Active Sensing for Epidemic State Estimation Using ABM-guided Machine Learning

Sami Saliba[1], Faraz Dadgostari[2], Stefan Hoops[1], Henning S. Mortveit[1], and Samarth Swarup[1]

[1] University of Virginia, Charlottesville VA 22904, USA
{sms8fr,shoops,Henning.Mortveit,swarup}@virginia.edu
[2] Montana State University, Bozeman, MT, USA
faraz.dadgostari@montana.edu

Abstract. During an epidemic, it can be difficult to get an estimate of the actual number of people infected at any given time. This is due to multiple reasons, including some cases being asymptomatic and sick people not seeking healthcare for mild symptoms, among others. Large scale random sampling of the population for testing can be expensive, especially in the early stages of an epidemic, when tests are scarce. Here we show how an adaptive prevalence testing method can be developed to obtain a good estimate of the disease burden by learning to intelligently allocate a small number of tests for random testing of the population. Our approach uses a combination of an agent-based simulation and deep learning in an active sensing paradigm. We show that it is possible to get a good state estimate with relatively minimal prevalence testing, and that the trained system adapts quickly and performs well even if the disease parameters change.

Keywords: Computational Epidemiology · Neural Networks · Agent-based Modeling and Simulation · Active Sensing.

1 Introduction

Our goal is to get a good estimate of the burden of disease, i.e., the actual number of infections, as an epidemic progresses through a population. In this paper, we will refer to this as the *epidemic state estimation problem.*

This is a hard problem because a large fraction of cases can be asymptomatic, as in the case of COVID-19, where up to $\sim$40% of cases are believed to be asymptomatic [10]. In certain scenarios, if enough tests are available, prevalence testing is done to track the progression of the disease through the population, where a random sample of the population is tested at regular intervals, regardless of whether they are symptomatic [7]. This is easier to do in controlled settings, such as universities, army bases, etc., where the population can be required to go to a testing location [14]. However, prevalence testing of the general population is much harder, so other approaches are used, such as targeted testing [3] and contact tracing [1, 8]. Alternatively, attempts are made to estimate the burden of the disease from available testing data in combination with hospital visit data, mortality data, and more [6].

In this work, we show that an adaptive approach to prevalence testing can be effective. Informally, our problem is to estimate the number of infections in each of N regions over time, by intelligently choosing the number of random tests done in each region on each day. The main idea behind our approach is to use a very detailed, data-driven agent-based model to train a recurrent neural network to solve this problem. Traditionally, state estimation of a dynamical system is done using filtering. Particle filters are a popular choice when the system is non-Gaussian. Particle Filter Recurrent Neural Networks (PF-RNNs) have recently been developed to combine filtering with deep learning [11], which allows the underlying dynamical system model to be learned from data. This line of work assumes a fixed observation process, which provides information about the state of the system at each time step, but is not under the control of the modeler/state estimator.

In our case, in contrast, the state estimator has to choose the observations that are made, by choosing how many tests to assign to each region at each time step. This is known as active sensing [16]. Our main contribution, thus, is an innovative application of an (existing) agent-based model: to train an active sensing system for doing epidemic state estimation. While machine learning, filtering methods, and agent-based models have been combined in various ways recently, such as learning emulators [2], doing model comparison [15], and also doing state estimation [9, 12] we believe this is the first application for active sensing.

The rest of this paper is organized as follows. We give a precise problem formulation in Section 2. After that we present the relevant background on filtering, PF-RNNs, active learning, and active sensing. Then, in Section 4, we present our approach, which uses an agent-based model to train a PF-RNN in an active sensing setting. This is followed by a detailed description of the data and agent-based model used here (Section 4.2). Then we present a series of experiments with simulated epidemics in twelve counties of the US state of Virginia, where we evaluate the performance of our approach. We end with a discussion of related work and possible extensions.

