Recovery of Noisy Pooled Tests via Learned Factor Graphs with Application to COVID-19 Testing

Eyal Fishel Ben-Knaan, Yonina C. Eldar, and Nir Shlezinger

Abstract—**The ongoing pandemic and the necessity of frequent testing have spurred a growing interest in pooled testing. Conventional recovery methods from pooled tests are based on group testing or compressed sensing tools which rely on simplistic modeling of the pooling process, and may not be reliable in the presence of complex and noisy measurement procedures and highly infected populations. In this work, we propose a strategy for pooled testing designed for noisy settings, which bypasses the need for a tractable acquisition model. This is achieved by combining deep learning, for implicitly learning the measurement relationship from data, with factor graph inference, which exploits the structured known pooling pattern. Learned factor graphs provide a quantitative readout corresponding to the infection severity, as opposed to group testing which only detects the presence of infection. The proposed scheme is shown to achieve improved robustness to noise compared with previous approaches and to reliably estimate in highly infected populations.**

Index terms— **Factor Graphs, deep learning, pooling.**

I. Introduction

As of March 2020, the Coronavirus disease 2019 (COVID-19) has been declared a health pandemic claiming millions of lives worldwide. A key strategy for combating COVID-19 is to mitigate the virus spread. Mitigation requires frequent testing [1], for which the prevalent approach to date is based on examining swab measurements via reverse transcription-polymerase chain reaction (RT-qPCR) [2]. RT-qPCR, however, gives rise to several challenges when used for frequent testing. Mainly, RT-qPCR requires long testing duration, approximately 3 to 4 hours;limited capacity, being restricted to screen a fixed number of samples; and requires skilled technicians [3]. These challenges motivate exploring pooling methods for increasing the throughput of such tests.

Pooling allows to improve the throughput of infection tests, such as those based on RT-qPCR, by inspecting a mixture of samples, termed pools, rather than each individual sample. Recovering the viral load of individual samples from a group sample is an inverse problem that shares similarities with two mathematical frameworks, group testing (GT) and compressed sensing (CS). GT focuses on identifying whether a sample is infected or not. The rationale is that a negative pooled outcome implies that all samples corresponding to that specific pool are negative, thereby saving tests and resources when the samples are sparse. If a pooled outcome returns positive, GT traditionally operates in an adaptive manner [4], requiring additional testing for complete detection, though GT schemes can also be applied in a one-shot fashion [5]. GT tools were proposed to detect the presence of infection in pooled COVID-19 tests [6]–[8], and recently also to identify the level of infection [9]. CS focuses on reconstruction of high-dimensional real-valued sparse signals from a set of low-dimensional projections [10]. For pooled testing, CS allows to recover the viral loads of each sample, and such methods were used for COVID-19 tests in [11]–[16].

This work is supported in part by QuantiERA grant C'MON-QSENS!. E. F. Ben-Knaan and N. Shlezinger are with the School of ECE, Ben-Gurion University of the Negev, Be'er-Sheva, Israel (e-mail: eyalfish@post.bgu.ac.il; nirshl@bgu.ac.il). Y. C. Eldar is with the Faculty of Math and CS, Weizmann Institute of Science, Rehovot, Israel (e-mail: yonina@weizmann.ac.il).

While both CS and GT are established theoretical frameworks [4], they share several properties which limit their applicability in some scenarios. For one, they typically assume a linear measurement model, where each observation is given by a linear combination of some of the samples, with possibly limited noise. Furthermore, both CS and GT are reliable when the samples are highly sparse, i.e., mostly not infected. GT and CS based recovery methods are likely to fail when these requirements do not hold, e.g., when the measurement procedure or the pooling operation induce some non-linear and possibly intractable manipulation of the samples corrupted by non-negligible noise, which is often the case in RT-qPCR based pooling, or when a substantial portion of the samples are infected (e.g., [11] requires that at most 1.3% of the samples be infected.) This motivates the derivation of a robust recovery method which is still capable of identifying infection levels in the presence of non-linear noisy and not-too-sparse pooled tests.

