them into $m < n$ pools using a binary pooling matrix $\mathbf{\Phi}$ such that $\Phi_{ij} = 1$ if the jth image takes part in the ith pool and $\Phi_{ij} = 0$ otherwise. The QMPNN is trained to predict the number of objectionable images in each pool, producing vector $\boldsymbol{y} \in \mathbb{Z}_+^m$. For details of QMPNN, IFM, SFM, refer to Sec. 3.3.1. The IFM is the output of first l layers of the QMPNN, and the SFM for the ith is the entry-wise maximum of the IFM of images belonging to that pool. Given $\boldsymbol{y}, \mathbf{\Phi}$, a CS decoding step with suitable binarization obtains the status (objectionable or not) of each of the n images.

in [42]. Thus, methods to reduce the heavy load on moderation servers are the need of the hour. Besides reduction in power consumption, it is also helpful if the computational cost for image moderation engines is reduced significantly.

In this part of the thesis, we present a CS approach to speed up image moderation. Consider a set of n images, each of which may independently have objectionable content with a small probability p, called the *prevalence rate* of objectionable images (henceforth referred to as OIs). A small value for p is justified by independent reports from forums such as Facebook $(0.03 - 0.14\%)$ [43] or Reddit (6%) [44], where millions of images are regularly uploaded and where image or content moderation is an important requirement. Instead of invoking a NN separately on each image to classify it as objectionable or non-objectionable, we introduce the *quantitative matrix-pooled neural network* (QMPNN), which takes in a specification of $m < n$ pools of images – each pool being a selection of r out of n images – and efficiently predicts the count of OIs in these pools. Similar to [45], the QMPNN runs the first few layers of the

NN on each image of a pool and computes their Intermediate Feature Maps (IFMs), computes the superposed feature map (SFM) of the pool from its constituent IFMs, and processes only the SFM using the remaining layers of the network (see Fig. 3.1 and the example in Fig. 3.6 in Appendix 3.B). However, unlike [45], the IFM for each image is computed only once, and all n images are processed in a single forward pass to produce the m pool outputs. Such a design ensures that the computational cost of running the network on the m pools is lower than that of running the network on the n images individually. The pool specification is given by an $m \times n$ binary pooling matrix. The output of the QMPNN is thus the product of the pooling matrix and the *sparse* unknown binary vector specifying whether each image was objectionable or not. This enables the usage of CS in the domain of image classification. We employ CS algorithms followed by *binarization* using a threshold pre-determined by a validation set, to determine the status of each of the n images from the QMPNN's results on the m pools. The CS algorithms run at very little additional computational cost as compared to the QMPNN.

The above method and the one used in [45] are applicable only to classification problems, where the objectionable images belong to a single or a limited number of classes. However, in many commonly occurring cases, the the objectionable images may belong to a large number, in fact even an unknown number of classes. For example, on a topical forum dedicated to sharing information about tennis, any image whose content is not related to tennis can be considered objectionable. Hence, this is an outlier detection problem, as opposed to a classification problem. We extend our earlier approach of combining NNs and CS algorithms to deal with this application via a novel technique.

Our work is inspired by that in [45] which was the first work to apply GT principles to the problem of binary classification in the presence of class imbalance. However, we significantly build up on their setup and technique, and make the following contributions (see also Sec. 3.3.5 for more details):

1. We show that the problem of classifying the original n images (as OI or not) from these m counts (each count belonging to a pool) can be framed as a CS problem with some additional constraints (Sec. 3.3.2). We employ various algorithms from the CS and GT literature for this purpose (Sec. 3.3.2). Via an efficient implementation of our network called QMPNN, we enable usage of pooling matrices in which one image can be a part of *many* pools, bringing in innate resilience to any errors in the network outputs. As

opposed to this, in the approach in [45], each new pool that an image contributes to, incurs an additional cost, which makes it impractical to have an image contribute to more than two pools.

2. We test our method for a wide range of prevalence rates p and show that it achieves significant reduction in computation cost (number of floating point operations or the computation time) compared to individual moderation of images, while remaining competitive in terms of classification accuracy despite noisy outputs by the pooled NN (Sec. 3.4).

3. We show that our method outperforms binary GT methods, as used in [45], for high p, where quantitative information is more important due to a higher probability of more than one objectionable images being part of the same pool (Sec. 3.4).

4. Our decoding algorithms are designed to be noise-tolerant, as opposed to existing GT algorithms used for this problem which do not handle the case of a pool falsely testing negative for the presence of an objectionable image as done in [45] (also see Sec. 3.4).

5. Finally, we present a novel *pooled deep outlier detection* method for computationally efficient automatic identification of off-topic images on internet forums such as Reddit (Sec. 3.3.4). Moderation of such off-topic content is currently either done manually or via text-based tools [47]. Note that off-topic images do not belong to a set of pre-defined classes, making this problem different from the one where OIs belonged to a single class. To the best of our knowledge, the approach proposed in this work is the first one in the literature to present this problem in the context of CS or GT.

In this work, we combine the capabilities of NNs and CS algorithms in a specific manner. NNs have been used in recent times for end-to-end image recovery from CS measurements [62] with excellent results. However, in this paper, we are using NNs for either a noisy classification or a noisy outlier detection task, retaining the usage of classical CS or GT algorithms.

This chapter is organized as follows. A brief background of group testing is presented in Sec. 3.2. The main approach is presented in Sec. 3.3: the pooled NN architectures are described in Sec. 3.3.1, and the decoding algorithms are presented in Sec. 3.3.2. A non-trivial extension of the technique to pooled outlier detection is presented in Sec. 3.3.4. Comparisons to related work are presented in Sec. 3.3.5. Experimental results are for the one-class OI and multi-class

Table 3.1: List of Abbreviations

List of Abbreviations	
BMPNN	Binary Matrix-Pooled Neural Network
BPNN	Binary Pooled Neural Network
CLASSO	Constrained Least Absolute Shrinkage and Selection Operator
COMP	Combinatorial Orthogonal Matching Pursuit
CS	Compressed Sensing
FLOPs	Floating Point Operations
GFLOPs	Giga Floating Point Operations
GMM	Gaussian Mixture Model
GT	Group Testing
IFM	Intermediate Feature Map
INN	Individual Neural Network
KFLOPs	Kilo Floating Point Operations
MFLOPs	Mega Floating Point Operations
MIP	Mixed Integer Programming Method
NCOMP	Noisy Combinatorial Orthogonal Matching Pursuit
NLPD	Negative Log Probability Density
NN	Neural Network
OD	Outlier Detection
OI	Objectionable Image
QGT	Quantitative Group Testing
QMPNN	Quantitative Matrix-Pooled Neural Network
QPNN	Quantitative Pooled Neural Network
SFM	Superposed Feature Map

OI are presented in Sec. 3.4.1 and Sec. 3.4.2 respectively. We conclude in Sec. 3.5. A list of useful abbreviations is presented in Table 3.1.

Author Contributions: The work in this chappter has been done under the guidance of Prof. Ajit Rajwade, in collaboration with one more researcher. The contents of this chapter will be submitted to a reputed peer-reviewed journal for publication. A preprint version is available at [17]. The author of this thesis has made the following contributions to this work: conceptualization and implementation of the problem and the basic framework for the classification setting, formulation of, partial implementation of and guidance on the methods for the outlier detection setting, proof of concept and experiments with the IMFDB dataset for classification, instrumentation for the experiments, guidance for experimentation with the remaining datasets.

3.2 Group Testing Background

We first define some important terminology from the existing literature, which will be used in this chapter. In **binary group testing**, there are n items, of which $k \ll n$ unknown items are defective. Instead of individually testing each item, a pooled test (or group test) is performed on a pool/group of items to determine whether there exists at least one defective item in the pool, in which case the test is said to be 'positive' (otherwise 'negative'). The goal of group testing is to determine which k items are defective with as few tests as possible. In **quantitative group testing**, pooled tests give the *count* of the number of defective items in the pool being tested. In **non-adaptive group testing**, the number of tests and the pool memberships do not depend on the result of any test, and hence all tests can be performed in parallel. In **adaptive group testing**, the tests are divided into two or more rounds, such that the pool memberships for tests belonging to round $t_r + 1$ depend on the test outcomes from rounds $[t_r] \triangleq \{1, \cdots, t_r\}$. **Dorfman Testing** is a popular 2-round binary, adaptive algorithm [28], widely used in COVID-19 testing. In round 1 of Dorfman testing, the n items are randomly assigned to one out of n/g groups, each of size g. All items which are part of a negative pool are declared non-defective. All items which are part of a positive pool are then tested individually in round 2. The optimal pool size g which minimizes the number of tests in the worst case is $\sqrt{n/k}$ and the number of tests is at most $2\sqrt{nk}$ [28].

A **pooling matrix** $\mathbf{\Phi}$ is a $m \times n$ binary matrix, where $\mathbf{\Phi}_{ij} = 1$ indicates that item j was tested as part of pool i and $\mathbf{\Phi}_{ij} = 0$ otherwise. Let $\boldsymbol{x}$ be the unknown binary vector with $x_j = 1$ if item j is defective, 0 otherwise. Then the outcome of the pooled tests in binary GT may be represented by the binary vector $\boldsymbol{y}$, such that $y_i = \bigvee_{i=1}^{n} \mathbf{\Phi}_{ij} \wedge x_j$ where $\vee$ and $\wedge$ denote the Boolean OR and AND operations respectively. The outcome of pooled tests in quantitative GT may be represented by the integer-valued vector $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}$ obtained by multiplication of the matrix $\mathbf{\Phi}$ with the vector $\boldsymbol{x}$. In both binary and quantitative GT, a **decoding algorithm** recovers $\boldsymbol{x}$ from known $\boldsymbol{y}$ and $\mathbf{\Phi}$. GT is termed **noisy** or **noiseless** depending on whether or not the test results $\boldsymbol{y}$ contain noise.

3.3 Main Approach

Algorithm 3.1 Pseudocode for Pooled Classification using CS/non-adaptive GT

Input: set of n images $\mathcal{I}$, a trained QMPNN or BMPNN (see abbreviations in Table 3.1 and Sec. 3.3.1) parameterized by $\tilde{\mathbf{\Theta}}$ or $\mathbf{\Theta}$, $m \times n$ pooling matrix $\mathbf{\Phi}$, CS/non-adaptive GT decoding algorithm $\mathcal{F}$

Output: binary vector $\boldsymbol{x}$ representing whether each of the images is OI or not

1: If $\mathcal{F}$ is not a binary GT algorithm, run the QMPNN on the n images and obtain the predicted vector $\boldsymbol{y}$ of the number of OIs in each pool:

$y_q \leftarrow \underset{s \in \{0 \ldots r\}}{\arg\max}\, \texttt{QMPNN}_{\tilde{\mathbf{\Theta}}}(\mathcal{I}, \mathbf{\Phi})(q, s)\ \forall q \in \{1 \ldots m\}$,

2: Otherwise, run the BMPNN on the n images and obtain the predicted vector $\bar{\boldsymbol{y}}$ indicating whether each pool contains an OI or not:

$\bar{y}_q \leftarrow \underset{s \in \{0,1\}}{\arg\max}\, \texttt{BMPNN}_{\mathbf{\Theta}}(\mathcal{I}, \mathbf{\Phi})(q, s)\ \forall q \in \{1 \ldots m\}$,

3: Decode $\boldsymbol{y}$ or $\bar{\boldsymbol{y}}$ using the CS/non-adaptive GT algorithm $\mathcal{F}$ and obtain $\boldsymbol{x} \leftarrow \mathcal{F}(\boldsymbol{y}, \mathbf{\Phi})$ from the QMPNN or $\boldsymbol{x} \leftarrow \mathcal{F}(\bar{\boldsymbol{y}}, \mathbf{\Phi})$ from the BPNN.

4: **return** $\boldsymbol{x}$

Fig. 3.1 shows the main components of our approach to image moderation (see also pseudocode in Alg. 3.1). Consider a set of n images, $\mathcal{I} = \{I_1, \ldots, I_n\}$, and an unknown n-dimensional binary vector $\boldsymbol{x}$, with $x_i = 1$ if the i^{th} image has objectionable content and $x_i = 0$ otherwise. The binary *pooling matrix* $\mathbf{\Phi}$ of dimensions $m \times n$ specifies the m image pools $\{P_1, \ldots, P_m\}$, to be created from the images in $\mathcal{I}$. Each row of $\mathbf{\Phi}$ has sum equal to r, which

means that each pool has r images. The pooling matrix $\mathbf{\Phi}$ and the n images in $\mathcal{I}$ are passed as input to a so-called *quantitative matrix-pooled neural network* (QMPNN, see Sec. 3.3.1) which is specifically trained to output the number of OIs in each pool in the form of a m-element vector $\boldsymbol{y}$. If the output of the QMPNN is perfect, then clearly, for all $q \in \{1, \ldots, m\}$, $y_q = \sum_{I_i \in P_q} \mathbb{1}(x_i = 1) = \sum_{i=1}^{n} \mathbf{\Phi}_{qi} x_i$, which gives us $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}$ just as in Sec. 3.2. Hence, recovery of the unknown vector of classifications $\boldsymbol{x}$ given the output $\boldsymbol{y}$ of QMPNN (the vector of m different quantitative group tests) and the pooling matrix $\mathbf{\Phi}$ can be framed as a CS problem. Such an approach is non-adaptive because the pool memberships for each image are decided beforehand, independent of the QMPNN outputs. Furthermore, comparing with Eqn. 3.1, we see that this CS problem has boolean constraints on each x_i, and with $y_q \in \{0, \ldots, r\}$ for each q. In general the QMPNN will not be perfect in reporting the number of OIs, hence the $\boldsymbol{y}$ vector predicted will be noisy. This is represented as:

$$\boldsymbol{y} = \mathfrak{N}(\mathbf{\Phi}\boldsymbol{x}), \tag{3.3}$$

where $\mathfrak{N}(.)$ is a noise operator, which may not necessarily be additive or signal-independent unlike the case in Eqn. 3.1. However, we experimentally find that CS algorithms coupled with appropriate binarization techniques, which are specifically designed for signal-independent additive noise, are effective even in the case of noise in the outputs of the QMPNN.

3.3.1 Pooled Neural Network for Classification

The idea of a pooled NN was first proposed in [45]. We make two key changes to their design – (1) incorporate the pooling scheme *within* the NN which enables efficient non-adaptive (one-round) testing, and (2) have quantitative outputs instead of binary, which enables CS decoding. We discuss the significance of our architectural changes in detail in Sec. 3.3.5. Below, we describe the pooled NN architecture from [45] and then our network models.

3.3.1.1 Neural Group Testing [45] architecture

Consider a feed-forward deep NN which takes as input a single image for classifying it as objectionable or not. We term such a network an *individual neural network*, or INN. Furthermore, let this NN have the property that it may be decomposed into L blocks parameterized by the (mutually disjoint) sets of parameters $\theta_j, j \in \{1, \ldots, L\}$, with $\mathbf{\Theta} \triangleq \bigcup_{j=1}^{L} \theta_j$, so that for any input image I,

$$\texttt{INN}_{\mathbf{\Theta}}(I) \triangleq f_{\theta_L}^{(L)} \circ f_{\theta_{L-1}}^{(L-1)} \circ \cdots \circ f_{\theta_1}^{(1)}(I), \tag{3.4}$$

where $\circ$ represents function composition, and where each block computes a function $f_{\theta_j}^{(j)}, j \in \{1, \ldots, L\}$. Furthermore, the L^{th} block is a linear layer with bias, and has two outputs which are interpreted as the unnormalized log probability of the image being objectionable or not. We derive *pooled neural networks* based on an INN parameterized by $\mathbf{\Theta}$, and which take as input more than one image.

The output of the first $l < L$ blocks of the INN on an input image I is called its *intermediate feature-map* (IFM), with

$$\texttt{IFM}_{\mathbf{\Theta}}(I) = f_{\theta_l}^{(l)} \circ \cdots \circ f_{\theta_1}^{(1)}(I). \tag{3.5}$$

A *superposed feature map* (SFM, known as 'aggregated features' in [45]) of a pool of r images $P \triangleq \{I_1, I_2, \ldots, I_r\}$ is obtained by taking an entry-wise max over the feature-maps of individual images, i.e.

$$\texttt{SFM}_{\mathbf{\Theta}}(P)(.) \triangleq \max_{I \in P} \texttt{IFM}_{\mathbf{\Theta}}(I)(.). \tag{3.6}$$

In a *binary pooled neural network* (BPNN), the SFM of a pool of images P is processed by the remaining $L - l$ blocks of the INN, i.e., $\texttt{BPNN}_{\mathbf{\Theta}}(P) \triangleq f_{\theta_L}^{(L)} \circ f_{\theta_{L-1}}^{(L-1)} \circ \cdots \circ f_{\theta_{l+1}}^{(l+1)} \circ \texttt{SFM}_{\mathbf{\Theta}}(P)$. The two outputs of a BPNN are interpreted as the unnormalized log probability of the pool P containing an image belonging to the OI class or not. A BPNN has the same number of parameters as the corresponding INN. This is referred to as 'Design 2' in [45].

3.3.1.2 Our Modifications

In this work, we introduce a quantitative matrix-pooled neural network (QMPNN). First, we define a *quantitative pooled neural network* (QPNN), which is the same as a BPNN except that it has $r+1$ outputs instead of 2. Due to this there are more parameters in the last linear layer (the L^{th} block). Let these parameters be denoted by $\tilde{\theta}_L$, with the L^{th} block computing a function $f^{(L)}_{\tilde{\theta}_L}$, and

$$\texttt{QPNN}_{\tilde{\mathbf{\Theta}}}(P) \triangleq f^{(L)}_{\tilde{\theta}_L} \circ \ldots f^{(l+1)}_{\theta_{l+1}} \circ \texttt{SFM}_{\mathbf{\Theta}}(P), \tag{3.7}$$

with $\tilde{\mathbf{\Theta}} \triangleq \bigcup_{j=1}^{L-1} \theta_j \cup \tilde{\theta}_L$. The $r+1$ outputs of a QPNN are interpreted as the unnormalized log probabilities of the pool P containing 0 through r OIs. Finally, we introduce the *quantitative matrix-pooled neural network* (QMPNN), which takes as input a set of n images $\mathcal{I} \triangleq \{I_1, \ldots, I_n\}$, a specification of m pools containing r out of n images in each pool via a binary matrix $\mathbf{\Phi}$, and outputs $r+1$ real numbers for each of the m pools, in a single forward pass. That is,

$$\begin{aligned} \texttt{QMPNN}_{\tilde{\mathbf{\Theta}}}(\mathcal{I}, \mathbf{\Phi}) &\triangleq (\texttt{QPNN}_{\tilde{\mathbf{\Theta}}}(P_1), \ldots, \texttt{QPNN}_{\tilde{\mathbf{\Theta}}}(P_m)) \\ \text{where } P_q &\triangleq \{I_i | \Phi_{qi} = 1, I_i \in \mathcal{I}\} \; \forall q \in \{1, \ldots, m\}. \end{aligned} \tag{3.8}$$

For a pool q, the s^{th} output ($s \in \{0, \ldots, r\}$) is interpreted as the unnormalized log probability of the pool q containing s images belonging to the OI class. *As illustrated in Fig. 3.1, the IFM for each of the n images is computed only once by the* QMPNN*, and is re-used for each pool that it takes part in.* An alternative implementation, wherein the IFM for an image is re-computed for each pool that it is a part of (such as in the implementation of 'Algorithm 3' of [45]) would be inefficient, and non-adaptive group testing or compressed sensing with such an implementation would not be viable. For efficiency, all of the QMPNN outputs are computed in a single forward pass on the GPU. In particular, all the SFMs for the pools specified by the matrix $\mathbf{\Phi}$ are computed in a single forward pass on the GPU. Otherwise, some IFMs would need to either be re-computed or cached, which would require the transfer of large amounts of data between the GPU memory and the system memory.

We note that while the first l blocks of the NN are run for each of the n images, the remaining $L-l$ blocks are only run for each of m SFMs. However, if an individual neural network (INN) is run on the n images, then both the first l blocks and the last $L-l$ blocks will be run n times. Hence, since $m < n$, the QMPNN requires significantly less computation than running the INN on the n images, which we verify empirically in Sec. 3.4.

Analogous to the QMPNN, we also introduce a *binary matrix-pooled neural network* (BMPNN) for efficient non-adaptive binary group testing, which is exactly the same as a QMPNN except that each pool has only two outputs.

Training: Let $\mathcal{D}$ be a dataset of images labelled as OI or non-OI. A *pooled dataset* $\mathcal{D}_{\text{pooled}}$ is obtained from $\mathcal{D}$ in the following manner: each entry of $\mathcal{D}_{\text{pooled}}$ is a pool of r images, and has a label s in $\{0,\ldots,r\}$, equal to the number of images in that pool which belong to the OI class. The s OIs and the $r-s$ non-OIs in each pool are chosen uniformly at random from $\mathcal{D}$. A total of $N = N_0 + \cdots + N_r$ pools each of type $0 \ldots r$ respectively are created. Due to the rare occurrence of OIs at test time, the fraction of pools which contain s images at test time will be small for large values of s. Hence we maintain such imbalance while creating the pooled training dataset $\mathcal{D}_{\text{pooled}}$, with $N_0 \geq \cdots \geq N_r$. The parameters $\tilde{\mathbf{\Theta}}$ of a QPNN are trained via supervised learning on $\mathcal{D}_{\text{pooled}}$ using the cross-entropy loss function, $L(\tilde{\mathbf{\Theta}}) = \underset{(P,c)\sim\mathcal{D}_{\text{pooled}}}{E}[-\log \sigma(\texttt{QPNN}_{\tilde{\mathbf{\Theta}}}(P))_c]$, where $\sigma(.)$ is the softmax function, given by $\sigma(\boldsymbol{z})_s \triangleq \frac{e^{z_s}}{\sum_{t=0}^{r} e^{z_t}}$. The parameters Θ of a BPNN are trained similarly, except that each training pool has a label in $\{0,1\}$, with 1 indicating pools which contain at least one OI.

Inference: The trained parameters $\tilde{\mathbf{\Theta}}$, and a pooling matrix $\mathbf{\Phi}$ are then used to instantiate a QMPNN. The entries of the vector $\boldsymbol{y}$, containing pool-level predictions of the count of OIs in each pool, are obtained from a QMPNN as:

$$y_q = \underset{s\in\{0,1,\ldots,r\}}{\arg\max}\ \texttt{QMPNN}_{\tilde{\mathbf{\Theta}}}(\mathcal{I},\mathbf{\Phi})(q,s)\ \forall q \in \{1,\ldots,m\}. \tag{3.9}$$

In Sec. 3.3.2, we present CS algorithms to decode this vector $\boldsymbol{y}$ to recover the vector $\boldsymbol{x}$ for classification of each image in $\mathcal{I}$. Similarly, a BMPNN may be instantiated using the trained parameters $\mathbf{\Theta}$ of a BPNN, and its prediction vector – representing whether each pool contains an OI or not – can be decoded by the non-adaptive binary GT decoding algorithms presented in Sec. 3.3.2.

3.3.2 CS/GT Decoding Algorithms

Given Eqn. 3.3, the main aim is to recover $\boldsymbol{x}$ from $\boldsymbol{y}$ (the prediction of the QMPNN, Eqn. 3.9) and $\boldsymbol{\Phi}$. For this, we propose to employ the following two algorithms:

1. Mixed Integer Programming Method (MIP): We minimize the objective in Eqn. 3.2 with boolean constraints on entries of $\boldsymbol{x}$, i.e. $\forall i \in [n], x_i \in \{0, 1\}$, i.e., we employ the following estimator:

$$\hat{\boldsymbol{x}} \triangleq \underset{\boldsymbol{x}}{\text{argmin}} \ \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \ \text{s.t.} \ \forall i \in [n], x_i \in \{0, 1\}. \tag{3.10}$$

   We use the CVXPY [115] package with the Gurobi solver, which uses a branch and bound method for optimization [116, 117].

2. Constrained LASSO (CLASSO): This is a variant of the LASSO from Eqn. 3.2, with the values in $\boldsymbol{x}$ constrained to lie in $[0, 1]$, yielding the following estimator:

$$\begin{aligned} &\bar{\boldsymbol{x}} \triangleq \underset{\boldsymbol{x}}{\text{argmin}} \ \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_1, \ \text{s.t.} \ \forall i \in [n], x_i \in [0, 1] \\ &\hat{\boldsymbol{x}} = [\hat{x}_1 \dots \hat{x}_n]^T \text{ where } \forall i \in [n], \ \hat{x}_i = \begin{cases} 1 & \text{if } \bar{x}_i \geq \tau \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.11}$$

   The relaxation from $\{0, 1\}$ to $[0, 1]$ is done for computational efficiency. The final estimate $\hat{\boldsymbol{x}}$ regarding whether each of the images is an OI is obtained by thresholding $\bar{\boldsymbol{x}}$, i.e. $\forall i \in [n]$, I_i is considered to be an OI if $\bar{x}_i \geq \tau$, where τ is the threshold.

The hyperparameters in our algorithms (λ for MIP, and λ and τ for CLASSO) are chosen via grid search by maximizing the product of specificity and sensitivity (both defined in Sec. 3.4) on a representative validation set of images.

We also compare MIP and CLASSO with some popular algorithms from the binary GT literature, which act as a baseline. For the non-adaptive methods, the binary vector $\bar{\boldsymbol{y}}$ containing predictions of a BMPNN is provided to the decoding algorithms. The algorithms are:

1. Dorfman Testing: See Sec. 3.2. This is referred to as Two-Round testing in [45, Algorithm 1]. In the first stage, a binary pooled neural network (BPNN) is used to classify a pool of images as containing at least one objectionable image or not. If the BPNN predicts that the pool contains an objectionable image, then each image is tested individually in a second round using an individual neural network (INN). Otherwise, all images in the pool are declared as not being objectionable, without testing them individually. Because of a second round of testing, Dorfman Testing is inherently resistant to a pool being falsely declared as positive, but not to a pool being falsely declared as negative.

2. Combinatorial Orthogonal Matching Pursuit (COMP) [29]: This is a simple decoding algorithm for noiseless non-adaptive binary GT, wherein any image which takes part in at least one pool which tests negative (i.e., no OIs in the pool) is declared as a non-OI, and the remaining image are declared as OIs. The One-Round Testing method in [45] uses COMP decoding, albeit with a different pooling matrix than ours.

3. Noisy COMP (NCOMP) [29]: a noise-resistant version of COMP. An image is declared to be OI if it takes part in strictly greater than t positive pools (or equivalently in strictly less than $c - t$ negative pools, where c is the number pools that each image takes part in).

3.3.3 Choice of Pooling Matrix

A good pooling matrix $\mathbf{\Phi}$ is crucial for the performance of these algorithms. We choose random binary matrices with the following constraints: (*i*) equal number of ones in each column, (*ii*) equal number of ones in each row, (*iii*) with the dot product of each pair of columns and each pair of rows at most 1. We argue that such a matrix obeys three key properties of a good sensing matrix from the point of view of CS/GT theory: (1) k-disjunctness, (2) low mutual coherence, and (3) RIP-1 (ℓ_1 restricted isometry property) with high probability. The definitions of these properties and arguments that our matrix satisfies these properties are presented in Appendix 3.A.

3.3.4 Pooled Deep Outlier Detection

Algorithm 3.2 Pseudocode for Pooled Outlier Detection using CS/non-adaptive GT

Inputs: set of n images $\mathcal{I}$, trained QMPNN parameterized by $\tilde{\mathbf{\Theta}}$, $m \times n$ pooling matrix $\mathbf{\Phi}$,
GMM G fit to feature vectors of training pools with no off-topic images,
histogram of training pool anomaly scores H_G,
CS/non-adaptive GT decoding algorithm $\mathcal{F}$

Output: binary vector $\boldsymbol{x}$ representing whether each of the images are off-topic or on-topic

1: Run the QMPNN with pooling matrix $\mathbf{\Phi}$ on the n images and obtain feature vectors $\{\phi(P_1), \ldots, \phi(P_m)\}$ for the m pools $P_1, \ldots, P_m$ specified by $\mathbf{\Phi}$.
2: Use the GMM G to obtain the vector of anomaly scores for the m pools,
$S := \{\texttt{NLPD}_G(\phi(P_1)), \ldots, \texttt{NLPD}_G(\phi(P_m))\}$
3: Find the bin indices of the anomaly scores in S in the histogram H_G and use Eqn. 3.14 to obtain each entry of the predicted vector $\boldsymbol{y}$ of the number of off-topic images in the pools
4: If $\mathcal{F}$ is a binary GT algorithm, binarize $\boldsymbol{y}$ by setting each non-zero entry to 1
5: Decode $\boldsymbol{y}$ using the CS/non-adaptive GT algorithm and obtain $\boldsymbol{x} \leftarrow \mathcal{F}(\boldsymbol{y}, \mathbf{\Phi})$
6: **return** $\boldsymbol{x}$

Here, we consider the case where the set of allowed images belong to one underlying class, and any image not belonging to that class is considered 'off-topic'. Such a situation arises in the moderation of topical communities on online forums such as Reddit (e.g. see [47]). For example, an image of a baseball game is off-topic on a forum meant for sharing of tennis-related images. As the off-topic images could belong to an unbounded variety of classes, instances of all or some of which may not be available for 'training', *classification* based approaches (such as the QMPNN from Sec. 3.3.1, and [45]) may not be directly applicable, and instead, *deep outlier detection* based methods are more suitable.

Recent deep outlier detection methods [118, 119] approximate a class of interest, represented by suitable feature vectors in $\mathbb{R}^d$, using a high dimensional Gaussian distribution characterized by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and a covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$. Then the Mahalanobis distance between the feature vectors of a test image and the mean vector $\boldsymbol{\mu}$ is computed. This Mahalanobis distance acts as the anomaly score to perform outlier detection, i.e. test feature vectors with a Mahalanobis distance greater than some threshold (say $\bar{\tau}_{md}$) are considered out-

Algorithm 3.3 Pseudocode for Dorfman Pooled Outlier Detection

Inputs: Pool $P := \{I_1, \ldots, I_r\}$ of r images, trained QPNN parameterized by $\tilde{\boldsymbol{\Theta}}_{\mathbf{1}}$, trained INN parameterized by $\boldsymbol{\Theta}_{\mathbf{2}}$, GMM G_1 fit to feature vectors of training image pools with no off-topic images, Histogram H_{G_1} of anomaly scores of training image pools, GMM G_2 fit to feature vectors of on-topic training images, Histogram H_{G_2} of anomaly scores of training images,

Output: binary vector $\boldsymbol{x}$ representing whether each of the images are off-topic or on-topic

1: Run the QPNN on pool P and obtain its feature vector $\phi(P)$

2: Use the GMM G_1 to obtain the anomaly score for the pool, $\texttt{NLPD}_{G_1}(\phi(P))$,

3: Find the bin index of the anomaly score $\texttt{NLPD}_{G_1}(\phi(P))$ in the histogram H_{G_1} and use Eqn. 3.14 to obtain the number of off-topic images in pool P, $L(P)$

4: If $L(P)$ is 0, output is vector of all zeros, i.e., $\boldsymbol{x} \leftarrow \mathbf{0}$ and **return** $\boldsymbol{x}$

5: Otherwise, run the INN on each image $\{I_1, \ldots, I_r\}$ to obtain r image feature vectors, $\{\phi(I_1), \ldots, \phi(I_r)\}$

6: Obtain the vector of anomaly scores of the images using the GMM G_2, $S := \{\texttt{NLPD}_{G_2}(\phi(I_1), \ldots, \texttt{NLPD}_{G_2}(\phi(I_r)\}$

7: Find the bin indices of the anomaly scores in S in the histogram H_{G_2} and obtain the set of labels of each image $\{L(I_1), \ldots, L(I_r)\}$, similar to Eqn. 3.14

8: Set $\boldsymbol{x} \leftarrow (L(I_1), \ldots, L(I_r))$ and **return** $\boldsymbol{x}$

liers. The feature vectors can be represented by the outputs of a suitably trained NN. One may think of using such an approach with a pooled NN to detect pools containing outlier images. However a single Gaussian distribution is often inadequate to represent a sufficiently diverse class, and hence we resort to using a Gaussian Mixture Model (GMM). We put forward a novel approach which uses a pooled NN combined with a GMM, to detect anomalous pools, i.e., pools which contain at least one off-topic image, which we term *pooled deep outlier detection*. This enables us to use binary GT methods from Sec. 3.3.2 - such as COMP, NCOMP and Dorfman Testing - for pooled outlier detection. Furthermore, we also detect the number of outlier images in a pool, thus enabling CS methods to be used for pooled outlier detection. To the best of our knowledge, this is the first such work in the literature, which enables GT and CS methods to be used for outlier detection.

We consider the setting where for training we have available a set of images which are either on-topic or are off-topic images from known classes, but at test time the off-topic images may be from unknown classes. We train a QPNN using the method in Sec. 3.3.1.2, with the label $L(P)$ for any training pool P set to the number of off-topic images in it. The output of the last-but-one layer of the QPNN (i.e. the $(L-1)^{\text{th}}$ layer – see Sec. 3.3.1) is considered to be the feature vector $\phi(P)$ for the pool of images P which is input to it. First, we create M pools from the on-topic training data images (i.e. these pools have 0 off-topic images), and fit a GMM G with some K clusters to the feature vector $\phi(.)$ of all these pools, using the well-known expectation-maximization (EM) algorithm. The optimal value of K is chosen via cross-validation, by selecting the K with the maximum likelihood given a held-out set of pools containing only on-topic images.

For any pool P, we use the negative log probability density of its feature vector $\texttt{NLPD}_G(\phi(P))$ (negative log probability density) under the GMM G as an anomaly score, given by:

$$\texttt{NLPD}_G(\phi(P)) := -\log\left(\sum_{j=1}^{K} p_j \mathcal{N}(\phi(P)|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})\right), \tag{3.12}$$

where $\{\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_K}\}$ are the K mean vectors, $\{\boldsymbol{\Sigma_1}, \boldsymbol{\Sigma_2}, \cdots, \boldsymbol{\Sigma_K}\}$ are the K covariance matrices and $\{p_1, p_2, \cdots, p_K\}$ are the K membership probabilities of the GMM G, and for any vector $\mathbf{z}$, $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})$ is its probability density given the multivariate normal distribution with

mean $\boldsymbol{\mu_j}$ and covariance matrix $\boldsymbol{\Sigma_j}$.

It is intuitive to imagine that $\texttt{NLPD}_G(\phi(P))$ values would generally be larger for pools containing a larger number of outlier images ($L(P)$), and we should be able to predict $L(P)$ given $\texttt{NLPD}_G(\phi(P))$. The main idea is to create a histogram of anomaly scores of pools with different number of off-topic images in them, and assign to a new pool the label which is most common in its bin. This is accomplished as follows. From the *validation* set of images, some N pools $\{P_i\}_{i=1}^N$ are created, each containing r images. Since off-topic images are expected to be rare at test time, we only consider pools which contain upto some $t < r$ off-topic images. Thus $L(P) \in \{0, 1, \ldots, t\}$. For each pool P_i, the value $\texttt{NLPD}_G(\phi(P_i))$ is computed. These NLPD values are divided into some Q bins/intervals and an anomaly score histogram H_G is created in the following manner:

$$\begin{aligned} &\forall j \in \{1, 2, \cdots, Q\}, \forall l \in \{0, 1, \cdots, t\}, \\ &H_G(j, l) = \frac{\#\text{pools in bin } j \text{ containing } l \text{ outlier images}}{N}. \end{aligned} \tag{3.13}$$

That is, $H_G(j, l)$ represents the fraction of pools from $\{P_i\}_{i=1}^N$ which contain l outlier images and whose NLPD value falls into the j^{th} bin. This histogram H_G is created at the time of training.

Each non-empty bin j is assigned a label $\mathcal{L}(j)$ equal to $\text{argmax}_{l \in \{0,1,\cdots,t\}} H_G(j, l)$, i.e., the label with the most number of pools in that bin. An empty bin is assigned the label of its nearest non-empty bin. If there is more than one nearest non-empty bin, then the bin is assigned the label of the bin with the larger index. At test time, given the feature vectors $\phi(P_{\text{test}})$ of a pool P_{test} created from some r test images, first its anomaly score $\texttt{NLPD}_G(\phi(P_{\text{test}}))$ is computed using the GMM. If the anomaly score lies in bin j_{test}, then P_{test} is assigned the label $\mathcal{L}(j_{\text{test}})$. If the anomaly score is greater than the right bound (s_{max}) of the rightmost bin, then it is assigned the label t. If the anomaly score is less than the left bound (s_{min}) of the leftmost bin, then it is

assigned the label 0. That is,

$$L(P_{\text{test}}) = \begin{cases} \mathcal{L}(j_{\text{test}}) & \text{if } \texttt{NLPD}_G(\phi(P_{\text{test}})) \in \text{ bin } j_{\text{test}} \\ 0 & \text{if } \texttt{NLPD}_G(\phi(P_{\text{test}})) < s_{\min} \\ t & \text{if } \texttt{NLPD}_G(\phi(P_{\text{test}})) > s_{\max} \end{cases} . \tag{3.14}$$

At the time of deployment, m pools are created from n images to be tested for off-topic content using a QMPNN and a pooling matrix $\mathbf{\Phi}$, and feature vectors of the m pools are obtained. For each pool, we determine the number of off-topic images it contains using the GMM and histogram-based method just described. Given these m numbers and the pooling matrix $\mathbf{\Phi}$, we predict whether or not each of the n images is off-topic using the CS and non-adaptive GT decoding algorithms described in Sec. 3.3.2. See Alg. 3.2 for a pseudocode.

We also perform Dorfman Testing for outlier detection. Feature vectors of a pool of r images are obtained by running a QPNN, and the GMM and histogram-based method is used to predict the number of off-topic images in the pool. If the pool is predicted to contain at least one off-topic image, then each image is individually tested for being off-topic using the same method, but using an individual neural network (INN). That is, the features of each image are obtained using the INN, and a previously trained GMM and histogram (of validation *image* anomaly scores) are used to label the images as off-topic or not. See Alg. 3.3 for a pseudocode.

3.3.5 Comparison with Related Work

There exists very little literature on the combination of GT algorithms with NNs. The sole published work on this topic (to our knowledge) can be found in [45]. The NNs in [45] use only binary outputs, while the NNs proposed in our methods have quantitative outputs, and are trained to predict the *number of OIs* in a given SFM. This enables usage of CS decoding algorithms. CS methods have better recovery guarantees than non-adaptive binary GT (see Appendix 3.A.3). In [45], non-adaptive pooling is implemented by *separately* running a BPNN for each pool, due to which the IFM for an image is re-computed for each pool that it takes part in, making it computationally inefficient. Due to this, the scheme in [45] is limited to

using pooling matrices in which each image takes part in at most two pools. Such matrices can be at most 1-disjunct (see Appendix 3.A.1 for the definition), and hence guarantees of exact recovery via COMP is for only 1 OI per matrix (see [120, Prop. 2.1] and Appendix 3.A.4 for an explanation). Thus their scheme is applicable for only very low prevalence rates (p) of OIs. In our implementation, an IFM for each image is computed only once, and is re-used for each pool that it takes part in. This enables usage of pooling matrices with 3 or more entries per column. A larger number of 1 entries per column are necessary for handling a higher number of defective items, since it enables high 'disjunctness' for GT decoding [121], or restricted isometery property (RIP) of high orders – see Appendix 3.A.1 for definitions of disjunctness and RIP, and Appendix 3.A.4 for an upper bound on the disjunctness of a matrix, given a fixed number of 1 entries per column. Hence our method is applicable for high values of p as well. Furthermore, there are no guarantees for binary GT for recovery from noisy pool observations using matrices which have only two 1 entries per column. This is because if one test involving an item is incorrect but the other one is correct, there is no way to determine which of them is the incorrect one. Noise-tolerant recovery algorithms for binary GT exists for suitable matrices with 3 or more 1 entries per column. For example, consider NCOMP with a pooling matrix with 3 tests per item and which has the properties such as disjunctness. When testing in the presence of upto one defective item, NCOMP can recover from an error in one test of that item by declaring it defective if two of the tests are positive, or non-defective if two of its tests are negative. Moreover, higher number of 1 entries per column implies better noise tolerance for CS algorithms for the matrices described in Appendix 3.A.5.