2 Problem Formulation

Suppose that there is an infectious disease spreading through the population of a region. For simplicity, at present we assume that the population is naïve, i.e., that no one has prior immunity to the disease. We also assume that there isn't a vaccine available yet. This describes the early stages of the COVID-19 epidemic reasonably well (first wave). Our approach is general enough that these assumptions aren't necessary, as we discuss in the Conclusion (Section 7).

Epidemiologists generally divide a population into compartments, such as **S**usceptible, **E**xposed, **I**nfectious, **R**ecovered, etc., when modeling the progression of a disease through a population. However, the main statistic of interest at any time is the proportion of the population that is currently infectious, as this guides intervention policy (mask mandates, school closures, etc.) as well as planning for healthcare resources.

In practice, it helps to be able to estimate disease prevalence in sub-regions within larger regions, e.g., in individual counties within a state. So, for generality, we define the true state of the epidemic at time t to be, $\mathbf{y}_t = [y_{0,t}, y_{1,t}, \ldots, y_{N,t}]$, where $y_{i,t}$ is the proportion of the population in (sub-)region i that is infectious at time t. N is the total number of regions. Let $\mathbf{x}_t$ be our estimate of $\mathbf{y}_t$, where $\mathbf{x}_t$ is an N-dimensional vector analogous to $\mathbf{y}_t$. Note that $\mathbf{y}_t$ is never directly observed.

The system has to assign a vector of numbers of tests, $\mathbf{n}_t = [n_{0,t}, n_{1,t}, \ldots, n_{N,t}]$, at each time step. We assume that we have a budget on tests, such that $1 \leq n_{i,t} \leq M$. In words, the system has to assign at least one test to each region at each time step (for reasons explained in Section 4) and can assign a maximum of M tests to each region at each time step.

Prevalence testing is expensive because it requires testing a random sample of the population, which means sending testers in the field to find the randomly selected people and test them. This requires much more time and effort than, e.g., opportunistically testing symptomatic people who come into healthcare facilities. However, it has the advantage of being unbiased. Our objective, therefore, is to minimize both the number of tests used and the error in the state estimate:

$$R(t) = |\mathbf{y}_t - \mathbf{x}_t| + \mathbf{n}_t, \tag{1}$$

where $|\cdot|$ is an appropriate vector norm, such as the L_2 norm.

3 Background

To estimate the unobserved underlying state of a dynamical system at time t, filtering methods maintain a posterior distribution over the state, given the prior estimate at time $t-1$ and the observation at time t. This distribution is also known as the belief state, $b(y_t)$, where y_t is the state at time t. In Bayes filtering, the belief state is updated in two steps. In the prediction step, the previous belief state is used to create a predicted belief state,

$$\hat{b}(y_t) = \int p(y_t|y_{t-1}) b(y_{t-1}) dy_{t-1}. \tag{2}$$

In the correction step, an observation model is used to convert the predicted belief state into a "corrected" belief state at time step t,

$$b(y_t) = \mu p(o_t|y_t)\hat{b}(y_t), \tag{3}$$

where o_t is the observation at time t and μ is a normalization factor. This class of models is also known as predictor-corrector filters due to the two step nature of this approach. The Kalman filter is the classic example, which assumes Gaussian distributions and linear dynamics.

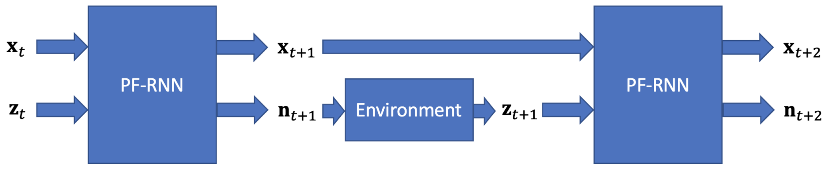

Fig. 1: The PF-RNN takes the previous state estimate, $\mathbf{x}_t$, as input, along with results of the tests (proportion positive), $\mathbf{z}_t$, in the previous round. It has to generate a prediction of the epidemic state for the next time step, $\mathbf{x}_{t+1}$, as well as the number of tests to be assigned to each region (county or health district), $\mathbf{n}_{t+1}$. The environment converts tests to test results, $\mathbf{z}_{t+1}$. The process continues with $\mathbf{x}_{t+1}$ and $\mathbf{z}_{t+1}$ forming the input for the next time step.