In this work we propose a recovery method for pooled testing designed to cope with noisy non-linear measurements and with relatively large percentages of infected samples. We account for the structural relationship induced by the pooling procedure using factor graphs, which are representations of structured joint distributions that facilitate accurate inference at controllable complexity via message passing algorithms [17], [18]. Unlike CS and GT solutions, which rely on a linear measurement model, the usage of factor graphs allows us to handle arbitrary measurement forms, and use the known pooling pattern, i.e., knowledge of which samples are mixed together, to determine the interconnections over the graph. Once the factor graph is established, one can recover the infection level of each sample using loopy belief propagation (LBP) [19].

In order to utilize the factor graph representation for inference, one must provide a statistical model relating the desired information and the observations. To handle complex and possibly unknown models, such as those induced by noisy RT-qPCR measurements, we utilize learned factor graphs, previously proposed in the context of communications systems and sleep pattern detection [20]. By incorporating deep neural networks (DNNs) to capture the measurement effect as a form of model-based deep learning [21], one can bypass the need to explicitly know the underlying statistics, and learn how to carry out such inference from a limited amount of samples. Our numerical results demonstrate that by applying LBP over learned factor graphs, one can achieve notably improved recovery accuracy in noisy settings compared with the schemes considered in [12].

The rest of this paper is organized as follows: In Section II we review the noisy pooled testing system model. Section III presents the proposed learned factor graphs inference method. Section IV details the numerical study, and Section V concludes the paper.

II. System Model

Here, we present the system model for which we derive our recovery algorithm. We begin by describing the measurement process,

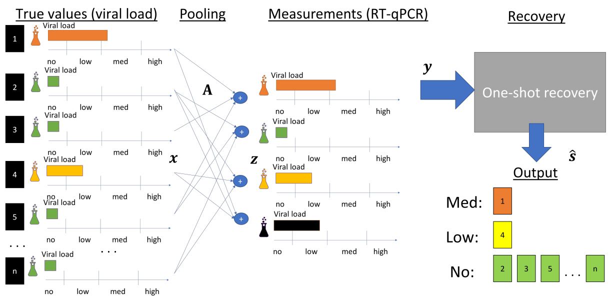

Fig. 1. Pooled testing with one-shot recovery illustration.

relating to the application of COVID-19 tests in Subsection II-A, and formulate the recovery problem in Subsection II-B.

A. Pooled Tests

Pooling considers a small mixture of pooled samples rather than the total number of samples. To model this procedure in a generic manner, let $\boldsymbol{x} \in \mathcal{R}_+^n$ represent the samples vector, i.e., the viral loads of each of the subjects, whose entries $x_1, \ldots, x_n$ denote the viral loads. If $x_i = 0$, then the i^{th} individual is not infected. Pooled testing is comprised of two stages: pooling and measurement.

Pooling refers to the mapping of the n samples $\boldsymbol{x}$ into a lower-dimensional vector $\boldsymbol{z} \in \mathcal{R}_+^m$, with $m < n$. Here, each entry of $\boldsymbol{z}$ is obtained by mixing a subset of the samples in $\boldsymbol{x}$ determined by the binary valued pooling matrix $\mathsf{A} \in \{0,1\}^{m \times n}$, such that

$$z_i = f_{\rm pool}\left(\{x_j | \mathsf{A}_{i,j} = 1\}\right), \quad i \in \{1, \ldots, m\}. \tag{1}$$

In (1), $f_{\rm pool}(\cdot)$ represents how the viral loads of the mixed samples are mapped into the viral load of the pool. While pooling is commonly modelled as averaging, i.e., $f_{\rm pool}(\mathcal{X}) = \frac{1}{|\mathcal{X}|}\sum_{x \in \mathcal{X}} x$, in many applications, including COVID-19 testing, the relationship between the pool and the mixed samples is more complex and includes stochasticity and dilution [6].