Over and above these differences, we also extend binary GT and CS to deep outlier detection as earlier described in Sec. 3.3.4, whereas the approach of [45] is limited to the classification setting. In the case of OIs with a single class, our test datasets also contain a much larger number of objectionable images (see details in Appendix 3.B), and these are mixed with non-OIs to form datasets of size 1M or 100K for a wide range of prevalence rates p, so that each OI gets tested many times in combination with other OIs and different non-OIs. Moreover in [45], prediction measures are presented only for a single dataset for the case when $p = 0.001$, with the number of unique OIs being only 50 and each OI being tested only once.

3.4 Experiments

We present an extensive set of experimental results, first focusing on the classification case in Sec. 3.4.1, and then on the outlier detection case in Sec. 3.3.4.

3.4.1 Image Moderation using Pooled Classification

Here the objectionable images belong to a single underlying class, and the goal is to classify each image as objectionable or not objectionable.

Tasks We use the following two tasks for evaluation: (*i*) firearms classification using the Internet Movie Firearms Database (IMFDB) [122], a popular dataset used for firearms classification [39]; and (*ii*) Knife classification using the Knives dataset [38], a dataset of CCTV images, labelled with the presence or absence of a handheld knife. In (*i*), the firearm images from IMFDB are mixed with non-firearm images from ImageNet-1K [123] as IMFDB contains only firearm images. The firearms and the knife images are respectively considered as objectionable images (OIs) in the two tasks.

Data Splits, Training Details, Data Transformations Details of the number of images in the training/validation/test splits in each dataset and the choice of various hyper-parameters for training are provided in Appendix 3.B.

Confusion Matrix As can be seen in Fig. 3.2 (Left), if g is the ground truth number of OIs, the predictions of the QMPNN lie in the interval $[\max(0, g-1), \min(g+1, r)]$ with high probability. This is presented as a confusion matrix of size $(r+1) \times (r+1)$ of the ground truth number of OIs in a pool versus the predicted number. In Fig. 3.2 (Left), we have $r = 8$.

Neural Network Implementation We use the official PyTorch implementation of ResNeXt-101 $(32 \times 8d)$[124, 125], a highly competitive architecture which has produced compelling results in many image classification tasks, with weights pre-trained on ImageNet-1K. The last

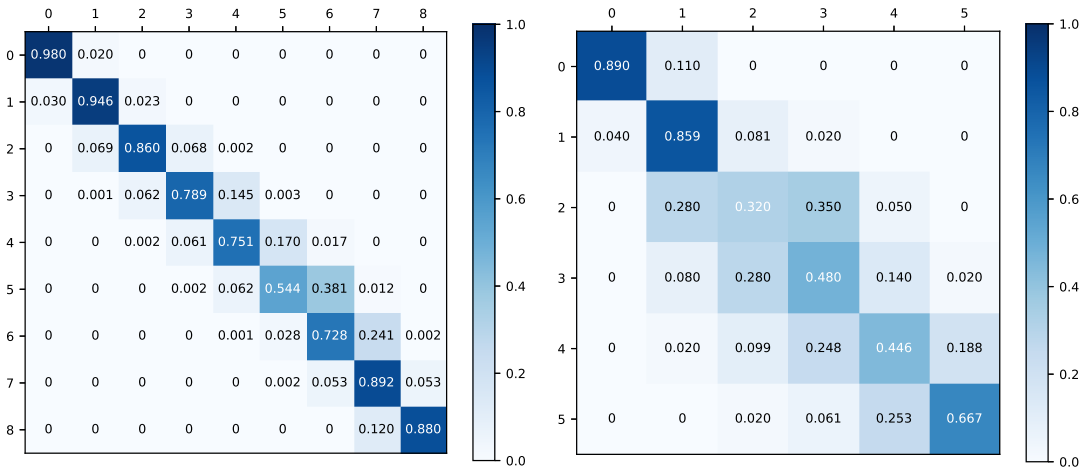


Figure 3.2: Pool-level Confusion Matrices for Classification (Left) and Pooled Outlier Detection (Right) on IMFDB pooled validation data : true #OIs (rows) versus predicted #OIs (columns). Observe that in most cases, the predicted and actual number of OIs are very close.

linear layer of this network has 1000 outputs by default. We replace the last layer to have only 9 outputs for the QPNN or QMPNN (since pool size $r = 8$), and 2 outputs for the BPNN or BMPNN (recall discussion in Sec. 3.3.1). Each image is passed through first three stages of the ResNeXt-101 to create IFMs, which are then combined to create SFMs for the pools in the QMPNN, QPNN, BMPNN, or BPNN. The remaining two stages of the ResNeXt-101 process only the SFMs in QMPNN, QPNN, BMPNN, or BPNN. This configuration is exactly the same as Design 2 in [45, Sec. II.A]. In initial experiments, we also tried using Design 3 of [45], but the accuracy of this configuration was not good on the IMFDB dataset. The networks are trained using the method specified in Sec. 3.3.1, with training data and epoch, learning rates etc as specified in Appendix 3.B.

Prediction We randomly sampled pN OIs and $(1-p)N$ non-OIs from the test split and shuffled these N images to create our test datasets. Here, $p \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1\}$ is the prevalence rate of OIs, and N is 1M for IMFDB and 100K for Knives. We use a 50×100 balanced pooling matrix $\mathbf{\Phi}$ with the construction as described in Appendix 3.A, with $r = 8$ ones per row and $c = 4$ ones per column. That is, 50 SFMs are created from 100 IFMs, with each SFM being created from 8 different IFMs, and each IFM contributing to 4 different SFMs. Each test set is divided into chunks of size 100 each and passed to the QMPNN to retrieve the prediction vector $\boldsymbol{y}$ (of length 50) for each chunk. As can be seen

in Fig. 3.2 (left), if g is the ground truth number of OIs, the predictions lie in the interval $[\max(0, g-1), \min(g+1, r)]$ with high probability. These pool-level predictions are decoded using the algorithms CLASSO and MIP. We compare them with the baselines of COMP, Dorfman Testing with pool-size 8 (DORFMAN-8, same as Design 2 + Algorithm 1 of [45]), NCOMP with $t = 2$ (NCOMP-2) and individual image testing (referred to as INDIVIDUAL) on images of the same size. The numerical comparison is based on the following two performance measures:

1. *Sensitivity* (also called *Recall* or True Positive rate) $\triangleq \dfrac{\#\text{correctly detected OIs}}{\#\text{actual OIs}}$
2. *Specificity* (also called True Negative Rate) $\triangleq \dfrac{\#\text{correctly detected non-OIs}}{\#\text{actual non-OIs}}$

For both the measures, larger values indicate better performance.

Discussion on Results Fig. 3.3 shows the sensitivity and specificity of each of these algorithms for different values of p, for the two classifications tasks (*i*) and (*ii*) defined earlier. We see that for IMFDB and Knives, the CS/GT methods remain competitive with INDIVIDUAL. In general, CLASSO and MIP have *much higher sensitivity* than COMP and DORFMAN-8. This is because if a pool gets falsely classified as negative (i.e. containing no OIs), then all OIs in that pool get classified as negative by COMP and DORFMAN. However, CLASSO and MIP can recover from such errors as they inherently consider the results of other pools that the OI takes part in. Such false negative pools are more common at lower p values, where most positive pools have only a single OI, which may remain undetected by the pooled NN (see also the confusion matrix of the pooled NN in Fig 3.2). NCOMP-2 improves upon the sensitivity of COMP and DORFMAN-8 by allowing for one false negative pool for each OI. However, it incurs a corresponding drop in specificity because non-OIs which take part in three positive pools incorrectly get classified as OI. When p is high, it is likely that such non-OIs are common, and hence the steep drop in specificity makes NCOMP-2 unviable. At high prevalence rates of OIs, it is common for a non-OI to test positive in all of its pools. Since COMP uses only such binary information, it incorrectly declares all such images as OIs, and we observe a sharp decline in specificity. However, CLASSO and MIP use the quantitative information of the number of OIs in the pools, and can eliminate such false positives. For example, if it is known that a positive pool has only two OIs, then in general only two images from that pool will be declared as being an OI, whereas COMP may declare anywhere from 1 through 8 positives for such a pool.

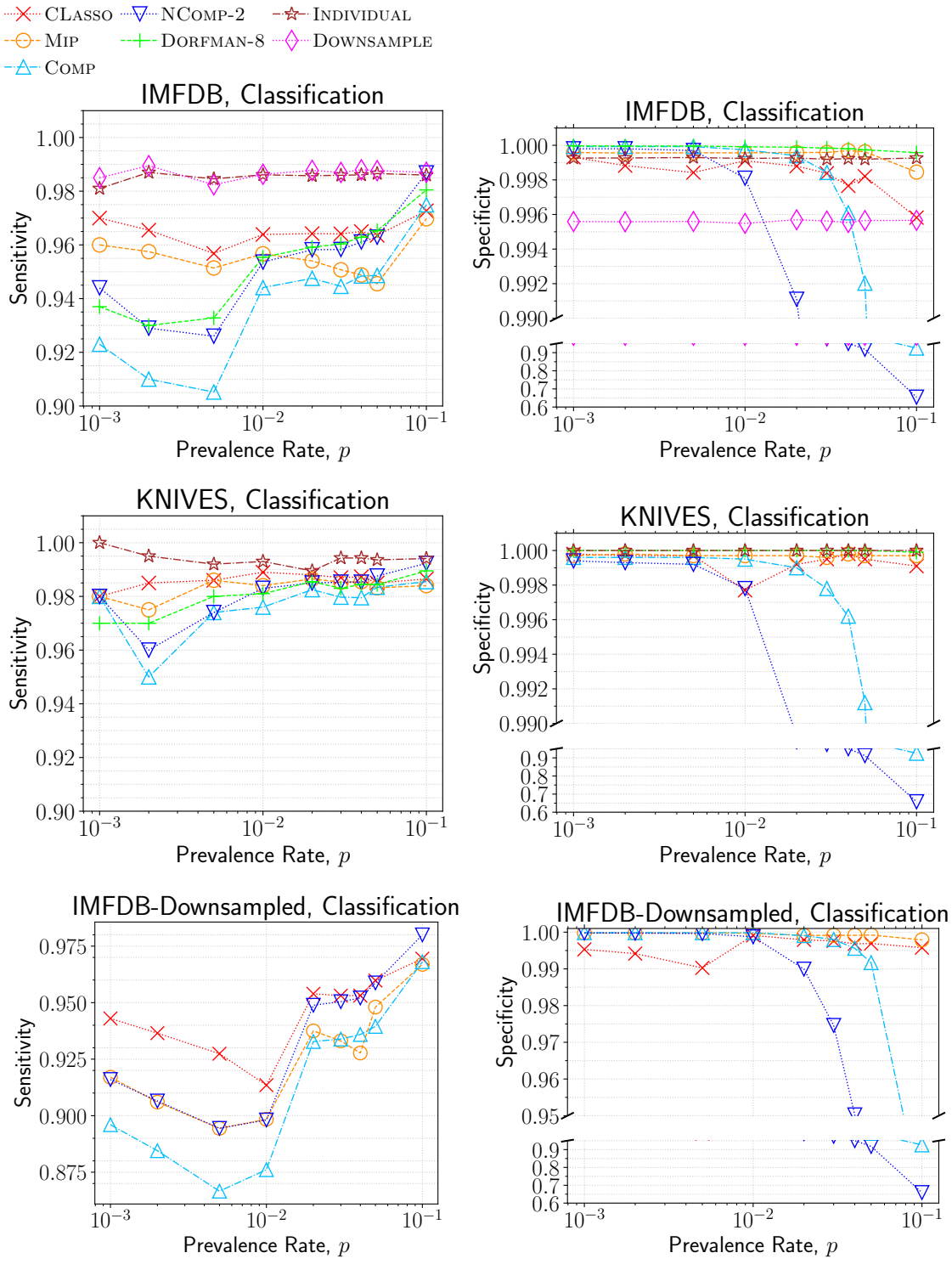


Figure 3.3: Performance of Image Moderation via Pooled Classification methods for various prevalence rates (p), over 1M images for IMFDB and 100K images for Knives. Sensitivity (left) and Specificity (right) on IMFDB (top row), Knives (middle row), and IMFDB-Downsampled (bottom row).

Among the CS algorithms, CLASSO has better sensitivity, while MIP has better specificity.

Comparison with Downsampling In Fig. 3.3, we also compare our method to individual testing of images downsampled by a factor of 4 (size 112×112) (referred to in Fig. 3.3 as DOWNSAMPLE) for the firearms task. We find that downsampling lowers the specificity of prediction of individual testing, and both MIP and CLASSO have better specificity for both IMFDB and Knives.

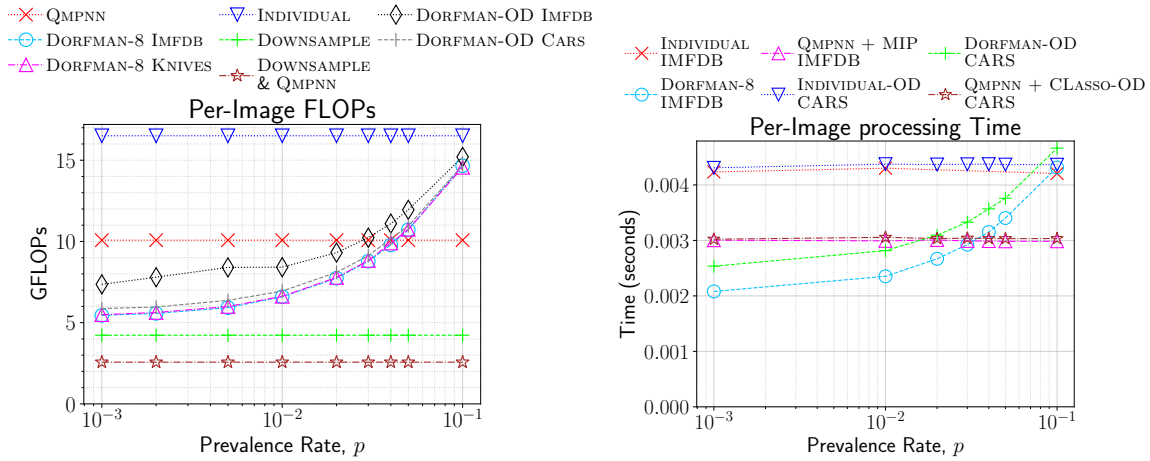


Figure 3.4: Image Moderation cost in (Left) giga floating point operations (GFLOPs), (Right) Computation Time (see Sec. 3.4.1 and Sec. 3.4.2, paragraphs titled 'Computational Cost'). GFLOPs for running non-adaptive or individual methods is the same for classification and outlier detection. GFLOPs for DORFMAN-8 and DORFMAN-8-OD and computation times for all methods averaged over 1M images for IMFDB and 100K images for Knives and Cars. Timing experiments performed with one NVIDIA GeForce RTX 2080 Ti GPU (11GB RAM) and one AMD Ryzen Threadripper 2920X 12-Core CPU, 64GB RAM. Batch-sizes chosen to fill GPU RAM.

Computational Cost Fig. 3.4 (Left) compares the average number of floating point operations (FLOPs) needed for processing an image by the QMPNN against those needed by DORFMAN-8 and INDIVIDUAL, for different prevalence rates on IMFDB. We used the Python package *ptflops* [1] for estimation of FLOPs used by the NNs. The FLOPs needed by the INN to process one image is 16.5 GFLOPs, whereas for QMPNN with 50×100 pooling matrix it is 10.1 GFLOPs, i.e. only 61% of the INN. While DORFMAN-8 uses less computation for small p, the QMPNN is significantly more efficient for $p > 0.05$.

[1] `https://github.com/sovrasov/flops-counter.pytorch`

In Fig. 3.4 (Right), we present the end-to-end wall-clock time (normalized by the number of images) for processing 1M images from IMFDB using QMPNN with MIP (our most compute-intensive method), for a range of prevalence rates. We see that this is significantly faster than individual testing of the images for all p, and for $p \geq 0.04$ than DORFMAN-8.

CS decoding time is much smaller than the overall time taken by the pipeline. The overall time taken by the QMPNN + MIP pipeline is around 3×10^{-3} seconds per image for various prevalence rates. CS decoding for $p = 0.1$ takes the most time – CLASSO took around 1.1×10^{-5} seconds per image and MIP around 1.6×10^{-4} seconds per image, using a parallelized implementation with 12 processes. An alternate implementation of CLASSO using FISTA (fast iterative shrinkage thresholding algorithm), which has a theoretically optimal rate of convergence [126], takes only 4.4×10^{-6} seconds per image. Even single-threaded implementations of our methods take only 9.6×10^{-5} seconds using the default CLASSO, 2.1×10^{-3} seconds using MIP, and 2.3×10^{-5} seconds using FISTA, per image. In general these times are between one to three orders of magnitude lower than the time taken by the QMPNN, except for the single-threaded version of MIP which takes around two-thirds of the time required by a QMPNN. Moreover, a sensible implementation of our pipeline will perform the CS decoding for one batch of images in parallel with the QMPNN decoding for a second batch of images as we have done in our experiments, so that overall, no additional time is needed for CS decoding. The savings attained in GPU usage translate to monetary cost – for example, on Google Cloud Platform, NVIDIA T4 GPU usage costs 0.35 U.S. dollars per hour, whereas an N1 machine type single-core CPU usage costs only 0.032 U.S. dollars per hour [127, 128].

We also find that the FLOPs required for all of our CS methods are more than four orders of magnitude lower than the FLOPs required for QMPNN. FLOPs needed for CS decoding was measured by counting the number of floating point operations retired by the processor, using the AMD μProf [129] and linux `perf` [130] tools. The FLOPs needed for the methods for $p = 0.1$ are 26.3 KFLOPs (*Kilo* FLOPs) for our default CLASSO implementation, 162 KFLOPs for MIP, and 10.2 KFLOPs for FISTA, per image. We see that the additional compute requirements and the time taken by the CS methods are negligible as compared to those needed by the QMPNN.

Combining Pooling with Downsampling GT and CS methods may be combined with downsampling for further reduction in computation cost while maintaining acceptable performance.

Fig. 3.3 (sub-figures labelled IMFDB-Downsampled, rightmost column, top and bottom rows) shows the performance of non-adaptive GT and CS methods on IMFDB images downsampled by a factor of 4 to 112×112. We note that while all the algorithms observe a drop in sensitivity, it may be within an acceptable range depending on the particular use-case, especially for CLASSO. Specificity also drops, but less severely. CLASSO and MIP present the best balance of specificity and sensitivity across the entire range of p.

Multiple Objectionable Elements in the *Same Image* We created 100 random pools containing one OI (firearm image) each, but where *each OI depicted two firearms*. Out of 100, the QPNN correctly reported 97 times that the pool contained only one OI, and incorrectly reported two OIs 3 times. We conjecture that the QPNN detects unnatural edges between firearms in the SFM, due to which it does not get confused by a single image containing multiple firearms.

3.4.2 Off-topic Image Moderation using Pooled Outlier Detection

Here, the on-topic images are from a single underlying class and off-topic images may be from classes not seen during training. The goal is to detect such off-topic images, which do not belong to the underlying class. We do this by treating this as an outlier detection task, where the on-topic images are considered 'normal' and off-topic images are considered to be outliers.

Tasks We consider two off-topic image moderation tasks, wherein the on-topic images are (*i*) firearm images from IMFDB used in Sec. 3.4.1, and (*ii*) car images from the Stanford Cars Dataset [2]. During training, we use non-firearm or non-car images from ImageNet-1K as off-topic images from known classes for the two tasks, respectively. In Appendix 3.C, we provide additional details such as training/validation/test data splits and hyper-parameter tuning for these experiments. Off-topic images for testing are from some 182 non-firearm or 179 non-car randomly selected classes which are in ImageNet-21K [123] but not in ImageNet-1K.

GMM and Histogram Creation Recall the steps for GMM fitting and creation of histograms of anomaly scores in Sec. 3.3.4. We created 40850 and 18750 pools respectively from the

[2] https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset

training splits of IMFDB and Cars datasets and obtained their feature vectors by passing them to the trained ResNeXt-101 QPNN. For each dataset, we fit a GMM on these feature vectors. The optimal number of clusters in the GMM for the pooled case was 30 and 6 respectively for the IMFDB and Cars dataset, and 5 and 1 respectively for the individual case. We used $N = 100K$ pools from the validation split to compute the anomaly score histogram, with pool size $r = 8$ and the number of outliers t between 0 and 5. The number of outliers was capped, since for small prevalence rates, having more than 5 outlier images out of 8 is improbable. Recall from Sec. 3.3.4 that the same histogram method can be used for testing individual images as well. For this we used 5000 off-topic and 5000 on-topic images, and used $Q = 500$ bins.

Confusion Matrix Fig. 3.2 (Right) presents the $(t+1) \times (t+1)$ confusion matrix of the ground truth number of OIs in a pool versus the number predicted by the histogram method. The method predicts upto $t = 5$ outlier images in a pool. We see that if g is the ground truth number of OIs, the predictions of our proposed method lie in the interval $[\max(0, g-1), \min(g+1, t)]$ with high probability.

Prediction For each dataset, we created 9 test sample sets (one per prevalence rate) each of size 100K with different prevalence rates of off-topic images, similar to Sec. 3.4.1. For pooled outlier detection using CS or non-adaptive GT methods in Sec. 3.3.2, Algorithm 3.2 was used, with the same 50×100 pooling matrix $\mathbf{\Phi}$ as in Sec. 3.4.1, processing $n = 100$ images at a time. For Dorfman pooled outlier detection, Algorithm 3.3 was used, processing $r = 8$ images at a time. We refer to these outlier detection methods as CLASSO-OD, COMP-OD, NCOMP-2-OD, and DORFMAN-8-OD in the discussion that follows. Individual testing using our outlier detection method is referred to as INDIVIDUAL-OD, and individual testing of downsampled images as DOWNSAMPLE-OD.

Discussion on Results Fig. 3.5 shows the sensitivity and specificity of each of these algorithms for different values of p, for the two off-topic image detection tasks defined earlier. We see that for both IMFDB and Cars dataset, the CS/GT methods for oulier detection remain competitive with INDIVIDUAL. We observe that CLASSO-OD has higher sensitivity than COMP-OD for the entire range of prevalence rates considered and higher than DORFMAN-8-OD for $p \leq 0.01$. The reason for this has been discussed in Sec. 3.4.1: when a pool gets

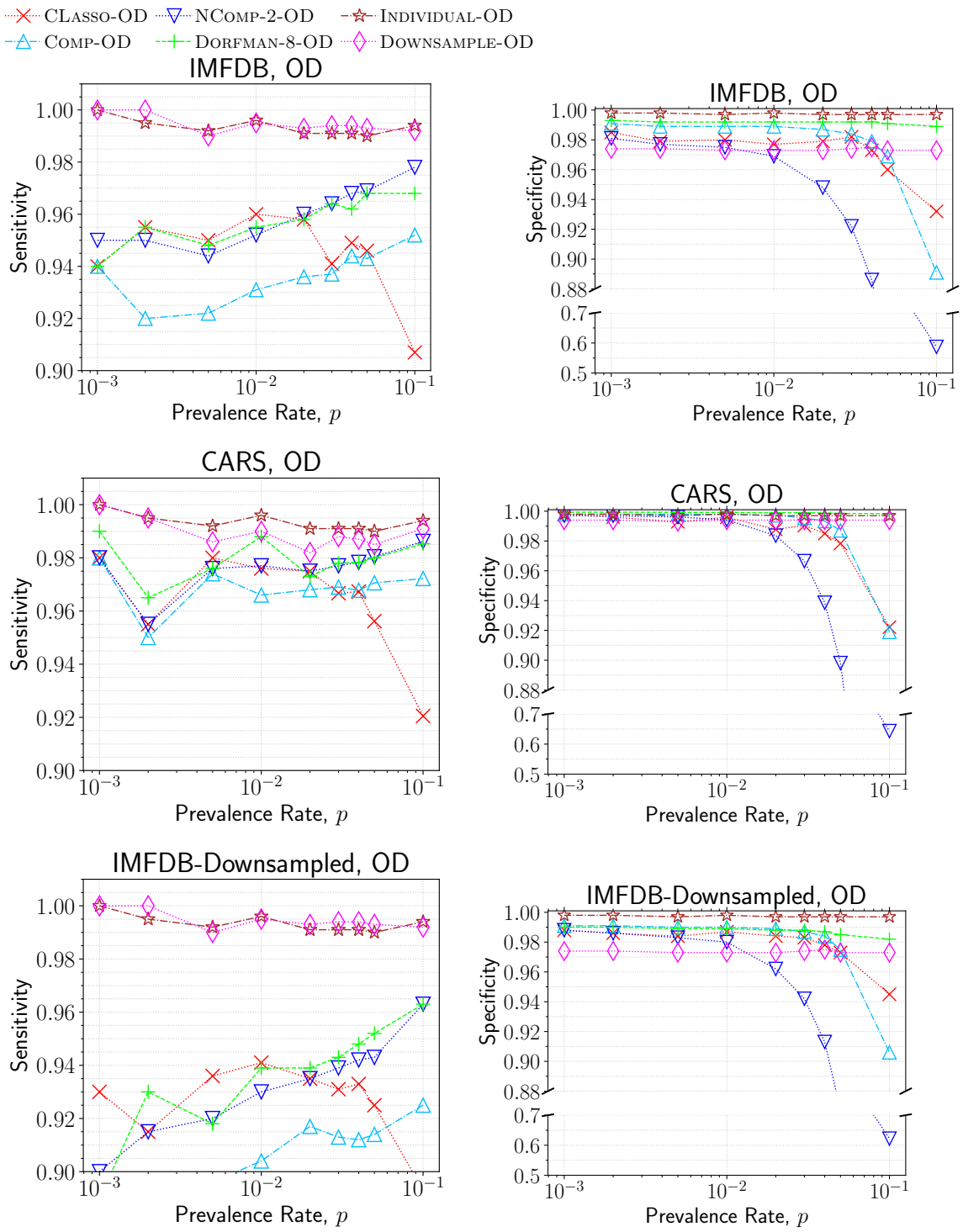


Figure 3.5: Performance of Off-topic Image Moderation via Pooled Outlier Detection methods for various prevalence rates (p), over 100K images each for IMFDB and Cars. Sensitivity (top row) and Specificity (bottom row) on IMFDB (left), Cars (middle), and IMFDB-Downsampled (right).

falsely classified as negative (i.e., containing no outlier image), then all outlier images in that pool get classified as negative by COMP-OD and DORFMAN-8-OD. In the higher p regions, DORFMAN-8-OD starts to perform better than CLASSO-OD. This is because, as we can see in Fig. 3.2 (Right), the prediction model gets confused for pools having ≥ 2 outlier images. At high prevalence rates of off-topic images, it is common for an on-topic image to test positive in all of its pools. As discussed in Sec. 3.4.1, COMP-OD uses only such binary information and declares incorrectly all such images as off-topic, so we observe a sharp decline in specificity. However, CLASSO-OD uses the quantitative information of the number of off-topic images in the pools, and can eliminate such false positives.

Comparison with Downsampling In Fig. 3.5, we compare our method to individual testing of images downsampled by a factor of 4 (size 112×112) (referred to in Fig. 3.5 as DOWNSAMPLE-OD). We find that both CLASSO-OD and DORFMAN-8-OD show better specificity than individual testing of downsampled images.

Computational Cost Fig. 3.4 (Left) shows the computational cost in terms of number of floating point operations (FLOPs). The amount of computation for CS or non-adaptive GT methods is the same in the classification and outlier detection case since the same NN is used. The GMM and histogram steps in the pooled outlier detection case need only 9.9 MFLOPs (*Mega* FLOPs) per image and in the individual outlier detection case need 3.2 MFLOPs per image (measured using AMD μProf as in Sec. 3.4.1). The FLOPs needed for CS decoding are given in Sec. 3.4.1. These are orders of magnitude smaller than the FLOPs needed by the QMPNN and may be neglected. For DORFMAN-8-OD, the amount of computation needed is slightly higher than for DORFMAN-8. This is because more pools falsely test positive in case of pooled outlier detection. For example, the top-left entries in the confusion matrices presented in Fig. 3.2 indicate that in case of pooled outlier detection, 11% of negative pools falsely test as positive, as compared to only 2% for pooled classification. However, we see that the computation costs for both adaptive and non-adaptive pooled outlier detection methods are well below those of individual testing.

The end-to-end wall-clock time (normalized by the number of images) for pooled outlier detection of 100K images of the Cars Dataset with CLASSO decoding and using DORFMAN-8-OD, are presented for a range of prevalence rates in Fig. 3.4 (Right). We find that CLASSO-

OD is significantly faster than individual testing of the images for all p, and DORFMAN-8-OD is significantly faster than individual testing for $p \leq 0.05$. CLASSO-OD is also faster than DORFMAN-8-OD for $p \geq 0.02$. The presented times include the execution of QMPNN, the GMM and histogram steps, as well as CS decoding. The GMM and histogram steps take 2.1×10^{-4} seconds per image for pooled outlier detection using the 50×100 pooling matrix, and 9.9×10^{-5} seconds per image for outlier detection on individual images. Computation time and GFLOPs for CS decoding are given in Sec. 3.4.1. Overall, these are orders of magnitude lower than the time required by the QMPNN step or by an INN.

Combining Pooling with Downsampling We show the performance of our pooled outlier detection methods on IMFDB images downsampled by a factor of 4 to 112×112 in Fig. 3.5 (sub-figures labelled IMFDB-Downsampled, rightmost column, top and bottom row). We note that while all the algorithms observe some drop in sensitivity, it remains above 90% across all prevalence rates for all algorithms. The specificity also drops, but to an even smaller extent.

3.5 Conclusion and Future Work

We presented a novel CS based approach for efficient automated image moderation, achieving significant reduction in computational cost for image moderation using deep NNs with very little loss of accuracy. We also presented a novel method for pooled deep outlier detection for off-topic image moderation, bringing CS and GT methods to the problem of outlier detection for the first time in the literature. A key strength of our approach is that it could, in principle, be combined with any good NN model for classification or outlier detection.

Our approach currently has some limitations, dealing with which presents several directions for possible future work. The optimal number of pools m (smallest number to yield a desired accuracy) depends on the prevalence rate p; exploring the exact relationship between the two may improve the throughput. Because we output the number of OIs for each pool, recovering the original OIs is also a quantitative group testing (QGT) [113] problem. For this QGT decoding algorithms [131] may also be used. Moreover, in this work, we assumed that the errors in the different elements of $\boldsymbol{y}$ are independent of each other and the underlying $\boldsymbol{x}$,

neither of which is strictly true for NN outputs. Though our algorithms still produce good results, incorporating a more sophisticated noise model may improve the decoding, and will also indirectly contribute to the CS and GT literature, where such noise models have not been analyzed. Another avenue of improvement might be the incorporation of the confidence values output by the softmax layer of the QMPNN or BMPNN in a downstream CS or GT algorithm – i.e. using the full probability distribution over the possible count values instead of taking the argmax. To the best of our knowledge, existing formulations of the CS or GT problems do not take into account such probabilistic measurements or tests. Our approach reduces the number of tests and is orthogonal to approaches such as network pruning or weights quantization which reduce the complexity of each test. We could combine these two approaches, just as we (already) explored the combination of pooled testing and image downsampling. In practice, image moderation engines may have access to side information such as IP addresses or past user history, incorporation of which can further improve decoding. Finally, it will also be of interest to extend our work to higher level semantic problems such as violence detection or detection of a wide variety of objectionable items in cluttered backgrounds.

Appendix 3.A Choice of Pooling/Sensing Matrix

3.A.1 Properties of Good Sensing Matrices

In a k**-disjunct pooling matrix**, the support of any column is not a subset of the union of supports of any k other columns. In noiseless binary group testing, COMP can identify up to k defectives exactly if the pooling matrix is k-disjunct [121, Sec 2.2].

Mutual Coherence of a matrix is the maximum value of the dot product of any of its two columns. Sensing matrices with low mutual coherence are preferred for compressed sensing as they allow for reduction of the upper bounds on recovery errors [94, Theorem 1].

The 1**-norm Restricted Isometry Property (RIP-1)** of order $2k$ is a sufficient condition on the sensing matrix $\mathbf{\Phi}$ for recovery of k-sparse $\boldsymbol{x}$ from noisy $\boldsymbol{y}$ via LASSO [132]. A pooling matrix $\mathbf{\Phi}$ satisfies RIP-1 of order $2k$ if it holds for some constants $2k$ and δ that

$\|\boldsymbol{x}\|_1 \leq \|\boldsymbol{\Phi x}\|_1 \leq (1+\delta)\|\boldsymbol{x}\|_1$ for all $2k$-sparse vectors $\boldsymbol{x}$.

3.A.2 Choice of Pooling Matrix

Randomly generated matrices from zero-mean Gaussian or Rademacher distributions have been popular in CS because they are known to obey the Restricted Isometry Property (RIP) with high probability [11], which is a well-known sufficient condition to guarantee accurate recovery [4]. RIP-obeying matrices also satisfy condition $\mathcal{C}2$ mentioned in Sec. 3.1. Despite this, such matrices are not suitable for our image moderation application, as they will lead to all n images contributing to every SFM – that too in unequal or both positive and negative amounts – and complicate the job of the QMPNN that predicts $\boldsymbol{y}$. Random Bernoulli ($\{0,1\}$) matrices have been known to allow for very good recovery [12] due to their favorable null-space properties. But randomly generated matrices will contain different number of ones in each row, due to which each pool would contain contributions from a different number of images. Instead, we use binary matrices which are constrained to have an equal number of ones in each of the n columns, denoted by c (i.e. 'column weight' c), and an equal number of ones in each of the m rows, denoted by r (i.e. 'row weight' r). This implies that $nc = mr$. Such a construction ensures that the QMPNN must produce outputs constrained to lie in $\{0, 1, ..., r\}$, and also ensures that each of the n images contributes to at least one pool (actually to exactly c different pools, to be precise). These matrices can be made very sparse ($r \ll n$), which ensures that noise in the output of the QMPNN can be controlled. We put an additional constraint, that the dot product of any two rows must be at most 1, and similarly the dot product of any two columns must be at most 1. It can be shown that such matrices are $(c-1)$-disjunct, obey the 'RIP-1' property of order $2c$ [14, Sec. III.F], and have low mutual coherence [133], which makes them beneficial for GT as well as CS based recovery. These properties are defined in Sec. 3.A.1.

3.A.3 Recovery Guarantees for CS and binary GT

Quantitative group testing (see Sec. 3.2), which is closely related to CS, has some conceptual advantages over binary group testing, as it uses more information inherently. We summarize

arguments from [121, Sec. 2.1], [134, Sec. 1.1] regarding this: Consider binary group testing with m pools from n items with k defectives. The total number of outputs is 2^m. The total number of ways in which up to k defective items can be chosen from n is $\sum_{i=0}^{k} C(n,i)$. We must have $2^m \geq \sum_{i=0}^{k} C(n,i)$, which produces $m \geq k\log(n/k)$. If we instead consider quantitative group testing, then the output of each test is an integer in $\{0,1,...,k\}$. The total number of outputs would thus be $(k+1)^m$. Thus $(k+1)^m \geq \sum_{i=0}^{k} C(n,i)$, which produces $m \geq \frac{k\log(n/k)}{\log(k+1)}$. Thus, the lower bound on the required number of tests is smaller in the case of quantitative group testing by a factor of $\log(k+1)$. Moreover, work in [135] argues that in the linear regime where $k = O(n)$ (even if $k \ll n$), individual testing is the optimal scheme for binary non-adaptive group testing. This is stated in Theorem 1 of [135] which argues that if the number of tests is less than n, then the error probability is bounded away from 0. On the other hand CS recovery with binary matrices is possible with $m = O(k\log(n/k))$ measurements [132, Theorem 9].

3.A.4 Upper bound on disjunctness of column-regular matrices

We explain in this section that matrices with column weight k (i.e., with k ones per column, equivalent with each image participating in k pools) which have strictly fewer rows than columns cannot be k-disjunct. The work in [120, Prop. 2.1] gives an upper bound on the cardinality of a uniform r-cover-free family of sets. They consider the power set of a set X of n elements. A k-uniform family of sets is any set of subsets of X containing exactly k elements. Let $\mathcal{F} = \{S_1, \ldots, S_K\}$ be a k-uniform family of sets of cardinality K. If $\mathcal{F}$ is r-cover-free, it means that no element of $\mathcal{F}$ is a subset of the union of any r other sets in $\mathcal{F}$. The work in [120, Prop. 2.1] gives the following bound on the cardinality of F:

$$K \leq \frac{\binom{n}{t}}{\binom{k-1}{t-1}}, \tag{3.15}$$

where $t = \lceil \frac{k}{r} \rceil$.

We can construct a $n \times K$ pooling matrix $\mathbf{\Phi}$ from $\mathcal{F}$, where each column is a $0/1$-binary

vector with column weight k. If we have $\Phi_{ij} = 1$, it indicates that the element i is in the set S_j (i.e. the jth image participates in the ith pool). Recall from Appendix 3.A.1 that in an r-disjunct matrix, the support of any column is not a subset of the union of supports of any r other columns. From the definition of disjunctness and r-cover-free families, if the matrix $\mathbf{\Phi}$ is to be r-disjunct, then the family of sets $\mathcal{F}$ must be r-cover free. Moreover, if $r = k$, then $t = 1$, and $K \leq \frac{n}{k-1}$. Hence, matrices with column weight k and fewer rows than columns i.e. $n < K$ cannot be k-disjunct.

For the case from Sec. 3.3.5 where column weight can be *upto* $k = 2$, we note that for any column with column weight 1, the corresponding row in which it has a 1 entry also has a row weight equal to 1, otherwise the matrix cannot be 2-disjunct. Thus, we can remove such rows and columns from the matrix, and the reduced matrix will also be 2-disjunct and will have column weight equal to 2 for all columns. From the above result, the number of columns in this reduced matrix must be less than or equal to the number of rows. Since we removed an equal number of rows and columns from the original matrix, the original matrix also has number of columns less than or equal to the number of rows, and hence cannot be used for achieving a reduction in number of tests using group testing.

3.A.5 Noise-tolerance of balanced binary matrices with row and column dot product at most 1

The work in [14, Sec. III.F.4] shows that such matrices with column weight c are adjacency matrices of (k, ϵ)-unbalanced expander graphs, and consequently [132, Theorem 1] have RIP-1 for any $k < 2c + 1$, with $\epsilon = \frac{k-1}{2c}$, and $0 \leq \epsilon < 1$. Robust recovery guarantees using the RIP-1 for such adjacency matrices exist – for example, [136, Theorem 6.1] states that the ℓ_1-norm of the recovery error of [136, Algorithm 1] is upper-bounded by $\frac{7-4\epsilon}{1-2\epsilon}\eta$, where η is the ℓ_1-norm of the noise in measurements. For a fixed magnitude of noise in the measurements, the amount of error in the recovered vector depends on the value of ϵ, with smaller values being better. For a fixed k, ϵ can be made smaller by having a larger value of c, the column weight of the matrix. Hence matrices with higher values of the column weight i.e. ones in which each item takes part in a larger number of pools are more robust to noise for CS recovery.

Appendix 3.B Experimental Details: Pooled Classification

Individual Dataset Splits For the firearm classification task, we use firearm images from IMFDB and Imagenet-1K [123] as the OI class, with 9458 and 3617 (total 13 075) images in the training set from the two datasets respectively. Additionally, 667 firearam images from IMFDB were used in the validation set, and 931 images were used in the test set. The validation and test images were manually cleaned to remove images which did not contain a firearm, or images in which the firearm was not the primary object of interest. For the non-OI images, we used images from 976 classes in ImageNet-1K [123], which did not belong to a firearm class – 1 200 764 in the training set, 50 000 in the validation set, and 48 800 in the test set (leaving out weapon images from ImageNet-1K such as 'tank', 'holster', 'cannon', etc., following [45]). During training of the INN, the classes were balanced by selecting all 13 075 OIs and the same number of random non-OIs for training at each epoch. Classes were balanced in the validation split as well, by using all 50 000 non-OIs and randomly sampling with replacement 50 000 OIs.