3.1 Particle Filtering

Particle filtering [5] is an approximate approach to Bayes filters, where the distributions are maintained non-parametrically as a collection of *particles* $\{y^i\}_t$ with associated weights $\{w^i\}_t$. This allows particle filters to be applied in very general settings, as this method makes no assumptions about the distribution over states. In the prediction step, each particle is sampled from the transition distribution,

$$y_t^i \sim p(y_t|y_{t-1}^i). \tag{4}$$

The weights are updated using the observations and the observation distribution,

$$w_t^i = \mu p(o_t|y_t)w_{t-1}^i. \tag{5}$$

In particle filtering, this is followed by an additional resampling step, where the particles are resampled with probabilities proportional to their weights. The weights of these resampled particles are set to $1/K$, where K is the total number of particles. This is done to avoid an empirical problem of "particle degeneracy", where the weights of many particles can rapidly fall to be close to zero, which causes the particle set to perform poorly at estimating the belief distribution.

Filtering methods generally assume that two things are known: the transition model $p(y_t|y_{t-1})$ and the observation model $p(o_t|y_t)$. In practice, this may often not be the case (as is also true in our application).

3.2 Particle Filter Recurrent Neural Networks

A recently developed method to address model learning along with state estimation is the Particle Filter Recurrent Neural Network (PF-RNN) [11], which combines particle filtering with recurrent neural networks (RNN), specifically Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) networks. This combination brings together the powerful time series learning and prediction capabilities of RNNs with the capability of particle filters to maintain an estimate of a time-dependent state with uncertainty. In particular, the PF-RNN maintains a collection of hidden layer particles, $\{h^i, w^i\}_t$, and computes the output as a function of the average or expectation of the particles, $f(\sum_i h_t^i w_t^i)$.

The PF-RNN functions similarly to traditional RNNs, although they differ on input information. Where RNNs are given historical state information to inform their prediction; the PF-RNN utilizes the previous state estimates created by particle filtering as input information. During training, the PF-RNN learns to approximate the transition model and the observation model through end-to-end discriminative training. An additional advantage of the PF-RNN is that the output $f(\sum_i h_t^i w_t^i)$ can be a task-specific control output (not just an estimate of the system state). We make use of this feature in our application for assigning tests to regions.

3.3 Active Sensing

This is essentially a reinforcement learning problem, where the actions are sensing actions (i.e., they do not change the environment) and are costly, so must be optimized to maximize some reward [16]. As in the case of filtering, the true state of the environment is never observed directly, so active sensing also operates on belief states. Since the goal is to choose actions so as to come up with the best estimate of the environmental state while minimizing cost, the reward $R(a, y)$ of action a in state y is generally specified as a combination of expected information gain and the cost.

We now describe how we bring together these ideas and couple them with an agent-based model for our problem.

4 Approach

The general idea for the approach is shown in Figure 1. Continuing with our notation from Section 2, we assume that the environment transforms the tests into test results, giving us a vector $\mathbf{z}_t = [z_{0,t}, z_{1,t}, \ldots, z_{N,t}]$, where each element corresponds to the proportion of tests assigned to that region that turn out positive (indicating an infected person). In this work, we use an agent-based model (ABM) for the environment. The idea is that we can generate many realistic simulated epidemic instances using a very detailed data-driven ABM, as described in Section 4.2, which can be used to train the PF-RNN on the underlying epidemic dynamics (i.e., the transition model), and we can use the ABM itself to provide the results of prevalence testing (i.e., to map $\mathbf{n}_t$ to $\mathbf{z}_t$ as shown in Figure 1). However, we would like the learned model to be applicable to a real-world setting, e.g., to be used when a new epidemic of the same kind (but with different transmission parameters) appears, so the true epidemic state cannot be used for training.