Measurement refers to reading of the pooled vector $\boldsymbol{z}$ into the observed vector $\boldsymbol{y} \in \mathcal{R}_+^m$. This is carried out via the element-wise mapping $f_{\rm meas}(\cdot)$, such that $y_i = f_{\rm meas}(z_i)$ for each $i \in \{1, \ldots, m\}$. For RT-qPCR, the mapping $f_{\rm meas}(\cdot)$ represents a procedure which involves the conversion of the sample into complementary DNA through reverse transcription, followed by examining its reaction in response to the heating and cooling cycle through a chemical process to quantify the number of virus particles. The measurement procedure typically involves some level of uncertainty and distortion, and thus $f_{\rm meas}(\cdot)$ is typically stochastic. For instance, [12] approximated RT-qPCR measurements as

$$y_i = (1+q)^{\epsilon_i} z_i, \tag{2}$$

where $q \in [0,1]$ is an amplification parameter associated with the increase in the viral particles in each cycle, and ϵ_i represents measurement fluctuations and is modeled as Gaussian noise.

B. Problem Formulation

Our goal is to design an algorithm for recovering the infected samples from the measurements $\boldsymbol{y}$, as illustrated in Fig. 1 for pooled RT-qPCR testing, while meeting the following requirements:

R1: The number of infected samples, denoted by k, is smaller than the number of tested individuals n. In contrast to previous GT and CS based studies, e.g., [11], we allow the infected samples to constitute a notable portion of the overall samples, e.g., up to 20% of the tested population is infected.

R2: The desired output is not the viral load, but a discrete score on the level of infection. Possible outputs may be *no virus*, *low* (borderline), *mid* (infected), and *high* (highly infected).

R3: The pooling and measurement procedures $f_{\rm pool}(\cdot)$ and $f_{\rm meas}(\cdot)$ may be noisy, unknown, and intractable. However, one can acquire data corresponding to these procedures, i.e., past measurements with ground truth from the pooling setup.

R4: Recovery should operate in one-shot, identifying all subjects from $\boldsymbol{y}$, as opposed to adaptive strategies which carry out additional tests based on the results.

R5: One cannot mix more than $L < n$ samples together in each pool due to, e.g., dilution [6].

Per *R2* our objective is to recover a discrete representation of the viral load directly in a one-shot fashion. To achieve this, we define a quantization mapping, $Q : \mathcal{R}_+ \mapsto \mathcal{Q}$, where $\mathcal{Q}$ is a finite set describing all possible infection status. The goal is to design an algorithm that estimates the discretized viral loads, $\hat{\boldsymbol{s}}$, from the pooled measurements $\boldsymbol{y}$, to minimize the error probability

$$e(\hat{s}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \Pr\left(Q(x_i) \neq \hat{s}_i\right). \tag{3}$$

The fact that $\hat{\boldsymbol{s}}$ is obtained directly from $\boldsymbol{y}$ indicates that the algorithm operates in a one-shot fashion, as required in *R4*. To exploit the pooling structure induced by the known A while complying with *R3*, we propose using learned factor graphs, detailed in the sequel.

III. Learned Factor Graphs

Factor graph methods are a family of algorithms that incorporate an encoded factorization structure of a multivariate function, such as a joint probability distribution, to compute variables of interest at reduced complexity via message passing [18]. While message passing methods are commonly used in GT [22] and CS [23], their generic formulation does not share the conventional assumption of highly sparse linear measurements as in GT and CS, which we avoid adopting here due to *R3*. Our proposed method combines knowledge of the pooling pattern A with DNNs to capture the underlying factorizable distribution as a factor graph, which in turn is used for recovery via the LBP algorithm [19]. To review the proposed method, we first discuss LBP inference in Subsection III-A, after which we present its usage with learned factor graphs in Subsection III-B and provide a discussion in Subsection III-C.