The Knives dataset [38] used for the knife classification task consists of 12,899 images of which 3559 are knife images and 9340 are non-knife images. Some images were taken indoors, while some were taken through a car window in the street. We randomly selected 2159 knife and 6140 non-knife images for training, 700 knife and 1600 non-knife images for validation, and 700 knife and 1600 non-knife images for testing. During training of the individual neural network, classes were balanced by sampling (without replacement) 2159 random non-knife images at each epoch. The validation set was also balanced by sampling (with replacement) 1600 random knife images. See Fig. 3.6 for examples of images from the IMFDB and Knives datasets, and also channels from the SFM (superposed feature map) for an example pool from the IMFDB dataset.

Pooled Dataset Splits For firearms classification, during training of the QPNN, at each epoch, 6248 pools of size $r = 8$ were created, each pool containing k randomly selected OIs and $8 - k$ randomly selected non-OIs, for each k in $\{0 \dots 8\}$. Of these, 40% pools contained no OIs, 24% contained 1 OI, 12% contained 2 OIs, 6% each containing 3 and 4 OIs, and 3% each contained 5 through 8 OIs. Such a distribution was chosen because OIs are assumed to have low prevalence rate in the test dataset. Hence greater accuracy is desired on pools with low number

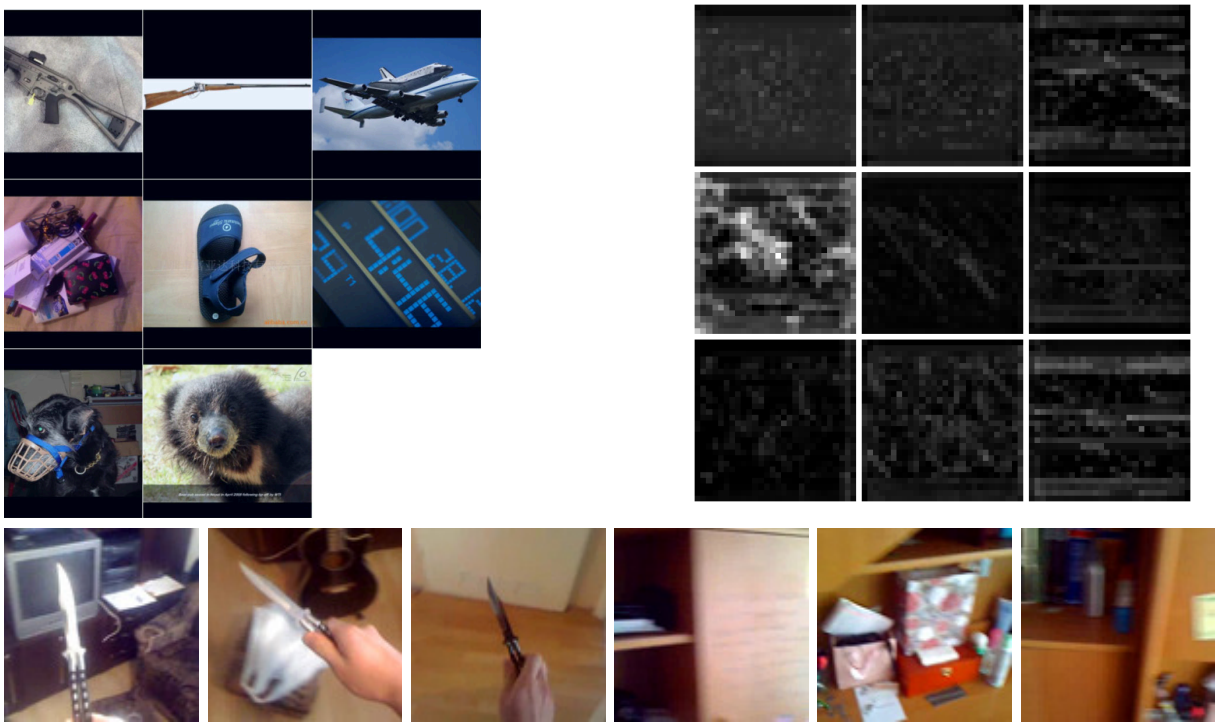


Figure 3.6: Top Row (left): A pool of 8 images from the IMFDB dataset, of which two are firearm images. Top row (right): The first 9 channels of Superposed Feature Maps (SFM) for this pool. The SFM has size $512 \times 28 \times 28$. Second row: Data Samples from Knives dataset [38]. First three images: Images containing a knife; Next three images: Images not containing a knife.

of OIs – however, enough instances of pools with high number of OIs must be seen during training for good accuracy. The pooled validation set used for selection of neural network weights contained $63\,325$ pools of size 8, created with the same method as the pooled training set. For testing, we created 9 mixtures of the test data split of the individual dataset, each containing $N = 1$ Million images, with each mixture containing pN OIs, with the prevalence rate $p \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1\}$.

For knife classification, 12496 pools of size 8 were created for training following the same distribution of OIs as for the firearms classification case. The pooled validation set used for selection of neural network weights contained 2873 pools of size 8, created with the same method as the pooled training set. The pooled test dataset consisted of 9 mixtures of 100K images, created with the same method as for the firearms classification task.

Data Transformations Images were padded with zeros to equalize their height and width, and resized to 224×224, during both training and testing. We also added a random horizontal

flip during training. For the pooled datasets, a random rotation rotation between $[-30°, 30°]$ was applied to each training image. This rotation was also applied to test data for the knife classification task.

Training We train all our networks for 90 epochs using stochastic gradient descent with learning rate 0.001 and momentum 0.9, using cross-entropy loss and weight decay of 0.0001. Model weights are checkpointed after every epoch. The best model weights are selected from these 90 checkpoints using classification accuracy on the validation set. For the firearms classification task, a weighted accuracy is used for selection of the best checkpoint of the pooled neural networks, viz $\texttt{Acc}_{\text{weighted}} = \sum_{k=0}^{8} \texttt{Binom}(8, p, k)\texttt{Acc}_k$, where $\texttt{Acc}_k$ is the classification accuracy on pools with k OIs in them, and $\texttt{Binom}(8, p, k)$ is the probability of having k OIs in a pool of size 8 if the prevalence rate of OIs is p. This accounts for the fact that accuracy on pools with higher number of OIs is more important for high values of p, while pools with 0 or 1 OI are more important for low values of p.

Hyperparameter Selection We created mixtures of images from the validation split of the same sizes and the same prevalence rates as for the test split. For a given prevalence rate p, grid search was performed on the corresponding validation split mixture with the same value of p to determine the optimum value of λ and τ for CLASSO and λ for MIP (see Sec. 3.3.2). The hyperparameter values which maximized the product of sensitivity and specificity on the validation mixture were chosen.

Appendix 3.C Experimental Details: Pooled Outlier Detection

Individual Dataset Splits The firearm images in training, validation and test splits of IMFDB as described in Appendix 3.B are taken to be on-topic images for the respective splits for the off-topic image moderation task. From the 976 non-firearm classes of ImageNet-1K in Appendix 3.B, we randomly sample $10\,000$ and 1000 images and add them to the training and validation split, respectively, as off-topic images. For testing, we choose 1000 images from 182

random non-firearm classes of ImageNet-21K as off-topic images.

For the Cars dataset, we take 8041 cars images for testing, 2144 cars images for validation, and 6000 cars images for training from the Stanford cars dataset as on-topic images. The 1000 off-topic images for testing come from 179 non-car classes of ImageNet-21K, whereas for training and validation, we sample 10000 and 1000 images from 955 non-cars classes of Imagenet-1K. We created 9 test sample sets (one per prevalence rate) of the test data split, each containing 100K images, using the same procedure and the same prevalence rates p as in Appendix 3.B. At each epoch of INN training, classes in the training split were balanced by using all 10 000 off-topic images from known classes and randomly sampling (with replacement) an equal number of on-topic images. The validation split classes were not balanced since the imbalance in class proportions was not too much.

Pooled Dataset Splits The pooled dataset training split was created using the same procedure as in Appendix 3.B, with the same pool size (i.e. 8), number of pools, and distribution of pools with different number of off-topic images in them. The pooled dataset validation split had 625 pools and was also created using the same procedure.

Neural Network, Data Transformation, Training We use the same neural network (ResNeXt-101 [124]) and train an INN and a QPNN using the same procedure as for the classification case (Appendix 3.B) on the IMFDB and Cars datasets. The data transformation used was the same as for the classification case, except that random rotation transformations were also applied to the images during testing to create more artificial edges in the SFMs and aid in detection of outliers in a pool.

Hyperparameter Selection The hyperparameters for CLASSO-OD were chosen in exactly the same manner as for the classification case (Appendix 3.B).

This page was intentionally left blank.

Chapter 4

Compressive Perturbed Graph Recovery

4.1 Introduction

In many applications domains, data are naturally arranged on the nodes of a graph, such as in sensor networks, transport networks, biological networks, social networks, and epidemiology. In some other domains, such a latent graph may be derived or inferred from available measurements, such as in image processing, computer graphics, and statistics. Such data are known as graph signals – a vector of values indicating a mapping of each node of the graph to a real number. Typically, the data on such graphs are results of processes on the graph, and its characteristics are intimately tied with the structure of the graph. For example, consider a contact tracing graph tracking the spread of a viral infection. Here, people who come in contact with each other frequently are more likely to either both be positive or both be negative for the virus. Each person represents a node, the data at the node represents the infection status or extent, and the edges represent the contact between the people. Similarly, in a grayscale 2D image, the image intensity value at neighbouring pixels are correlated. Here, each pixel is a node and neighboring pixels are connected by an edge. Such graph signals can often be approximated by

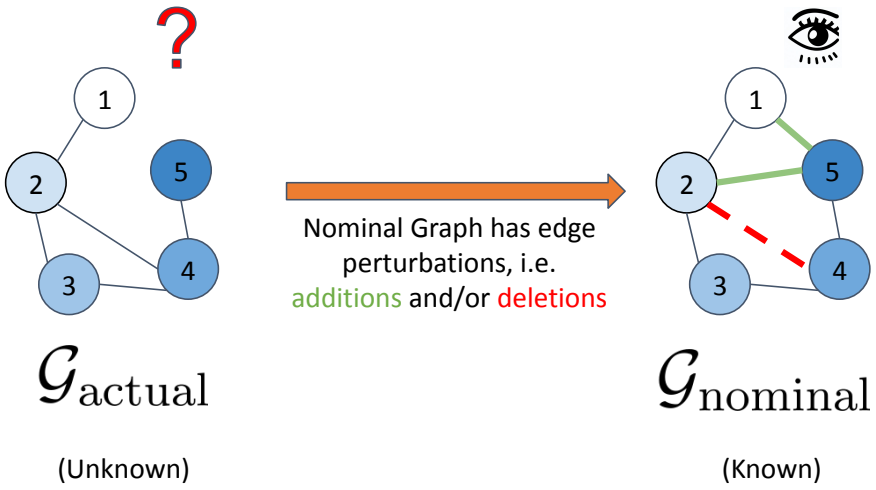


Figure 4.1: Illustration of Actual Graph (left) vs Nominal Graph (right) in the Compressive Perturbed Graph Recovery problem (Sec. 4.3). The actual Graph is not known, but the nominal graph is known. The nominal graph has a small number of edge perturbations relative to the actual graph.

the linear combination of a small number of eigenvectors of the Laplacian matrix of the graph – a key property which is used in the field of Graph Signal Processing [48, 49, 50] to process such data. For example, the 2-D Discrete Cosine Transform (DCT) basis is an eigenbasis of the Laplacian matrix of a 2-D lattice graph where the nodes of the graph represent pixels and the edges connect neighbouring pixels [52], and is commonly used in image compression [51]. The representation of a graph signal in the domain of the eigenvectors of the Laplacian matrix of the graph is known as the Graph Fourier Transform (GFT), analogous to the Fourier Transform in classical signal processing. Using the GFT, the field of GSP brings classical signal processing techniques such as filtering, convolution, sampling, etc., to signals on graphs [48].

One such technique is the compressive acquisition of graph signals. Instead of recording the data at each node of the graph separately, a small number of random linear measurements of the graph signal may be taken. These measurements may later be decoded to recover the original graph signal by using a Compressed Sensing (CS) decoding algorithm by exploiting the sparsity of the GFT of the graph signal. CS techniques allow the recovery of a high-dimensional vector from a small number of linear measurements, provided the vector has a sparse representation in a known orthonormal basis, and the measurements obey certain properties [3, 4].

In this part of the thesis, we consider the case when a graph signal has been compressively acquired, but there is some uncertainty in the knowledge of the graph from which it was ac-

quired. A *nominal* graph is known, which has the same nodes as the *actual* (unknown) graph, but has a small number of edge perturbations (edge additions and deletions) relative it (see Fig. 4.1). Due to this, some uncertainty is induced in the GFT basis of the graph, since the perturbation of a few edges of the graph will perturb its Laplacian matrix, and correspondingly its eigenvectors. Hence the graph signal will not be recovered accurately by a CS decoding algorithm if the GFT basis of the nominal graph is used, and some alternative approach is needed for recovery of the graph signal. In some applications, it may be desirable to recover the actual graph as well. We refer to the problem of recovery of the actual graph and the graph signal from compressive measurements as the *Compressive Perturbed Graph Recovery* problem.

We present a method called Greedy Edge Selection (GES), which solves this problem by refining the nominal graph one edge at a time, based on the cross-validation (CV) errors of the signals recovered using these graphs on a held-out set of measurements. The algorithm keeps proceeding as long as the CV errors of the successive graphs keep decreasing. Finally, the refined graph as well as the graph signal recovered using it are output. To summarize, we make the following main contributions in this work:

- We present the novel problem of Compressive Perturbed Graph Recovery, in which there is uncertainty in the orthonormal basis used for CS recovery, induced via perturbations of a graph.
- We present the Greedy Edge Selection (GES) method (Sec. 4.4.2) and its brute-force version (Sec. 4.4.1) for solving the compressive perturbed graph recovery problem.
- Based on the same idea – i.e. of comparison of CV errors of signals recovered using candidate graphs – we propose Inferred Linear-Edge Compressive Image Recovery (ILECIR, Sec. 4.4.3), an algorithm which performs the recovery of patch-wise compressively acquired images (such as in [62] and [63]) along with inferring linear image edges in the patches of the image via structured perturbation of the edges of a 2-D lattice graph (unlike unstructured edge perturbations done in GES).
- Using CV theory for compressed sensing [60], we prove sufficient conditions for signal and graph recovery using the brute-force algorithm with high probability (Sec. 4.4.4) . We also provide similar conditions for solution improvement in ILECIR and at each step of the GES.

- We empirically validate the GES algorithm on signals on a variety of graphs commonly used in the Network Science [64] literature, such as the Erdos-Renyi graph, the Planted Partition Model, the Barabasi-Albert graph, etc, and also validate the ILECIR algorithm on 40 images of various kinds – natural images, synthetically generated piece-wise smooth images, cartoon images, and depth-maps of indoor scenes (Sec. 4.5 and Sec. 4.6). Our algorithms outperform the baseline method of using the GFT basis of the nominal graph in a standard CS decoding algorithm. We also evaluate the suitability of replacing an eigendecomposition step in the GES algorithm with a faster approximation (Sec. 4.4.5 and Sec. 4.4.5).

- We demonstrate that our method is applicable to graph signals which admit some other forms of structure dependent on the graph, apart from just the sparsity of the GFT of the signal. Examples of such structural forms include the sparsity of the difference of the graph signal value across the edges of the graph (Sec. 4.4.6 and Sec. 4.6.2.6), which is often called graph total variation.

The remainder of this chapter is organised as follows: Sec. 4.2 presents an overview of Graph Signal Processing and Compressed Sensing. Sec. 4.3 defines the Compressive Perturbed Graph Recovery problem formally and gives an overview of the literature tackling similar problems. Sec. 4.4 describes the methods presented in this chapter in detail. Sec. 4.5 details the setup of empirical evaluation of our methods. The results of the evaluation are presented in Sec. 4.6. Finally, conclusions and future work are discussed in Sec. 4.7.

Author Contributions: The work presented in this chapter has been performed entirely by the author of this thesis, under the guidance of Prof. Ajit Rajwade.

4.2 Background

This section presents relevant background in Graph Signal Processing (Sec. 4.2.1) and Compressed Sensing (Sec. 4.2.2) for readers unfamiliar with these areas.

4.2.1 Graph Signal Processing

We present a very brief overview of the concept of graph signal processing for completeness. Survey articles such [49, 50] contain a much more extensive exposition. Consider an undirected, unweighted graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of n vertices (also referred to as nodes) of $\mathcal{G}$, labelled as $\{1, \ldots, n\}$, and $\mathcal{E}$ denotes the set of edges in $\mathcal{G}$, where each edge $e \in \mathcal{E}$ is an unordered pair of nodes in $\mathcal{V}$, i.e. $e = \{i, j\}$ where $i, j \in \mathcal{V}$. Let $n = |\mathcal{V}|$ be the number of nodes in $\mathcal{G}$, and $r = |\mathcal{E}|$ be the number of edges in $\mathcal{G}$. The adjacency matrix of $\mathcal{G}$ is denoted by the $n \times n$ binary matrix W, such that $w_{ij} = 1$ if $\{i, j\} \in \mathcal{E}$, else it is equal to 0. Note that W is a symmetric matrix because $\mathcal{G}$ is an undirected graph. The degree matrix of $\mathcal{G}$ is a $n \times n$ diagonal matrix D whose diagonal entries are the degrees of the corresponding node in $\mathcal{V}$, i.e. $d_{ij} = 0$ if $i \neq j$ and $d_{ii} = |\mathcal{N}(i)|$, where $\mathcal{N}(i) = \{j : \{i, j\} \in \mathcal{E}\}$ denotes the set of neighbours of the node i. The Laplacian matrix of the graph is the $n \times n$ matrix $L := D - W$. Besides being symmetric, it turns out that the Laplacian matrix is positive semi-definite. This can be proved by a straightforward application of the Gershgorin circle theorem, which states that the eigenvalues of a $n \times n$ complex matrix lie in the union of n closed discs on the complex plane, with each disc centered at a diagonal entry of the matrix, and the radius of the disc being equal to the sum of the absolute values of the off-diagonal entries of the corresponding row of the matrix. The eigendecomposition of the Laplacian matrix is denoted by

$$L = V \Lambda V^T, \tag{4.1}$$

where V is an orthonormal matrix of the eigenvectors of L (since L is symmetric), and Λ is the diagonal matrix of the eigenvalues of L. That is, if $\boldsymbol{v}_1, \ldots \boldsymbol{v}_n$ denote the eigenvectors of L (which are also the n columns of V in that order), and $\lambda_1, \ldots, \lambda_n$ denote the corresponding eigenvalues, then $L\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$. Also, $\Lambda_{ii} = \lambda_i$ and $\Lambda_{ij} = 0$ if $i \neq j$. Since L is a real, symmetric matrix, all of its eigenvalues and eigenvectors are real-valued. Moreover, it can be easily verified that since $L = D - W$, the vector of all 1's is an eigenvector of L with eigenvalue 0, hence the smallest eigenvalue is exactly equal to 0. For convenience, we assume that the eigenvalues are

listed in increasing order of magnitude, that is,

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n. \tag{4.2}$$

A **graph signal** on the graph $\mathcal{G}$ is a mapping from the nodes of a graph to the set of real numbers, and can be represented as a vector $\boldsymbol{x}$ of length n, with x_i being the value at node i. The vector $\boldsymbol{x}$ is known as the **vertex domain** representation of the graph signal. The **Graph Fourier Transform (GFT)** of the graph signal $\boldsymbol{x}$ is a a vector $\boldsymbol{\theta}$ which is the representation of $\boldsymbol{x}$ in the basis of the eigenvectors of the Laplacian matrix L of the graph $\mathcal{G}$. That is,

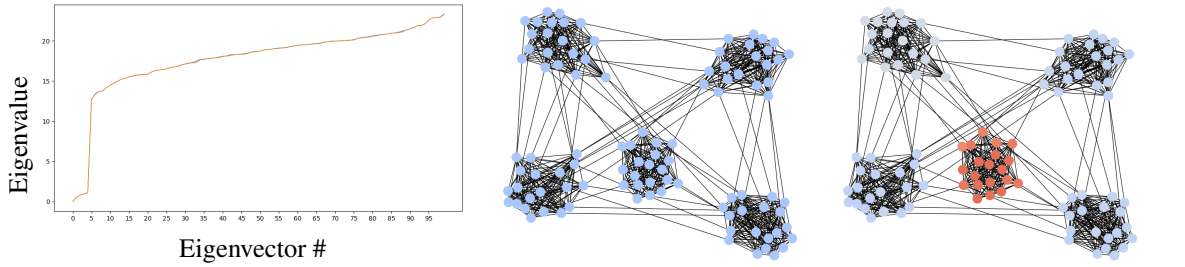
$$\boldsymbol{\theta} = V^T \boldsymbol{x}, \tag{4.3}$$

$$\text{and } \boldsymbol{x} = V\boldsymbol{\theta}. \tag{4.4}$$

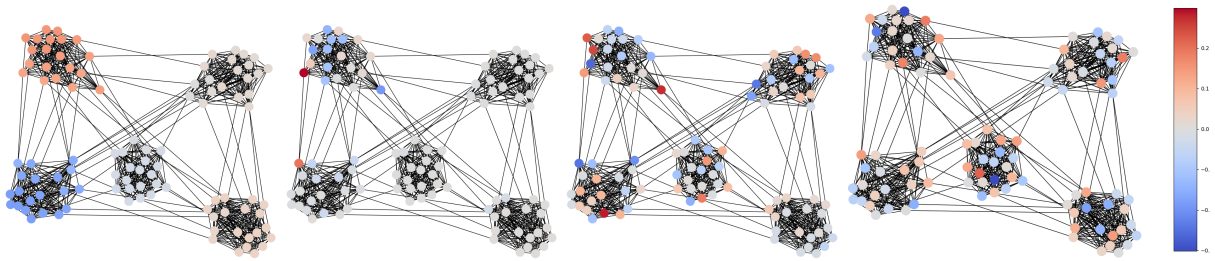
In this context, the Laplacian matrix L is known as the **graph shift operator**, and the set of eigenvectors of L are known as the **GFT basis** of the graph $\mathcal{G}$.

The Laplacian matrix on graphs is an analogue of the Laplace operator in Euclidean space [137, Slides 20-24], defined as $\nabla^2 \boldsymbol{z} := \sum_{i=1}^{d} \frac{\partial^2}{\partial z_i^2}$ where $\boldsymbol{z} \in \mathbb{R}^d$ represents the d-dimensional coordinate. Just as the eigenfunctions of the Laplace operator form the Fourier basis, the eigenvectors of the Laplacian matrix forms the Graph Fourier basis. In fact, the Fourier basis vectors of the n-element 1-D Discrete Fourier Transform are eigenvectors of the Laplacian matrix of the n-node ring graph. There exists a formal relationship between Laplacian matrices and the Laplace-Beltrami operator, a generalization of the Laplace operator for Riemannian Manifolds. Namely, the Laplacian matrices of a particular infinite sequence of weighted, undirected graphs defined on a manifold converge to the Laplace-Beltrami operator of the manifold [138]. There may be other reasonable choices for the graph shift operator than the Laplacian matrix, such as the adjacency matrix, normalized Laplacian matrix, etc.

We note that the eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ are also graph signals on $\mathcal{G}$. Each eigenvector has a notion of frequency associated with it, with low eigenvalues signifying low frequency, and high eigenvalues signifying high frequency – thus eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ are ordered in increasing order of frequency. The value at each node of a high frequency eigenvector varies rapidly when moving along an edge of the graph, whereas for a low frequency eigenvector it



(a) Eigenvalue vs Eigenvector # **(b)** Eigenvector #1 **(c)** Eigenvector #2



(d) Eigenvector #5 **(e)** Eigenvector #6 **(f)** Eigenvector #20 **(g)** Eigenvector #40

Figure 4.2: Eigenvalues and Eigenvectors of a community-structured graph with 100 nodes, 5 communities, and 20 nodes per community, generated using a Planted Partition Model (Sec. 4.5.1.1). (a) – Eigenvalue vs Eigenvector #. First 5 eigenvectors have small eigenvalue, after which there is a large jump in the eigenvalue. (b) to (g) – Eigenvectors. Node color represents the value of the eigenvector at that node, indicated by the colorbar.

varies more smoothly.

This is because the eigenvectors of the Laplacian are successive orthogonal minimizers of the Rayleigh quotient $\boldsymbol{x}^T L \boldsymbol{x}$ (given $\|\boldsymbol{x}\|_2 = 1$). The semi-norm $S(x) \triangleq \sqrt{\boldsymbol{x}^T L \boldsymbol{x}}$ may be interpreted as a measure of the variation of the signal $\boldsymbol{x}$ across the edges, since $\boldsymbol{x}^T L \boldsymbol{x} = \sum_{\{i,j\} \in \mathcal{E}} (x_i - x_j)^2$ [49, 139]. See [50, Sec. II-D, II-E, and III-A], for a detailed discussion on frequency in graph signal processing. Indeed, as illustrated in [49, Fig. 2] and also here in Fig. 4.2, the number of zero crossings of the values of the eigenvector at the endpoints of each edge increases as the eigenvalue corresponding to the eigenvector increases. Hence, analogous to the classical Fourier Transform literature, the representation of a graph signal in the basis formed by the eigenvectors of the graph's Laplacian matrix (Eqn. 4.3) is also known as its **frequency domain** representation.

For some intuition of the frequency domain of a graph, see Fig. 4.2, which shows some eigenvectors of a community-structured graph with 100 nodes and 5 communities, as well as

the plot of the eigenvalues versus the eigenvector index, with eigenvectors sorted in ascending order of eigenvalue. In a community-structured graph, the set of nodes of the graph may be partitioned into two or more communities, such that most edges are between two nodes of the same community, and very few edges have endpoints in two different communities. We see that the first eigenvector has the same value at all nodes. Analogous to Fourier Transform, this eigenvector represents the DC component of the graph signal. The next 4 eigenvectors vary smoothly within a community, but the values on edge endpoints across two communities vary more. In general, for a graph with k communities, the first k eigenvectors vary smoothly within a community. For some intuition regarding this, note that if there were no edges going across communities, then the membership indicator vector of a community (having a 1 if a node is a member, and 0 otherwise) will be an eigenvector of this graph, having eigenvalue exactly equal to 0. This property has been used widely in spectral clustering of graphs and in image segmentation [140]. From eigenvector 6 onwards, the eigenvalue rises sharply, and we start seeing variation even within a community, which becomes more and more pronounced for eigenvectors with larger eigenvalues.

The action of certain naturally occurring discrete-time linear dynamical systems defined on the graph $\mathcal{G}$ run for an arbitrary amount of time T, transforming an initial graph signal into another graph signal, can be expressed as a polynomial of the Laplacian matrix of the graph. Graph signals typically under consideration are outputs of such dynamical systems, including diffusion processes [141, 142], defined on the graph. By the Cayley-Hamilton theorem, which states that a matrix satisfies its own characteristic polynomial, the matrix powers of degree n or higher for an $n \times n$ matrix may be written as polynomials of degree $n-1$, and thus any matrix polynomial of arbitrary degree T may be converted to a polynomial of degree at most $n-1$ (even if $T = \infty$). A **linear graph filter** H is an arbitrary polynomial of the graph's Laplacian matrix of degree at most $n-1$:

$$H = \sum_{i=0}^{n-1} a_i L^i. \tag{4.5}$$

Hence, the action of a discrete-time linear dynamical system is a linear graph filter. Using the eigendecomposition of the Laplacian matrix L from Eqn. 4.1, the linear graph filter expression

in Eqn. 4.5 can be rewritten as:

$$H = V\Big(\sum_{i=0}^{n-1} a_i \Lambda^i\Big) V^T. \tag{4.6}$$

Hence, the action of the graph filter on a graph signal $\boldsymbol{z}$ with GFT $\boldsymbol{\phi} = V^T \boldsymbol{z}$ is:

$$H\boldsymbol{z} = V\Big(\sum_{i=0}^{n-1} a_i \Lambda^i\Big) V^T \boldsymbol{z} = V\Big(\sum_{i=0}^{n-1} a_i \Lambda^i\Big) \boldsymbol{\phi}. \tag{4.7}$$

We note that since Λ is a diagonal matrix, any polynomial of Λ is a matrix with each diagonal entry being equal to the polynomial of the corresponding entry of Λ. Thus, a linear graph filter suppresses or amplifies the j^{th} frequency component of a graph signal by multiplying it with a value $h(\lambda_j)$, where

$$h(\lambda) = \sum_{i=0}^{n-1} a_i \lambda^i \tag{4.8}$$

is known as the **generating function** of the graph filter.

A **sparse-spectrum** graph signal is one for which most of the frequency components are equal to 0. Such a graph signal may arise naturally as the output of linear graph filters for which most of the roots of their generating functions are equal to some eigenvalues of the Laplacian matrix L, i.e. for which $h(\lambda)$ is equal to 0 for most of the eigenvalues in $\{\lambda_1, \ldots, \lambda_n\}$. In fact, any generating function of the form

$$h(\lambda) = \prod_{i \in [n]/S} (\lambda - \lambda_i) \sum_{j=0}^{|S|-1} b_j \lambda^j, \tag{4.9}$$

where S is a set of indices and $[n] = \{1, \ldots, n\}$, will produce output signals for which the frequency components in $[n]/S$ are always 0. A **band-limited** graph signal is a special case of a sparse-spectrum graph signal, for which all frequency components higher than some given frequency are equal to 0. Sparse-spectrum and band-limited signals may be recovered from compressive measurements using compressed sensing techniques if the GFT matrix V is known.

4.2.2 Compressed Sensing

Compressed Sensing (or Compressive Sensing) [3] is a technique for acquiring a signal $\boldsymbol{x}^* \in \mathbb{R}^n$ from fewer than n linear measurements, by exploiting its sparsity in some orthonormal basis $\boldsymbol{\Psi}$ (i.e. $\boldsymbol{\Psi}^T = \boldsymbol{\Psi}^{-1}$). The signal $\boldsymbol{x}^*$ is known to be sparse in the basis $\boldsymbol{\Psi}$, i.e., $\boldsymbol{x}^* = \boldsymbol{\Psi}\boldsymbol{\theta}^*$, such that $\|\boldsymbol{\theta}\|_0 = s \ll n$. Typically, the sparsity s is not known, but the orthonormal basis $\boldsymbol{\Psi}$ is known. The measurements are acquired via a known sensing matrix (or a measurement matrix) $\boldsymbol{\Phi}$ of dimensions $m \times n$, with $m \ll n$ (typically $m = O(s \log \frac{n}{s})$) is used to acquire the measurement vector $\boldsymbol{y}$, with the relationship:

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + \boldsymbol{\eta} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}^* + \boldsymbol{\eta}, \tag{4.10}$$

where $\boldsymbol{\eta}$ is a noise vector, whose entries are typically assumed to be i.i.d. Gaussian with mean 0 and variance σ^2. The acquisition of the measurements via Eqn. 4.10 is implemented in some domain-specific hardware such as the Rice Single Pixel Camera [53], and is typically much cheaper than acquiring the entire signal $\boldsymbol{x}^*$ in terms of number of hardware elements needed, time of acquisition, or monetary cost. The original signal needs to be recovered from the compressive measurements using a decoding algorithm which incurs some cost, the aim here is to save *acquisition* resources including acquisition time. The signal in its transform domain representation may be estimated via Basis Pursuit Denoising (BPDN) [3], which is the following ℓ_1-norm minimization problem with a quadratic constraint:

$$\hat{\boldsymbol{\theta}}_{\text{bpdn}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_1, \quad \text{s.t. } \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}\|_2 \leq \varepsilon \tag{4.11}$$

where $\varepsilon \geq \|\boldsymbol{\eta}\|_2$ is an upper bound on the magnitude of the noise vector $\boldsymbol{\eta}$. Due to the method of Lagrangian multipliers for strictly convex functions, Eqn. 4.11 may be expressed in a penalized form, known as the LASSO (Least Absolute Shrinkage and Selection Operator) [5], as follows:

$$\hat{\boldsymbol{\theta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\theta}\|_2^2 + \mu\|\boldsymbol{\theta}\|_1. \tag{4.12}$$

with some value of the regularization parameter $\mu \geq 0$ corresponding to ε in Eqn. 4.11. The ℓ_1 penalty on $\boldsymbol{\theta}$ again promotes the sparsity of $\hat{\boldsymbol{\theta}}$. The canonical domain representation of the estimated signal may be retrieved as $\hat{\boldsymbol{x}}_{\text{lasso}} = \boldsymbol{\Psi}\hat{\boldsymbol{\theta}}_{\text{lasso}}$. Alternatively, LASSO may be performed directly in terms of $\boldsymbol{x}$ as:

$$\text{LASSO:} \qquad \hat{\boldsymbol{x}}_{\text{lasso}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1. \tag{4.13}$$

The estimate $\hat{\boldsymbol{x}}_{\text{bpdn}} := \boldsymbol{\Psi}\hat{\boldsymbol{\theta}}_{\text{bpdn}}$ is close to $\hat{\boldsymbol{x}}^*$ if the matrix $\boldsymbol{A} := \boldsymbol{\Phi}\boldsymbol{\Psi}$ possesses the so-called Restricted Isometry Property [4]:

$$\exists \delta_{2s} \in (0,1) \text{ s.t. } \forall \boldsymbol{\theta} \text{ with } \|\boldsymbol{\theta}\|_0 \leq 2s, \quad (1-\delta_{2s})\|\boldsymbol{\theta}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{\theta}\|_2^2 \leq (1+\delta_{2s})\|\boldsymbol{\theta}\|_2^2. \tag{4.14}$$

The constant δ_{2s} is known as the Restricted Isometry Constant (RIC) of order $2s$. If the entries of $\boldsymbol{\Phi}$ are i.i.d. Gaussian or sub-Gaussian with mean 0 and variance m, and $m \geq O(s \log \frac{n}{s})$, then with high probability, the matrix $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$ has the RIP [11].

Any compressive recovery algorithm is typically evaluated in terms of the mean squared error given by

$$\varepsilon_{\text{x}} \triangleq \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2^2. \tag{4.15}$$

The recovery error for BPDN (Eqn. 4.11) has the following guarantee [4]:

$$\|\hat{\boldsymbol{x}}_{\text{bpdn}} - \boldsymbol{x}^*\|_2 = \|\hat{\boldsymbol{\theta}}_{\text{bpdn}} - \boldsymbol{\theta}^*\|_2 \leq C\varepsilon, \tag{4.16}$$

where the constant C depends only on δ_{2s} and is a monotonically increasing function of it. A stronger guarantee holds for LASSO for some μ which is lower bounded as $\Omega(\sigma\sqrt{s \log n/m})$ [6, Thm. 11.1]. The guarantee itself is given by

$$\|\hat{\boldsymbol{x}}_{\text{lasso}} - \boldsymbol{x}^*\|_2 = \|\hat{\boldsymbol{\theta}}_{\text{lasso}} - \boldsymbol{\theta}^*\|_2 \leq C_l\sigma\sqrt{\frac{s \log n}{m}}, \tag{4.17}$$

where $C_l > 0$ is a constant. The recovery guarantees for the LASSO do not use the RIP but

a somewhat weaker condition on $\boldsymbol{A}$ called the Restricted Eigenvalue Condition (REC). This imposes that $\|\boldsymbol{A\nu}\|_2^2 \geq \gamma\|\boldsymbol{\nu}\|_2^2$ for all error vectors $\boldsymbol{\nu}$ which are naturally restricted to lie in a cone-shaped set. Here $\gamma > 0$ is the so-called order-s restricted eigenvalue constant. See [6, Sec. 11.2.2] for more details.

Cross-validation for selection of the regularization parameter: Typically, the best regularization parameter μ for LASSO is not known in advance, as the signal sparsity is often unknown. Hence μ is determined via grid search over a set of possible values Γ_μ, using a method called cross-validation [60]. Cross-validation in the context of compressed sensing [59, 60] is executed in a slightly different manner as compared to its conventional usage in machine learning. Out of a total of m compressive measurements, some m_{r} randomly chosen measurements are used for signal reconstruction, and the remaining m_{cv} measurements are chosen for cross-validation. Let the sub-vector $\boldsymbol{y}_{\text{r}}$ contain the m_{r} measurements chosen for reconstruction and $\boldsymbol{\Phi}_{\text{r}}$ be the sub-matrix containing the corresponding rows of the sensing matrix. Let $\boldsymbol{y}_{\text{cv}}$ be the sub-vector of measurements used for cross-validation, and let $\boldsymbol{\Phi}_{\text{cv}}$ be the corresponding sub-matrix. Then the cross-validation using LASSO (termed LASSO-CV) proceeds as follows:

LASSO-CV$(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, \Gamma)$:

$$\hat{\boldsymbol{x}}_{\text{r}}^{\mu} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y}_{\text{r}} - \boldsymbol{\Phi}_{\text{r}}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1, \quad \epsilon_{\text{cv}}^{\mu} = \|y_{\text{cv}} - \boldsymbol{\Phi}_{\text{cv}}\hat{\boldsymbol{x}}_{\text{r}}^{\mu}\|_2^2, \tag{4.18}$$

$$\hat{\mu} = \arg\min_{\mu\in\Gamma} \epsilon_{\text{cv}}^{\mu}, \quad \hat{\boldsymbol{x}}_{\text{lasso-cv}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \hat{\mu}\|\boldsymbol{\Psi}^T\boldsymbol{x}\|_1. \tag{4.19}$$

Note that the subsets of measurements used for reconstruction and validation are completely *disjoint*. Since only m_{r} measurements are used for recovery, we need $m_{\text{r}} \geq O(s\log\frac{n}{s})$ for REC-based recovery guarantees to hold for LASSO-CV. The work in [60, 61] presents a probabilistic guarantee for the success of using CV error comparison as a proxy for recovery error comparison in the compressed sensing setting when the matrix $\boldsymbol{\Phi}$ is Gaussian, which we use in Sec. 4.4.4 and the Appendix for proving bounds on the recovery error for the methods presented in our work.

4.3 Problem Statement

Consider that we are given linear, and possibly noisy, compressive measurements $\boldsymbol{y} \in \mathbb{R}^m$ of an unknown signal $\boldsymbol{x}^*$ defined on the vertices of an undirected, unweighted graph $\mathcal{G}_{\text{actual}} := (\mathcal{V}, \mathcal{E}_{\text{actual}})$ where $|\mathcal{V}| = n$. Hence, we have

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + \boldsymbol{\eta}, \tag{4.20}$$

where $\boldsymbol{\Phi}$ is a $m \times n$ measurement matrix with $m \ll n$ (since we are in the compressive regime), and $\boldsymbol{\eta}$ is a vector of noise, each of whose entries is i.i.d. Gaussian with mean 0 and variance σ^2. In some scenarios, the graph $\mathcal{G}_{\text{actual}}$ is not known with full accuracy. Instead, we have access to the graph $\mathcal{G}_{\text{nominal}} = (\mathcal{V}, \mathcal{E}_{\text{nominal}})$, such that $\rho(\mathcal{E}_{\text{actual}}, \mathcal{E}_{\text{nominal}}) = d$, where $\rho(\mathcal{E}_{\text{actual}}, \mathcal{E}_{\text{nominal}}) \triangleq |\mathcal{E}_{\text{actual}} - \mathcal{E}_{\text{nominal}}| + |\mathcal{E}_{\text{nominal}} - \mathcal{E}_{\text{actual}}|$ is the number of edge additions or removals needed to obtain the actual graph from the nominal graph, and d is an unknown but *small* positive integer. That is, while the set of nodes of $\mathcal{V}$ is fully known, the set of edges $\mathcal{E}_{\text{actual}}$ is known only upto a few perturbations. This means that while most of the edges in $\mathcal{E}_{\text{actual}}$ and $\mathcal{E}_{\text{nominal}}$ are the same, a few edges present in $\mathcal{E}_{\text{actual}}$ may not be present in $\mathcal{E}_{\text{nominal}}$, and a few edges not present in $\mathcal{E}_{\text{actual}}$ may be present in $\mathcal{E}_{\text{nominal}}$. While d is not known, we can assume that an upper bound d_0 is known (i.e. $d \leq d_0$)). Let L_{actual} be the unknown Laplacian matrix of $\mathcal{G}_{\text{actual}}$, and let $L_{\text{actual}} = V_{\text{actual}}\Lambda_{\text{actual}}V_{\text{actual}}^T$ be its eigendecomposition. In addition, we will assume that the unknown signal $\boldsymbol{x}^* = V_{\text{actual}}\boldsymbol{\theta}^*$ is sparse in V_{actual}. That is $\boldsymbol{\theta}^*$ is the GFT of $\boldsymbol{x}^*$, and $\|\theta^*\|_0 = k \ll n$ for some unknown k. The goal is to recover the original signal $\boldsymbol{x}^*$, as well as the actual graph $\mathcal{G}_{\text{actual}}$ given just $\boldsymbol{y}$, $\boldsymbol{\Phi}$ and $\mathcal{G}_{\text{nominal}}$. This is formally defined as follows:

Problem 4.1 (Compressive Perturbed Graph Recovery Problem)**.** *Given $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\mathcal{E}_{nominal}$ and d_0, find $\boldsymbol{x}^*$ and $\mathcal{E}_{actual}$.*

This is a novel computational problem, and has not been explored in the literature. In the following section, we present a literature review of closely related work.