Therefore, instead of using Equation 1 directly as our objective function for training, we change it to,

$$r(t) := |\mathbf{x}_t - \mathbf{z}_t| + \eta|\mathbf{n}_t| + H(\mathbf{h}_t), \tag{6}$$

where $H(\mathbf{h}_t)$ is the entropy of the hidden state particles in the PF-RNN. The intuition is that we wish to have the state estimate match the test results (first term), while using as few tests as possible (second term), and also minimizing our uncertainty about the state (third term). The entropy term is optional and we present results with and without this term in Section 6. η is a constant that lets us adjust the relative importance of accuracy and the numbers of tests used. To avoid the degenerate solution where the system assigns no tests and predicts that no one is infected, we constrain the system to assign at least one test to every region at each time step.

4.1 PF-RNN

Since the original PF-RNN code is designed for use with image data, we have adapted it to work with vectors as needed for our application. Our modified version of PF-RNNs works in exactly the same way for training and will be made available via GitHub once the paper is accepted.

We did simple hyperparameter optimization to determine the best learning rate, the final value being 5×10^{-5}. We also tried various values for the number of particles and the number of hidden layer nodes and determined the optimal number of particles to be 64, with a hidden layer of 256 nodes.

For the actual PF-RNN training, following the approach of Ma et al. [11], we used a modified version of Equation 6 as the objective function:

$$a(t) := \sum_{n=1}^{N}(\mathbf{x}_t - \mathbf{z}_t) + \eta \sum_{n=1}^{N}(\mathbf{p}_t) \tag{7}$$

$$b(t) := \sum_{n=1}^{N}(\mathbf{x}_t - \mathbf{z}_t)^2 + \eta \sum_{n=1}^{N}(\mathbf{p}_t)^2 \tag{8}$$

$$\hat{a}(t) := \sum_{n=1}^{N}(\hat{\mathbf{x}}_t - \mathbf{z}_t) + \eta \sum_{n=1}^{N}(\hat{\mathbf{p}}_t) \tag{9}$$

$$\hat{b}(t) := \sum_{n=1}^{N}(\hat{\mathbf{x}}_t - \mathbf{z}_t) + \eta \sum_{n=1}^{N}(\hat{\mathbf{p}}_t) \tag{10}$$

$$r(t) := \alpha a(t) + \beta b(t) + \gamma(\log \hat{a}(t) + \hat{b}(t)) \tag{11}$$

The first of these equations $a(t)$ represent the mean error of the predicted proportion, and the tested proportion; with a secondary term representing the number of tests assigned. The intuition is that we wish to have the state estimate match the test results (first term), while using as few tests as possible (second term). $\sum_{n=1}^{N}(\hat{\mathbf{x}}_t - \mathbf{z}_t)$ is in the range of 0 - 1, however the testing component is limited by the total number of tests, therefore can be far greater than 1. To counter act this, the testing error was multiplied by discount factor η which changed the range to 0 - 1. $a(t)$ is then added to $b(t)$, which is very similar to $a(t)$ although using mean square error. An additional component is added to the first two functions to minimize the *evidence lower bound* (ELBO). The ELBO is utilized to determine the probability that a model is likely to predict, in this case, the desired output. The method utilized by Ma et al. is also utilized here, and completes our objective function with the addition of $\log \hat{a}(t) + \hat{b}(t)$ with constant γ to lessen the impact on the reward. Equations $a\hat{(t)}$ and $b\hat{(t)}$ are equivalent to $a(t)$ and $b(t)$, although in place of the PF-RNN prediction (x), the average of the hidden particle filter layer for prediction and test assigning is taken, $(\hat{x})$ and $(\hat{p})$ respectively. This process was shown to be more robust to noise and variation than those using simply a mean square error reward [11].