A. Loopy Belief Propagation

As detailed in Subsection II-B, recovery accuracy is measured in terms of the error rate (3), which is minimized by the maximum a-posteriori (MAP) rule. While computing MAP estimates is a computationally prohibitive task whose burden grows exponentially with n, when the underlying statistical relationship is factorizable, the MAP rule is approximated at linear complexity via LBP [19].

Consider our pooled testing setup, where the task is to estimate the discrete-valued vector $\boldsymbol{s} \in \mathcal{Q}^n$ with entries $s_i = Q(x_i)$ from the observed $\boldsymbol{y} \in \mathcal{R}_+^m$. Given a pooling pattern matrix A, the joint probability distribution of $\boldsymbol{y}, \boldsymbol{s}$ can be expressed as a partition

$$P(\boldsymbol{y}, \boldsymbol{s}) = P(\boldsymbol{y}|\boldsymbol{s})P(\boldsymbol{s}) = P(\boldsymbol{s})\prod_{i=1}^{m} P(y_i | \mathcal{S}_i), \tag{4}$$

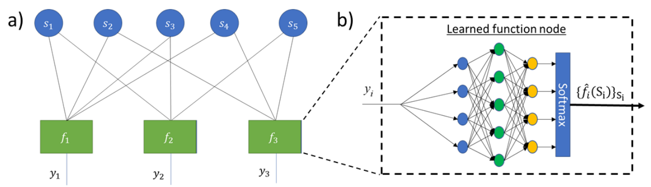

Fig. 2. Factor graph (a) with learned function node (b).

where $\mathcal{S}_i = \{s_j | \mathsf{A}_{i,j} = 1\}$. Note that the factorization in (4) is dictated by the pooling pattern A, while the combined stochastic effect of the pooling and measurement procedures is encapsulated in the conditional distribution $P(y_i|\mathcal{S}_i)$. By defining $f_i(\mathcal{S}_i) := P(y_i|\mathcal{S}_i)$, (4) can be represented as a factor graph as illustrated in Fig. 2(a), with m *function nodes* $\{f_i\}$ and n *variable nodes* $\{s_i\}$.

LBP is an iterative algorithm that approximates the marginal a-posteriori probabilities by propagating messages in the factor graph domain according to a flooding schedule [24]. Messages are initialized $\{\mu^{(0)}_{f_i \to s_j}(s), \mu^{(0)}_{s_j \to f_i}(s)\}$ for each $s \in \mathcal{Q}$ and for every function node f_i and variable node s_j which share an edge in the factor graph, i.e., $\mathsf{A}_{i,j} = 1$. Then, for each vertex, the messages are shared downstream with their neighbors in an iterative fashion. For iteration index t, the variable-to-function messages are computed via

$$\mu^{(t+1)}_{s_j \to f_i}(s) = \prod_{f \in \mathcal{N}(s_j)/f_i} \mu^{(t)}_{f \to s_j}(s), \tag{5}$$

and the function-to-variable messages are obtained as

$$\mu^{(t+1)}_{f_i \to s_j}(s) = \sum_{\mathcal{S}_i/s_j} f_i(\mathcal{S}_i) \prod_{z \in \mathcal{N}(f_i)/s_j} \mu^{(t)}_{z \to f_i}(z), \tag{6}$$

where $\mathcal{N}(v)$ denotes the set of neighbors of a given node v. Note that the neighbours of variable nodes are all function nodes, while the neighbors of function nodes are all variable nodes.

Assuming convergence after T iterations, the a-posteriori probability can be approximated as the product of all incoming messages

$$\hat{P}(s_j = s|\boldsymbol{y}) \propto \prod_{f \in \mathcal{N}(s_j)} \mu^{(T)}_{f \to s_j}(s). \tag{7}$$

When the graph is cycle-free, (7) is the true posterior distribution, whose maximization is the MAP rule. When cycles are present, as typically occurs in pooled testing, (7) is an approximation of the posterior, approaching the true value when the graph satisfies some requirements on its number of cycles and girth; see [19] for details.