4.3.1 Related Work

4.3.1.1 Graph Spectral Compressed Sensing

Note that here we are concerned with compressive measurements of signals defined on a graph, and wish to make use of the graph structure in signal recovery via V_{actual}. This is sometimes called graph spectral compressed sensing. There exists only a moderately sized literature on this specific topic: [143, 144] are two references. In both these papers and in a large body of graph signal processing literature in the non-compressive regime (see for example references in [49]), the signal $\boldsymbol{x}^*$ is considered to be band-limited, i.e. it is considered to be a linear combination of only low frequency eigenvectors in V_{actual}. In a departure from this, we consider the signal $\boldsymbol{x}^*$ to be just *sparse* in the graph spectral domain in the work presented here, and band-limited signals are a special case of this more general model. Furthermore, the work in [143, 144] assumes a measurement model where the data at only a subset of all nodes of a graph are observed. The purpose of compressive recovery is then to fill in the missing data at other nodes. Another orthogonal set of works in the network tomography literature focuses on performing compressed sensing on graphs with matrices which may be defined via random walks on the graph [145, 146]. In such works, the signals represent properties of the edges of the graphs rather than the nodes.

As against this, our work focuses on compressive measurements which are random linear combinations of the values at the different nodes, more in line with traditional compressed sensing as described in Sec. 4.2.2. To the best of our knowledge, the only work which uses linear combinations of the values at different nodes along with using the sparsity of GFT in the decoding algorithm is [147], which performs compressive recovery of multi-channel EEG signals. In their case, the graph connects channels whose electrodes are physically closer than some threshold distance, and the edge weights decrease exponentially as the square of the distance.

4.3.1.2 Compressed Sensing of Signals over Graphs with Partly Erroneous Topology

There exists a decent body of literature on inferring the graph toplogy (i.e. the edges and their respective weights) from a large number of signals defined at each node of the graph.

These signals could represent information at each node that varies dynamically across time. The graph Laplacian or the underlying adjacency matrix are then inferred from these signals under the assumption that the signals would be *smooth*, i.e. expressible accurately as a linear combination of low-frequency eigenvectors of the Laplacian. This is the approach followed in a large variety of papers – see for example references such as [148, 149, 150], and other references in [151]. Our work here differs from these approaches in three ways:

1. First, we operate in the compressive regime, and hence have access to significantly less data, as opposed to a large number of signals at each node.

2. Second, we assume sparsity of the signal in the eigenvectors of the graph as opposed to smoothness or band-limitedness.

3. Last, we are not operating in the regime when the graph is completely unknown. Rather, we are given a graph $\mathcal{G}_{\text{nominal}}$ with error in a *small* number of edges, and wish to infer the erroneous edges along with the signal coefficients from compressive measurements. Thus, we are interested in correcting a small number of errors from the given graph topology. These errors manifest themselves as either missing edges or extra edges as compared to an underlying 'ground truth' graph.

The approach of correcting a small number of edges has been followed in a small number of papers, albeit in the non-compressive regime given a single signal vector – [152, 153, 154]. In [153], the effect of perturbations in the form of deletion or addition of a small number of edges on the eigenvectors and eigenvalues of the Laplacian matrix is studied, and scenarios where the effect is negligible or very adverse are analyzed. The derived results are compared to those produced in matrix perturbation theory. Moreover, the effect of the perturbations on applications such as (non-compressive) signal and graph recovery with known signal priors, clustering and label propagation, are examined. Furthermore, approximate closed form expressions for the perturbations of eigenvectors and eigenvalues are derived, under the assumption that the perturbation to the Laplacian matrix is small. A total least squares approach to robust graph signal recovery given structural equation models is presented in [152]. The work in [154] develops models for perturbations to edges in a graph and examines their effect on graph signal filtering and independent components analysis. The stability of polynomial graph filters to edge perturbations is proved in [155], but the importance of the location of the perturbed edge

is emphasized. As opposed to this, in our work, we present a method for robust graph signal recovery from *compressive measurements* given a small number of perturbations in the graph topology, and assuming the signal has a *sparse*, but not necessarily band-limited, representation in the eigenvectors of the underlying graph Laplacian.

Lastly, the work in [156] performs joint recovery of an undirected, weighted, data-dependent graph and a graph signal in the compressive regime. They use graph total variation based regularization along with wavelet-based regularization for tomographic reconstruction of a 2-D image. The graph in consideration is an undirected, data-dependent weighted graph of overlapping patches of the images, wherein the weights may be directly computed from the pixel values of the patches of the unknown image, with only the top few weighted edges for each patch being retained. Graph total variation regularization using such a graph exploits similarity in parts of the image which are not spatially close to each other. They alternately compute an approximation of the graph from an approximation of the image, and compute an approximation of the image from the approximation of the graph, in a fixed-point iteration. The initial approximation for the image is found using filtered back-projection. While there is some similarity in the problem setting – since they are in the compressive regime with an unknown graph (or an initial approximation of the graph) – their method is not directly applicable to a case where the graph is unweighted and is not data-dependent. In our setting, the underlying graph cannot be directly computed from the graph signal. Moreover, each node of the graph in [156] maps to a vector of values – the constituent pixel values of the patch, whereas in our case each node maps to only a single value in the graph signal. Furthermore, regularization via graph total variation implies that the signal is smooth over the graph, whereas in our setting the signal is a linear combination of a small number of arbitrary eigenvectors of the Laplacian matrix of the graph. We provide sufficient conditions for successful signal and graph recovery using one of our methods in Sec. 4.4.4, whereas no such guarantees are given for the algorithm in [156].

4.3.1.3 Compressed Sensing with Perturbed Models

We conclude this literature review, with an overview of techniques in compressed sensing that deal with model mismatches. Referring to Eqn. 4.10, we could have perturbations in either the sensing matrix $\mathbf{\Phi}$ or the representation matrix $\mathbf{\Psi}$ or both. The former problem has been

explored in terms of perturbations to specified Fourier frequencies in Fourier sensing matrices in magnetic resonance imaging in work such as [157, 158]. More unstructured and dense perturbations to $\mathbf{\Phi}$ are considered in [58, 159, 160, 161, 162, 163]. The focus of the work here, however, is related to perturbations in $\mathbf{\Psi}$, because perturbations to the graph topology induces changes in the eigenvectors of the Laplacian matrix, which is the signal *representation* matrix (and not the sensing matrix which is completely independent). The problem of perturbations in $\mathbf{\Psi}$ has been extensively explored in the context of perturbations to frequency specifications in sinusoidal bases such as the Fourier or discrete cosine transform in work such as [55, 56, 57]. Various approaches such alternating minimization [56], perturbation correction in greedy algorithms like orthogonal matching pursuit (OMP) [9], or structured sparsity [57, 159] have been considered. The problem of estimating a small number of complex sinusoids with off-the-grid frequencies from a subset of regularly spaced samples has been explored in [164]. Many applications of this framework in radar signal processing for problems such as direction or arrival (DoA) estimation have been explored in [57, 159, 165]. The problem of *additive* perturbations in both the sensing matrix as well as the representation matrix has been analyzed in [58], using several assumptions on both perturbations. In contrast to this, in this work, we explore perturbations in the representation matrix in an *indirect* manner. That is, we explore methods to correct for perturbations in edge specifications in the adjacency matrix within our optimization framework. Given changes to the adjacency matrix, the Laplacian matrix and its eigenvectors are recomputed. We also note that small perturbations in the Laplacian matrix may lead to large perturbations of some of its eigenvectors – especially in those with high frequency. Hence the methods which make the assumption of small perturbation in $\mathbf{\Psi}$ are not directly applicable to the problem considered in our work.

4.4 Method

We note that if $\mathcal{E}_{\text{actual}}$ was known in Problem 4.1, then it becomes a standard compressed sensing problem (Sec. 4.2.2), with the orthonormal basis $\mathbf{\Psi}$ being equal to the GFT basis of the actual graph, V_{actual}. Hence an estimate of $\boldsymbol{x}^*$ could be recovered via LASSO (Eqn. 4.13) or LASSO-CV (Eqn. 4.18) by putting $\mathbf{\Psi} = V_{\text{actual}}$, with recovery guarantees from compressed sensing theory. A naive method of estimating $\boldsymbol{x}^*$ when $\mathcal{E}_{\text{actual}}$ is not known is to use the GFT matrix of the

nominal graph, V_{nominal} as the orthonormal basis $\mathbf{\Psi}$ in LASSO or LASSO-CV from Eqn. 4.18. Thus we have the following estimates:

$$\hat{\boldsymbol{x}}_{\text{actual}} = \text{LASSO-CV}(\boldsymbol{y}, \mathbf{\Phi}, V_{\text{actual}}, \Gamma), \quad \hat{\boldsymbol{x}}_{\text{nominal}} = \text{LASSO-CV}(\boldsymbol{y}, \mathbf{\Phi}, V_{\text{nominal}}, \Gamma). \tag{4.21}$$

We refer to the technique of using the GFT basis in LASSO as GFT-LASSO. If the actual graph is used, we call it AGFT-LASSO, and if the nominal graph is used, it is called NGFT-LASSO . If cross-validation is used to determine the value of the parameter μ, then these techniques are termed GFT-LASSO-CV , AGFT-LASSO-CV , and NGFT-LASSO-CV , respectively.

Since the set of edges $\mathcal{E}_{\text{nominal}}$ differs slightly from $\mathcal{E}_{\text{actual}}$, the GFT matrix of the nominal graph, V_{nominal} will be a perturbed version of the actual GFT matrix V_{actual}. From CS Theory, for an appropriate value of μ, the estimate $\hat{\boldsymbol{x}}_{\text{nominal}}$ will be an s-term approximation of $\boldsymbol{x}^*$ in the eigenbasis V_{nominal}, as given by Eqn. 4.16. Due to this, the estimate $\hat{\boldsymbol{x}}_{\text{nominal}}$ will not be accurate, depending on the amount of perturbation of the eigenvectors constituting $\boldsymbol{x}^*$. Indeed, $\boldsymbol{x}^*$ may not even be sparse in V_{nominal} if the perturbation is significant, and hence the best s-term approximation itself may yield a large error. In the following subsections, we present methods which use the cross-validation error of GFT-LASSO-CV to select from potential refinements of the nominal graph.

4.4.1 Brute-Force Method

First, we consider a brute-force approach (illustrated in Fig. 4.3 and detailed in Alg. 4.1) which is guaranteed to recover the original signal with high probability. While this brute-force approach is not practical, it will aid in understanding the other (more computationally efficient) algorithms which will follow. We consider only the *noiseless* setting to begin with, i.e. the case where there is no additive noise in the compressive measurements. The main idea is to compare the GFTs of all graphs which are at most d_0 edge perturbations away from the nominal graph $\mathcal{G}_{\text{nominal}}$, using the cross-validation error of the estimated graph signals on a held out set of measurements. Here, a perturbation of some $s = s_1 + s_2$ edges means that a new graph was obtained from the nominal graphs, such that some s_1 edges originally not present in the nominal graph are present in the new graph, and some s_2 edges originally present in the nominal graph

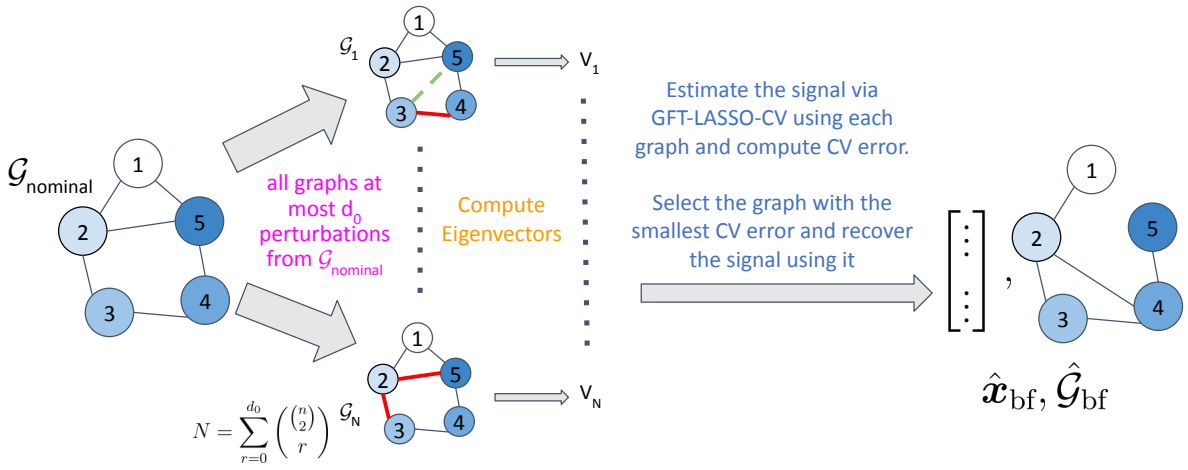


Figure 4.3: Main steps of the Brute-Force algorithm

are not present in the new graph. All other edges which were originally present or absent in the nominal graph are also respectively present or absent in the new graph. The nodes in the new graph are the same as in the nominal graph. From all such graphs which are at most d_0 perturbations away from the nominal graph, the graph with the lowest cross-validation error (see Eqn. 4.18) is chosen, and the signal is estimated using it. For cross-validation, the last m_{cv} rows of $\boldsymbol{\Phi}$ and the corresponding entries of $\boldsymbol{y}$ are held out – they are *not* used for reconstruction but only for computing the validation error, as described in the last paragraph of Sec. 4.2.2 and also mentioned in Eqn. 4.18. (Since the entries of the matrix $\boldsymbol{\Phi}$ are chosen randomly, any m_{cv} rows can be chosen as the validation set. We refer to the last m_{cv} rows only for convenience.) The ideal value of the LASSO regularization parameter μ for each GFT matrix is also found via cross-validation, using grid search over a list of possible values Γ. The algorithm is presented in Alg. 4.1, and the cross-validation method is presented in Alg. 4.2.

Each cross-validation invocation involves solving the LASSO optimization problem once. Since there are $\binom{n}{2}$ possible edges, hence the total number of graphs enumerated via brute-force is $\binom{\binom{n}{2}}{0} + \binom{\binom{n}{2}}{1} + \cdots + \binom{\binom{n}{2}}{d_0} = O(n^{2d_0})$. For grid search over μ, $|\Gamma|$ values are tried. Hence a total of $O(|\Gamma| n^{2d_0})$ LASSO optimization steps are performed by the brute-force algorithm. This is clearly too expensive, which motivates the need for a greedy edge selection procedure.

Algorithm 4.1 Brute-force enumeration

Input: $\boldsymbol{y}$: Compressive measurements, $\boldsymbol{\Phi}$: measurement matrix, $\mathcal{E}_{\text{nominal}}$: nominal graph edge set, d_0 : upper bound on number of edge perturbations needed to obtain the actual graph from the nominal graph, Γ : list of values for grid search of the hyperparameter μ, `CVE` : the cross-validation function, m_{cv} : number of measurements to use for the held-out set

Output: Estimated graph signal $\hat{\boldsymbol{x}}_{\text{bf}}$

1: **for** each edge set $\mathcal{P}$ of size in $\{1, \ldots, d_0\}$ **do**
2: Compute perturbed edgeset $\mathcal{E}_{\mathcal{P}}$ as follows:
3: Initialize $\mathcal{E}_{\mathcal{P}} \leftarrow \mathcal{E}_{\text{nominal}}$
4: **for** each edge $e \in \mathcal{P}$ **do**
5: **if** $e \in \mathcal{E}_{\text{nominal}}$ **then** $\mathcal{E}_{\mathcal{P}} \leftarrow \mathcal{E}_{\mathcal{P}} - \{e\}$
6: **else** $\mathcal{E}_{\mathcal{P}} \leftarrow \mathcal{E}_{\mathcal{P}} \cup \{e\}$
7: **end if**
8: **end for**
9: Compute Laplacian matrix $L_{\mathcal{P}}$ from $\mathcal{E}_{\mathcal{P}}$
10: Compute GFT matrix $V_{\mathcal{P}}$ by eigendecomposition of $L_{\mathcal{P}}$
11: Compute cross-validation loss: $\epsilon_{\text{cv}}^{\mathcal{P}} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V_{\mathcal{P}}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$
12: Compute best value of LASSO regularization parameter:
13: $\mu_{\mathcal{P}} \leftarrow \arg\min_{\mu \in \Gamma} \texttt{CVE}(V_{\mathcal{P}}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$
14: **end for**
15: Compute the best set of perturbations: $\mathcal{P}_{\text{best}} \leftarrow \arg\min_{\mathcal{P}} \epsilon_{\text{cv}}^{\mathcal{P}}$. Let $\mu_{\mathcal{P}_{\text{best}}}$ denote the associate regularization parameter value.
16: Estimate $\hat{\boldsymbol{x}}_{\text{bf}}$ via LASSO using $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $V_{\mathcal{P}_{\text{best}}}$, and $\mu_{\mathcal{P}_{\text{best}}}$:

$$\hat{\boldsymbol{x}}_{\text{bf}} \leftarrow \arg\min_{x \in \mathbb{R}^n} \|\boldsymbol{y} - \boldsymbol{\Phi} x\|_2^2 + \mu_{\mathcal{P}_{\text{best}}} \|V_{\mathcal{P}_{\text{best}}}^T \boldsymbol{x}\|_1$$

17: **return** $\hat{\boldsymbol{x}}_{\text{bf}}$

Algorithm 4.2 Cross validation error computation: CVE

Input: $\boldsymbol{y}$: Compressive measurements, $\boldsymbol{\Phi}$: $m \times n$ Measurement Matrix, V : GFT Matrix, m_{cv} : number of measurements to use for the held-out set, μ : LASSO regularization parameter

Output: Cross-validation error, ϵ_{cv}

1: Let $\boldsymbol{\Phi}_{\text{r}}$ be the first $m_r = m - m_{\text{cv}}$ rows of $\boldsymbol{\Phi}$, and $\boldsymbol{y}_{\text{r}}$ be the first m_{r} entries of $\boldsymbol{y}$

2: Let $\boldsymbol{\Phi}_{\text{cv}}$ be the last m_{cv} rows of $\boldsymbol{\Phi}$, and $\boldsymbol{y}_{\text{cv}}$ be the last m_{cv} entries of $\boldsymbol{y}$

3: Estimate the signal via LASSO: $\hat{x}_r \leftarrow \underset{x \in \mathbb{R}^n}{\arg\min} \|\boldsymbol{y}_r - \boldsymbol{\Phi}_r \boldsymbol{x}\|_2^2 + \mu \|V^T \boldsymbol{x}\|_1$

4: **return** Cross validation error, $\epsilon_{\text{cv}} = \|\boldsymbol{y}_{\text{cv}} - \boldsymbol{\Phi}_{\text{cv}} \hat{\boldsymbol{x}}_r\|_2^2$

4.4.2 Greedy Edge Selection

We propose a greedy exploration (illustrated in Fig. 4.4 and detailed in Alg. 4.3) of the search-space explored by the brute-force algorithm in Sec. 4.4.1. The main idea is to keep refining the edges of a candidate graph – initialized with the edges of the nominal graph – by perturbing one edge at a time, as long as the cross-validation error of the retrieved signal on a held-out set of measurements keeps decreasing. The hope is that each greedy refinement of the candidate graph brings it closer to the actual graph, such that eventually the actual graph as well as the original graph signal are recovered.

The algorithm is presented in Alg. 4.3. The algorithm performs at most d_0 greedy refinement steps. The edges of the candidate graph after greedy refinement step t are referred to as $\mathcal{E}^{(t)}_{\text{candidate}}$. The initial set of edges, $\mathcal{E}^{(0)}_{\text{candidate}}$, is initialized with the edges of the nominal graph, $\mathcal{E}_{\text{nominal}}$. In the greedy step t of the algorithm, all graphs which can be obtained by perturbing one edge of the candidate graph $\mathcal{E}^{(t-1)}_{\text{candidate}}$ (except those which have already been perturbed in greedy steps $1, \ldots, (t-1)$) are considered. There are $\binom{n}{2} - (t-1)$ such graphs. The Laplacian matrices of these graphs are computed. The GFT matrices of these graphs are obtained via eigendecomposition of the respective Laplacian matrices. The ideal value of the LASSO regularization parameter μ is chosen via grid search of the minimum of the cross-validation loss from Eqn. 4.18 over a list of parameter values Γ. For each GFT matrix and each value of the LASSO regularization parameter in Γ, the cross-validation error is computed using the method presented in Alg. 4.2 The edge $e^{(t)}_{\text{best}}$, for which the corresponding GFT matrix has the smallest value of cross-validation error, is greedily chosen in step t, conditioned on the fact

that the cross-validation error shows substantial improvement over the cross-validation error obtained for $\mathcal{E}^{(t-1)}_{\text{candidate}}$. That is, the cross-validation error must decrease by more than a suitably chosen factor $\tau \in (0, 1]$. In such a case, the set of edges $\mathcal{E}^{(t-1)}_{\text{candidate}}$ is perturbed with $e^{(t)}_{\text{best}}$ to obtain $\mathcal{E}^{(t)}_{\text{candidate}}$, and the algorithm proceeds to the next greedy refinement step. Otherwise, the algorithm stops, and an estimate of the signal is returned by performing GFT-LASSO using the GFT matrix obtained from $\mathcal{E}^{(t-1)}_{\text{candidate}}$.

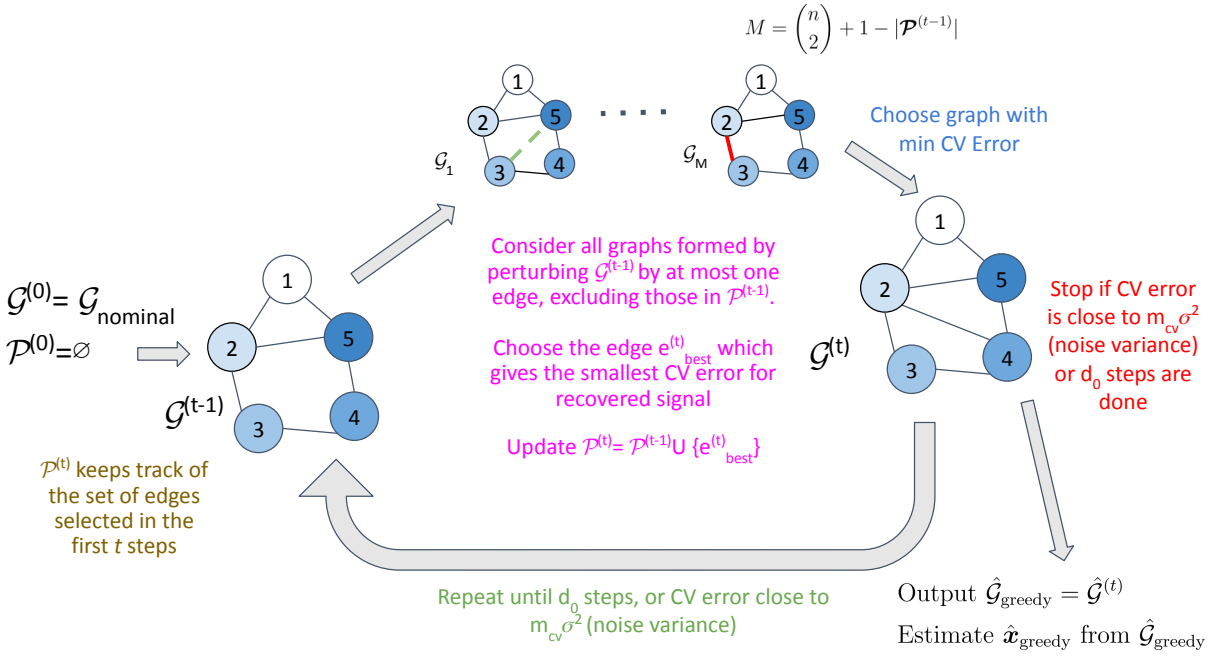


Figure 4.4: Main steps of the Greedy Edge Selection algorithm

Noise-based Stopping Criterion: Additionally, before the beginning of a greedy step, the algorithm checks whether the cross-validation loss is within a high confidence interval of the noise variance σ^2. In this case, the algorithm stops and returns an estimate of the signal by performing LASSO using the GFT matrix obtained from $\mathcal{E}^{(t-1)}_{\text{candidate}}$. Such a stopping criterion is needed in order to prevent estimation of the signal on the noise in the measurements. We note from Eqn. 4.20 that the cross-validation error using the ground-truth signal $\boldsymbol{x}^*$ is:

$$\epsilon^*_{\text{cv}} = \|\boldsymbol{y}_{\text{cv}} - \boldsymbol{\Phi}_{\text{cv}}\boldsymbol{x}^*\|_2^2 = \|\boldsymbol{\eta}\|^2 = \sum_{i=1}^{m_{\text{cv}}} \eta_i^2, \tag{4.22}$$

a sum of m_{cv} random variables η_i^2 for $i \in \{1 \dots m_{\text{cv}}\}$. Since each η_i is i.i.d. Gaussian with mean 0 and variance σ^2, hence η_i^2 are i.i.d. χ^2, with mean σ^2, and standard deviation $\sqrt{2}\sigma^2$. Hence

the expected value of ϵ^*_{cv} is $m_{\text{cv}}\sigma^2$, and its standard deviation is $\sqrt{2m_{\text{cv}}}\sigma^2$. For sufficiently large m_{cv}, the estimator may be taken to be normally distributed, using the Central Limit Theorem. Thus if the cross-validation error is some g standard deviations higher than the expected value of ϵ^*_{cv}, the underlying signal is unlikely to be close to $\boldsymbol{x}^*$, and we continue with the next greedy step, otherwise the algorithm is stopped. That is if at the beginning of the t^{th} step, the CV error $\epsilon^{(t-1)}_{\text{cv}}$ is such that $\epsilon^{(t-1)}_{\text{cv}} \leq (m_{\text{cv}}\sigma^2 + g\sqrt{2m_{\text{cv}}}\sigma^2)$, then the algorithm stops.

Running time: The algorithm performs a maximum of d_0 greedy steps. In each greedy step, GFT-LASSO is performed for a maximum of $\binom{n}{2}$ graphs. Grid search for μ is performed over $|\Gamma|$ values. Hence the total number of GFT-LASSO optimizations performed is $O(|\Gamma|d_0n^2)$, which is a significant improvement over the $O(|\Gamma|n^{2d_0})$ GFT-LASSO optimizations performed by the brute-force algorithm.

Signal Recovery Accuracy: While the greedy algorithm is not guaranteed to recover the original graph and the signal, we find empirically (Sec. 4.6.1) that the mean error in the recovered signal is much lower than that using the nominal graph, and in many cases, the original graph is recovered. In Theorem 4.3, we present conditions under which the solution is guaranteed to improve at any step of the GES algorithm.

4.4.2.1 Practical Modifications

We use a few heuristics to make the greedy edge selection method more practical for real-world settings.

Efficient LASSO regularization parameter selection: In real-world settings, even the greedy edge selection method may be performing too many LASSO optimization steps to be feasible. Hence, instead of choosing the LASSO regularization parameter μ separately for each perturbed graph (steps 20 and 21 in Alg. 4.3), we reuse the value chosen for the candidate set of edges $\mathcal{E}^{(t-1)}_{\text{candidate}}$. That is, steps 20 and 21 of Alg. 4.3 set $\mu = \mu^{(t-1)}$, instead of choosing the minimum over all values of μ in the set Γ. Step 29 of Alg. 4.3 is modified to give $\mu^{(t)}$ by performing cross-validation for all values of μ in the set Γ, with the GFT matrix fixed to $V^{(t)}$, i.e. $\mu^{(t)} \leftarrow \underset{\mu\in\Gamma}{\arg\min}\ \texttt{CVE}(V_e^{(t)}, \mu|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$. The best cross-validation loss for greedy step t is also

Algorithm 4.3 Greedy Edge Selection

Input: $\boldsymbol{y}$: Compressive measurements, $\boldsymbol{\Phi}$: measurement matrix, $\mathcal{E}_{\text{nominal}}$: nominal graph edge set, σ^2 : variance of pool noise, Γ : list of values for grid search of the hyperparameter μ, `CVE` : function to compute the CV error, $\tau \in (0, 1]$: CV error improvement factor, m_{cv} : Number of CV measurements, d_0 : upper bound on number of edge perturbations needed to obtain the actual graph from the nominal graph, g : CV error confidence interval factor

Output: Estimated graph signal $\hat{\boldsymbol{x}}_{\text{greedy}}$

Initialize $\mathcal{E}^{(0)}_{\text{candidate}} \leftarrow \mathcal{E}_{\text{nominal}}$

Initialize Laplacian and GFT matrices: $L^{(0)} \leftarrow L_{\text{nominal}}$ and $V^{(0)} \leftarrow V_{\text{nominal}}$

Initialize cross-validation error : $\epsilon^{(0)}_{\text{cv}} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V^{(0)}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

Initialize LASSO regularization parametere: $\mu^{(0)} \leftarrow \arg\min_{\mu \in \Gamma} \texttt{CVE}(V^{(0)}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

Initialize set of perturbed edges: $\mathcal{P}^{(0)} = \emptyset$

Initialize iteration number to use for final estimate: $T_{\text{est}} \leftarrow 0$

for t in $1, \ldots, d_0$ **do**

  if $\epsilon^{(t-1)}_{\text{cv}} \leq (m_{\text{cv}}\sigma^2 + g\sqrt{2m_{\text{cv}}}\sigma^2)$ i.e. CV error is close to noise **then**

    break, since further improvement is likely to fit noise

  end if

  for each possible edge e which is not in $\mathcal{P}^{(t-1)}$ **do**

    if $e \in \mathcal{E}^{(t-1)}_{\text{candidate}}$ **then**

      Remove edge e to get the set of edges of the perturbed graph:

      $\mathcal{E}^{(t)}_e \leftarrow \mathcal{E}^{(t-1)}_{\text{candidate}} - \{e\}$

    else

      Add edge e to get the set of edges of the perturbed graph: $\mathcal{E}^{(t)}_e \leftarrow \mathcal{E}^{(t-1)}_{\text{candidate}} \cup \{e\}$

    end if

    Compute the Laplacian matrix of perturbed graph, $L^{(t)}_e$, from $\mathcal{E}^{(t)}_e$

    Compute the GFT matrix of perturbed graph, $V^{(t)}_e$, by eigendecomposition of $L^{(t)}_e$

    Compute the best cross-validation loss over Γ using $V^{(t)}_e$:

    $\epsilon^{(e,t)}_{\text{cv}} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V^{(t)}_e, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

    Compute the best LASSO regularization parameter value using $V^{(t)}_e$:

    $\mu^{(t)}_e \leftarrow \arg\min_{\mu \in \Gamma} \texttt{CVE}(V^{(t)}_e, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

  end for

23: **if** $\min_e \epsilon_{\text{cv}}^{(e,t)} < \tau \epsilon_{\text{cv}}^{(e,t-1)}$ **then**

24: Select edge: $e_{\text{best}}^{(t)} \leftarrow \arg\min_e \epsilon_{\text{cv}}^{(e,t)}$

25: Update best CV error: $\epsilon_{\text{cv}}^{(t)} \leftarrow \min_e \epsilon_{\text{cv}}^{(e,t)}$

26: Update perturbed edge set: $\mathcal{P}^{(t)} \leftarrow P^{(t-1)} \cup \{e_{\text{best}}^{(t)}\}$

27: Update candidate edge set: $\mathcal{E}_{\text{candidate}}^{(t)} \leftarrow \mathcal{E}_{e_{\text{best}}^{(t)}}^{(t)}$

28: Update Laplacian and GFT matrices: $L^{(t)} \leftarrow L_{e_{\text{best}}^{(t)}}^{(t)}$ and $V^{(t)} \leftarrow V_{e_{\text{best}}^{(t)}}^{(t)}$

29: Update hyperparameter: $\mu^{(t)} \leftarrow \mu_{e_{\text{best}}^{(t)}}^{(t)}$

30: **else**

31: **break**, since solution doesn't improve significantly.

32: **end if**

33: Update iteration number to be used for $\hat{\boldsymbol{x}}_{\text{greedy}}$: $t_{\text{est}} \leftarrow t$

34: **end for**

35: Estimate $\hat{\boldsymbol{x}}_{\text{greedy}}$ via LASSO from $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{V}^{(t_{\text{est}})}$, and $\mu^{(t_{\text{est}})}$ using all the measurements:

$$\hat{\boldsymbol{x}}_{\text{greedy}} \leftarrow \arg\min_{x \in R^n} \|\boldsymbol{y} - \boldsymbol{\Phi} x\|_2^2 + \mu^{(t_{\text{est}})} \|V^{(t_{\text{est}})T} \boldsymbol{x}\|_1$$

36: **return** $\hat{\boldsymbol{x}}_{\text{greedy}}$

modified accordingly, i.e., step 25 of Alg. 4.3 becomes $\epsilon_{\text{cv}}^{(t)} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V_e^{(t)}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$. Due to this modification, each greedy step needs to perform only $O(|\Gamma| + n^2)$ LASSO optimizations instead of $O(|\Gamma| n^2)$ LASSO optimizations. However, since the best value of μ is not used for the best edge at step t for comparing CV errors, the error in the signal recovered using it may be higher than if μ was chosen by grid search. Consequently, the CV error for the best edge may also be higher. Hence, the probability that an incorrect edge gets selected at a greedy step increases due to this modification. Hence the final recovered signal may have higher error than if μ was chosen using grid search for each edge.

Efficient Edge Selection based on prior knowledge: In some real-world problems, it may be known as to which kinds of edges have a higher probability of having been added to or removed from the actual graph to create the nominal graph. For example, in the case of contact tracing for monitoring the spread of an infectious disease in a population, there might be a chance that if the phones of two people use the same WiFi access points, then those people may have come in contact, even though the Bluetooth-based contact tracing may suggest that those people may not have come in contact (e.g. by being located in the same office room at

Algorithm 4.4 Multi-fold Cross-validation

Input: $\boldsymbol{y}$: Compressive measurements, $\boldsymbol{\Phi}$: $m \times n$ Measurement Matrix, V : GFT Matrix, F : number of cross-validation folds, μ : LASSO regularization parameter

Output: Multi-fold Cross-validation error

1: Let $m_{\text{cv}} = \left\lfloor \frac{m}{F} \right\rfloor$

2: **for** $f \in \{1, \ldots, F\}$ **do**

3: Let $\boldsymbol{\Phi}_{\text{cv}}^{(f)}$ be the m_{cv} rows of $\boldsymbol{\Phi}$ with indices $\{(f-1)m_{\text{cv}} + 1, \ldots, fm_{\text{cv}}\}$, and $\boldsymbol{y}_{\text{cv}}^{(f)}$ be the corresponding m_{cv} entries of $\boldsymbol{y}$

4: Let $\boldsymbol{\Phi}_{\text{r}}^{(f)}$ be the remaining $m_r = m - m_{\text{cv}}$ rows of $\boldsymbol{\Phi}$, and $\boldsymbol{y}_{\text{r}}^{(f)}$ be the corresponding m_{r} entries of $\boldsymbol{y}$

5: Estimate the signal via LASSO using the remaining entries:

$$\hat{\boldsymbol{x}}_r^{(f)} \leftarrow \underset{\boldsymbol{x} \in R^n}{\arg\min} \, \|\boldsymbol{y}_r^{(f)} - \boldsymbol{\Phi}_r^{(f)} \boldsymbol{x}\|_2^2 + \mu \|V^T \boldsymbol{x}\|_1$$

6: **end for**

7: **return** $\sum_{f=1}^{F} \frac{1}{F m_{\text{cv}}} \|\boldsymbol{y}_{\text{cv}}^{(f)} - \boldsymbol{\Phi}_{\text{cv}}^{(f)} \hat{\boldsymbol{x}}_r^{(f)}\|_2^2$

a far-enough distance). In some other problems, certain kinds of edge perturbations might be more important than others. For example, if the graph has a community structure and the graph signal is band-limited in the GFT domain, the perturbations of intra-cluster edges may not lead to significant perturbations of the relevant (low-frequency) eigenvectors, and only perturbations of the inter-cluster edges may be important. In some other problems, it might be that edges only got *added* to the the actual graph to create the nominal graph (for example, see the edge-aware compressive image acquisition problem in Sec. 4.4.3), hence the only kinds of perturbations of the nominal graph that need to be considered are those which *remove* existing edges of the nominal graph. We modify our algorithm to take advantage of such prior knowledge by only considering perturbations of edges from a known set of edges Ω instead of all possible edges (step 10 of Alg. 4.3). With this modification, along with the heuristic for LASSO regularization parameter selection, only $O(|\Gamma|+|\Omega|)$ LASSO optimizations may be performed by the algorithm in each greedy edge selection step.

Multi-fold Cross-validation: While we assume that the number of linear measurements m is large enough such that m_{cv} measurements can be held-out for cross-validation, this may not be the case if n is small. In such cases, if too many measurements are used for cross-

validation, good quality recovery using the remaining (reconstruction) measurements may not be guaranteed by compressed sensing. On the other hand, if too few measurements are used for cross-validation, then the cross-validation error may not be a good approximation of the signal error (see Sec. 4.4.4 for relevant discussion). In such cases, we may resort to performing multi-fold cross-validation. In F-fold cross-validation, cross-validation is performed F times, each time with a different subset of the m measurements, and the average of the F cross-validation errors is returned. The measurement matrix $\mathbf{\Phi}$ and the vector of measurements $\boldsymbol{y}$ are divided into F folds each of size $\left\lfloor \frac{m}{F} \right\rfloor$. For each fold $f \in \{1, \ldots, F\}$, the rows $\{(f-1)\left\lfloor \frac{m}{F} \right\rfloor + 1, \ldots, f\left\lfloor \frac{m}{F} \right\rfloor\}$ of $\mathbf{\Phi}$ and the corresponding entries of $\boldsymbol{y}$ are held-out for computing the cross-validation error, whereas the remaining entries of $\boldsymbol{y}$ and the corresponding rows of $\mathbf{\Phi}$ are used for estimation of the signal via LASSO. The details of multi-fold cross-validation are provided in Alg. 4.4. If multi-fold cross-validation is used, then the number of folds F is passed to the greedy edge selection algorithm (Alg. 4.3), instead of m_{cv} (the number of measurements to be used for single-fold cross-validation). The stopping criterion in step 6 of Alg. 4.3 is modified to $\epsilon_{\text{cv}}^{(t-1)} \leq (\widetilde{m}\sigma^2 + g\sqrt{2\widetilde{m}}\sigma^2)$ since a total of $\widetilde{m} \triangleq F\left\lfloor \frac{m}{F} \right\rfloor$ measurements are used to compute the multi-fold cross-validation error. We note that the cross-validation error of some fold a is not independent of the cross-validation error of some other fold b, since the measurements used for computing the cross-validation error in one fold are included in the measurements used to estimate the signal in the other fold. Hence the Central Limit Theorem may not directly apply in this case (though there are variants of the Central Limit Theorem with weak dependence). However, in Sec. 4.6.1 and Sec. 4.5.2, we find that multi-fold cross-validation works well in practice, both for selection of the best GFT, as well as for the stopping criterion.