Finally, calculating the particle filter entropy is not straightforward due to the fact that a discrete set of particles is used to approximate a continuous probability distribution [4]. However, we do not need to calculate the entropy precisely. Since variance is monotonically related to entropy, we add the variance of the particles as an additional term to the objective function when we wish to minimize the entropy also.

Next we describe the agent-based model of epidemics used for training. This consists of two parts: a very detailed and realistic synthetic population of the region and a high-performance and flexible epidemic simulator that operates on the synthetic population.

4.2 The Epidemic Simulator

In this work, we made use of *EpiHiper* [8], which is a high-performance computational modeling framework supporting epidemic science. The design of EpiHiper comprises four crucial parts (*i*) a labeled, time-varying social *contact network* over which contagions spread, (*ii*) fully *customizable disease models* capturing disease transmission between hosts as well as within-host disease progression; (*iii*) *user-programmable interventions* covering both pharmaceutical and non-pharmaceutical ones; and (*iv*) the discrete time, parallel simulator designed to take full advantage of modern high performance computing (HPC) hardware. This modeling framework has been used extensively by its creators throughout the ongoing COVID-19 pandemic by directly supporting planning and response efforts to support state, local, and federal authorities.

Disease model assumptions: It is assumed that (i) propensities for a person are independent across contact configurations, and (ii) that during any time step no person can change their health state. The first assumption is quite common and not unreasonable for the contact networks that are used. Violations of the second assumption can always be accommodated by reducing the size of the time step. Its real purpose is to ensure *order invariance* of contacts within a time step, thus providing the required guarantee for algorithm correctness.

To mathematically describe the *disease transmission model*, consider a situation where a susceptible person P is in contact with infectious person P'. Looking at the pair (P', P), we combine the *state susceptibility* and *state infectivity* of their respective health states X_k and X_i with the *infectivity scaling factor* of P'

and the *susceptibility scaling factor* of P to form the *propensity* associated with the *contact configuration* $T_{i,j,k} = T(X_i, X_j, X_k)$ for the potential transition of the health state of person P to X_j as:

$$\rho(P, P', T_{i,j,k}, e) = \big[T \cdot \tau\big] \times w_e \times \alpha_e \times \big[\beta_s(P) \cdot \sigma(X_i)\big] \times \big[\beta_i(P') \cdot \iota(X_k)\big] \times \omega(T_{i,j,k}) \tag{12}$$

Here, T is the duration of contact for the edge $e = (P', P, w, \alpha, T)$, w is an edge weight, and α is a Boolean value indicating whether or not the edge is active (e.g., not disabled because of an ongoing school closure). A complete list of parameters and notation can be found in Table 1.

Data sets for training and validation purposes have been created by running EpiHiper with an age stratified COVID-19 disease model. We consider five age groups (Preschool 0-4 years, Students 5-17, Adults 18-49, Older Adults 50-64, and Seniors 65+). The simulation was run for the entire state of Virginia. The initial ten cases of the outbreak where restricted to Lee County (FIPS: 51105). To evaluate different disease dynamics we created multiple trajectories with transmissibility values in the range of τ (0.025, 0.095). For each transmissibility value, 50 trajectories were created for each. All of the other parameters were kept fixed at values already set in EpiHiper.

Parameter	Description
P, P'	Persons/agents/nodes
X_i	Health state i
$\sigma(X_i)$	Susceptibility of health state X_i
$\iota(X_i)$	Infectivity of health state X_i
$\beta_\sigma(P)$	Susceptibility scaling factor for person P
$\beta_\iota(P)$	Infectivity scaling factor for person P
w_e	Weight of edge $e = (P, P')$
α_e	Flag indicating whether the edge e is active
$T(X_i, X_j, X_k)$	Contact configuration for a susceptible transition from X_i to X_j in the presence of state X_k
$\omega_{i,j,k}$	Transmission weight of contact configuration $T(X_i, X_j, X_k)$
τ	Transmissibility
$\rho(P, P', T_{i,j,k}, e)$	Contact propensity

Table 1: EpiHiper core model parameters.