B. LBP Pooled Recovery over Learned Factor Graphs

LBP carries out pooled recovery based on knowledge of the underlying distribution in (4). This knowledge is comprised of two components: the structure of the factor graph, which is dictated by the known pooling matrix A; and the function nodes, whose expression is unknown by *R3*. Our approach overcomes this challenge by using learned factor graphs, extending the approach proposed in [20] for Markovian distributions to enable LBP inference over loopy factor graphs encountered in pooled recovery.

Our recovery method builds upon the fact that given A, one needs only to evaluate the function nodes $\{f_i\}$ in order to carry out LBP inference. In pooled testing, the function nodes $f_i(\mathcal{S}_i) = P(y_i|\mathcal{S}_i)$ are dictated by $f_{\rm pool}$ and $f_{\rm meas}$, which do not depend on the pool index i. Consequently, the same function node computation can be reused for each pool index i, and it can be learned from data using DNNs and applied to the variables y_i and $\mathcal{S}_i$. Following [20], we learn to evaluate $f_i(\mathcal{S}_i)$ using a DNN classifier comprised of three dense layers and a softmax output layer, trained to predict $\mathcal{S}_i = \{s_j|\mathsf{A}_{i,j} = 1\} \in \mathcal{Q}^{|\mathcal{S}_i|}$ from y_i using the cross entropy loss. The network output is a $|\mathcal{S}_i| \times 1$ vector, where each entry represents the estimation of the function node at the corresponding input combination. An illustration of such a learned function node is depicted in Fig. 2(b). Since the cardinality of $\mathcal{S}_i$ may vary in the range $1, \ldots, L$ by *R5*, we train different DNNs for different possible cardinalities. These learned function nodes are trained offline using the available data corresponding to the pooling and measurement procedure. Once they are obtained, the resulting learned factor graph is used for LBP inference on each incoming observations $\boldsymbol{y}$.

To reduce the computational complexity of LBP over the learned factor graph and explicitly account for the sparsity of the samples *R1*, we include a preliminary processing for identifying definitely-non-defective subjects via column-matching pursuit (COMP) [25], which is commonly used in GT. This preliminary processing phase declares all samples pooled into a measurement as non-infected if the measurement is below some pre-defined threshold τ, i.e.,

$$y_i \le \tau \Rightarrow \hat{s}_j = Q(0), \qquad \forall s_j \in \mathcal{S}_i. \tag{8}$$

To summarize, the procedure consists of the two stages: It first removes definitely-non-defective subjects using COMP [25]. Then, the pooling mapping of the remaining observations is used to form the factor graph representing these relationships, over which inference is carried out for each subject individually via LBP over the learned graph. The outputs are the discrete values assigned to each sample. The procedure is summarized as Algorithm 1.

Algorithm 1: LBP over Learned Factor Graph

```
Init: Trained DNNs, pooling matrix A, measurements $\boldsymbol{y}$
Pre-processing:
1 Apply COMP to $\boldsymbol{y}$ to identify sure negative samples via (8);
2 Stack remaining measurements as vector $\tilde{\boldsymbol{y}}$ and form
    reduced $\tilde{\mathsf{A}}$ corresponding to possibly defective samples;
3 Form factor graphs where function node $f_i$ and variable
    node $s_j$ share an edge if $\tilde{\mathsf{A}}_{i,j} = 1$;
Loopy belief propagation:
4 Initialize $\{\mu^{(0)}_{f_i \to s_j}(s), \mu^{(0)}_{s_j \to f_i}(s)\}$ uniformly;
5 for $t = 0, \cdots T-1$ do
6     Update variable-to-function messages via (5).;
7     Update function-to-variable messages via (6) using
        trained DNN to compute $f(\mathcal{S}_i)$;
8 end
9 Recover infection levels of possibly defective samples by
```

$$\hat{s}_j = \mathop{\arg\max}_{s \in \mathcal{Q}} \prod_{f \in \mathcal{N}(s_j)} \mu^{(T)}_{f \to s_j}(s). \tag{9}$$