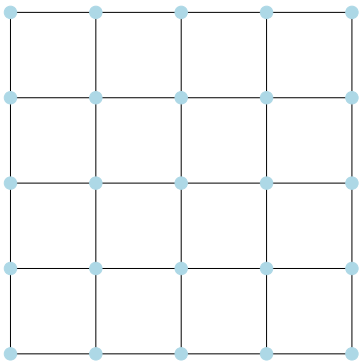
4.4.3 Inferred Linear-Edge Compressive Image Recovery

In this section, we use the term 'edge' in two ways – as an element of a graph connecting a pair of vertices, or as an image edge which separates two regions. For the sake of disambiguation, we will use the term 'image-edge' for the latter, and simply use the word 'edge' in the former case. The greedy edge selection method presented in Sec. 4.4.2 is a generally applicable technique on any graph. However, if the graph has some well-known structure to it, it may be possible to select the edges for perturbation in a more structured manner, instead of greedily one at a

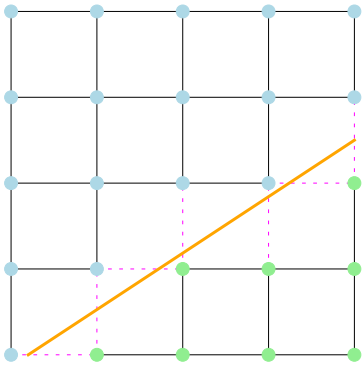
time. In this section, we present such a method for image reconstruction from compressive measurements. Any 2-D (two-dimensional) grayscale raster image can be considered to be a graph signal, with each pixel being a node and horizontal and vertical neighbours connected to each other via an edge, with each pixel node mapped to its value in the image. Such a graph is called a 'lattice graph' (Fig. 4.5(a)). The 2-D Discrete Cosine Transform (DCT) basis (Fig. 4.5(c)) – a commonly-used basis for sparse encoding of natural images – is known to form an eigenbasis of the Laplacian matrix of a 2-D lattice graph, even though it is not a unique eigenbasis due to eigenvalue multiplicity [52, Proposition 1].

Each edge of a graph naturally induces a correlation between the intensity values at the different nodes that it links together. Pixel values are spatially correlated in many image domains, such as natural images, and also in depth-maps or other piece-wise smooth images. Such spatial correlation can be exploited for compression by encoding the image in the DCT basis and keeping only the largest few coefficients. That is, an image of size $h \times w$ may be represented by only $k \ll hw$ coefficients of the DCT basis. This fact is in fact heavily exploited in the well known JPEG standard for image compression [51]. If compressive acquisition of such images is performed, the sparsity or compressibility of images in the DCT basis is a useful prior for reconstruction. That is, instead of acquiring data for hw pixels for an $h \times w$ image, only $m \ll hw$ linear combinations of the hw pixels may be acquired. Later, recovery of the original image from these m linear measurements may be done using a compressed sensing decoding algorithm such as LASSO using the DCT basis, exploiting the fact that the image is compressible in this basis. Compressive image acquisition saves on the number of sensor hardware elements needed to capture an image, thus lowering the cost of image acquisition (at the cost of some computation needed for decoding).

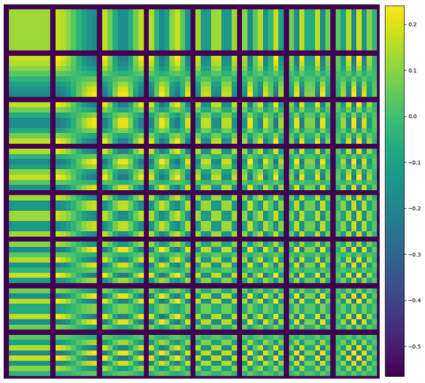
If the image being acquired has a sharp image-edge in it, then the pixel values across the image-edge will not be correlated to each other, even though they are spatially close to each other. Since the DCT-based compressive image acquisition takes advantage of correlation in pixel values which are spatially close to each other, DCT-based decoding of compressively acquired images which contain a sharp image-edge will not be accurate near the image-edge. In such cases, it may be better to use some other transform basis whose vectors maintain no correlation in their values across the image edge. Since the DCT basis is the GFT basis of the 2-D lattice graph, one possible basis is the GFT basis of the graph obtained by dropping those



(a) A 5×5 2-D lattice graph

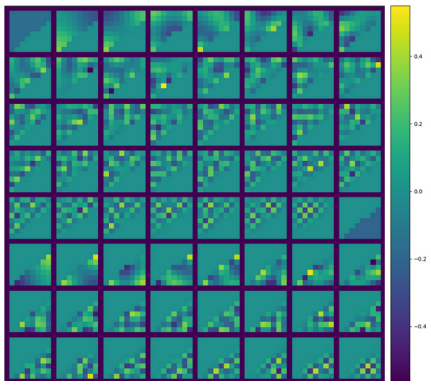


(b) An image-edge partitioned graph



(c) DCT basis vectors for an 8×8 patch

(d) A Patch with a sharp edge



(e) Segmentation-aware basis vectors

Figure 4.5: (a) A 5×5 2-D lattice graph. The nodes represent pixels of a 5×5 patch, which are connected to the four neighbouring pixels. This forms the nominal graph for the problem of recovery of an image patch from compressive measurements. (b) the lattice graph partitioned by an image edge (orange line). The graph edges going across the image edge are removed (dotted purple lines). Since the image edge is unknown before reconstruction, the image-edge partitioned graph is the (unknown) actual graph for the problem of recovery of an image patch from compressive measurements. (c) All 64 2-D DCT basis vectors of an 8×8 patch. (d) An 8×8 patch with a sharp edge (e) Segmentation-aware basis vectors for this patch, obtained by computing the eigenvectors of the Laplacian matrix of the graph created by dropping the edges of the 8×8 lattice graph whose endpoints lie in different segments of the patch.

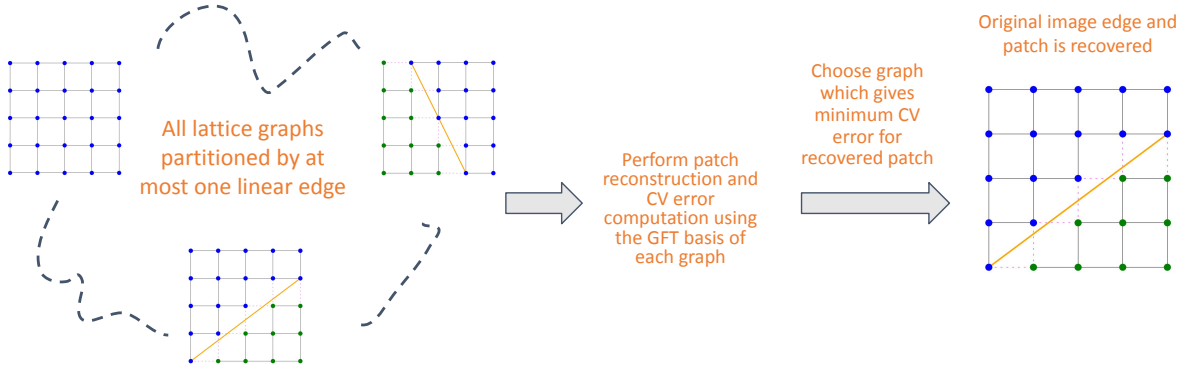


Figure 4.6: Illustration of the ILECIR algorithm (see Alg. 4.5)

edges of the 2-D lattice graph which link nodes that are located on two different sides of an image-edge. We may call this graph the 'image-edge partitioned graph' (see Fig. 4.5(b)). However, since the image is acquired compressively, it is not straightforward to know the location of the image-edges, and hence the image-edge partitioned graph is unknown. Hence the problem of decoding of the compressively acquired image using the GFT of the unknown image-edge-partitioned graph while knowing only the 2D lattice graph may be formulated as Problem 4.1. The 2D lattice graph is the nominal graph, the set of pixel values is the unknown graph signal, and the image-edge partitioned graph is the actual graph. The actual graph is obtained by perturbation of the nominal graph, i.e. by dropping appropriate edges of the nominal graph. Note that the graph signal is sparse or compressible in the GFT of the *actual* graph.

One may apply the greedy algorithm from Sec. 4.4.2 to solve this problem. However, the graph edges which should be dropped from the nominal graph to obtain the actual graph possess some additional structure. For example, image-edges have an inherent smoothness. Hence, it may be better to drop the edges in a different manner than what the greedy algorithm from Sec. 4.4.2 suggests. In Alg. 4.5 (also illustrated in Fig. 4.6), we present a method for patch-wise edge-aware compressive image recovery. The image is assumed to have been acquired in a patch-by-patch manner, each patch of size $h \times w$, with each patch being acquired compressively. This follows the architecture of a block-based version [62, 63] of the Rice Single Pixel Camera discussed in Sec. 4.2.2 [53]. Each image patch is assumed to either contain no image-edge or at most a single linear image-edge, i.e., a straight line with its endpoints at the boundary of the patch. While this assumption might not be perfect, it is a reasonable approximation for small-sized patches, and will yield better results than assuming no image-edge as done by the

Algorithm 4.5 Inferred Linear-Edge Compressive Image Recovery (ILECIR)

Input:

$\boldsymbol{y}$: vector of m Compressive measurements of the image patch of size $h \times w$,

$\boldsymbol{\Phi}$: $m \times n$ Measurement Matrix where $n = hw$ is the number of pixels in the patch and $m < n$; with columns of the matrix corresponding to the pixels in row-major order,

V_{DCT}: matrix of 2-D DCT basis vectors of an $h \times w$ patch; rows indices of this matrix correspond to pixels in the patch in row-major order,

ϕ : set of all possible linear image edges in an $h \times w$ patch with endpoints at the boundary,

$\mathcal{V}_{h,w}$: set of precomputed GFT matrices for all possible linear-image-edge-partitioned graphs for an $h \times w$ patch (each graph represents a particular partition of the patch via a single linear image edge),

σ^2 : variance of pool noise,

Γ : list of values for grid search of the hyperparameter μ,

`CVE` : the cross-validation function,

m_{cv} : number of measurements to use for the held-out set,

$\tau \in (0, 1]$: ratio by which the CV error should improve over DCT for a image-edge-partitioned graph to be selected.

g : CV error confidence interval factor

Output: $\hat{P}$: estimated image patch of size $h \times w$

Compute cross-validation error using DCT: $\epsilon_{\text{cv}}^{\text{DCT}} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V_{\text{DCT}}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

Compute LASSO regularization parameter using DCT:

$$\mu_{\text{DCT}} \leftarrow \arg\min_{\mu \in \Gamma} \texttt{CVE}(V_{\text{DCT}}, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$$

if $\epsilon_{\text{cv}}^{\text{DCT}} \leq (m_{\text{cv}}\sigma^2 + g\sqrt{2m_{\text{cv}}}\sigma^2)$ i.e. CV error is close to noise **then**

    $V_{\text{est}} \leftarrow V_{\text{DCT}}$

    $\mu_{\text{est}} \leftarrow \mu_{\text{DCT}}$

else

    for Each linear image edge $i \in \phi$ **do**

        Retrieve precomputed GFT matrix $V_i \in \mathcal{V}_{h,w}$ corresponding to i

        Compute best cross-validation error using i: $\epsilon_{\text{cv}}^{(i)} \leftarrow \min_{\mu \in \Gamma} \texttt{CVE}(V_i, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$

        Compute best LASSO regularization parameter using i:

$$\mu_i \leftarrow \arg\min_{\mu \in \Gamma} \texttt{CVE}(V_i, \mu | \boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$$

    end for

if $\min_i \epsilon_{\text{cv}}^{(i)} < \tau \epsilon_{\text{cv}}^{\text{DCT}}$ **then**

$V_{\text{est}} \leftarrow V_i$

$\mu_{\text{est}} \leftarrow \mu_i$

end if

end if

Estimate the vectorized patch using LASSO: $\hat{\boldsymbol{x}}_{\text{ilecir}} \leftarrow \underset{x \in R^n}{\arg\min} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu_{\text{est}} \|V_{\text{est}}^T \boldsymbol{x}\|_1$

Convert $\hat{\boldsymbol{x}}_{\text{ilecir}}$ to $h \times w$ patch $\hat{P}$, assuming row-major order

return $\hat{P}$

DCT implicitly. For each such linear image-edge, we consider the graph formed by dropping those graph edges of the 2-D lattice graph of size $h \times w$ whose nodes lie on opposite sides of the image-edge. That is, the lattice graph is partitioned into two components along the linear image edge, and the GFT of this partitioned graph is considered. Reconstruction of the patch is performed using the GFTs corresponding to each image-edge (as well as the nominal graph) using the reconstruction subset of the compressive measurements for that patch. Here again, we use the cross-validation technique to not only determine the best regularization parameter, but also the best graph representation. In other words, the GFT corresponding to the estimated signal with the lowest cross-validation error is selected. The final estimation of the signal is performed on all the available measurements (including those which were previously held out for cross-validation) using the selected image-edge and the selected regularization parameter. The complete image is output by stitching together all the reconstructed patches.

The noise-based stopping criterion used in Alg. 4.3 is also used in the edge-aware compressive image reconstruction algorithm. The algorithm first performs decoding using the GFT of the nominal graph (i.e. the 2-D DCT). If the cross-validation error using the DCT is within two-standard-deviations of its expected value, then recovery using image-edge-partitioned graphs is not performed. The DCT is used for final patch recovery in this case. The complete procedure is presented in Alg. 4.5.

We also incorporate the following practical modifications as discussed in Sec. 4.4.2.1. Firstly, multi-fold cross-validation is used instead of using only one fold for cross-validation, in order to handle small number of available measurements. Secondly, the LASSO regularization parameter μ is chosen only for the nominal graph. For comparing the image-edge-partitioned

graphs, the value of μ used for the nominal graph is reused. If a graph other than the nominal graph is selected, then for the final estimation of the signal, a better value of μ is selected via cross-validation. Hence, step 9 of Alg. 4.5 becomes $\epsilon_{\text{cv}}^{(i)} \leftarrow \texttt{CVE}(V_i, \mu_{\text{DCT}}|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$, step 10 is deleted, and in step 13, $\mu_{\text{est}} \leftarrow \underset{\mu \in \Gamma}{\arg\min}\, \texttt{CVE}(V_{\text{est}}, \mu|\boldsymbol{y}, \boldsymbol{\Phi}, m_{\text{cv}})$.

The idea of using the GFT of a modified lattice graph has been used in [166] for compression of piecewise smooth images. In this application, however, there is access to the entire graph. Graph edges whose nodes have significantly different intensities are deleted, and the GFT of the thus modified lattice graph are used as a representation basis for compression, yielding superior results as compared to DCT or the GFT of a standard lattice graph. We wish to emphasize that in compressed sensing, we do not have access to the graph, nor do we have to access to the intensity values at each pixel. Instead, these must be inferred on the fly from the compressive measurements.

4.4.3.1 Procedure for Choice of Linear Image-Edges

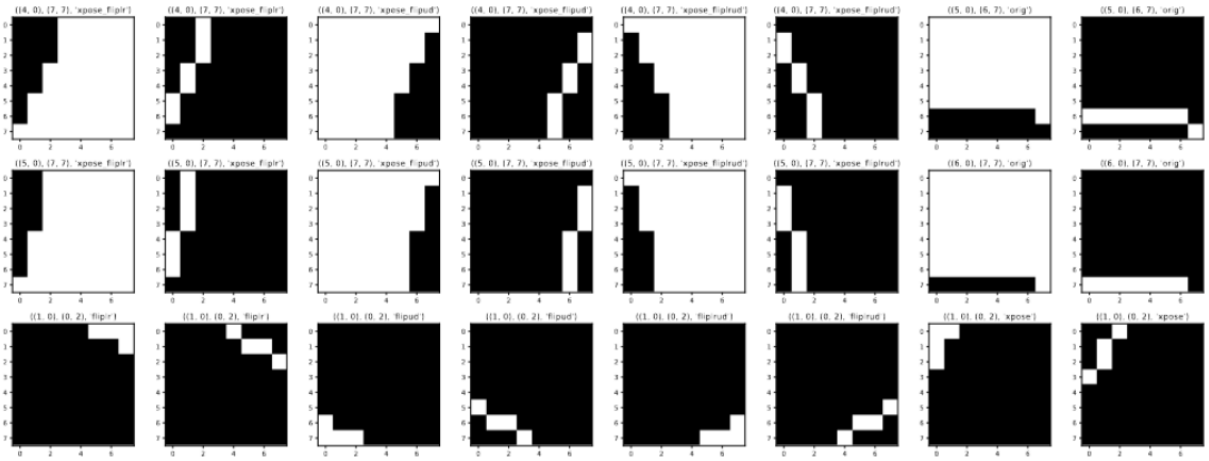


Figure 4.7: Some 8×8 patch segmentations (cols 1, 3, 5, 7) generated by linear edges (cols 2, 4, 6, 8).

The linear image-edges are chosen according to the following algorithm. All pairs of points $\boldsymbol{a} := (x_1, y_1)$ and $\boldsymbol{b} := (x_2, y_2)$, such that $\boldsymbol{a}$ and $\boldsymbol{b}$ are on different boundaries of the patch, are enumerated. For each such pair we have a straight line joining two boundary pixels (x_1, y_1) and (x_2, y_2). The patch is segmented into two regions – the pixels which satisfy $(x_2 - x_1)(y - y_1) \geq (y_2 - y_1)(x - x_1)$ form the first region, whereas the remaining pixels form the second region. Reflected versions of this segmentation are also considered – in general there are 8 such reflections (original, horizontal reflection, vertical reflection, horizontal and

vertical reflection, transposed, and the three reflections of the transposed version). We ignore the possible segmentation given by the equation $(x_2 - x_1)(y - y_1) > (y_2 - y_1)(x - x_1)$ (strict inequality) so as to keep the number of considered image edges small. Across all enumerated points $\boldsymbol{a}$ and $\boldsymbol{b}$, only unique segmentations are kept.

For each segmentation, an adjacency matrix is created by dropping the edges of the 2-D lattice graph whose two endpoints lie in two different segments. We compute the eigenvectors of Laplacian matrices of the two segments in each patch separately, and compute the eigenvectors of the Laplacian matrix of the complete segmented patch from those. Note that the eigenvectors of the Laplacian matrix of a graph with more than one connected components are the same as the eigenvectors of each component, with the entries corresponding to nodes of the remaining components set to zero. This is done in order to avoid introducing correlations between the two segments, which may happen in case there are repeated eigenvalues. Notably, the eigenvalue of 0 gets repeated k times if there are k connected components in a graph. These computed GFTs are used in Alg. 4.5.

4.4.4 Recovery Guarantees and Bounds

Let $\hat{\boldsymbol{x}}_{\mathcal{G}}$ denote the signal recovered using GFT-LASSO-CV with the GFT of graph $\mathcal{G}$. We present the following recovery guarantee for the brute-force algorithm presented in Alg. 4.1:

Theorem 4.2 (Brute-Force Algorithm Recovery Guarantee)**.** *If in the brute force algorithm in Algorithm 4.1 the number of CV measurements m_{cv} obeys*

$$m_{cv} \;\geq\; 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln(d_0 + 1) + 2d_0 \ln n + \ln\frac{1}{\delta}\Big\} \tag{4.23}$$

for arbitrary constants $c \in (1, \infty)$ and $\delta \in (0, 1)$, then its recovery error is bounded as

$$\|\hat{\boldsymbol{x}}_{bf} - \boldsymbol{x}^*\|_2^2 \;<\; c\|\hat{\boldsymbol{x}}_{actual} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{4.24}$$

with probability more than $1-\delta$*. As a consequence, if*

$$\|\hat{\boldsymbol{x}}_{\mathcal{G}} - \boldsymbol{x}^*\|_2^2 \;\geq\; c\|\hat{\boldsymbol{x}}_{actual} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{4.25}$$

for all graphs $\mathcal{G} \neq \mathcal{G}_{actual}$ *which are upto* d_0 *perturbations from* $\mathcal{G}_{nominal}$*, then with probability more than* $1-\delta$*,* $\hat{\boldsymbol{x}}_{bf} = \hat{\boldsymbol{x}}_{actual}$*, and the actual graph is recovered.*

The proof is provided in the appendix. It is based on a theorem from [60] regarding how well cross-validation error predicts the recovery error for compressed sensing with Gaussian random matrices. We use this theorem and the union bound to lower bound the probability that the CV error of signal recovered using GFT-LASSO-CV with the actual graph is lower than all CV errors for signals which have a recovery error higher than that specified in Eqn. 4.24.

The result shows that there is a tradeoff between the number of CV measurements used and the quality of recovery, encapsulated by the parameter c. If c is close to 1, the recovery error is close to or equal to that achieved using GFT-LASSO-CV on the actual graph. However in such a case, m_{cv} will be large. On the other hand, a large value of c will allow for a smaller value of m_{cv} – however in that case, the recovered signal may have error higher than if the actual graph was known. It also illustrates that if the recovery error using the nominal graph is close to that using the actual graph or if the noise level is too high, then it is harder to distinguish between the two. This may be case when perturbing the actual graph did not significantly perturb the eigenvectors on which $\boldsymbol{x}^*$ had support. Dependence of the bound on d_0 in Eqn. 4.23 shows that the method works well if the nominal graph and the actual graph are only a few perturbations away from each other. In particular, if a nominal graph were not known, then the brute-force algorithm must enumerate all graphs, making $d_0 = \binom{n}{2}$. In this case, $m_{\text{cv}} = \Omega(n^2 \ln n)$, and we are no longer in the regime of compressive sensing. The dependence of the bound in Eqn. 4.23 on $|\Gamma|$ seems to be an artifact of our proof technique. Our proof does not exploit the fact that there may be many 'bad' graphs in the search space, for which the recovery error (and the CV error) will be high regardless of the value of $\mu \in \Gamma$ used. Instead, the proof proceeds by assuming that the CV errors for a 'bad' graph with different values of μ are not correlated, leading to an overestimation of the bound for m_{cv}. Perhaps some other proof technique might be used which removes or weakens the dependence of the bound on $|\Gamma|$.

Remark on Graph Recovery: While the brute-force method is guaranteed to recover the signal, we do not know if it always recovers the actual graph, even in the absence of measurement noise. For example, there might be cases wherein a graph signal has a sparser or equally-sparse representation in the GFT basis of a graph which is not the actual graph, which might get chosen by the algorithm. In particular, it is known that the eigenvectors of the Laplacian matrix of a graph and its complement graph[1] are the same. The author does not know if there exist two graphs which are a small number of perturbations from each other and share some eigenvectors or have eigenvectors which are the linear combination of a small number of the eigenvectors of the other graph. Interestingly, the signal recovery bound in Eqn. 4.24 does not depend on the structure of the graph – it only depends on m_{cv} and σ.

Analogous to Theorem 4.2, we present two theorems for solution improvement using the greedy GES method (Algorithm 4.3) and ILECIR (Algorithm 4.5). Let $\hat{\boldsymbol{x}}^{(t)}$ be the estimate by the greedy edge selection algorithm after t steps, and let $\hat{\boldsymbol{x}}^{(t)}_{\text{best}}$ be the estimate with the lowest recovery error amongst the signals recovered at step t.

Theorem 4.3 (Greedy Edge Selection Solution Improvement Guarantee)**.** *If in the greedy edge selection algorithm (Algorithm 4.3) with $\tau = 1$ the number of CV measurements m_{cv} obeys*

$$m_{cv} \;\geq\; 4\Big(1+\frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln\frac{n(n+1)}{2} + \ln\frac{1}{\delta}\Big\} \tag{4.26}$$

for arbitrary constants $c \in (1,\infty)$ and $\delta \in (0,1)$, then the recovery error after step t is bounded as

$$\|\hat{\boldsymbol{x}}^{(t)} - \boldsymbol{x}^*\|_2^2 \;<\; c\|\hat{\boldsymbol{x}}^{(t)}_{best} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{4.27}$$

with probability more than $1-\delta$. As a consequence, if

$$\|\hat{\boldsymbol{x}}^{(t-1)} - \boldsymbol{x}^*\|_2^2 \;\geq\; c\|\hat{\boldsymbol{x}}^{(t)}_{best} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{4.28}$$

then the solution is guaranteed to improve at step t with probability more than $1-\delta$.

[1]The complement of a graph is a graph in which every edge present in the original graph is absent in the complement graph and every edge which is absent in the original graph is present in the complement graph.

Let $\hat{\boldsymbol{x}}_{\text{best-edge}}$ denote the (vectorized) patch estimate with the lowest recovery error amongst all the patch estimates in the ILECIR algorithm for a single patch.

Theorem 4.4 (ILECIR Solution Improvement Guarantee)**.** *If in the* ILECIR *algorithm (Algorithm 4.3) with* $\tau = 1$ *the number of CV measurements* m_{cv} *obeys*

$$m_{cv} \geq 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln(|\phi|+1) + \ln\frac{1}{\delta}\Big\} \tag{4.29}$$

for arbitrary constants $c \in (1,\infty)$ *and* $\delta \in (0,1)$ *and where* ϕ *is the set of all possible linear image edges in an* $h \times w$ *patch with endpoints at the boundary, then the recovery error is bounded as*

$$\|\hat{\boldsymbol{x}}_{ilecir} - \boldsymbol{x}^*\|_2^2 < c\|\hat{\boldsymbol{x}}_{best\text{-}edge} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2 \tag{4.30}$$

with probability more than $1-\delta$*. As a consequence, if*

$$\|\hat{\boldsymbol{x}}_{dct} - \boldsymbol{x}^*\|_2^2 \geq c\|\hat{\boldsymbol{x}}_{best\text{-}edge} - \boldsymbol{x}^*\|_2^2 + (c-1)\sigma^2, \tag{4.31}$$

then the solution is guaranteed to improve over DCT-LASSO-CV *with probability more than* $1-\delta$.

We omit the proofs for Theorems 4.3 and 4.4 since they are very similar to that for Theorem 4.2. The main difference is in the lower bound for m_{cv}, which is due to the different number of perturbed graphs considered in each algorithm.

4.4.5 Alternatives to Eigendecomposition for GFT basis computation

The Greedy Edge Selection algorithm requires performing the eigendecomposition (step 16 in Alg. 4.3) of the Laplacian matrix $L_e^{(t)}$ of the perturbed graph obtained by perturbing the candidate edge set $\mathcal{E}_{\text{candidate}}^{(t-1)}$ with edge e, for each possible edge e, at each time step t. This step may take a long time for large graphs. It may be made more efficient by using an approximate formula given by Ceci and Barbarossa in [153], to obtain the eigenvectors of $L_e^{(t)}$ from

the eigenvectors of L_{nominal}, and the set of perturbations $\mathcal{P}^{(t-1)} \cup \{e\}$, instead of performing eigendecomposition of $L_e^{(t)}$. If an original Laplacian matrix L is perturbed by a set of edges $\mathcal{P}$ to obtain a perturbed Laplacian matrix $\tilde{L}$, and their eigenvectors are $\boldsymbol{v}_1 \dots \boldsymbol{v}_n$ and $\tilde{\boldsymbol{v}}_1 \dots \tilde{\boldsymbol{v}}_n$ respectively, then from [153, Eqn. 7 and 11] we have the approximation:

$$\tilde{\boldsymbol{v}}_k \simeq \boldsymbol{v}_k + \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j)) \sum_{\substack{l=2 \\ l\neq k}}^{n} \frac{\boldsymbol{v}_l(i) - \boldsymbol{v}_l(j)}{\lambda_k - \lambda_l} \boldsymbol{v}_l, \tag{4.32}$$

where $\sigma_{i,j} = 1$ for edge addition, and $\sigma_{i,j} = -1$ for edge deletion. The approximation for $\tilde{\boldsymbol{v}}_k$ is valid only under the condition that

$$\lambda_{k-1} - \lambda_k \ll \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 \ll \lambda_{k+1} - \lambda_k. \tag{4.33}$$

We discuss some intuition behind this approximation and the condition under which it is valid. First, note that each new edge $\{i,j\}$ in $\mathcal{P}$ introduces a rank-one perturbation of L with the matrix $\sigma_{i,j}\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T$, where $\boldsymbol{a}_{i,j}$ is a vector with $\boldsymbol{a}_{i,j}(i) = 1, \boldsymbol{a}_{i,j}(j) = -1$, and the remaining entries of $\boldsymbol{a}_{i,j}$ are equal to zero. That is, $\tilde{L} = L + \Delta L$, where the perturbing matrix $\Delta L = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T$. Note that $\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = (\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))$ and hence $\boldsymbol{v}_k^T\Delta L\boldsymbol{v}_k = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}\boldsymbol{v}_k^T\boldsymbol{a}_{i,j}\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = \sum_{\{i,j\}\in\mathcal{P}} \sigma_{i,j}(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2$. It is easily verified that if $\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k = 0$ (i.e. $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j)) = 0$) for all the perturbing edges $\{i,j\} \in \mathcal{P}$, then $\boldsymbol{v}_k$ is also an eigenvector of $\tilde{L}$. Recall that the normalized eigenvectors of a matrix L are the critical points of the Rayleigh quotient $\boldsymbol{x}^T L\boldsymbol{x}$ on the unit sphere $\boldsymbol{x}^T\boldsymbol{x} = 1$, with the eigenvalue being the value of the Rayleigh quotient at the corresponding eigenvector. Eqn. 4.33 means that the perturbation of the Rayleigh quotient at $\boldsymbol{v}_k$ must be much smaller than the difference in values of the Rayleigh quotient at the closest critical points. In Eqn. 4.32, the perturbation of $\boldsymbol{v}_k$ by edge $\{i,j\}$ is negligible if $|\boldsymbol{a}_{i,j}^T\boldsymbol{v}_k|$ is small. Similarly, if $|\boldsymbol{a}_{i,j}^T\boldsymbol{v}_l|$ is small, then the perturbation of other eigenvectors in the direction of $\boldsymbol{v}_l$ due to the edge $\{i,j\}$ is small. Finally, a small eigengap $\lambda_k - \lambda_{k-1}$ or $\lambda_{k+1} - \lambda_k$ means that the Rayleigh quotient is relatively flat between the two critical points, and hence the critical point may be changed easily via a perturbation.

We evaluate the suitability of this approximation to our method in Sec. 4.6.1.5.

4.4.6 Alternatives to Graph Fourier Transform for Regularization

The methods presented in this section can be easily extended to settings where some other graph-based regularizer may be more appropriate than the ℓ_1-norm of the Graph Fourier Transform of the graph signal. For example, if the graph signal under consideration is known to be piece-wise constant (i.e. the graph can be partitioned into sets of topologically close nodes each having the same value of the graph signal), it may be more appropriate to use the Graph Total Variation of the signal as the regularization term. The graph total variation of a signal is the sum of the absolute differences of the value of the graph signal at the endpoints of all the edges of the graph. That is, we have:

$$\mathtt{TV}_{\mathcal{G}}(\boldsymbol{x}) = \sum_{\{a,b\}\in\mathcal{E}} |x_a - x_b|. \tag{4.34}$$

It is straightforward to see that piece-wise constant signals will have very small graph total variation.

For recovery of an unknown graph signal $\boldsymbol{x}^*$ from compressive measurements $\boldsymbol{y}$ (see Eqn. 4.20) using an arbitrary regularizer $\mathtt{R}_{\mathcal{G}} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ on some graph $\mathcal{G}$, one may use:

$$\hat{\boldsymbol{x}}_{\mathtt{R}_{\mathcal{G}}} = \arg\min_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu \mathtt{R}_{\mathcal{G}}(\boldsymbol{x}). \tag{4.35}$$

Accordingly, for a problem setting similar to Sec. 4.3, when only a nominal graph G_{nominal} is known, and the graph signal $\boldsymbol{x}^*$ has a small value of the regularizer $\mathtt{R}_{\mathcal{G}}(\boldsymbol{x}^*)$ for $\mathcal{G} = \mathcal{G}_{\text{actual}}$, the LASSO steps in the brute-force algorithm (Alg. 4.1), cross-validation and multi-fold cross-validation (Alg. 4.2 and Alg. 4.4), greedy edge selection (Alg. 4.3), or ILECIR (Alg. 4.5) may be replaced appropriately by Eqn. 4.35. When the underlying signal is known to be piece-wise constant, one may use the graph total variation for the respective graphs in each of these algorithms, i.e. with $\mathtt{R}_{\mathcal{G}}(\boldsymbol{x}^*) := \mathtt{TV}_{\mathcal{G}}(\boldsymbol{x}^*)$.

4.5 Empirical Evaluation

We performed an empirical evaluation of our greedy edge selection (GES) and the inferred linear-edge compressive image recovery (ILECIR) methods. We describe the experimental setup for both in this section. In each case, we compare with the baseline method of recovery using the nominal graph, as given in Eqn. 4.36 below:

$$\hat{\boldsymbol{x}}_{\boldsymbol{N}} := \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu_{\text{est}} \|\boldsymbol{V}_{\boldsymbol{N}}^T \boldsymbol{x}\|_1, \tag{4.36}$$

where $\boldsymbol{V_N}$ refers to the eigenvectors of the Laplacian of the nominal graph.

We also compare with the ideal case of recovery using the actual graph, i.e. as given in Eqn. 4.37 below:

$$\hat{\boldsymbol{x}}_{\boldsymbol{GT}} := \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \mu_{\text{est}} \|\boldsymbol{V}_{\boldsymbol{GT}}^T \boldsymbol{x}\|_1. \tag{4.37}$$

4.5.1 Greedy Edge Selection (GES)

First, we describe the experimental setup for evaluating the greedy edge selection algorithm.

4.5.1.1 Actual Graph Types

We test our greedy edge selection model on a number of commonly used random graph models in the network science literature. We also use a real social network graph used in the literature. For the random graphs, the number of nodes is set to $n = 100$. Efforts are made to set the model parameters in each case so that the expected number of edges for each random graph model is roughly equal (except in some cases as stated below).

Planted Partition Model (PPM): The Planted Partition Model (PPM) is a random graph with a community structure. The nodes of the graph are divided into some l communities (or clusters), with each community having an equal number of nodes, r. The total number of nodes

is thus $n = lr$. The edges can be thought of as belonging to two categories – intra-cluster, or inter-cluster. An intra-cluster edge has both of its endpoints in the same cluster, whereas an inter-cluster edge has its two endpoints in two different clusters. The edge between any given pair of nodes is included or not included in the graph at random, independent of other edges. Each intra-cluster edge is included in the graph with a probability p, whereas each inter-cluster edge is included with a probability q, with $p > q$. For our experiments, we set $n = 100$ nodes, $l = 5$ clusters, $r = 20$ nodes per cluster, intra-cluster edge probability $p = 0.9$, and inter-cluster edge probability $q = 0.01$. This PPM has 895 edges in expectation, out of which 855 are intra-cluster, and 40 are inter-cluster.

Stochastic Block Model (SBM): The Stochastic Block Model (SBM) is a generalization of the PPM. It is also a community-structured graph, but the community sizes may be different, and the edge probabilities depend on the which cluster(s) the endpoints of the edge belong to. There are l communities with $\{r_1, \ldots, r_l\}$ nodes respectively, so that the total number of nodes $n = r_1 + \cdots + r_l$. Each intra-cluster edge with both endpoints in some community $a \in \{1, \ldots, l\}$ is independently included in the graph with probability p_a. Each inter-cluster edge with one end-point in some community a and the second end-point in a different community b (i.e. $a, b \in \{1, \ldots, l\}, a \neq b$) is independently included in the graph with probability q_{ab} (note that $q_{ab} = q_{ba}$). We use an SBM with $n = 100$ nodes, with 5 communities, with community sizes being $\{r_1, \ldots, r_5\} = \{5, 10, 20, 25, 40\}$. The intra-cluster probabilities are the same for each cluster and is equal to $p = 0.9$, and the inter-cluster edge probabilities are also the same for each pair of clusters, and is equal to $q = 0.01$ (both parameters equal to the one used for the PPM). The expected number of edges for the SBM is 1228.75, which is much higher than in the PPM. The expected number of intra-cluster edges is 1192.5, whereas the expected number of inter-cluster edges is 36.25.

Erdős–Rényi Graph: The Erdős–Rényi Graph is the simplest random graph model, with only two parameters – the number of nodes n, and the probability of each edge independently being included in the graph or not, p. We set $n = 100$, and $p = \frac{895}{4950} \approx 0.18$, so that the expected number of edges is the same as for the PPM.

Random Geometric Graph (RGG): In a Random Geometric Graph (RGG), each node represents a point. Some n points in a unit square are selected uniformly at random, and points within a radius r from each other are connected by an edge, whereas others are not. RGGs model real-world graphs where nodes are on a plane, and spatially close nodes are connected (for example, the highway network of cities which are within 100km of each other by road, or the social network of people who know each other in a neighbourhood). We set $n = 100$ and $r = 0.27$ in our experiments. Empirically, we found that the expected number of edges in this graph was close to the expected number of edges of the PPM.

Barabasi-Albert Graph [167]: Barabasi-Albert Graphs model preferential attachment seen in social networks – for example, new customers are more likely to buy items from popular brands. The graph is generated iteratively, one node at a time. The graph is initialized as a star graph of $r + 1$ nodes. For each newly added node, connections are made from it to r existing nodes. The r existing nodes to which a new node connects, are chosen with a probability proportional to their current degree. The process continues until there are a total of n nodes in the graph. We set $n = 100$, and $r = 10$. The number of edges in this graph is 900, which is close to the expected number of edges of the PPM.

Zachary's Karate Club This is a real-world friend network of $n = 34$ members of a karate club, observed by a person named Wayne Zachary over a two-year period from 1970 to 1972. This graph has a community structure, with two clusters. At some time during the observation, the karate club split into two factions. Zachary was able to correctly predict the faction that each member would join (except for node 19 of the graph) by partitioning the graph into two clusters with the minimum cut (or ratio cut) [168, 169].

4.5.1.2 Nominal Graph Generation

Nominal graphs are generated by perturbing $d \in \{1, 2, 5, 10\}$ edges of the actual graph. For each signal, a different nominal graph is generated, even though the actual graph may be the same. The set of edges to be perturbed – known as the perturbation set ω, with $|\omega| = d$ – is chosen to be a subset of a prior set Ω of 100 "faulty edges" decided at the time of data generation. For each graph, all $\binom{n}{2}$ possible edges are categorized into different categories, and

an approximately equal number of edges from each of these categories are sampled to form Ω.

The edges are categorized in the following manner. For the Erdős–Rényi and random geometric graphs, the edges are categorized as `present` or `absent`, denoting whether the edge was present or absent in the actual graph. For the PPM, SBM, and Karate club graphs, i.e. graphs which have a community structure, there are four categories – `intra-cluster-present`, `intra-cluster-absent`, `inter-cluster-present`, and `inter-cluster-absent`. Intra-cluster and inter-cluster are as defined in Sec. 4.5.1.1 in the paragraph on Planted Partition Model (PPM). For the Barabasi Albert graphs, nodes are categorized as having high degree or low degree. Nodes with high degree have degrees strictly greater than the nodes with low degree, and account for approximately 50% of the edges of the graph. We empirically found that for the class of Barabasi Albert graphs instantiated with $n = 100$ and $r = 10$, around 31.5% of the nodes have high degree on an average. Edges are categorized by whether their two endpoints have low or high degrees, as well as whether they are present or absent in the actual graph, making a total of six categories: `low-low-present`, `low-low-absent`, `low-high-present`, `low-high-absent`, `high-high-present`, and `high-high-absent`.

Each perturbation set with $d > 1$ perturbations may contain edge removals, edge additions or both. Each perturbation set with $d = 1$ perturbation may contain either an edge removal or an edge addition.

4.5.1.3 Signal Model

We test both band-limited as well as sparse-spectrum signals (see Sec. 4.2.1 for definition of these signal models). For the signals on the random graphs with the number of nodes $n = 100$, we set sparsity to $k = 5$ for both models. For the signals on Zachary's Karate club graph, we set $k = 2$. For the sparse-spectrum signals, the underlying graph signal $\boldsymbol{x}^*$ is a linear combination of k randomly chosen eigenvectors of the Laplacian matrix. For the band-limited signals, $\boldsymbol{x}^*$ is a linear combination of the k eigenvectors with the smallest eigenvalues. The coefficient of each eigenvector is chosen independently from a standard Normal distribution.