5 Experiments

To evaluate the performance of our approach, we conducted a variety of experiments to determine its capabilities; specifically, modeling the pandemic within and across counties and testing strategy. Experiments were conducted on simulated trajectories of pandemic spread in 12 counties in the state of Virginia. The trajectories were extracted from EpiHiper runs for the full Virginia population so that the relative progression of the epidemic in the chosen counties is meaningful. These counties were selected for proximity, population size, and population density.

Additionally we tested the viability of modeling infection rates with previously unseen spread parameters. Each PF-RNN was trained for 500 epochs, each lasting approximately 180 ticks (simulated days), with 49 unique trajectories. In total, 90000+ days of pandemic spread were seen during each training session. Each experiment was evaluated on a previously unseen trajectory with transmissibility $\tau = 0.055$. Performance was assessed based on mean squared error and total number of tests used. We compared four different training scenarios:

1. **No action**: In this case, the number of tests assigned to each county was fixed at 5. The neural network only learned to use the test results in combination with its own prediction at time step $t - 1$ to estimate the epidemic state at time t.
2. **With action**: In this case, the neural network has to generate both the estimate of the epidemic state at time t and the number of cases to assign to each region in time step $t + 1$.
3. **No entropy**: This is the same as the previous case, but without the term for the variance of the set of particles in the objective function. This encourages the neural network to minimize prediction error without worrying about minimizing the uncertainty in the prediction.
4. **Multi-trajectory**: In this case, training was done on a range of transmissibility rates in the range of $(0.025 - 0.095)$ and evaluation was conducted on a trajectory with an unseen transmissibility value within this range. In total, 200 trajectories were seen in training, 50 per transmissibility value. The goal was to evaluate if the trained system can perform well on a new variant of the epidemic.

6 Results

Our main result is the ROC curve shown in Figure 2. The curve in the figure corresponds to the average RMSE if a fixed number of tests (the value on the x-axis) are assigned to each region on each day, and there is no learned model used to estimate the epidemic state based on the previous day. In other words, the estimate is generated independently on each day.

In this case, the estimate for each day is simply a draw from a binomial distribution with probability of success given by the proportion of the population that is infectious on that day. Thus, as expected, the average RMSE drops exponentially as the number of tests increases. The outcomes of the four experiments are plotted in the figure as points and there are distinct differences in the RMSE and numbers of tests for these scenarios.

The 'multi-trajectory' scenario has the highest average RMSE, while using the fewest (3) tests/day on average. The 'no entropy' scenario has the lowest RMSE, while using the most (16) tests/day on average. The 'with action' scenario provides the best tradeoff between the two, using only 4 tests/day, while

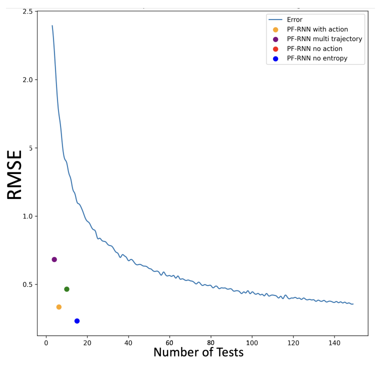

Fig. 2: ROC curve showing the trade-off between RMSE and number of tests.

generating the second-lowest average RMSE. The important thing to note overall, however, is that all the scenarios use far fewer tests than the independent case for the same level of error. For instance, for the 'with action' scenario, the corresponding point on the curve with the same average RMSE uses ∼120 tests/day, which is a ∼30x improvement for our method. Even for the 'multi-trajectory' case, we see a >10x improvement.

In Figure 3, we show the percent error over time in the epidemic state estimate for the 'with action' scenario. The average number of tests assigned on each day are also plotted. We see that the error is never more than ±10%.