C. Discussion

Algorithm 1 incorporates the structured relationship included by the known pooling pattern A while avoiding the need to impose a linear pooling and measurement model or even require explicit expression of its combined mapping. This is achieved by capturing the relationships induced by the pooling pattern in a principled manner by forming a factor graph, which is augmented with DNNs to fill in the missing domain knowledge *R3*. Inference over the learned factor inherently operates in a one-shot manner, satisfying *R4*, and is not restricted to binary outputs as in conventional GT. To exploit the known sparsity *R1*, Algorithm 1 utilizes COMP-based pre-processing, which is known to contribute to recovery accuracy of pooled recovery, particularly for COVID-19 testing [12].

An additional benefit of using COMP is the reduction in computational complexity. Specifically, LBP involves T iterations, each comprised of computing $|\mathcal{Q}|$ messages via (5) and $|\mathcal{Q}|$ messages via (6) for each edge in the graph, i.e., $2T|\mathcal{Q}|\|\tilde{\mathsf{A}}\|_1$ messages. The usage of COMP results in $\|\tilde{\mathsf{A}}\|_1$ being much smaller than $\|\mathsf{A}\|_1$, as we also numerically observe in Section IV, thus reducing the overall complexity. Note that while the function nodes are computed at each iteration, the DNN-based learned function node should only be applied once to each measurement in the reduced $\tilde{\boldsymbol{y}}$, and the results are reused over different iterations. Furthermore, unlike [9] which also used COMP as a preliminary step for pooled recovery with multiple infection levels, Algorithm 1 does not require the reduced setup obtained to represent an non-undetermined linear problem.

In order to train the learned function nodes, one must provide labeled data, including the ground truth score of each sample. For COVID-19 testing, this implies that data should be obtained by individually examining each subject before pooling and insertion into RT-qPCR measurements. Further, it is noted that the sparsity prior *R1*, accounted for in the usage of COMP, and is not explicitly incorporated into the computation of the function nodes, but rather learned from the data provided. While $P(\boldsymbol{s})$ can be estimated via histogram and incorporated into the function nodes, we numerically observed that it leads to instability when the data is highly sparse.

LBP inference only approximates the MAP rule, even given full accurate characterization of the underlying model. This indicates that one can possibly achieve improved performance, as well as reduce inference speed, by replacing the explicit LBP iterations in Algorithm 1 with a learned computation, possibly via unfolded weighted LBP [26] or using graph neural networks [27]. An additional aspect which is not treated in the current work is the design of the pooling matrix, assumed to be given here, while previous works have demonstrated that a proper design of A can notably facilitate recovery. We leave these extensions for future work.

IV. Numerical Results

Setup: We simulate a noisy setup where $n = 45$ samples are pooled in $m = 30$ measurements. The pooling and measurement procedures are based on (2) with parameters $q = 0.95$ and $\epsilon = 0.001$ as in [12] as well as additive Gaussian noise. The non-sparse entries of $\boldsymbol{x}$ are randomized uniformly from $[0, 40)$ while the additive noise is set to yield signal-to-noise ratio (SNR) in the range of $[-3, 5]$ dB. The matrix A is selected such that each sample is involved in two tests while the number of samples contributing to a single pool is $L = 3$. The conversion of viral loads to infection levels is obtained by $Q(x) = \lfloor 0.1x \rfloor$, such that $\mathcal{Q} = \{0, 1, 2, 3\}$. The error rates are averaged over 5000 Monte Carlo trials.