4.5.1.4 Sensing Matrix

We use a sensing matrix $\mathbf{\Phi}$ of size 50×100 when $n = 100$. The entries of the matrix are i.i.d. Gaussian with mean 0 and variance 1. The same sensing matrix is used for all experiments in order to minimize variance introduced by the choice of matrix. The 17×34 sub-matrix formed by the first 17 rows and first 34 columns of the 50×100 matrix is used as the measurement matrix when $n = 34$.

4.5.1.5 Measurement Noise

The sparse-spectrum signals are tested in the noiseless setting for all graphs. For the PPM, more extensive testing is done – both band-limited as well as the sparse-spectrum signals are tested on the PPM, under both noisy and noiseless settings. For the noisy setting for sparse-spectrum signals, the noise standard deviation is set to $\sigma = \beta \frac{\|\mathbf{\Phi} \boldsymbol{x}^*\|_1}{m}$, with $\beta \in \{0.01, 0.02, 0.05\}$, for each ground-truth signal $\boldsymbol{x}^*$, and the measurement vector $\boldsymbol{y}$ is set using Eqn. 4.20. For the noisy setting for band-limited signals, β is chosen from $\{0.0001, 0.001, 0.01\}$.

4.5.1.6 Experiment Setup

We generate 10 instances of each random graph type described in Sec. 4.5.1.1, with 10 sparse-spectrum graph signals for each graph, making it a total of $N = 100$ sparse-spectrum signals for each random graph type. We use the measurement matrix from Sec. 4.5.1.4 and add noise from $\mathcal{N}(0, \sigma^2)$ to each measurement. Corresponding to each signal, a nominal graph is generated, as described in Sec. 4.5.1.2. For each signal, the greedy edge selection algorithm (Alg. 4.3) with the practical modifications mentioned in Sec. 4.4.2.1 is run.

4.5.1.7 Algorithms Compared

The Greedy Edge Selection (GES) algorithm from Alg. 4.3 is run with the practical modifications mentioned in Sec. 4.4.2.1. Comparison is done with the baseline method of recovery using the nominal graph (NGFT-LASSO-CV), and also the ideal method of recovery using the actual graph (AGFT-LASSO-CV) from Sec. 4.4, Eqn. 4.21. See also Eqns. 4.36 and 4.37. Greedy Edge Selection using the Ceci-Barbarossa eigenvector perturbation approximation (Sec. 4.4.5)

is also evaluated on some graphs. We refer to this method as GES-CB.

4.5.1.8 Evaluation Metric

The metric used for evaluation of the graph signal recovery algorithms is RRMSE (Relative Root Mean Square Error):

$$\texttt{RRMSE}(\hat{\boldsymbol{x}}) = \frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}^*\|_2}, \tag{4.38}$$

where $\hat{\boldsymbol{x}}$ is the estimated signal, and $\boldsymbol{x}^*$ is the ground-truth signal.

4.5.1.9 Algorithm Settings

A prior set of edges Ω containing 100 edges is used as mentioned in Sec. 4.5.1.2. The set of LASSO regularization parameter values used for the grid search is: $\Gamma = \{10^{-3+\frac{6}{19}t} : t \in \{0, 1, 2, \ldots, 19\}\}$ (i.e. 20 values between 0.001 and 1000, evenly spaced in logarithm). The number of folds used for cross-validation is $F = 5$. The loss improvement factor is $\tau = 0.99$. The upper bound d_0 on the number of edges perturbed to obtain the nominal graph is set to twice the number of edges perturbed – i.e. $d_0 = 2d$, for each $d \in \{1, 2, 5, 10\}$.

4.5.2 Inferred Linear Edge Compressive Image Recovery

We simulate the patch-wise compressive acquisition of images from some real-world and synthetically generated image datasets, and evaluate the reconstruction performance of the Inferred Linear Edge Compressive Image Recovery (ILECIR) algorithm both quantitatively and qualitatively.

4.5.2.1 Datasets

We use the following image datasets for the evaluation. For each image dataset, either a ground-truth segmentation, a human-labelled segmentation, or a machine-computed segmentation of

the image is available. The segmentation information is used for patch recovery with the actual graph.

Synthetic Dataset: Union of Polynomial Intensity Regions We create a dataset of 10 synthetic images, each of size 256×256, with each image being the average image of 6 randomly generated component images. Each component image is generated in the following manner. First, a fourth degree polynomial of two variables, $g(x, y)$, is chosen by sampling its coefficients from the uniform distribution on the interval $(-1, 1)$. This function is rescaled to have values between 0 and 255 and rounded to the nearest integer when the variables x and y form a valid pixel in a 256×256 image, to get another function $h(x, y)$. That is, $h(x, y) = \lfloor 255 \frac{g(x,y)-a}{b-a} \rceil$, where $a = \min_{x,y \in \{0,\ldots,255\}} g(x, y)$, $b = \max_{x,y \in \{0,\ldots,255\}} g(x, y)$, and $\lfloor . \rceil$ denotes rounding to the nearest integer. In the final step of the component image generation, the intensity is set to $h(x, y)$ in regions where $h(x, y) \leq h(x_0, y_0)$ where (x_0, y_0) is a randomly chosen pixel, and 0 otherwise. This step ensures that a sharp edge is formed in the final image, given by the curve $h(x, y) = h(x_0, y_0)$, and not all of the polynomials are "active" at the same time in a given region of the final image, ensuring piece-wise smoothness of the intensity. We refer to each image in this dataset with an arbitrarily assigned 'Image ID', from 0 to 9.

Berkeley Segmentation Dataset [170] This is a popular image dataset used for segmentation tasks. We use 10 randomly chosen images from BSDS500, the version of the dataset containing 500 images. The images are of size 640×480 or 480×640. The provided images are color – we convert them to greyscale. The dataset provides segmentations for each image by upto five human labellers. For each image, we use the segmentation performed by the first human labeller for computing the actual graph. We refer to the file name (sans the file type suffix) of an image in this dataset as its 'Image ID'.

Tom and Jerry Dataset [171] This is a dataset consisting of video frames from the popular Tom and Jerry cartoon. The images are classified according to the presence of the main characters Tom and Jerry, as containing only Tom, only Jerry, both Tom and Jerry, or neither Tom nor Jerry. We use 10 randomly chosen images containing both Tom and Jerry from this dataset. We center-crop each image to make the height and width equal, and resize to 256×256. We segment the images using the Chan-Vese segmentation algorithm [172] to compute the segmen-

tation information for the actual graph. We refer to the file name (sans the file type suffix) of an image in this dataset as its 'Image ID'.

NYU Depth Dataset [173] These are depth maps of indoor scenes, collected using a Microsoft Kinect. The dataset contains segmentation information for each depth map, as well. The depth maps are of width 640 and height 480 pixels. We use the depth maps from the provided test dataset. The labelled dataset is provided as a single '.mat' file. We refer to the index (starting from 0) in the matlab variable 'depth' in the provided '.mat' file as the 'Image ID' of an image in this dataset.

4.5.2.2 Patch Size

The size of each patch used for simulation of patch-wise compressive image acquistion is 8×8. Thus there are 64 pixels in each patch.

4.5.2.3 Sensing Matrix

We create a 64×64 matrix whose entries are sampled independently from a standard Normal distribution. We perform our experiments with the first $m \in \{20, 30, 40\}$ rows of this matrix. The same matrix is used for all experiments.

4.5.2.4 Measurement Noise

Evaluation in the noiseless setting is performed for all images in all of the datasets. Evaluation in the noisy setting is performed for one image from the Synthetic dataset. We add noise to each measurement as described in Sec. 4.5.1.5, with $\beta \in \{0.01, 0.02, 0.05\}$.

4.5.2.5 Experiment Setup

Patch-wise compressive image acquisition is simulated for an image in the following manner. The image is divided into non-overlapping patches of size 8×8. Each patch is linearized in row-major order to a vector of $n = 64$ dimensions, which forms the signal $\boldsymbol{x}^*$. Measurements are generated using Eqn. 4.20, with $m \times n$ sensing matrix (Sec. 4.5.2.3), and noise is added to

the measurements. These measurements are then decoded using the ILECIR algorithm, and the 8×8 patch is estimated. For greyscale images, the estimated values are clipped to be between 0 and 255 and rounded to the nearest integer. For depth maps, negative values are set to 0. The recovered patches are stitched together to reconstruct the image.

4.5.2.6 Evaluation Metrics

We use RRMSE (Eqn. 4.38), as well as the Structural Similarity (SSIM) which has been shown to be superior to RRMSE [174], between the recovered and the original image, for quantitative evaluation of the recovery methods. For some common types of image defects, such as constant intensity shifts, sparse impulse noise, etc, SSIM is a better indicator of perceptual similarity than RRMSE (e.g. see Fig. 2 in [174]). Images are also visually inspected for recovery quality and to explore the strengths and weaknesses of each recovery method.

4.5.2.7 Algorithms Compared

We use the ILECIR algorithm presented in Alg. 4.5, with the practical modifications as mentioned in Sec. 4.4.3. Comparison is done with the baseline method of recovery using LASSO-CV with the GFT basis of the nominal graph (i.e. the 2-D DCT basis) as in Eqn. 4.36 – we refer to this method as DCT-LASSO-CV in figures. Comparison is also done with recovery using GFT-LASSO-CV with the GFT basis of the human-labelled or ground-truth segmented patch as in Eqn. 4.37 – this method is referred to as SEGGFT-LASSO-CV in figures. Performance of Greedy Edge Selection (GES, Alg. 4.3) on compressive image recovery is also evaluated, with perturbation budget $d_0 \in \{1, 2, 5, 10\}$, referred to as GES-1, GES-2, GES-5 and GES-10 respectively.

We also evaluate ILECIR with graph total variation regularization instead of GFT regularization on some images of the Synthetic dataset. We refer to this algorithm as GRAPHTV-ILECIR. It is compared with the baseline method of using graph total variation regularization on the nominal graph (i.e. the 2-D lattice graph), along with cross-validation to choose the regularization parameter, which is referred to as GRAPHTVREG-CV.

4.5.2.8 Algorithm Settings

For cross-validation, we use $F = 5$ folds. The set Γ of possible LASSO regularization parameter values used for grid search is same as for the greedy edge selection algorithm Sec. 4.5.1.9.

4.6 Results and Discussion

We discuss the results of the empirical evaluation of our algorithms.

4.6.1 Greedy Edge Selection on synthetic graphs

We discuss the results of empirical evaluation of Greedy Edge Selection (GES) on synthetic graphs in this section.

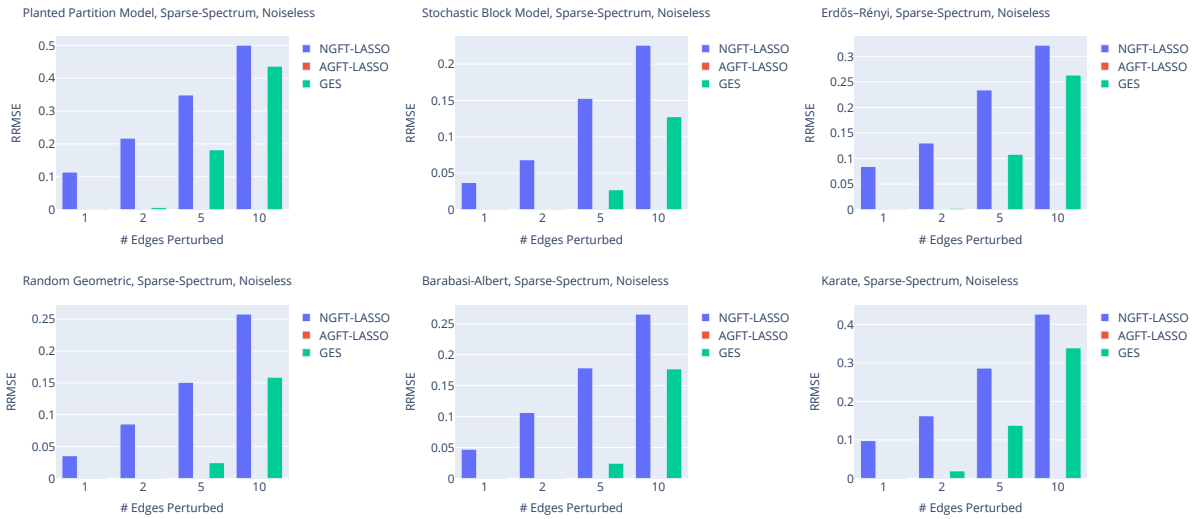


Figure 4.8: RRMSE of signal recovered via Greedy Edge Selection (GES), LASSO with the nominal graph (NGFT-LASSO), and LASSO with the actual graph (AGFT-LASSO), for various number of perturbed edges. RRMSE for AGFT-LASSO and in some cases for GES are close to zero, hence the bars are not visible.

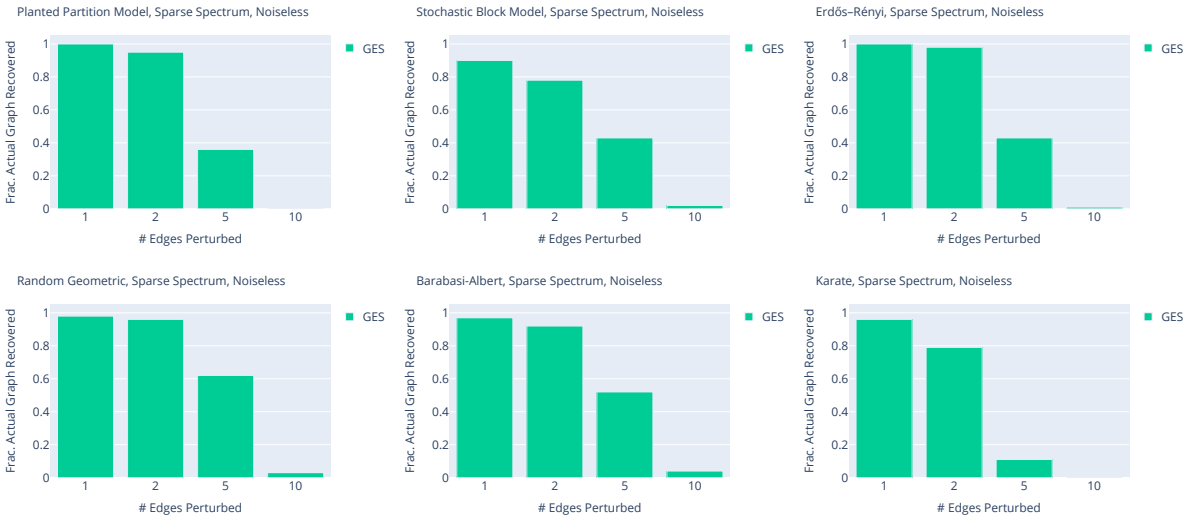


Figure 4.9: Fraction of cases in which actual graph was recovered by Greedy Edge Selection (GES), for various number of perturbed edges

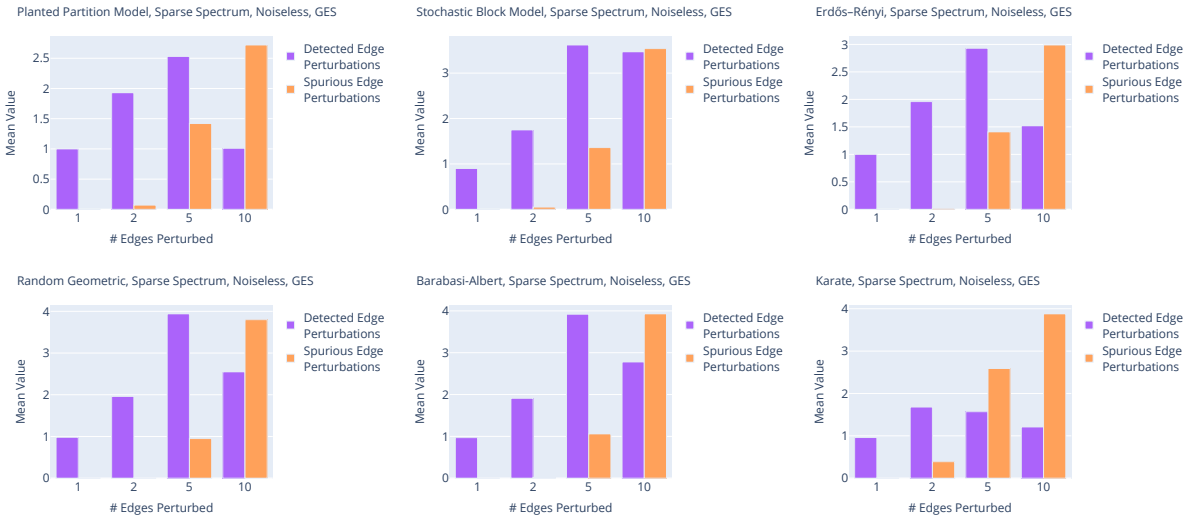


Figure 4.10: The number of edge perturbations successfully detected and the number of spurious edge perturbations reported by Greedy Edge Selection (GES), for various number of perturbed edges on different graph models.

4.6.1.1 Sparse-Spectrum signals

The RRMSE of the Greedy Edge Selection (GES) algorithm for sparse-spectrum signals is presented in Fig. 4.8, for each graph or graph model mentioned in Sec. 4.5.1.1, with $d = \{1, 2, 5, 10\}$ edges perturbed to create the nominal graph. We see that for each graph type, GES outperforms the baseline method of recovery using the nominal graph, i.e. NGFT-LASSO. As expected, the RRMSE increases as the number of perturbations are increased. For one or two edge perturbations, the RRMSE is close to zero or very small, whereas for upto five perturbations, the RRMSE is at most half of that obtained using NGFT-LASSO, and often much lower than that. Even for ten edge perturbations, the RRMSE of GES is significantly lower than NGFT-LASSO. Recovery using the actual graph gives an RRMSE close to zero, and hence the bars for it are not visible in Fig. 4.8.

Fig. 4.9 presents the fraction of cases in which GES successfully recovers the actual graph from the nominal graph. We see that for most graph types, the actual graph is recovered in close to 100% cases when up to two edges are perturbed. Despite being a greedy algorithm, the actual graph gets recovered in a significant fraction of cases even when five edges perturbations were performed. Surprisingly, in some cases the actual graph is recoverable via GES even if ten edge perturbations were induced.

Fig. 4.10 show the average number of edge perturbations *correctly* detected by GES, and the average number of spurious edge perturbations reported by it. Notably, the number of edge perturbations correctly detected falls down rapidly when more than five edge perturbations are made to the actual graph.

Overall, we find empirically that the technique of using cross-validation error to select the edges for refining the nominal graph has merit. Even in case of a large number of perturbations, the RRMSE of signal recovered by GES is significantly lower than the baseline of using LASSO with the GFT basis of the nominal graph. This is true despite the edge perturbations made to the actual graph not all being identified correctly and some spurious edge perturbations being reported. The greedy edge selection technique is most suitable when up to five edges perturbations are made to the actual graph.

4.6.1.2 Band-limited Signals

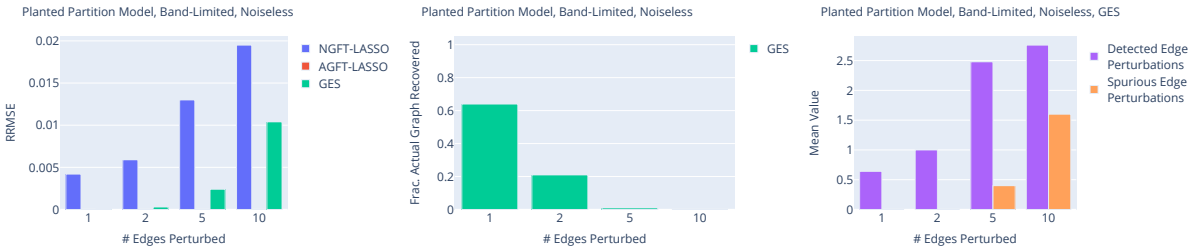


Figure 4.11: Performance of Greedy Edge Selection on band-limited signals on the Planted Partition Model. Top Left: RRMSE of recovery via GES, LASSO with the nominal graph (NGFT-LASSO-CV), and LASSO with the actual graph (AGFT-LASSO-CV), for various number of perturbed edges. The bar for AGFT-LASSO-CV is too small to be visible. Top Right: Fraction of cases in which the actual graph was recovered by GES. Bottom: Number of correctly detected and spuriously detected edge perturbations by GES.

We also evaluate the GES algorithm for compressive recovery of band-limited signals on the Planted Partition Model random graphs in the presence of edge perturbations. We again find significant reduction in RRMSE when using GES instead of NGFT-LASSO (Fig. 4.11, top-left). In Fig. 4.11 bottom, we find that even though the RRMSE of GES is close to zero for one or two edge perturbations, only around $50 - 60\%$ of the edge perturbations are successfully detected. Successful recovery of the full graph happens in only 60% of the cases (Fig. 4.11, top-right) when one edge perturbation is made, only 20% of the time when two edge perturbations are made, and close to 0% when 5 or more edge perturbations are made. In Sec. 4.6.1.4, we will discuss the possible reasons behind this anomaly. Regardless, we find that greedy edge selection using cross-validation is a competitive method for recovery of the signal.

4.6.1.3 Effect of Measurement Noise

Fig. 4.12 shows the results of performing recovery via GES on measurements which have added Gaussian noise as described in Sec. 4.5.1.5, with the actual graph being a Planted Partition Model, and the signals the nominal graphs generated in the same manner as the previous sections (Sec. 4.5.1). We find that the GES algorithm performs well even in the presence of noise. Its RRMSE is significantly less than the RRMSE of NGFT-LASSO-CV for upto 5 edge perturbations, and is comparable to that of AGFT-LASSO-CV when the number of edge perturbations

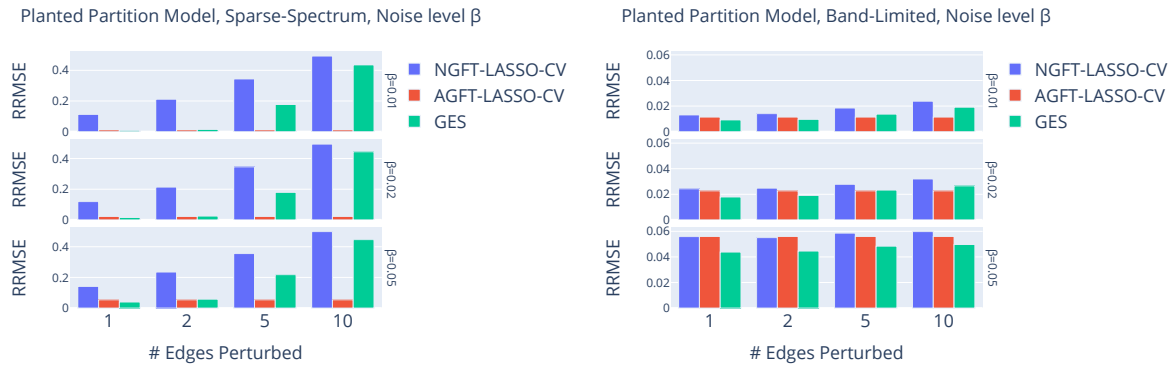


Figure 4.12: Performance of Greedy Edge Selection with measurement noise

is 1 or 2. Occasionally, we find that GES is able to do slightly better than AGFT-LASSO-CV. We suspect that this is because only 20 different values were tried during grid search for the LASSO regularization parameter μ, due to which GES could find a better graph than the actual graph for the given values of μ for the purpose of signal recovery. A more fine-grained grid search would lead to the expected behaviour of AGFT-LASSO-CV performing better than GES.

4.6.1.4 Recovery Analysis

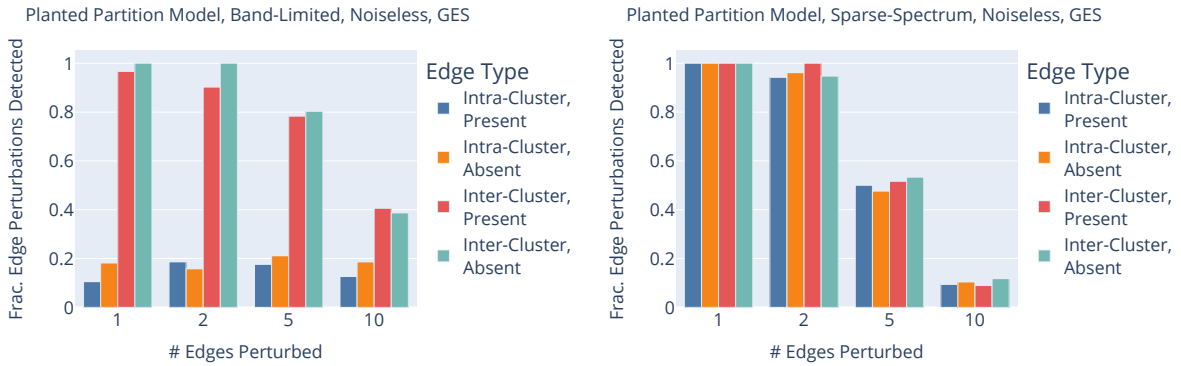


Figure 4.13: Fraction of Edge Perturbations of each type recovered via Greedy Edge Selection for the Planted Partition Model in the band-limited (left) and sparse-spectrum (right) cases.

Fig. 4.13 shows the fraction of edge perturbations which were detected by GES on the Planted Partition Model, categorized by edge type, for the band-limited (left) and the sparse-spectrum (right) cases. We note that for band-limited signals, inter-cluster edges are highly likely to be recovered by GES. For sparse-spectrum signals, all edge types are equally likely to be recovered. In general, an edge perturbation is more likely to be detected by GES if perturbing

that edge significantly perturbs a least one of the eigenvectors on which the original signal $\boldsymbol{x}^*$ has a non-zero component. From the formula in Eqn. 4.32 (Sec. 4.4.5), we note that an edge perturbs an eigenvector significantly only if the absolute difference of the values of that eigenvector on the two nodes of the edge is large. Since the PPM is a community-structured graph, hence for a PPM with k clusters, the value of the eigenvector within a cluster does not change much for the first k eigenvectors (see Fig. 4.2 and relevant background in Sec. 4.2.1).

Hence adding or removing an intra-cluster edge does not significantly perturb the first k eigenvectors of the PPM. On the other hand, the value of the first k eigenvectors of such PPMs vary a lot across clusters. Hence, adding or removing an inter-cluster edge causes a significant perturbation in the first k eigenvectors, which is easily detected by GES. For high-frequency eigenvectors, both inter-cluster and intra-cluster edges are highly likely to have different values at the endpoints. Hence for signals which are sparse-spectrum but not band-limited, GES is able to recover both inter-cluster and intra-cluster edges.

4.6.1.5 Using Ceci-Barbarossa approximation for eigenvector perturbation

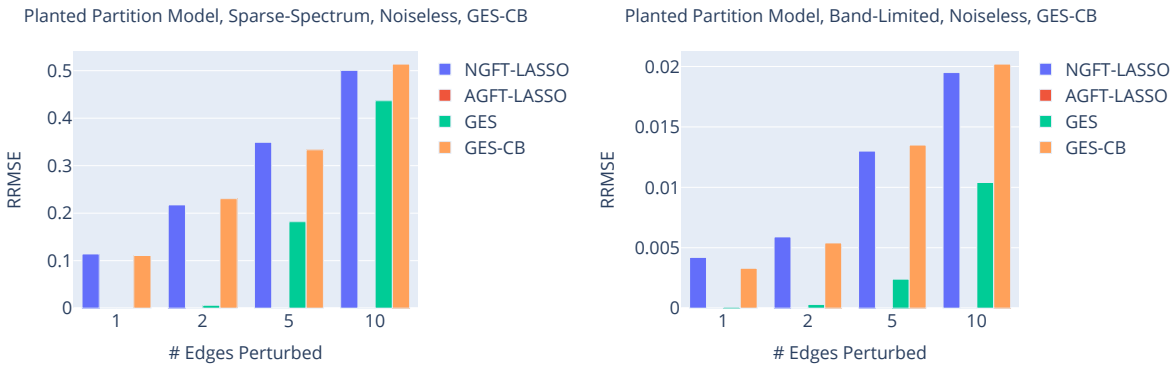


Figure 4.14: Performance of Greedy Edge Selection with the Ceci-Barbarossa approximation for eigenvector perturbation (GES-CB). In some cases, the error for GES or AGFT-LASSO is close to 0, and hence the error bars for them are not visible.

Fig. 4.14 shows the RRMSE obtained using the GES-CB algorithm, in which the Ceci-Barbarossa eigenvector perturbation approximation (Eqn. 4.32) is used in place of eigendecomposition in GES (see Sec. 4.4.5 and Sec. 4.5.1.7), for compressive recovery of sparse-spectrum and band-limited signals on PPMs with a few edge perturbations. We find that for both sparse-spectrum and band-limited signals, the RRMSE of GES-CB is significantly higher than that for

GES, across a range of edge perturbations. Moreover, the RRMSE of GES-CB is close to the RRMSE of Nominal, and sometimes slightly worse. Hence we do not find the Ceci-Barbarossa eigenvector perturbation approximation (a first order approximation) to be useful when applied to greedy edge selection with cross-validation. Perhaps a better approximation with higher order terms in it might be more useful.

One reason for the poor performance of GES-CB could be that the condition under which the Ceci-Barbarossa approximation is applicable – given by Eqn. 4.33 – gets violated often. We empirically find the likelihood of this condition getting violated for single-edge perturbations. We deem the condition to be violated for an edge $\{i, j\}$ and the k^{th} eigenvector $\boldsymbol{v}_k$ if $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 > 4(\lambda_{k+1} - \lambda_k))$ for edge addition or $(\boldsymbol{v}_k(i) - \boldsymbol{v}_k(j))^2 > 4(\lambda_k - \lambda_{k-1})$ in case of edge deletion. The fraction of times this condition is violated for at least one eigenvector out of the first five (i.e. the band-limited eigenvectors) for a PPM graph with $n = 100, l = 5, p = 0.9, q = 0.01$ is 87.7%, if an inter-cluster edge is perturbed. Similarly, we found that the condition in Eqn. 4.33 is violated 74.7% of the times for at least one eigenvector out of five for a random choice of five eigenvectors and any kind of edge. While this condition does not get violated for intra-cluster edges in the band-limited case, the amount of perturbation in the low-frequency eigenvectors by such edges is very small, such that the RRMSE for these cases is close to zero even for NGFT-LASSO-CV, and hence these cases do not contribute much to the average RRMSE.

4.6.2 Comparison of Recovery Algorithms for Compressive Image Acquisition

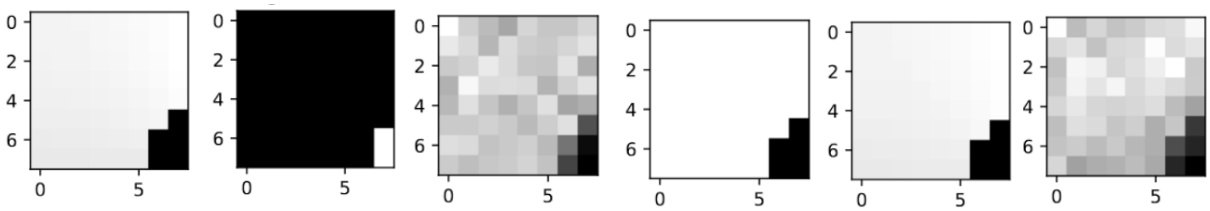
We present the results of evaluation of the Inferred Linear-Edge Compressive Image Recovery (ILECIR) algorithm and comparison with the baseline methods as described in Sec. 4.5.2.

4.6.2.1 Quantitative comparison in the noiseless setting

Fig. 4.15 shows the RRMSE and SSIM of images recovered using ILECIR, the baseline method DCT-LASSO-CV and the SEGGFT-LASSO-CV method which uses knowledge of either ground-truth, machine-generated or human-labelled segmentation of the images, from $m \in \{20, 30, 40\}$

Figure 4.15: RRMSE and SSIM using ILECIR, DCT-LASSO-CV and SEGGFT-LASSO-CV (Segmentation-aware recovery) in the noiseless regime

linear measurements, for 10 images from the Synthetic, Berkeley, Tom and Jerrry, and the NYU Depth datasets, as discussed in Sec. 4.5.2. We note that ILECIR substantially improves over DCT-LASSO-CV in terms of the RRMSE and SSIM of the recovered images on the Synthetic and the NYU Depth datasets. It also shows some improvement for the images in the Tom and Jerry dataset. For Tom and Jerry and the NYU Depth datasets, the improvement is most pronounced when $m = 20$ linear measurements are used. Such improvement is expected since these are piece-wise smooth images – while the smooth regions are accurately recovered by DCT-LASSO-CV, the patches containing sharp edges are not recovered very accurately. ILECIR performs better on such patches, while maintaining the accuracy afforded by DCT-LASSO-CV on smooth patches. The SEGGFT-LASSO-CV method has near-perfect recovery for the Synthetic dataset. This substantiates the rationale behind using the GFT basis of the 2-D lattice graph partitioned according to the segments of a patch for recovery. Since ILECIR only models linear image edges and does not model more than one image edge in a patch, recovery using it is not perfect. Notably, ILECIR often performs better than the SEGGFT-LASSO-CV method for the Tom and Jerry and NYU Depth Datasets. This means that it is able to perform edge detection slightly better than the algorithm used to perform edge detection for the images from the Tom and Jerry dataset. For the NYU Depth Dataset, we found that the provided segmentation



(a) Patch, Original **(b)** Seg, Original **(c)** Patch, SEGGFT **(d)** Seg, ILECIR **(e)** Patch, ILECIR **(f)** Patch, DCT

Figure 4.16: An 8×8 patch (a), and its *incorrectly* labelled segmentation (b), from an image in the NYU depth dataset, resulting in poor signal recovery by SEGGFT-LASSO-CV (c). ILECIR recovers the correct segmentation (d) and patch (e). Patch recovered via DCT-LASSO-CV (f) is also shown for comparison. Note: the two regions of the segmentation maps in (b) and (d) are coloured arbitrarily.

.

maps were not properly aligned with the depth map images (see Fig. 4.16 for an example of an incorrectly labelled segmentation map of a patch from this dataset). Due to this, the SEGGFT-LASSO-CV method performs *worse* than DCT-LASSO-CV for the NYU Depth dataset. This demonstrates a strength of the ILECIR method – since the underlying segmentation is automatically inferred, it circumvents possible errors in the segmentation map if it were available.

RRMSE and SSIM of images recovered using ILECIR are marginally better than those recovered using DCT-LASSO-CV for some images of the Berkeley dataset. For some other images, they are marginally worse. Since natural images contain vast regions of detailed texture and not just gradient of intensity, it is hard to improve upon the baseline DCT-LASSO-CV method by modelling just a single linear edge. This is also substantiated by the performance of the SEGGFT-LASSO-CV method, which is similar to DCT-LASSO-CV and ILECIR. However, a visual inspection of the images recovered by ILECIR, SEGGFT-LASSO-CV and DCT-LASSO-CV would reveal that ILECIR and SEGGFT-LASSO-CV perform better along the edges in these images (See Sec. 4.6.2.2).

4.6.2.2 Qualitative Comparison in the noiseless setting

The results of image recovery via SEGGFT-LASSO-CV, DCT-LASSO-CV, and ILECIR for one image of each dataset are shown in Fig. 4.17, along with the original image in each case. A careful inspection of the images reveals that in each case, the images recovered by ILECIR have higher fidelity than the corresponding image recovered by DCT-LASSO-CV along sharp edges in the image. The quality of the recovered images remains the same in regions where there are no sharp edges. In Fig. 4.17 (Row 1), for the image from the Synthetic dataset, we clearly see that ILECIR has good recovery in regions where there is only a single edge which is close to linear. In contrast, DCT-LASSO-CV does not do very well in regions where there are edges, and many artifacts appear along the edges. The image recovered by ILECIR also has some artifacts along the edges, but much fewer than the one recovered by DCT-LASSO-CV. These artifacts may appear for ILECIR whenever the ground-truth image edge in the patch does not exactly match any of the image edges that ILECIR tests for. ILECIR does not do well around corners, due to the presence of more than one image edge in the patch. However, a careful inspection reveals that for some patches, the recovery quality for ILECIR seems to be better than DCT-LASSO-CV even in the presence of two edges, and there does not appear to be a case where ILECIR has worse recovery quality than DCT-LASSO-CV. This is due to the high-probability RMSE improvement guarantee of ILECIR (due to cross-validation) as established by Thm. 4.3.

In Fig. 4.17 (Row 2), there are much fewer artifacts around the face of the polar bear in the image recovered by ILECIR as compared to the one recovered by DCT-LASSO-CV. In the same region, we also see that while there are fewer artifacts for SEGGFT-LASSO-CV compared

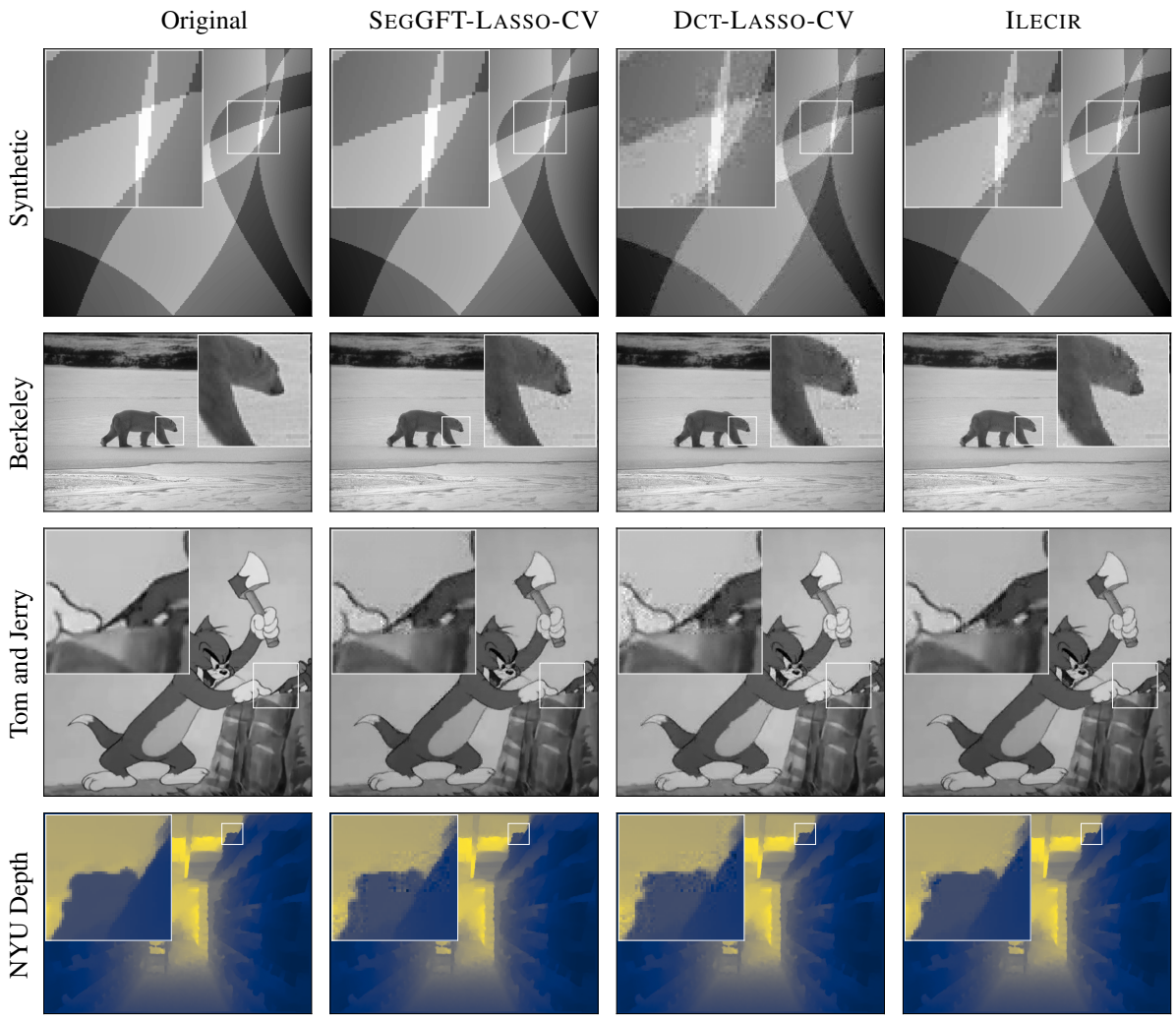


Figure 4.17: Zoomed view of images from the Synthetic (top row), Berkeley (second row), Tom and Jerry (third row), and NYU Depth (bottom row) datasets, recovered patch-wise from $m = 40$ compressive measurements per 8×8 patch via various algorithms in the noiseless setting. The columns denote, from left to right: Col 1 – the original image, Col 2 – image recovered via SEGGFT-LASSO-CV, Col 3 – image recovered via DCT-LASSO-CV, Col 4 – image recovered via ILECIR.

to DCT-LASSO-CV, the recovery by ILECIR is much better than for SEGGFT-LASSO-CV. This once again highlights the strength of ILECIR compared to using human-labelled segmentation (which may contain errors) for recovery.[2] In Fig. 4.17 (Row 3), there are much fewer artifacts for ILECIR around the ears and the head of Tom (the cat), compared to for DCT-LASSO-CV. However, there are many artifacts on the face of Tom for both ILECIR and DCT-LASSO-CV. This highlights the inability of ILECIR to improve upon DCT-LASSO-CV in regions containing many edges. In Fig. 4.17 (Row 4), we observe the same pattern – ILECIR performs better than DCT-LASSO-CV and even SEGGFT-LASSO-CV around the edges in the depth map.