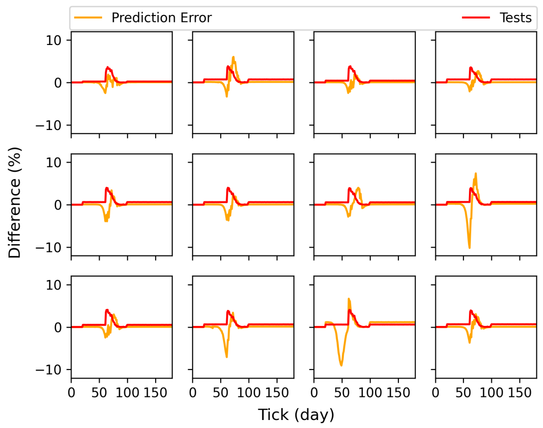

Fig. 3: PF-RNN prediction and testing strategy on 12 individual counties.

We note that we also tried other RNNs, such as LSTMs, but they performed similarly to the PF-RNN without entropy.

7 Conclusion

This study demonstrated the viability and success in implementing recurrent neural network algorithms in both predicting epidemic prevalence and optimal testing strategy. In total, over 300,000 days of simulated epidemics were trained and evaluated on. We evaluate the performance of PF-RNN on predicting prevalence, assigning tests, and transfer learning capabilities. Each model evaluated was accurately able to estimate infection rates in a 12 county area; minimizing mean squared error while testing less than 0.1 % of the population.

A number of extensions are possible to improve the applicability of this approach. In particular, the epidemic simulator is capable of modeling a range of interventions, so our approach could be trained on scenarios where different interventions are applied at different times. Other simulators can also be used, for examples ones that model human behavior and decision-making in greater detail [13]. Vaccination can also be incorporated into these simulators, providing another layer of realism and extending the duration for which our method would be applicable. The same is true of scaling to larger regions, as the simulators are designed to be high performance, and easily scaling to populations of several millions of agents.

Beyond the straightforward extensions that increase the applicability and realism of our approach, there are a few methodological directions of research also. In particular, we can explore dynamically varying the parameter η, which is essentially equivalent to the exploration-exploitation trade-off in active sensing since it controls the relative importance of cost vs. accuracy. This might be useful in triggering additional sensing when evidence suggests the emergence of new disease variants, for example. Going further, we can explore a control setting, where the goal is not just epidemic state estimation, but also containment.

In the world of post COVID-19, epidemic strategy should be in constant consideration to prevent another large scale pandemic. This study is a step towards more efficient AI-supported decision making; preventing the next epidemic from becoming a pandemic.

Acknowledgements This work was supported by NSF Expeditions in Computing grant CCF-1918656, Virginia Department of Health award UVABIO610-GY23, University of Virginia Strategic Investment Fund award number SIF160, and the Global Infectious Diseases Institute grant “Machine Learning Efficient Behavioral Interventions for Novel Epidemics” at the University of Virginia.

References

1. Ahmed, N., Michelin, R.A., Xue, W., Ruj, S., Malaney, R., Kanhere, S.S., Seneviratne, A., Hu, W., Janicke, H., Jha, S.K.: A survey of COVID-19 contact tracing apps. IEEE Access **8**, 134577–134601
2. Angione, C., Silverman, E., Yaneske, E.: Using machine learning to emulate agent-based simulations. arXiv:2005.02077 [cs.MA] (2020)
3. Bastani, H., Drakopoulos, K., Gupta, V., Vlachogiannis, I., Hadjichristodoulou, C., Lagiou, P., Magiorkinis, G., Paraskevis, D., Tsiodras, S.: Efficient and targeted

COVID-19 border testing via reinforcement learning. Nature **599**(7883), 108–113 (2021)