Learned factor graphs: We utilize three DNNs (since $L = 3$) for the trainable function nodes. Each DNN is comprised of three fully-connected layers with intermediate sigmod and ReLU activiations and a softmax output layer of sizes $\{4, 16, 64\}$. These networks are trained using 10^4 generated labeled pairs $\{y_i, \mathcal{S}_i\}$,constituting all

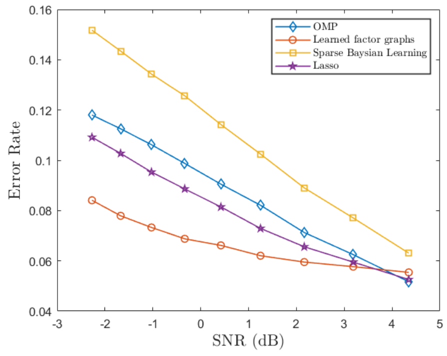

Fig. 3. Error rate as a function of SNR for sparsity level $k = 4$.

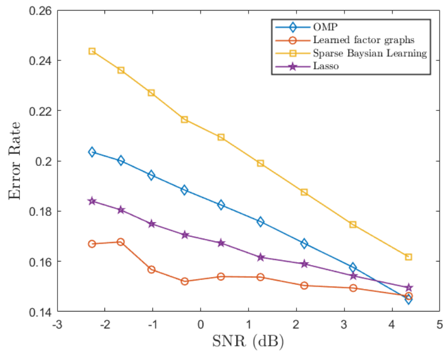

Fig. 4. Error rate as a function of SNR for sparsity level $k = 10$.

possible aggregates, $\boldsymbol{z} = \mathsf{A}x$, with the Adam optimizer [28] with learning rate 0.005, batch size 60, and 100 epochs. For COMP pre-processing, we use the threshold $\tau = 0.5$.

Results: We compare the accuracy of Algorithm 1 to three strictly model-based algorithms benchmarks considered in [12]: Least absolute shrinkage and selection operation (Lasso), with regularization coefficient optimized to minimize (3); Orthogonal Matching Pursuit (OMP), and Sparse Bayesian Learning (SBL). All recovery algorithms use the same preliminary COMP step, as also proposed in [12]. The viral loads predicted by these benchmarks are converted to infection levels $Q(\cdot)$ when computing the error rate. We specifically focus on harsh not-too-sparse setups, setting $k = 4$ ($\approx 9\%$ sparsity) and $k = 10$ ($\approx 23\%$ sparsity), and the error rates are depicted in Figs. 3-4, respectively. We observe in Figs 3-4 that since the noisy measurement procedure is implicitly incorporated into the factor graph, Algorithm 1 is notably more robust in predicting infection levels in such noisy settings, and its error rate grows less dramatically as the SNR decreases compared to the competing approaches.

V. Conclusions

In this work we presented a pooled recovery algorithm for noisy complex settings. Our proposed algorithm combines model-based factor graph inference for capturing the structured relationship induced by the pooling pattern with deep learning techniques for handling complex measurement procedures. Our simulations shows that the usage of learned factor graphs facilitates robust recovery of the infection status under contaminated environments, i.e., high noise level or lab pipetting errors, in highly infected populations.

References

[1] M. Salathé, C. L. Althaus, R. Neher, S. Stringhini, E. Hodcroft, J. Fellay, M. Zwahlen, G. Senti, M. Battegay, A. Wilder-Smith *et al.*, "COVID-19 epidemic in Switzerland: on the importance of testing, contact tracing and isolation." *Swiss medical weekly*, vol. 150, no. 11-12, p. w20225, 2020.

[2] T. Nolan, R. E. Hands, and S. A. Bustin, "Quantification of mRNA using real-time RT-PCR," *Nature protocols*, vol. 1, no. 3, pp. 1559–1582, 2006.

[3] E. J. Emanuel, G. Persad, R. Upshur, B. Thome, M. Parker, A. Glickman, C. Zhang, C. Boyle, M. Smith, and J. P. Phillips, "Fair allocation of scarce medical resources in the time of Covid-19," 2020.