The reconstruction by each algorithm on an individual patch of an image from the NYU Depth dataset is also shown in Fig. 4.16, which clearly demonstrates the recovery of the ground-truth segmentation and the original patch using ILECIR.

4.6.2.3 Edgemap Recovered by ILECIR

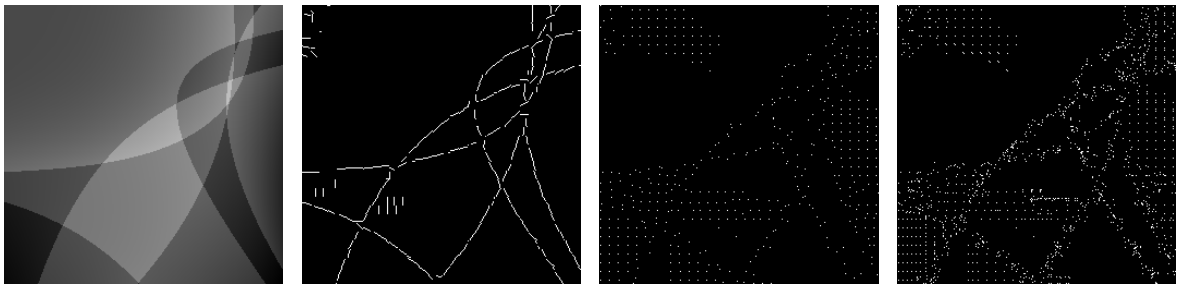


Figure 4.18: A synthetic image (col 1), and its edgemap as recovered patch-wise from $m = 30$ compressive measurements per 8×8 patch via the ILECIR (col 2), GES-1 (col 3), and GES-10 (col 4) algorithms.

Since the ILECIR algorithm discovers the best linear image edge for each patch, an edgemap of the original image may be recovered by stitching together the edgemaps of each patch. Fig 4.18 shows an image from the synthetic dataset (col 1), and its edgemap (col 2) recovered by the ILECIR algorithm. The original edges are correctly detected for most patches.

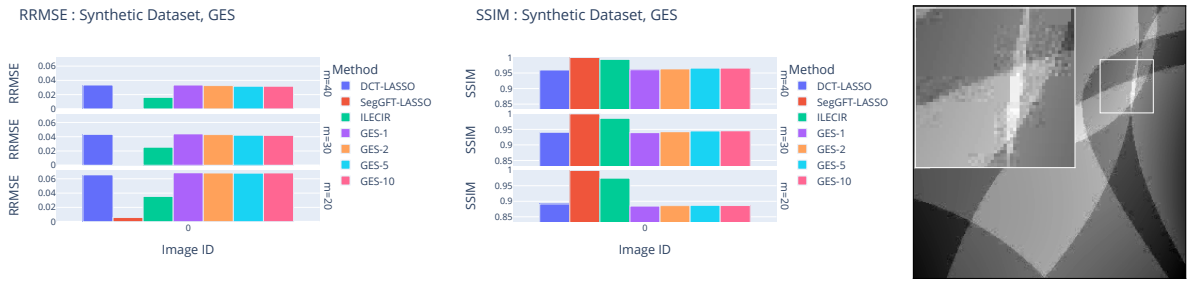


Figure 4.19: Performance of Greedy Edge Selection (GES) on Image ID 0 of the Synthetic dataset in the noiseless regime. RRMSE (left) and SSIM (middle) with perturbation budget 1, 2, 5, and 10 (GES-1, GES-2, GES-5, GES-10). Plots show comparison with DCT-based recovery (DCT-LASSO-CV), Segmentation-aware recovery (SEGGFT-LASSO-CV) and ILECIR. Right: The image recovered using GES-10.

4.6.2.4 Performance of Greedy Edge Selection on compressive image recovery

We test the effectiveness of GES on the problem of patch-wise compressive image recovery on an image from the Synthetic dataset. Fig. 4.19 shows the RRMSE and SSIM of the recovered images when GES is used for recovery, with a maximum of $d_0 \in \{1, 2, 5, 10\}$ edges of the 2-D lattice graph of a patch being allowed to be perturbed by the algorithm. We see that using GES (especially when $d_0 = 10$ graph edges are allowed to be perturbed) is marginally better than using DCT-LASSO-CV – however, it is much better to use ILECIR. This is evident in Fig. 4.19, as well – a careful inspection reveals that the image recovered by GES-10 ($d_0 = 10$) has slightly fewer artifacts than the one recovered by DCT-LASSO-CV, but recovery via ILECIR is much better. Fig. 4.18 shows the edgemap recovered by various algorithms, including GES-5 and GES-10 (cols 3 and 4). An edgemap for GES is recovered in the following manner – whenever the GES algorithm drops a (2-D lattice graph) edge between two pixels, the right pixel or the bottom pixel of the graph edge is marked with white color (denoting a pixel belonging to an image edge), and remaining pixels are marked with black color (denoting background pixels). We see that the edgemap recovered by GES follows the image edges very coarsely. It consists of many isolated dots, instead of groups of pixels connected as an edge. In comparison, the edgemap recovered by ILECIR consists of linear edges in each patch. The ILECIR algorithm takes advantage of the structure in perturbations of the graph edges, and is able to outperform GES, which perturbs graph edges in an unstructured manner.

[2]Note that in compressive image recovery, the segmentation map is usually not available.

4.6.2.5 Effect of Noise on ILECIR recovery performance

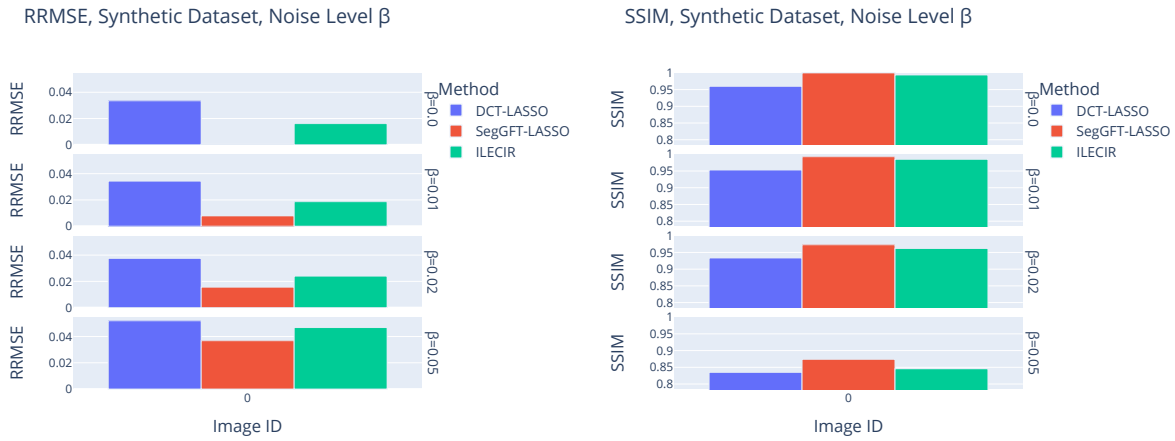


Figure 4.20: RRMSE and SSIM of ILECIR for Image ID 0 of the Synthetic dataset for different noise levels ($\beta \in \{0, 0.01, 0.02, 0.05\}$). Comparison with DCT-based recovery (DCT-LASSO-CV) and Segmentation-aware recovery (SEGGFT-LASSO-CV) is shown.

Fig. 4.20 shows the RRMSE and SSIM of images recovered using ILECIR from noisy measurements of an image of the Synthetic dataset, with different noise levels $\beta \in \{0, 0.01, 0.02, 0.05\}$, as discussed in Sec. 4.5.2.4. We find that ILECIR performs significantly better than DCT-LASSO-CV even in the presence of noise. In fact ILECIR on measurements with noise level 0.02 has better RRMSE and approximately the same SSIM as DCT-LASSO-CV with noiseless measurements. We also find that the RRMSE and SSIM of SEGGFT-LASSO-CV worsens significantly with measurement noise. Fig. 4.21 shows the images recovered using DCT-LASSO-CV and ILECIR for different noise levels. We note that the images recovered by ILECIR are of better quality than the corresponding images recovered by DCT-LASSO-CV, across all noise levels. The image recovered by ILECIR with noise level $\beta = 0.02$ looks better near the edges in the image than the one recovered by DCT from noiseless measurements.

4.6.2.6 Using Graph Total Variation regularization instead of GFT-LASSO

We demonstrate the effectiveness of ILECIR when used with some other graph-based regularizer than the Graph Fourier Transform. We use the Graph Total Variation regularizer, as discussed in Sec. 4.4.6 and Sec. 4.5.2.7. Fig. 4.22 (first column) shows the RRMSE and SSIM of six

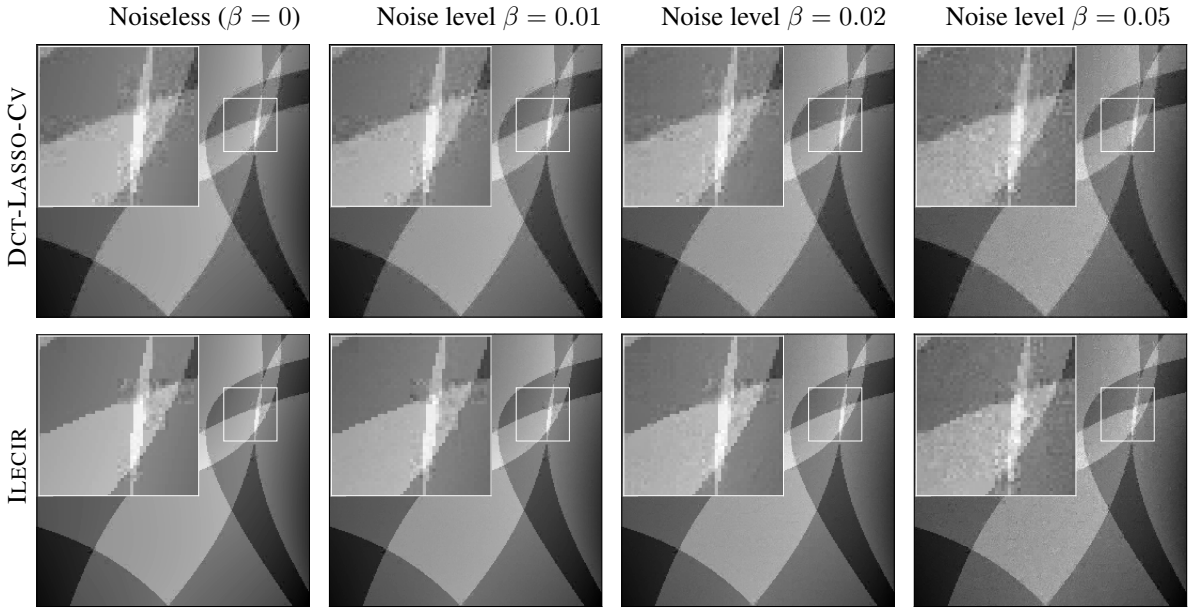
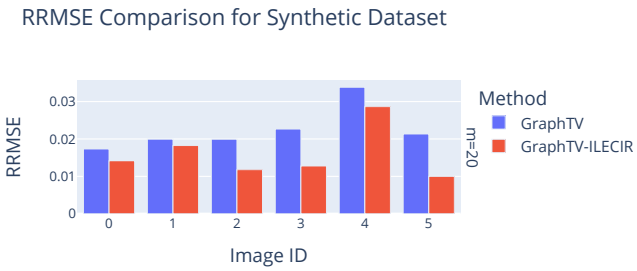
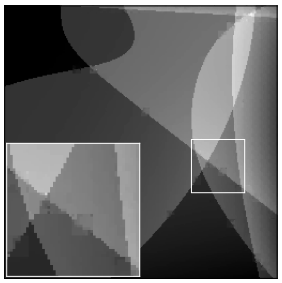


Figure 4.21: Zoomed view of Image ID 0 of the Synthetic dataset, recovered patch-wise from $m = 40$ compressive measurements per 8×8 patch using the DCT-LASSO-CV (top row) and the ILECIR (bottom row) algorithms for various noise levels ($\beta \in \{0, 0.01, 0.02, 0.05\}$).

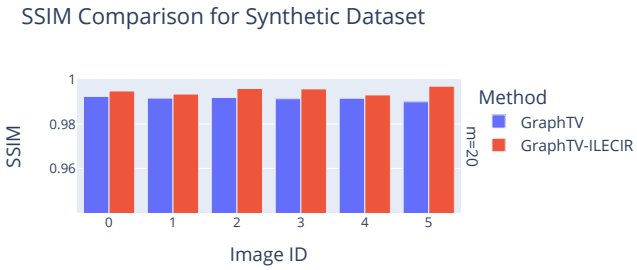
images from the Synthetic dataset recovered using GRAPHTV-ILECIR, and the baseline method of GRAPHTVREG-CV. We see that for all images, GRAPHTV-ILECIR performs better than GRAPHTVREG-CV. We also compare the images recovered by the two algorithms for Image ID 3 of the Synthetic dataset (Fig. 4.22, second column). Careful inspection of the two images reveals that there are fewer artifacts along the edges and corners in the image recovered by GRAPHTV-ILECIR than in the one recovered by GRAPHTVREG-CV.



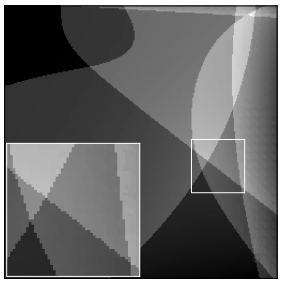
(a) GRAPHTVREG-CV RRMSE



(b) GRAPHTVREG-CV



(c) GRAPHTVREG-CV SSIM



(d) GRAPHTV-ILECIR

Figure 4.22: Column 1: RRMSE and SSIM of 6 images from the Synthetic dataset, recovered patch-wise from $m = 20$ compressive measurements per 8×8 patch using the GRAPHTVREG-CV and GRAPHTV-ILECIR algorithms in the noiseless setting; Column 2: Zoomed view of Image ID 0 of the Synthetic dataset, recovered patch-wise from $m = 20$ compressive measurements per 8×8 patch using the GRAPHTVREG-CV and the GRAPHTV-ILECIR algorithms in the noiseless setting.

4.7 Conclusion and Future Work

This work proposes a new technique of compressive graph signal recovery in cases where the graph topology is partly incorrectly specified. This is a novel computational problem, with no existing literature that targets this specific problem. At the core of the technique to correct for errors in graph topology, lies the concept of cross validation in compressed sensing. A novel algorithm is proposed and supportive theoretical results are stated and proved. A large number of computer simulations are presented, including results on compressive image recovery, where images or image patches are modeled as lattice graphs. Note that our technique is naturally applicable to image data that are defined on arbitrary non-Cartesian domains, for examples on the vertices of a 3D model. In such cases, our perturbed GFT technique fits in very naturally, even though the eigenvectors of the actual graph may not be the DCT. Furthermore, our technique can be easily extended to handle dropping of edges in weighted graphs. Since intensity values at pixels/nodes that lie on opposite sides of an image edge are uncorrelated, the presence of a graph edge that links such nodes is regarded as a perturbation that needs to be corrected. Thus edge-preserving compressive recovery is cast quite interestingly as a graph perturbation problem. This specific idea has been used in image compression before, where the application innately has access to the complete graph. However its application to compressive image reconstruction is quite novel. Last but not the least, the idea of signal sparsity in the GFT has not been extensively explored in the literature, with most papers focussing on bandlimited approximations unlike our approach. There are a few major avenues for future work listed below:

1. Exploration of other proxies for the reconstruction error apart from validation error, such as information theoretic criteria like the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC) and their application to our specific problem.

2. Research towards a more computationally efficient strategy to uncover all graph perturbations, over and above our greedy algorithms.

Appendices

Appendix 4.A Proof of Theorem 4.2

First, we state the relevant theorem and related concepts from [60]. For any recovered signal $\hat{\boldsymbol{x}}^p$ corresponding to a ground truth vector $\boldsymbol{x}^*$, the recovery error vector $\Delta\boldsymbol{x}^p$ and the recovery error ε_x^p are given by

$$\Delta\boldsymbol{x}^p \triangleq \hat{\boldsymbol{x}}^p - \boldsymbol{x}^*,\ \varepsilon_x^p \triangleq \|\Delta\boldsymbol{x}^p\|_2^2 = \|\hat{\boldsymbol{x}}^p - \boldsymbol{x}^*\|_2^2. \tag{4.39}$$

They introduce a generalized error vector, which includes the standard deviation of measurement noise, σ, and generalized error which is the square of its ℓ_2-norm. That is,

$$\Delta\boldsymbol{x}_g^p \triangleq \begin{bmatrix} \Delta\boldsymbol{x}^p \\ \sigma \end{bmatrix},\ \varepsilon_g^p \triangleq \|\Delta\boldsymbol{x}_g^p\|_2^2 = \varepsilon_x^p + \sigma^2. \tag{4.40}$$

The CV error for $\hat{\boldsymbol{x}}^p$ is given by $\epsilon_{\text{cv}}^p = \|\boldsymbol{y}_{\text{cv}} - \boldsymbol{\Phi}_{\text{cv}}\hat{\boldsymbol{x}}^p\|_2^2$. The CV comparison theorem from [60] states:

Theorem 4.5. *[60, Theorem 2] Let $\hat{\boldsymbol{x}}^p$ and $\hat{\boldsymbol{x}}^q$ be two recovered signals and assume there are enough measurements for cross-validation. If $\varepsilon_x^p \geq \varepsilon_x^q$, then it holds with probability $\Phi(\lambda_{p,q})$ that $\epsilon_{cv}^p \geq \epsilon_{cv}^q$, where $\lambda_{p,q}$ is given by*

$$\lambda_{p,q}^2 = \frac{m_{cv}}{2\left[1 + 2(1 - (\rho_g^{p,q})^2)\frac{\varepsilon_g^p \varepsilon_g^q}{(\varepsilon_g^p - \varepsilon_g^q)^2}\right]}, \tag{4.41}$$

$\Phi(u) \triangleq \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{u} e^{-\frac{t^2}{2}}dt$ is the cumulative distribution function of the standard Gaussian distribution, and

$$\rho_g^{p,q} \triangleq \frac{\langle \Delta\boldsymbol{x}_g^p, \Delta\boldsymbol{x}_g^q \rangle}{\|\Delta\boldsymbol{x}_g^p\|_2 \|\Delta\boldsymbol{x}_g^q\|_2} \tag{4.42}$$

is the correlation coefficient of the two generalized recovery error signals.

The condition of 'enough number of measurements for cross-validation' is needed for the application of the Central Limit Theorem in the proof of Theorem 4.5. We proceed with the proof of Theorem 4.2 under the same assumption.

Each pair $p \triangleq (\mathcal{P}, \mu)$ of a set of perturbations $\mathcal{P}$ and the LASSO regularization parameter μ results in a uniquely recovered signal $\hat{\boldsymbol{x}}^p$, with corresponding value of recovery error ε_x^p and generalized recovery error ε_g^p as defined earlier. Let $\mathcal{P}^*$ denote the set of edges upon perturbing which we recover the actual graph from the nominal graph. Let μ^* be the ideal value of the LASSO regularization parameter for the actual graph. Let $p^* \triangleq (\mathcal{P}^*, \mu^*)$. From Theorem 4.5, for any pair p such that $\varepsilon_x^p \geq \varepsilon_x^{p^*}$, we have

$$\Pr[\epsilon_{\text{cv}}^p \geq \epsilon_{\text{cv}}^{p^*}] = \Pr[u \leq \lambda_{p,p^*}], \tag{4.43}$$

where u is a standard Gaussian random variable, and λ_{p,p^*} is given by Eqn. 4.41. Hence the probability of the complement is given by

$$\Pr[\epsilon_{\text{cv}}^p \leq \epsilon_{\text{cv}}^{p^*}] = \Pr[u \geq \lambda_{p,p^*}] \leq e^{-\frac{\lambda_{p,p^*}^2}{2}}. \tag{4.44}$$

The inequality in Eqn. 4.44 is due to a tail bound on the distribution of u, since the standard Gaussian distribution is sub-Gaussian. Let

$$h_p \triangleq (1 - (\rho_g^{p,p^*})^2) \frac{\varepsilon_{\text{g}}^{p^*} \varepsilon_{\text{g}}^p}{(\varepsilon_{\text{g}}^{p^*} - \varepsilon_{\text{g}}^p)^2}. \tag{4.45}$$

Therefore

$$\Pr[\epsilon_{\text{cv}}^p \leq \epsilon_{\text{cv}}^{p^*}] \leq e^{-\frac{m_{\text{cv}}}{4(1+2h_p)}}. \tag{4.46}$$

Since $\rho_g^{p,p^*} \in [-1,1]$, we have

$$(1-(\rho_g^{p,p^*})^2) \leq 1 \tag{4.47}$$

$$\implies h_p \leq \frac{\varepsilon_{\mathrm{g}}^{p^*}\varepsilon_{\mathrm{g}}^{p}}{(\varepsilon_{\mathrm{g}}^{p^*}-\varepsilon_{\mathrm{g}}^{p})^2} \tag{4.48}$$

$$= \frac{(\varepsilon_{\mathrm{g}}^{p^*})^2\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}}{(\varepsilon_{\mathrm{g}}^{p^*})^2\left(1-\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}\right)^2} \tag{4.49}$$

$$= \frac{\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}}{\left(\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}-1\right)^2} \tag{4.50}$$

$$\implies h_p \leq f\left(\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}\right), \tag{4.51}$$

where

$$f(z) = \frac{z}{(z-1)^2}. \tag{4.52}$$

Now, we restrict our consideration to only those p for which $\varepsilon_{\mathrm{g}}^{p} \geq c\varepsilon_{\mathrm{g}}^{p^*}$ for some constant c. We denote this set by $S(c)$:

$$S(c) \triangleq \{p : \frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}} \geq c\} \tag{4.53}$$

We note that $f(z)$ is a monotonically decreasing function of z for $z > 1$, since

$$f'(z) = \frac{1}{(z-1)^2} - \frac{2z}{(z-1)^3} = -\frac{z+1}{(z-1)^3} < 0 \; \forall z > 1. \tag{4.54}$$

Hence if $c > 1$, then

$$f\left(\frac{\varepsilon_{\mathrm{g}}^{p}}{\varepsilon_{\mathrm{g}}^{p^*}}\right) \leq f(c) \;\; \forall p \in S(c) \tag{4.55}$$

$$\implies h_p \leq f(c) \;\; \forall p \in S(c) \;\; \text{[using Eqn. 4.51]} \tag{4.56}$$

We build on this to get an inequality for the R.H.S. of Eqn. 4.46:

$$\forall p \in S(c),$$

$$h_p \leq f(c) \tag{4.57}$$

$$\implies 4(1+2h_p) \leq 4(1+2f(c)) \tag{4.58}$$

$$\implies \frac{m_{\text{cv}}}{4(1+2h_p)} \geq \frac{m_{\text{cv}}}{4(1+2f(c))} \qquad [\ \because h_p \text{ and } f(c) \text{ are both } > 0\ ] \tag{4.59}$$

$$\implies -\frac{m_{\text{cv}}}{4(1+2h_p)} \leq -\frac{m_{\text{cv}}}{4(1+2f(c))} \tag{4.60}$$

$$\implies e^{-\frac{m_{\text{cv}}}{4(1+2h_p)}} \leq e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}} \tag{4.61}$$

$$\implies \Pr[\epsilon^p_{\text{cv}} \leq \epsilon^{p^*}_{\text{cv}}] \leq e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}}. \qquad [\text{ using Eqn. 4.46}] \tag{4.62}$$

Eqn. 4.62 bounds the probability of the events $\epsilon^p_{\text{cv}} \leq \epsilon^{p^*}_{\text{cv}}$ separately for each $p \in S(c)$. However, since we want $\epsilon^{p^*}_{\text{cv}}$ to be less than ϵ^p_{cv} for all $p \in S(c)$ at the same time i.e. we want the probability of the event $\bigcap_{p \in S(c)} (\epsilon^{p^*}_{\text{cv}} \leq \epsilon^p_{\text{cv}})$ to be large, we find an upper bound on the probability of the complement event $\bigcup_{p \in S(c)} (\epsilon^p_{\text{cv}} \leq \epsilon^{p^*}_{\text{cv}})$ using the union bound:

$$\Pr\left[\bigcup_{p \in S(c)} (\epsilon^p_{\text{cv}} \leq \epsilon^{p^*}_{\text{cv}})\right] \leq \sum_{p \in S(c)} \Pr[\epsilon^p_{\text{cv}} \leq \epsilon^{p^*}_{\text{cv}}] \qquad [\ \because \Pr[\bigcup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr[A_i]\ ] \tag{4.63}$$

$$\leq \sum_{p \in S(c)} e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}} \qquad [\text{ from Eqn. 4.62 }] \tag{4.64}$$

$$= |S(c)| e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}}. \qquad [\ \because \text{term inside the sum is a constant}] \tag{4.65}$$

In Eqn. 4.65, $|S(c)|$ denotes the cardinality of the set $S(c)$. This number is less than the total number of pairs of perturbations and LASSO regularization parameter values used for grid search (the pair p^* is not in $S(c)$ for $c > 1$). Hence,

$$|S(c)| < |\Gamma| N, \tag{4.66}$$

where N is the number of perturbations of size $\leq d_0$. Since there are $\binom{n}{2}$ possible edges, of

which s are chosen for perturbation, for all $s \in \{0, \ldots, d_0\}$, hence,

$$N = \binom{\binom{n}{2}}{0} + \binom{\binom{n}{2}}{1} + \cdots + \binom{\binom{n}{2}}{d_0} \tag{4.67}$$

$$\implies N < (d_0 + 1)n^{2d_0} \qquad [\text{ using } \binom{n}{2} < n^2 \text{ and } \binom{n^2}{k} \leq \binom{n^2}{d_0} < n^{2d_0} \text{ for } k \leq d_0 \text{ }] \tag{4.68}$$

$$\implies |S(c)| < |\Gamma|(d_0 + 1)n^{2d_0} \tag{4.69}$$

Putting this in Eqn. 4.65,

$$\Pr\Big[\bigcup_{p \in S(c)} (\epsilon_{\text{cv}}^{p} \leq \epsilon_{\text{cv}}^{p^*})\Big] < |\Gamma|(d_0 + 1)n^{2d_0} e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}} \tag{4.70}$$

We may upper bound the R.H.S. by a constant $\delta \in (0, 1)$ of our choosing:

$$|\Gamma|(d_0 + 1)n^{2d_0} e^{-\frac{m_{\text{cv}}}{4(1+2f(c))}} \leq \delta \tag{4.71}$$

$$\implies \ln(|\Gamma|(d_0 + 1)n^{2d_0}) - \frac{m_{\text{cv}}}{4(1 + 2f(c))} \leq \ln \delta \tag{4.72}$$

$$\implies m_{\text{cv}} \geq 4(1 + 2f(c))\Big\{\ln|\Gamma| + \ln(d_0 + 1) + 2d_0 \ln n - \ln \delta\Big\} \tag{4.73}$$

$$\implies m_{\text{cv}} \geq 4\Big(1 + \frac{2c}{(c-1)^2}\Big)\Big\{\ln|\Gamma| + \ln(d_0 + 1) + 2d_0 \ln n + \ln\frac{1}{\delta}\Big\}. \tag{4.74}$$

Hence, if the condition of Eqn. 4.74 is true, then

$$\Pr\Big[\bigcup_{p \in S(c)} (\epsilon_{\text{cv}}^{p} \leq \epsilon_{\text{cv}}^{p^*})\Big] < \delta. \tag{4.75}$$

Equivalently,

$$\Pr\Big[\bigcap_{p \in S(c)} (\epsilon_{\text{cv}}^{p^*} \leq \epsilon_{\text{cv}}^{p})\Big] > 1 - \delta. \tag{4.76}$$

Hence, with an arbitrarily high probability (more than $1 - \delta$), no pair in $S(c)$ gets chosen by the cross-validation procedure – either p^* gets chosen or some other pair p' for which $\varepsilon_{\text{g}}^{p'}$ is small gets chosen – provided the condition in Eqn. 4.74 is true. In such a case, for the chosen pair $\hat{p}_{\text{bf}}$,

since $\hat{p}_{\text{bf}} \notin S(c)$, it must be true that

$$\varepsilon_{\text{g}}^{\hat{p}_{\text{bf}}} < c\varepsilon_{\text{g}}^{p^*} \qquad \text{[from definition of } S(c) \text{ in Eqn. 4.53]}. \tag{4.77}$$

Using defintion of $\varepsilon_{\text{g}}^{p}$ and $\varepsilon_{\text{x}}^{p}$ (Eqns. 4.40 and 4.39) and putting in Eqn. 4.77,

$$\varepsilon_{\text{x}}^{\hat{p}_{\text{bf}}} + \sigma^2 < c\varepsilon_{\text{x}}^{p^*} + c\sigma^2 \tag{4.78}$$

$$\implies \varepsilon_{\text{x}}^{\hat{p}_{\text{bf}}} < c\varepsilon_{\text{x}}^{p^*} + (c-1)\sigma^2 \tag{4.79}$$

$$\implies ||\boldsymbol{x}^* - \hat{\boldsymbol{x}}_{\text{bf}}||_2^2 < c||\boldsymbol{x}^* - \hat{\boldsymbol{x}}_{\text{actual}}||_2^2 + (c-1)\sigma^2. \tag{4.80}$$

Eqns. 4.74, 4.76 and 4.80 together prove Theorem 4.2. □

This page was intentionally left blank.

Chapter 5

Conclusion

In this thesis, we have worked on three novel applications of compressed sensing, with unique challenges required to be solved for each. We have presented novel techniques for solving each problem, advancing the field of compressed sensing in a different aspect in each application. The measurement noise models and signal models are different for each of the three applications, and these differences influence the algorithms devised to solve the associated computational tasks. This point is elaborated upon in the following paragraphs.

For the problem of Pooled RT-PCR testing for COVID-19, there was additional information in the measurement noise due to it being heteroscedastic (i.e. different measurements have different noise variances and in fact, the negative pooled results are noiseless), which is typically not taken into account by standard CS recovery algorithms. We employed group testing methods to exploit this additional information, at the same time reducing the size of the problem to be solved by the CS algorithm such that the combined method Tapestry outperformed standalone group testing and CS methods. We proved theoretical recovery guarantees of this combined method, by showing that sufficient conditions for CS recovery still hold for the reduced problem. We used matrices derived from Kirkman triple systems and proved useful properties of it which make them especially suitable for our algorithm.

For the problem of Efficient Automated Image Moderation, we brought CS methods to neural network based imbalanced-class classification, using our Quantitative Matrix-Pooled Neural Network (QMPNN) to predict the number of objectionable images in a pool, which are then decoded using CS recovery algorithms. We extended our Compressed Sensing based image moderation methods to deep outlier detection, by extracting pool-level feature vectors from the QMPNN, learning the distribution of the feature vectors of training data pools of a Gaussian Mixture Model (GMM), and using the negative log probability density of the GMM on feature vectors at test time as anomaly scores to determine the number of outlier images in that pool. We demonstrated that our QMPNN-based methods need significantly lower computation and time than the method of processing each image separately using a neural network, while maintaining reasonable performance on the respective tasks.

Finally, we dealt with uncertainty in the orthonormal basis in the CS forward model in the novel problem of Compressive Perturbed Graph Recovery, in which graph signals sparse in the domain of the eigenvectors of the Laplacian matrix of a graph needed to be recovered from compressive linear measurements, but there was uncertainty in the eigenvectors due to the graph not being fully known. We solved this problem by using the cross-validation of recovered signals to compare graphs, and used a greedy strategy to refine the nominal graph one edge at a time. We proved bounds on the number of held-out measurements needed by a brute-force version of this algorithm to accurately recover the graph signal along with the actual graph. We framed the problem of recovery of compressively acquired images as a compressive perturbed graph recovery problem, and proposed an inferrred linear-edge compressive image recovery (ILECIR) algorithm which performs better recovery of images with sharp edges in them than the baseline method of using the Discrete Cosine Transform as the orthonormal basis in a CS recovery algorithm. In this method, partitioned graphs were created from a 2D lattice graph by dropping the edges going across hypothesis linear image edges in a patch, and were compared via cross-validation to infer the actual edge in the image patch.

To summarize, we note that the three applications deal with different noise models and signal models. The pooled RT-PCR testing application assumes non-negative, sparse and real-valued signals and signal-dependent measurement noise with negative (i.e. non-infected) pools yielding noiseless results. This influences our algorithm design which involves a traditional group testing algorithm like COMP followed by a traditional CS algorithm. The image mod-

eration application assumes binary-valued and sparse signals, and small levels of noise in the outputs of the QMPNN which again influences the choice of algorithms such as LASSO. Lastly, the graph signal recovery problem involves measurement noise that is induced by perturbations to the underlying graph and thereby to the eigenvectors of the Laplacian matrix. This is a very different form of (signal dependent) measurement noise which again influences the design of algorithms such as GES presented in the last chapter.

5.1 Future Work

The work done in this thesis opens up new questions and avenues of research in each of the three problems considered.

5.1.1 Pooled RT-PCR Testing for COVID-19

We presented the performance of our scheme for the case when there are no pools falsely negative or positive. However, due to dilution, contamination, or pooling errors, pooled tests may in some cases turn out to falsely be negative or positive. We have already presented a scheme to detect such errors in some cases using a COMP consistency check as discussed in Sec. 2.4.3. We have also shown that NCOMP may be used in the first stage of Tapestry, due to preservation of RIP (Sec. 2.3.5). Synthetic experiments are needed to verify the practicality of Tapestry with NCOMP, since we need reasonably-sized matrices which should be sufficiently sparse in order to be used for pooling the samples in practice. In general, pooling schemes which tolerate more error will use more pools in total, and each sample will go to more number of pools. Usage of matrices based on Steiner Systems or solutions of the Social Golfer problem may be considered, since they are generalizations of Steiner Triple Systems and Kirkman Triple Systems, respectively (Sec. 2.3.7.3 and Sec. 2.3.7.7).

Contamination or pooling errors may lead to measurement noise (in positive or negative pools) which is arbitrarily far from zero, whereas in CS a typical assumption is that the ℓ_2-norm of the measurement noise is bounded. The NNLAD objective (Sec. 2.3.4.4 and [84]) may

be suitable for handling such errors, since the minimizer of ean absolute deviation tends to be stable in the presence of outlier terms, whereas mean square error is easily influenced by outliers. New algorithms may be designed specifically with a model for pooling error in mind – for example, in case of only one pooling error, the amount of noise due to pooling error in a pool will be equal to the viral load of some sample which was erroneously added to or excluded from that pool.

CS algorithms typically assume additive Gaussian noise, whereas the noise model presented in our work was multiplicative. New CS algorithms which take into account such multiplicative noise may perform better.

We used matrices which have both disjunctness as well as RIP-1; however, a more precise formulation of the properties needed specifically for the Tapestry algorithm might be desirable. For example, perhaps such matrices should be (k, l)-list-disjunct [175] – i.e. if such a matrix is used for pooling, then in the presence of upto k defective items, COMP returns a set of at most $k+l-1$ items which contain all the defective items – *and* have RIP of order at least $2k$, enabling CS recovery in the second stage. Tighter bounds on the number of measurements needed and the error of the signal recovered by Tapestry may be found.

5.1.2 Efficient Automated Image Moderation

The main problem discussed in Chapter 3 is a quantitative group testing (QGT) problem [131], since the measurements give the count of objectionable images in a pool. Whether the pooling schemes and decoding algorithms from the noisy QGT literature are practically applicable to our problem is an open question and needs more exploration.

The studied problem considers the case when no correlation between two images exists. However, some side-information may be exploited to infer such correlation. For example, images uploaded by the same user, or from the same IP address, or by users who are friends with each other on the social media platform may be correlated in terms of having objectionable content. Exploitation of such side-information may enable more accurate detection of objectionable images.

Finally, the outputs of the softmax layer of the QMPNN may be used to obtain a probability

distribution over the possible values of the number of objectionable images in a pool, instead of obtaining a single value via argmax. Similarly, a probability distribution over the presence or absence of an objectionable image may be obtained from the neural network in case of binary group testing. To the best of our knowledge, such GT or CS problems with probabilistic measurements have not been studied. Such probabilistic outputs could possibly be incorporated in a message passing [176, 177] or Gibbs sampling [178] framework to improve upon the results presented in this thesis.

5.1.3 Compressive Perturbed Graph Recovery

The cross-validation based approaches presented in Chapter 4 are applicable to weighted graphs for the case when some edges have been added to the actual graph, but not in cases when some edges have been removed from it, since the weight of a new edge in a candidate graph will not be known. Such an extension would require a different approach – such as gradient descent of the GFT-LASSO objective (Sec. 4.4) in terms of the eigenvectors of the graph and the signal vector. The matrix could be parameterized in terms of its eigendecomposition. Additional constraints must be enforced to ensure that the resulting matrix remains a valid Laplacian matrix – i.e., the off-diagonal elements must be non-positive, and the rows must sum to zero. The ℓ_1-norm of the difference in weights of a candidate graph and the nominal graph could also be penalized in order to ensure that only a few edges are perturbed, and the total weighted perturbation is not too much.

Such a gradient-based approach may be used even for unweighted graphs by assuming that they are weighted, and binarizing the weights based on a threshold to obtain the final unweighted graph. This may present a faster approach than the cross-validation based approach considered in this thesis.

More applications could be framed as a compressive perturbed graph recovery problem. For example, consider pooled testing for COVID-19, with a contact trace graph available (i.e. who came into contact with who and for how long). Such a graph may be collected via an app installed on phones which uses bluetooth signal strength to determine proximity with other phones. The contact trace graph itself may have errors – some edges may be missing because of

bluetooth connectivity issues, while spurious edges might be present due to a user temporarily not carrying their phone. Pooled samples will contain the sum of viral loads in samples collected from people. The viral load vector will be sparse due to low infection rate, as assumed in Chapter 2. However, the viral loads of two people who have come in contact with each other will in general not have any correlation. Only the positivity vector – a binary vector indicating who is infected and who is not – will have such correlation, and will be sparse or compressible in the GFT domain of the actual contact trace graph. Thus this is another compressive perturbed graph recovery problem, with a slight twist, and new methods will need to be developed in order to solve it. Promising approaches might be to either combine our cross-validation based approach with the approximate message passing (AMP) based approach in [54], or incorporate random edge perturbations in the AMP framework.