4. Boers, Y., Driessen, H., Bagchi, A., Mandal, P.: Particle filter based entropy. In: Proceedings of the 13th International Conference on Information Fusion (2010)
5. Chopin, N., Papaspiliopoulos, O.: An Introduction to Sequential Monte Carlo. Springer International Publishing (2020)
6. Cramer, E.Y., Huang, Y., Wang, Y., Ray, E.L., Cornell, M., Bracher, J., Brennen, A., Castro Rivadeneira, A.J., Gerding, A., House, K., Jayawardena, D., Kanji, A.H., Khandelwal, A., Le, K., Niemi, J., Stark, A., Shah, A., Wattanachit, N., Zorn, M.W., Reich, N.G., US COVID-19 Forecast Hub Consortium: The United States COVID-19 Forecast Hub dataset. medRxiv:10.1101/2021.11.04.21265886v1 (2021)
7. Franceschi, V.B., Santos, A.S., Glaeser, A.B., Paiz, J.C., Caldana, G.D., Lessa, C.L.M., Mayer, A., Küchle, J.G., Zen, P.R.G., Vigo, A., Winck, A.T., Rotta, L.N., Thompson, C.E.: Population-based prevalence surveys during the Covid-19 pandemic: A systematic review. Reviews in Medical Virology **31**(4)
8. Hoops, S., Chen, J., Adiga, A., Lewis, B., Mortveit, H., Crow, J., Diskin, E., Levine, S., Tazelaar, H., Rossheim, B., Ghaemmaghami, C., Price, C., Baek, H., Early, R., Wilson, M., Xie, D., Swarup, S., Venkatramanan, S., Barrett, C., Marathe, M.V.: High performance agent-based modeling to study realistic contact tracing protocols. In: Proceedings of the Winter Simulation Conference (2021)
9. Lueck, J., Rife, J.H., Swarup, S., Uddin, N.: Who goes there? Using an agent-based simulation for tracking population movement. In: Mustafee, N., Bae, K.H., Lazarova-Molnar, S., Rabe, M., Szabo, C., Haas, P., Son, Y.J. (eds.) Proceedings of the Winter Simulation Conference (WSC). National Harbor, MD, USA (2019)
10. Ma, Q., Liu, J., Liu, Q., Kang, L., Liu, R., Jing, W., Wu, Y., Liu, M.: Global percentage of asymptomatic SARS-CoV-2 infections among the tested population and individuals with confirmed COVID-19 diagnosis. JAMA Network Open **4**(12), e2137257
11. Ma, X., Karkus, P., Hsu, D., Lee, W.S.: Particle filter recurrent neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5101–5108 (2020)
12. Malleson, N., Minors, K., Kieu, L.M., Ward, J.A., West, A., Heppenstall, A.: Simulating crowds in real time with agent-based modelling and a particle filter. Journal of Artificial Societies and Social Simulation **23**(3) (2020)
13. de Mooij, J., Dell'Anna, D., Bhattacharya, P., Dastani, M., Logan, B., Swarup, S.: Quantifying the effects of norms on COVID-19 cases using an agent-based simulation. In: Van Dam, K.H., Verstaevel, N. (eds.) Multi-Agent-Based Simulation XXII, pp. 99–112 (2022)
14. Rennert, L., McMahan, C., Kalbaugh, C.A., Yang, Y., Lumsden, B., Dean, D., Pekarek, L., Colenda, C.C.: Surveillance-based informative testing for detection and containment of SARS-CoV-2 outbreaks on a public university campus: an observational and modelling study. The Lancet Child & Adolescent Health **5**(6), 428–436 (2021)
15. Thorve, S., Hu, Z., Lakkaraju, K., Letchford, J., Vullikanti, A., Marathe, A., Swarup, S.: An active learning method for the comparison of agent-based models. In: Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2020)
16. Yang, S.C.H., Wolpert, D.M., Lengyel, M.: Theoretical perspectives on active sensing. Current Opinion in Behavioral Sciences **11**, 100–108 (2016)