[4] A. C. Gilbert, M. A. Iwen, and M. J. Strauss, "Group testing and sparse signal recovery," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 1059–1063.

[5] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri, "Non-adaptive group testing: Explicit bounds and novel algorithms," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 3019–3035, 2014.

[6] I. Yelin, N. Aharony, E. Shaer-Tamar, A. Argoetti, E. Messer, D. Berenbaum, E. Shafran, A. Kuzli, N. Gandali, T. Hashimshony *et al.*, "Evaluation of COVID-19 RT-qPCR test in multi-sample pools," *MedRxiv*, 2020.

[7] R. Hanel and S. Thurner, "Boosting test-efficiency by pooled testing strategies for SARS-CoV-2," *arXiv preprint arXiv:2003.09944*, 2020.

[8] R. Ben-Ami, A. Klochendler, M. Seidel, T. Sido, O. Gurel-Gurevich, M. Yassour, E. Meshorer, G. Benedek, I. Fogel, E. Oiknine-Djian *et al.*, "Large-scale implementation of pooled RNA extraction and RT-PCR for SARS-CoV-2 detection," *Clinical Microbiology and Infection*, 2020.

[9] A. Cohen, N. Shlezinger, A. Solomon, Y. C. Eldar, and M. Médard, "Multi-level group testing with application to one-shot pooled COVID-19 tests," in *Proc. IEEE ICASSP*, 2021, pp. 1030–1034.

[10] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*. Cambridge university press, 2012.

[11] N. Shental, S. Levy, S. Skorniakov, V. Wuvshet, Y. Shemer-Avni, A. Porgador, and T. Hertz, "Efficient high throughput SARS-CoV-2 testing to detect asymptomatic carriers," *medRxiv*, 2020.

[12] S. Ghosh, R. Agarwal, M. A. Rehan, S. Pathak, P. Agrawal, Y. Gupta, S. Consul, N. Gupta, R. Goyal, A. Rajwade, and M. Gopalkrishnan, "A compressed sensing approach to pooled RT-PCR testing for COVID-19 detection," *IEEE Open Journal of Signal Processing*, vol. 2, pp. 248–264, 2021.

[13] J. Yi, R. Mudumbai, and 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.

[14] H. B. Petersen, B. Bah, and P. Jung, "Practical high-throughput, non-adaptive and noise-robust SARS-CoV-2 testing," *arXiv preprint arXiv:2007.09171*, 2020.

[15] J. Yi, M. Cho, X. Wu, W. Xu, and R. Mudumbai, "Error correction codes for COVID-19 virus and antibody testing: Using pooled testing to increase test reliability," *arXiv preprint arXiv:2007.14919*, 2020.

[16] J. Zhu, K. Rivera, and D. Baron, "Noisy pooled PCR for virus testing," *arXiv preprint arXiv:2004.02689*, 2020.

[17] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, 2004.

[18] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, 2007.

[19] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 736–744, 2001.

[20] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Inference from stationary time sequences via learned factor graphs," *IEEE Trans. Signal Process.*, 2022.

[21] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *arXiv preprint arXiv:2012.08405*, 2020.

[22] M. Hahn-Klimroth, P. Loick, and M. Penschuck, "Efficient and accurate group testing via belief propagation: an empirical study," *arXiv preprint arXiv:2105.07882*, 2021.

[23] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[24] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 219–230, 1998.

[25] M. Aldridge, L. Baldassini, and O. Johnson, "Group testing algorithms: Bounds and simulations," *IEEE Trans. Inf. Theory*, vol. 60, no. 6, pp. 3671–3687, 2014.

[26] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, 2018.

[27] Z. Zhang, F. Wu, and W. S. Lee, "Factor graph neural network," in *Advances in Neural Information Processing Systems*, 2020, pp. 8577–8587.

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