On the theoretical side, whether the recovery of the actual graph from noiseless measurements is possible for all actual graphs given a small number of perturbations is not clear – perhaps in some graphs, some eigenvectors do not get perturbed upon edge perturbations due to some symmetries in the graph structure. Recovery of the actual graph from noisy measurements is only guaranteed if recovery using the nominal graph and other candidate graphs are sufficiently bad, as stated in the condition in Eqn. 4.25. We see this in practice in Sec. 4.6.1.4, where intra-cluster edges are harder to detect due to not causing a large enough perturbation of the eigenvectors. A refinement of the condition in Eqn. 4.25 in terms of eigenvector perturbation will lead to better insight into this problem.

5.2 Closing Remarks

We have made significant advancements in the field of compressed sensing while proposing novel solutions for each problem considered in this thesis. We have laid out future work stemming from the contributions in this thesis. We have proven theoretical guarantees for two out the three methods proposed, and have demonstrated the effectiveness of our techniques via extensive experiments.

References

[1] D. Donoho, Compressed sensing, IEEE Transactions on Information Theory 52 (4) (2006) 1289–1306. doi:10.1109/TIT.2006.871582. 1

[2] E. Candes, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Transactions on Information Theory 52 (2) (2006) 489–509. doi:10.1109/TIT.2005.862083. 1

[3] E. Candes, M. Wakin, An introduction to compressive sampling, IEEE Signal Processing Magazine 25 (2) (2008) 21–30. 1, 24, 31, 32, 57, 93, 94, 134, 142

[4] E. Candes, The restricted isometry property and its implications for compressive sensing, Comptes Rendus Mathematiques (2008). 2, 25, 31, 32, 125, 134, 143

[5] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) 58 (1) (1996) 267–288. 2, 142

[6] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity: The LASSO and Generalizations, CRC Press, 2015. 2, 36, 93, 94, 143, 144

[7] M. Tipping, Sparse Bayesian learning and the relevance vector machine, Journal of Machine Learning Research (2001). 2, 37

[8] D. Wipf, B. D. Rao, Sparse Bayesian learning for basis selection, IEEE Trans. Signal Processing 52 (8) (2004). 2, 37

[9] O. Teke, A. C. Gurbuz, O. Arikan, Perturbed orthogonal matching pursuit, IEEE Transactions on Signal Processing 61 (24) (2013) 6220–6231. 2, 149

[10] M. Lotfi, M. Vidyasagar, Compressed sensing using binary matrices of nearly optimal dimensions, IEEE Transactions on Signal Processing 68 (2020) 3008–3021. 2, 32, 33, 44, 45, 48, 57

[11] R. Baraniuk, M. Davenport, R. DeVore, M. Wakin, A simple proof of the restricted isometry property for random matrices, Constr Approx 28 (2008) 253–263. 3, 32, 125, 143

[12] R. Kueng, P. Jung, Robust nonnegative sparse recovery and the nullspace property of 0/1 measurements, IEEE Transactions on Information Theory 64 (2) (2018) 689–703. 3, 24, 33, 38, 47, 73, 125

[13] Wikipedia contributors, Steiner triple systems, https://en.wikipedia.org/wiki/Steiner_system#Steiner_triple_systems (2021). 3, 42

[14] S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agarwal, Y. Gupta, S. Consul, N. Gupta, Ritika, R. Goenka, A. Rajwade, M. Gopalkrishnan, A compressed sensing approach to pooled rt-pcr testing for covid-19 detection, IEEE Open Journal of Signal Processing 2 (2021) 248–264. doi:10.1109/OJSP.2021.3075913. 3, 10, 26, 94, 125, 127

[15] S. Foucart, H. Rauhut, A Mathematical Introduction to Compressive Sensing, Birkhauser, 2013. 3

[16] S. Ghosh, A. Rajwade, S. Krishna, N. Gopalkrishnan, T. E. Schaus, A. Chakravarthy, S. Varahan, V. Appu, R. Ramakrishnan, S. Ch, M. Jindal, V. Bhupathi, A. Gupta, A. Jain, R. Agarwal, S. Pathak, M. A. Rehan, S. Consul, Y. Gupta, N. Gupta, P. Agarwal, R. Goyal, V. Sagar, U. Ramakrishnan, S. Krishna, P. Yin, D. Palakodeti, M. Gopalkrishnan, Tapestry: A single-round smart pooling technique for COVID-19 testing, medRxiv https://www.medrxiv.org/content/10.1101/2020.04.23.20077727v2 (2020). 3, 10, 24, 26, 58, 63

[17] S. Ghosh, S. Saxena, A. Rajwade, Efficient neural network based classification and outlier detection for image moderation using compressed sensing and group testing (2023). arXiv:2305.07639. 3, 15, 99

[18] S. Ghosh, A. Rajwade, Compressive perturbed graph recovery, in preparation (2023). 3

[19] M. Aldridge, O. Johnson, J. Scarlett, et al., Group Testing: An Information Theory Perspective, Foundations and Trends® in Communications and Information Theory 15 (3-4) (2019) 196–392. 4, 6

[20] World Health Organisation, Coronavirus disease (COVID-19) pandemic, https://www.who.int/emergencies/diseases/novel-coronavirus-2019. 5

[21] American Society for Microbiology Communications, Shortages of COVID-19 and other testing supplies identified by ASMs data collection tool, https://asm.org/Press-Releases/2020/October/Shortages-of-COVID-19-and-Other-Testing-Supplies-I. 5

[22] WOWT 6 News, Symptomatic people question why they are denied COVID-19 tests, https://www.1011now.com/content/news/Symptomatic-people-question-why-they-are-denied-COVID-19-tests-569119301.html. 5

[23] P. Habibzadeh, M. Mofatteh, M. Silawi, S. Ghavami, M. A. Faghihi, Molecular diagnostic assays for covid-19: an overview, Critical reviews in clinical laboratory sciences 58 (6) (2021) 385–398. 6

[24] N. Jawerth, How is the COVID-19 virus detected using real time RT-PCR?, https://www.iaea.org/newscenter/news/how-is-the-covid-19-virus-detected-using-real-time-rt-pcr. 6, 26

[25] V. M. Corman, O. Landt, M. Kaiser, R. Molenkamp, A. Meijer, D. K. Chu, T. Bleicker, S. Brünink, J. Schneider, M. L. Schmidt, et al., Detection of 2019 novel coronavirus (2019-ncov) by real-time rt-pcr, Eurosurveillance 25 (3) (2020) 2000045. 6

[26] K. Guo, S. Wustoni, A. Koklu, E. Díaz-Galicia, M. Moser, A. Hama, A. A. Alqahtani, A. N. Ahmad, F. S. Alhamlan, M. Shuaib, et al., Rapid single-molecule detection of covid-19 and mers antigens via nanobody-functionalized organic electrochemical transistors, Nature biomedical engineering 5 (7) (2021) 666–677. 6

[27] D. Du, F. K. Hwang, F. Hwang, Combinatorial group testing and its applications, World Scientific, 2000. 6, 66

[28] R. Dorfman, The detection of defective members of large populations, The Annals of Mathematical Statistics 14 (4) (1943) 436–440. 6, 24, 40, 55, 81, 94, 99

[29] C. L. Chan, P. H. Che, S. Jaggi, V. Saligrama, Non-adaptive probabilistic group testing with noisy measurements: Near-optimal bounds with efficient algorithms, in: 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2011, pp. 1832–1839. 6, 10, 25, 30, 40, 47, 62, 73, 106

[30] P. M. Beldomenico, Do superspreaders generate new superspreaders? a hypothesis to explain the propagation pattern of COVID-19, International Journal of Infectious Diseases 96 (2020) 461–463. 7, 24, 62, 76

[31] Y. Liu, et al., Viral dynamics in mild and severe cases of COVID-19, The Lancet Infectious Diseases (2020). 7, 24

[32] D. K. Ray-Chaudhuri, R. M. Wilson, Solution of Kirkman's schoolgirl problem, in: Proc. symp. pure Math, Vol. 19, 1971, pp. 187–203. 9, 24, 42, 43

[33] S. J. Johnson, S. R. Weller, Construction of low-density parity-check codes from kirkman triple systems, in: GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270), Vol. 2, IEEE, 2001, pp. 970–974. 9, 43, 44

[34] S. Ghosh, V. Appu, R. Ramakrishnan, S. Ch, M. Jindal, V. Bhupathi, A. Gupta, A. Jain, M. Gopalkrishnan, Byom app, https://rebrand.ly/byom-app. 10, 25, 65

[35] M. Gopalkrishnan, S. Ghosh, A. V. Rajwade, D. Palakodeti, S. Krishna, Methods and systems for determining viruses in biological samples using a single round based pooling, US Patent Application 20220170118 (pending), https://www.freepatentsonline.com/y2022/0170118.html (2022). 10

[36] Algorithmic Biologics, https://algorithmicbiologics.com. 10

[37] J. Ruiz-Santaquiteria, et al., Improving handgun detection through a combination of visual features and body pose-based data, Pattern Recognition 136 (2023) 109252. 11, 94

[38] M. Grega, A. Matiolański, P. Guzik, M. Leszczuk, Automated detection of firearms and knives in a cctv image, Sensors 16 (1), 47 (2016). doi:10.3390/s16010047. 11, 94, 113, 128, 129

[39] R. Debnath, M. K. Bhowmik, A comprehensive survey on computer vision based concepts, methodologies, analysis and applications for automatic gun/knife detection, Journal of Visual Communication and Image Representation 78 (2021) 103165. 11, 94, 113

[40] A. M. Atto, A. Benoit, P. Lambert, Timed-image based deep learning for action recognition in video sequences, Pattern Recognition 104 (2020) 107353. 11, 94

[41] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access (2018). 11, 94

[42] A. Badar, A. Varma, A. Staniec, M. Gamal, O. Magdy, H. Iqbal, E. Arani, B. Zonooz, Highlighting the importance of reducing research bias and carbon emissions in CNNs, in: International Conference of the Italian Association for Artificial Intelligence, Springer, 2022, pp. 515–531. 11, 95

[43] Meta, Inc, Facebook Community Standards Enforcement Report - Q4 2021 - Adult Nudity and Sexual Activity, https://transparency.fb.com/data/community-standards-enforcement/adult-nudity-and-sexual-activity/facebook/#prevalence. 11, 95

[44] Reddit Inc, Transparency Report 2020, https://www.redditinc.com/policies/transparency-report-2020-1. 11, 13, 95

[45] W. Liang, J. Zou, Neural group testing to accelerate deep learning, in: IEEE International Symposium on Information Theory (ISIT), 2021, pp. 958–963. 12, 14, 15, 95, 96, 97, 101, 102, 103, 106, 107, 111, 112, 114, 115, 128

[46] Wikipedia contributors, Sensitivity and specificity, https://en.wikipedia.org/wiki/Sensitivity_and_specificity. 13

[47] S. Jhaver, I. Birman, E. Gilbert, A. Bruckman, Human-machine collaboration for content regulation: The case of reddit automoderator, ACM Transactions on Computer-Human Interaction (TOCHI) 26 (5) (2019) 1–35. 13, 97, 107

[48] A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, IEEE Transactions on Signal Processing 61 (7) (2013) 1644–1656. doi:10.1109/TSP.2013.2238935. 16, 134

[49] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE signal processing magazine 30 (3) (2013) 83–98. 16, 134, 137, 139, 146

[50] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (5) (2018) 808–828. 16, 20, 134, 137, 139

[51] G. Wallace, The jpeg still picture compression standard, IEEE Transactions on Consumer Electronics 38 (1) (1992) xviii–xxxiv. doi:10.1109/30.125072. 16, 134, 160

[52] G. Fracastoro, S. M. Fosson, E. Magli, Steerable discrete cosine transform, IEEE Transactions on Image Processing 26 (1) (2016) 303–314. 16, 134, 160

[53] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, R. G. Baraniuk, Single-pixel imaging via compressive sampling, IEEE signal processing magazine 25 (2) (2008) 83–91. 16, 142, 162

[54] R. Goenka, S. Cao, C. Wong, A. Rajwade, D. Baron, Contact tracing enhances the efficiency of covid-19 group testing, in: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021, 2021, pp. 8168–8172. 16, 94, 210

[55] Y. Chi, L. L. Scharf, A. Pezeshki, A. R. Calderbank, Sensitivity to basis mismatch in compressed sensing, IEEE Trans. Signal Processing 59 (5) (2011) 2182–2195. 17, 149

[56] J. M. Nichols, A. K. Oh, R. M. Willett, Reducing basis mismatch in harmonic signal recovery via alternating convex search, IEEE Signal Process. Lett. 21 (8) (2014) 1007–1011. 17, 149

[57] Z. Tan, Y. Peng, A. Nehorai, Joint sparse recovery method for compressed sensing with structured dictionary mismatches, IEEE Transactions on Signal Processing 62 (19) (2014) 4997–5008. 17, 149

[58] A. Aldroubi, X. Chen, A. Powell, Perturbations of measurement matrices and dictionaries in compressed sensing, Appl. Comput. Harmon. Anal. 33 (2) (2012). 17, 149

[59] R. Ward, Compressed sensing with cross validation, IEEE Transactions on Information Theory 55 (12) (2009) 5773–5782. 18, 144

[60] J. Zhang, L. Chen, P. T. Boufounos, Y. Gu, On the theoretical analysis of cross validation in compressive sensing, in: ICASSP, 2014, pp. 3370–3374. 18, 20, 67, 135, 144, 167, 198

[61] J. Zhang, L. Chen, P. T. Boufounos, Y. Gu, Cross validation in compressive sensing and its application of omp-cv algorithm, arXiv preprint arXiv:1602.06373 (2016). 18, 20, 144

[62] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, A. Ashok, Reconnet: Non-iterative reconstruction of images from compressively sensed measurements, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 449–458. 18, 97, 135, 162

[63] R. Kerviche, N. Zhu, A. Ashok, Information-optimal scalable compressive imaging system, in: Computational Optical Sensing and Imaging, Optica Publishing Group, 2014, pp. CM2D–2. 18, 135, 162

[64] M. Newman, Networks, Oxford University Press, 2018. 20, 136

[65] D. Benatia, R. Godefroy, J. Lewis, Estimating COVID-19 prevalence in the united states: A sample selection model approach, https://www.medrxiv.org/content/10.1101/2020.04.20.20072942v1 (2020). 23, 27

[66] ISRAEL21c Staff, Israelis introduce method for accelerated covid-19 testing, https://www.israel21c.org/israelis-introduce-method-for-accelerated-covid-19-testing/. 24

[67] Healthcare-in-Europe.com (HiE), Corona 'pool testing' increases worldwide capacities many times over, https://healthcare-in-europe.com/en/news/corona-pool-testing-increases-worldwide-capacities-many-times-over.html. 24

[68] E. W. Weisstein, Kirkman's schoolgirl problem, https://mathworld.wolfram.com/KirkmansSchoolgirlProblem.html, from MathWorld–A Wolfram Web Resource. 24, 42, 43

[69] A. Chakravarthy, S. Krishna, S. Ghosh, A. Tomar, S. Varahan, A. Rajwade, S. Ghosh, N. Gupta, R. Agarwal, H. Payal, P. Chakraborty, K. V. Vemula, A. Vyas, R. Goru, S. Krishna, D. Palakodeti, M. Gopalkrishnan, Large-scale testing for sars-cov-2 using tapestry pooling, medRxiv (2020). doi:10.1101/2020.10.09.20209742. 26

[70] Thermo Fisher Scientific Inc, Efficiency of Real-Time PCR, https://www.thermofisher.com/in/en/home/life-science/pcr/real-time-pcr/real-time-pcr-learning-center/real-time-pcr-basics/efficiency-real-time-pcr-qpcr.html. 28, 54, 77

[71] M. Aldridge, L. Baldassini, O. Johnson, Group testing algorithms: Bounds and simulations, IEEE Transactions on Information Theory 60 (6) (2014) 3671–3687. 30, 45

[72] A. Gilbert, M. Iwen, M. Strauss, Group testing and sparse signal recovery, in: Asilomar Conference on Signals, Systems and Computers, 2008, p. 1059–1063. 31, 94

[73] N. Zhao, D. O'Connor, A. Basarab, D. Ruan, K. Sheng, Motion compensated dynamic mri reconstruction with local affine optical flow estimation, IEEE Trans. on Biomedical Engg. 66 (11) (2019). 31

[74] Z. Zhang, T.-P. Jung, S. Makeig, B. D. Rao, Compressed sensing for energy-efficient wireless telemonitoring of noninvasive fetal ECG via block sparse bayesian learning, IEEE Trans. Biomedical Engg. 60 (2) (2013). 31

[75] Y. Liu, M. D. Vos, S. V. Huffel, Compressed sensing of multichannel EEG signals: The simultaneous cosparsity and low-rank optimization, IEEE Trans. Biomedical Engg. 62 (8) (2015). 31

[76] R. DeVore, Deterministic construction of compressed sensing matrices, J. Complexity 23 (2007) 918–925. 32, 57

[77] M. Davenport, M. Duarte, Y. Eldar, G. Kutyniok, Introduction to compressed sensing, in: Y. Eldar, G. Kutyniok (Eds.), Compressed Sensing: Theory and Applications, Cambridge University Press, 2012, p. 1–64. 32

[78] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, M. J. Strauss, Combining geometry and combinatorics: A unified approach to sparse signal recovery, in: 46th Annual Allerton

Conference on Communication, Control, and Computing, 2008, pp. 798–805. 32, 41, 42, 44

[79] Y. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with application to wavelet decomposition, in: Asilomar Conf. On Signals, Systems and Computing, 1993, pp. 40–44. 36

[80] T. T. Cai, L. Wang, Orthogonal matching pursuit for sparse signal recovery with noise, IEEE Transactions on Information Theory 57 (7) (2011) 4680–4688. 36

[81] M. Yaghoobi, D. Wu, M. Davies, Fast non-negative orthogonal matching pursuit, IEEE Signal Processing Letters 22 (9) (2015) 1229–1233. 37

[82] E. Crespo Marques, N. Maciel, L. Naviner, H. Cai, J. Yang, A review of sparse recovery algorithms, IEEE Access 7 (2019) 1300–1322. 37

[83] A. Nalci, I. Fedorov, M. Al-Shoukairi, T. T. Liu, B. D. Rao, Rectified gaussian scale mixtures and the sparse non-negative least squares problem, IEEE Transactions on Signal Processing 66 (12) (2018) 3124–3139. 37

[84] H. B. Petersen, B. Bah, P. Jung, Efficient tuning-free l1-regression of nonnegative compressible signals, Frontiers in Applied Mathematics and Statistics 7 (2021). doi:10.3389/fams.2021.615573. 38, 207

[85] M. Lotfi, M. Vidyasagar, A fast noniterative algorithm for compressive sensing using binary measurement matrices, IEEE Transactions on Signal Processing 66 (15) (2018). 41, 42

[86] M. Raginsky, S. Jafarpour, Z. T. Harmany, R. Marcia, R. Willett, R. Calderbank, Performance bounds for expander-based compressed sensing in poisson noise, IEEE Trans. Sig. Processing 59 (9) (2011). 41

[87] E. J. Pegg, Social golfer problem, https://mathworld.wolfram.com/SocialGolferProblem.html, from MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. (2021). 43, 47

[88] E. J. Pegg, Math games: Social golfer problem, http://www.mathpuzzle.com/MAA/54-Golf%20Tournaments/mathgames_08_14_07.html. 43

[89] M. Triska, Solution Methods for the Social Golfer Problem, Master's thesis, Vienna University of Technology (2008). 43

[90] I. Dotú, P. Van Hentenryck, Scheduling social golfers locally, in: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, Springer, 2005, pp. 155–167. 43

[91] V. Vermeirssen, B. Deplancke, M. I. Barrasa, J. S. Reece-Hoyes, H. E. Arda, C. A. Grove, N. J. Martinez, R. Sequerra, L. Doucette-Stamm, M. R. Brent, et al., Matrix and steiner-triple-system smart pooling assays for high-performance transcription regulatory network mapping, Nature methods 4 (8) (2007) 659–664. 43

[92] Wikipedia contributors, Steiner systems, `https://en.wikipedia.org/wiki/Steiner_system` (2021). 43

[93] V. D. Tonchev, Steiner systems for two-stage disjunctive testing, Journal of combinatorial optimization 15 (1) (2008) 1–6. 44

[94] C. Studer, R. Baraniuk, Stable restoration and separation of approximately sparse signals, Applied and Computational Harmonic Analysis 37 (1) (2014) 12 – 35. 46, 124

[95] V. Abdoghasemi, S. Ferdowsi, B. Makkiabadi, S. Sanei, On optimization of the measurement matrix for compresive sensing, in: EUSIPCO, 2010. 48

[96] V. Bioglio, T. Bianchi, E. Magli, On the fly estimation of the sparsity degree in compressed sensing using sparse sensing matrices, in: ICASSP, 2015, p. 3801–3805. 57

[97] C. Ravazzi, S. Fosson, T. Bianchi, E. Magli, Sparsity estimation from compressive projections via sparse random matrices, EURASIP J. Adv. Signal Process. 56 (2018). 57

[98] U.S. Food and Drug Adminstration, In vitro diagnostics euas, section: Templates for eua submissions/diagnostic templates (molecular and antigen), bullet: Molecular diagnostic template for laboratories, `https://www.fda.gov/medical-devices/coronavirus-disease-2019-covid-19-emergency-use-authorizations-medical-devices/in-vitro-diagnostics-euas#covid19ivdtemplates`, retrieved 27-Mar-21. 61, 62

[99] J. Yi, R. Mudumbai, W. Xu, Low-cost and high-throughput testing of COVID-19 viruses and antibodies via compressed sensing: System concepts and computational experiments, arXiv preprint arXiv:2004.05759 (2020). 63

[100] N. Shental, S. Levy, V. Wuvshet, S. Skorniakov, B. Shalem, A. Ottolenghi, Y. Greenshpan, R. Steinberg, A. Edri, R. Gillis, M. Goldhirsh, K. Moscovici, S. Sachren, L. M. Friedman, L. Nesher, Y. Shemer-Avni, A. Porgador, T. Hertz, Efficient high-throughput SARS-CoV-2 testing to detect asymptomatic carriers, Science Advances (2020). doi:10.1126/sciadv.abc5961. 63, 64, 94

[101] H. Petersen, B. Bah, P. Jung, Practical high-throughput, non-adaptive and noise-robust SARS-CoV-2 testing, https://arxiv.org/abs/2007.09171 (2020). 63, 64

[102] J. Zhu, K. Rivera, D. Baron, Noisy pooled PCR for virus testing, https://arxiv.org/abs/2004.02689 (2020). 63

[103] J.-T. Seong, Group testing-based robust algorithm for diagnosis of COVID-19, Diagnostics 10 (6) (2020) 396. 63, 64

[104] M. Täufer, Rapid, large-scale, and effective detection of COVID-19 via non-adaptive testing, Journal of Theoretical Biology 506 (2020). 63, 64

[105] H. Nida, S. Blum, D. Zielinski, D. A. Srivastava, R. Elbaum, Z. Xin, Y. Erlich, E. Fridman, N. Shental, Highly efficient de novo mutant identification in a sorghum bicolor tilling population using the comseq approach, The Plant Journal 86 (4) (2016) 349–359. 63

[106] D. J. MacKay, Good error-correcting codes based on very sparse matrices, IEEE transactions on Information Theory 45 (2) (1999) 399–431. 63

[107] R. R. Naidu, P. Jampana, C. S. Sastry, Deterministic compressed sensing matrices: Construction via euler squares and applications, IEEE Transactions on Signal Processing 64 (14) (2016) 3566–3575. 63

[108] N. Thierry-Mieg, A new pooling strategy for high-throughput screening: the shifted transversal design, BMC bioinformatics 7 (1) (2006) 28. 63

[109] A. Heidarzadeh, K. R. Narayanan, Two-stage adaptive pooling with rt-qpcr for COVID-19 screening, arXiv preprint arXiv:2007.02695 (2020). 64

[110] S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agarwal, Y. Gupta, S. Consul, N. Gupta, Ritika, R. Goenka, A. Rajwade, M. Gopalkrishnan, Tapestry code, https://github.com/atoms-to-intelligence/tapestry. 65

[111] Algorithmic Biologics, Tapestry website, https://algorithmicbiologics.com/technology.html. 65

[112] Y. Li, G. Raskutti, Minimax optimal convex methods for Poisson inverse problems under lq-ball sparsity, IEEE Trans. Information Theory 64 (8) (2018). 67

[113] J. Scarlett, V. Cevher, Phase transitions in the pooled data problem, in: Advances in Neural Information Processing Systems, Vol. 30, 2017. 94, 123

[114] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, 2012. 94

[115] S. Diamond, S. Boyd, CVXPY: A Python-embedded modeling language for convex optimization, Journal of Machine Learning Research 17 (83) (2016) 1–5. 105

[116] Gurobi Optimization, LLC., Gurobi Optimizer Reference Manual, https://www.gurobi.com/documentation/current/refman/index.html (2022). 105

[117] Gurobi Optimization, LLC, Mixed-integer programming (MIP) – a primer on the basics, https://www.gurobi.com/resource/mip-basics. 105

[118] K. Lee, K. Lee, H. Lee, J. Shin, A simple unified framework for detecting out-of-distribution samples and adversarial attacks, Advances in neural information processing systems 31 (2018). 107

[119] V. Sehwag, M. Chiang, P. Mittal, SSD: a unified framework for self-supervised outlier detection, in: ICLR, 2021. 107

[120] P. Erdős, P. Frankl, Z. Füredi, Families of finite sets in which no set is covered by the union of r others, Israel J. Math 51 (1-2) (1985) 79–89. 112, 126

[121] A. Mazumdar, On almost disjunct matrices for group testing, in: International Symposium on Algorithms and Computation, 2012. 112, 124, 126

[122] IMFDB Contributers, Internet movie firearms database – guns in movies, tv and video games, http://www.imfdb.org. 113

[123] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV) 115 (3) (2015) 211–252. doi:10.1007/s11263-015-0816-y. 113, 119, 128

[124] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: CVPR, 2017, pp. 5987–5995. 113, 131

[125] Pytorch Team, Resnext pytorch, https://pytorch.org/hub/pytorch_vision_resnext. 113

[126] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences 2 (1) (2009) 183–202. doi:10.1137/080716542. 118

[127] Google Cloud Platform, GPU pricing, https://cloud.google.com/compute/gpus-pricing, online, accessed 20-Dec-2023. 118

[128] Google Cloud Platform, VM instance pricing, https://cloud.google.com/compute/vm-instance-pricing, online, accessed 20-Dec-2023. 118

[129] Advanced Micro Devices, Inc, AMD μProf, https://www.amd.com/en/developer/uprof.html. 118

[130] A. Riesen, C. Couder, D. Potapov, et al., perf: Linux profiling with performance counters , https://perf.wiki.kernel.org/index.php/Main_Page. 118

[131] E. Karimi, F. Kazemi, A. Heidarzadeh, K. R. Narayanan, A. Sprintson, Non-adaptive quantitative group testing using irregular sparse graph codes, in: ACCC, 2019, pp. 608–614. 123, 208

[132] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, M. Strauss, Combining geometry and combinatorics: A unified approach to sparse signal recovery, in: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, IEEE, 2008, pp. 798–805. 124, 126, 127

[133] R. Goenka, S.-J. Cao, C.-W. Wong, A. Rajwade, D. Baron, Contact tracing information improves the performance of group testing algorithms, https://arxiv.org/abs/2106.02699 (2021). 125

[134] O. Gebhard, M. Hahn-Klimroth, D. Kaaser, P. Loick, On the parallel reconstruction from pooled data, in: IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2022, pp. 425–435. doi:10.1109/IPDPS53621.2022.00048. 126

[135] M. Aldridge, Individual testing is optimal for nonadaptive group testing in the linear regime, IEEE Transactions on Information Theory 65 (4) (2019) 2058–2061. doi:10.1109/TIT.2018.2873136. 126

[136] M. A. Khajehnejad, A. G. Dimakis, W. Xu, B. Hassibi, Sparse recovery of nonnegative signals with minimal expansion, IEEE Transactions on Signal Processing 59 (1) (2011) 196–208. doi:10.1109/TSP.2010.2082536. 127

[137] X. Dong, Graph signal processing: Concepts, tools and applications in neuroscience, https://web.media.mit.edu/~xdong/talk/BDI_GSP.pdf. 138

[138] A. Singer, From graph to manifold laplacian: The convergence rate, Applied and Computational Harmonic Analysis 21 (1) (2006) 128–134. 138

[139] X. Zhu, M. Rabbat, Approximating signals supported on graphs, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 3921–3924. doi:10.1109/ICASSP.2012.6288775. 139

[140] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on pattern analysis and machine intelligence 22 (8) (2000) 888–905. 140

[141] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, M. G. Rabbat, Characterization and inference of graph diffusion processes from observations of stationary signals, IEEE transactions on Signal and Information Processing over Networks 4 (3) (2017) 481–496. 140

[142] D. Thanou, X. Dong, D. Kressner, P. Frossard, Learning heat diffusion graphs, IEEE Transactions on Signal and Information Processing over Networks 3 (3) (2017) 484–499. 140

[143] X. Zhu, M. Rabbat, Graph spectral compressed sensing for sensor networks, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2012, pp. 2865–2868. 146

[144] A. Jung, N. Tran, A. Mara, When is network lasso accurate?, Frontiers in Applied Mathematics and Statistics 3 (2018) 28. 146

[145] W. Xu, E. Mallada, A. Tang, Compressive sensing over graphs, in: 2011 Proceedings IEEE INFOCOM, 2011, pp. 2087–2095. doi:10.1109/INFCOM.2011.5935018. 146

[146] X. Qi, Y. Wang, Y. Wang, L. Xu, Compressive sensing over strongly connected digraph and its application in traffic monitoring, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 1222–1230. doi:10.1109/INFOCOM.2014.6848054. 146

[147] X. Zou, L. Feng, H. Sun, Compressive sensing of multichannel eeg signals based on graph fourier transform and cosparsity, Neural Processing Letters 51 (2020) 1227–1236. 146

[148] V. Kalofolias, How to learn a graph from smooth signals, in: Artificial intelligence and statistics, PMLR, 2016, pp. 920–929. 147

[149] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Learning laplacian matrix in smooth graph signal representations, IEEE Transactions on Signal Processing 64 (23) (2016) 6160–6173. 147

[150] S. P. Chepuri, S. Liu, G. Leus, A. O. Hero, Learning sparse graphs under smoothness prior, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 6508–6512. 147

[151] X. Dong, D. Thanou, M. Rabbat, P. Frossard, Learning graphs from data: A signal representation perspective, IEEE Signal Processing Magazine 36 (3) (2019) 44–63. 147

[152] E. Ceci, Y. Shen, G. B. Giannakis, S. Barbarossa, Graph-based learning under perturbations via total least-squares, IEEE Transactions on Signal Processing 68 (2020) 2870–2882. 147

[153] E. Ceci, S. Barbarossa, Graph signal processing in the presence of topology uncertainties, IEEE Transactions on Signal Processing 68 (2020) 1558–1573. doi:10.1109/TSP.2020.2976583. 147, 169, 170

[154] J. Miettinen, S. A. Vorobyov, E. Ollila, Modelling and studying the effect of graph errors in graph signal processing, Signal Processing 189 (2021) 108256. 147

[155] H. Kenlay, D. Thanou, X. Dong, On the stability of polynomial spectral graph filters, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 5350–5354. 147

[156] F. Mahmood, N. Shahid, U. Skoglund, P. Vandergheynst, Adaptive graph-based total variation for tomographic reconstructions, IEEE Signal Processing Letters 25 (5) (2018) 700–704. 148

[157] H. Pandotra, E. Malhotra, A. Rajwade, K. S. Gurumoorthy, Dealing with frequency perturbations in compressive reconstructions with Fourier sensing matrices, Signal Process. 165 (2019) 57–71. 149

[158] J. Ianni, W. Grissom, Trajectory auto-corrected image reconstruction, Magnetic Resonance in Medicine 76 (3) (2016). 149

[159] H. Zhu, G. Leus, G. Giannakis, Sparsity-cognizant total least-squares for perturbed compressive sampling, IEEE Trans. Signal Process. 59 (11) (2011). 149

[160] T. Ince, A. Nacaroglu, On the perturbation of measurement matrix in non-convex compressed sensing, Signal Processing (98) (2014) 143–149. 149

[161] J. Parker, V. Cevher, P. Schniter, Compressive sensing under matrix uncertainties: An approximate message passing approach, in: Asilomar Conference on Signals, Systems and Computers, 2011, pp. 804–808. 149

[162] M. Herman, T. Strohmer, General deviants: an analysis of perturbations in compressed sensing, IEEE Journal on Sel. Topics Signal Process. 4 (2) (2010). 149

[163] S. Fosson, V. Cerone, D. Regruto, Sparse linear regression from perturbed data, Automatica 122 (2020). 149

[164] G. Tang, B. N. Bhaskar, P. Shah, B. Recht, Compressed sensing off the grid, IEEE Trans. Information Theory 59 (11) (2013) 7465–7490. 149

[165] A. Fannjiang, H.-C. Tseng, Compressive radar with off-grid targets: a perturbation approach, Inverse Problems 29 (5) (2013). 149

[166] W. Hu, G. Cheung, A. Ortega, O. C. Au, Multiresolution graph fourier transform for compression of piecewise smooth images, IEEE Transactions on Image Processing 24 (1) (2014) 419–433. 165

[167] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of modern physics 74 (1) (2002) 47. 174

[168] W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of anthropological research 33 (4) (1977) 452–473. 174

[169] M. Girvan, M. E. Newman, Community structure in social and biological networks, Proceedings of the national academy of sciences 99 (12) (2002) 7821–7826. 174

[170] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 898–916. doi:10.1109/TPAMI.2010.161. 178

[171] B. Baskar, Tom and jerry image classification, https://www.kaggle.com/datasets/balabaskar/tom-and-jerry-image-classification (2022). 178

[172] T. F. Chan, L. A. Vese, Active contours without edges, IEEE Transactions on image processing 10 (2) (2001) 266–277. 178

[173] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid (Eds.), Computer Vision – ECCV 2012, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 746–760. 179

[174] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612. doi:10.1109/TIP.2003.819861. 180

[175] P. Indyk, H. Q. Ngo, A. Rudra, Efficiently decodable non-adaptive group testing, in: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2010, pp. 1126–1142. 208

[176] A. Coja-Oghlan, M. Hahn-Klimroth, P. Loick, M. Penschuck, Efficient and Accurate Group Testing via Belief Propagation: An Empirical Study, in: C. Schulz, B. Uçar (Eds.), 20th International Symposium on Experimental Algorithms (SEA 2022), Vol. 233 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 8:1–8:18. doi:10.4230/LIPIcs.SEA.2022.8. 209

[177] N. Tan, J. Scarlett, R. Venkataramanan, Approximate message passing with rigorous guarantees for pooled data and quantitative group testing (2023). arXiv:2309.15507. 209

[178] E. Romanov, O. Ordentlich, On compressed sensing of binary signals for the unsourced random access channel, Entropy 23 (5) (2021) 605. 209

List of Publications

Pooled RT-PCR Testing for COVID-19

The work done in this part of the thesis has been **published** in the following paper, which was one of top 25 most downloaded papers of the IEEE Open Journal of Signal Processing in 2021:

- **S. Ghosh**, R. Agarwal, M. A. Rehan, S. Pathak, P. Agarwal, Y. Gupta, S. Consul, N. Gupta, Ritika, R. Goenka, A. Rajwade, and M. Gopalkrishnan. *A Compressed Sensing Approach to Pooled RT-PCR Testing for COVID-19 Detection*. In: IEEE Open Journal of Signal Processing 2 (2021), pp. 248–264.

The following are **preprints** written for a medical science and biology audience (reverse chronological order):

- **S. Ghosh**, A. Rajwade, S. Krishna, N. Gopalkrishnan, T. E. Schaus, A. Chakravarthy, S. Varahan, V. Appu, R. Ramakrishnan, S. Ch, M. Jindal, V. Bhupathi, A. Gupta, A. Jain, R. Agarwal, S. Pathak, M. A. Rehan, S. Consul, Y. Gupta, N. Gupta, P. Agarwal, R. Goyal, V. Sagar, U. Ramakrishnan, S. Krishna, P. Yin, D. Palakodeti, and M. Gopalkrishnan. *Tapestry: A Single-Round Smart Pooling Technique for COVID-19 Testing*. medRxiv preprint. (2020). `https://www.medrxiv.org/content/early/2020/05/02/2020.04.23.20077727`

- A. Chakravarthy, S. Krishna, **S. Ghosh**, A. Tomar, S. Varahan, A. Rajwade, S. Ghosh, N. Gupta, R. Agarwal, H. Payal, P. Chakraborty, K. V. Vemula, A. Vyas, R. Goru, S. Krishna, D. Palakodeti, and M. Gopalkrishnan. *Large-scale Testing for SARS-CoV-2 using Tapestry Pooling*. medRxiv preprint. (2020). `https://www.medrxiv.org/content/early/2020/10/13/2020.10.09.20209742`

The above papers and preprints have a combined citation count of more than 130 as of December 2023.

A **patent** has been filed for the technology arising from this part of the thesis:

- M. Gopalkrishnan, **S. Ghosh**, A. V. Rajwade, D. Palakodeti, and S. Krishna. *Methods and systems for determining viruses in biological samples using a single round based pooling*. US Patent App. 17/535,998. 2022. (pending)

The author of this thesis has given the following **invited talks** on the Tapestry method:

- *Tapestry: A Compressed Sensing Approach to Pooled RT-PCR Testing for COVID-19 Detection* – Signal Processing Society Webinar, 2022.
- *Tapestry: A Compressed Sensing Approach to Pooled RT-PCR Testing for COVID-19 Detection* – International Workshop on Bio-Design Automation, 2021.

This part of the thesis resulted in the following **software** has been released, to which the author of this thesis has either been the primary contributor or has overseen and made key contributions:

- Tapestry method: `https://github.com/atoms-to-intelligence/tapestry`
- BYOM App for manual pooling: `https://rebrand.ly/byom-app`

The work in this part of the thesis directly led to the founding of a molecular diagnostics **startup** by Prof. Manoj Gopalkrishnan, to which the author of this thesis made some code contributions as well as performed the first deployment:

- Algorithmic Biologics. `https://algorithmicbiologics.com`

Efficient Automated Image Moderation

The following manuscript is **to be submitted** to a well-reputed journal in the field of deep learning:

- **S. Ghosh**, S. Saxena, and A. Rajwade. *Efficient Neural Network based Classification and Outlier Detection for Image Moderation using Compressed Sensing and Group Testing*. arXiv preprint arXiv:2305.07639 (2023). `https://arxiv.org/abs/2305.07639`

Compressive Perturbed Graph Recovery

The following manuscript is under preparation and is **to be submitted** to a well-reputed journal in the field of signal processing:

- S. Ghosh and A. Rajwade. "Compressive Perturbed Graph Recovery". manuscript in preparation. 2023

Not Part of Thesis

The following publications by the author are not part of this thesis:

- A. Awasthi, **S. Ghosh**, R. Goyal, and S. Sarawagi. *Learning from rules generalizing labeled exemplars*. In: International Conference on Learning Representations. 2020.
- A. Awasthi, S. Sarawagi, R. Goyal, **S. Ghosh**, and V. Piratla. *Parallel iterative edit models for local sequence transduction*. In: Proceedings of the 2019 Conference on Empirical

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019, pp. 4260–4270.

- **S. Ghosh**, M. Redekopp, and M. Annavaram. *Knightshift: shifting the i/o burden in datacenters to management processor for energy efficiency*. In: Proceedings of the 2010 International Symposium on Computer Architecture (ISCA). Workshop on Energy Efficient Design (WEED). 2010, pp. 183–197.

- G. Thatte, V. Rozgic, M. Li, **S. Ghosh**, U. Mitra, S. Narayanan, M. Annavaram, and D. Spruijt-Metz. *Optimal time-resource allocation for activity-detection via multimodal sensing*. In: 4th International ICST Conference on Body Area Networks. 2010.

- G. Thatte, V. Rozgic, M. Li, **S. Ghosh**, U. Mitra, S. Narayanan, M. Annavaram, and D. Spruijt-Metz. *Optimal allocation of time-resources for multihypothesis activity-level detection*. In: International Conference on Distributed Computing in Sensor Systems. Springer Berlin Heidelberg Berlin, Heidelberg. 2009, pp. 273–286.