Differentiable Agent-based Epidemiology

Ayush Chopra∗
Massachusetts Institute of Technology
USA

Alexander Rodríguez∗
Georgia Institute of Technology
USA

Jayakumar Subramanian
Adobe MDSR
India

Arnau Quera-Bofarull
University of Oxford
United Kingdom

Balaji Krishnamurthy
Adobe MDSR
India

B. Aditya Prakash
Georgia Institute of Technology
USA

Ramesh Raskar
Massachusetts Institute of Technology
USA

ABSTRACT
Mechanistic simulators are an indispensable tool for epidemiology to explore the behavior of complex, dynamic infections under varying conditions and navigate uncertain environments. Agent-based models (ABMs) are an increasingly popular simulation paradigm that can represent the heterogeneity of contact interactions with granular detail and agency of individual behavior. However, conventional ABM frameworks not differentiable and present challenges in scalability; due to which it is non-trivial to connect them to auxiliary data sources. In this paper, we introduce GradABM: a scalable, differentiable design for agent-based modeling that is amenable to gradient-based learning with automatic differentiation. GradABM can quickly simulate million-size populations in few seconds on commodity hardware, integrate with deep neural networks and ingest heterogeneous data sources. This provides an array of practical benefits for calibration, forecasting, and evaluating policy interventions. We demonstrate the efficacy of GradABM via extensive experiments with real COVID-19 and influenza datasets.

CCS CONCEPTS
• Computing methodologies → Machine learning; Agent / discrete models; • Applied computing → Epidemiology.

KEYWORDS
Differentiable Agent-based Modeling, Computational Epidemiology, Automatic Differentiation, Deep Neural Networks

ACM Reference Format:
Ayush Chopra, Alexander Rodriguez, Jayakumar Subramanian, Arnau Quera-Bofarull, Balaji Krishnamurthy, B. Aditya Prakash, and Ramesh Raskar. 2023. Differentiable Agent-based Epidemiology. In Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023, IFAAMAS, 13 pages.

1 INTRODUCTION
Agent-based models (ABMs) are discrete simulators which comprise a collection of agents which can act and interact within a computational world [19, 37, 50, 61, 72]. They can explicitly represent heterogeneity of the interacting population via underlying contact networks and model the adaptability of individual behavior to enable more realistic simulations. As a result, ABMs are increasingly popular in epidemiology in place of classical ODE-based models, such as the SIR (susceptible-infected-recovered) model [28, 33]. In recent efforts to contain the spread of COVID-19, ABMs have been used to evaluate the benefit of delaying 2nd dose of the vaccine [61], deploying mobile apps for digital contact tracing [6], and prioritizing test speed over specificity [45]. The utility of such simulators for practical decision making depends upon their ability to recreate the population with great detail, integrate with real-world data streams, and analyze the sensitivity of results.

However, ABMs are conventionally slow, difficult to scale to large population sizes [52] and tough to calibrate with real-world data [30]. Conventional ABM frameworks follow an object-oriented (agent-centered) design where the agents are modeled as objects. While conceptually appealing, object-oriented approaches often represent inter-agent infection transmission (over contact networks) and within-agent disease progression inefficiently, resulting in poor scalability as the agent population and number of interactions grows. Performing a single forward simulation over a large ABM can take several days [12, 18], making calibration results subpar where approaches require multiple forward passes [22, 61, 68]. This is an issue since simulation results (emergent behavior) can be highly sensitive to the scale of the input population and parameters. In addition, incorporating novel sources of data that could inform calibration and other downstream tasks (e.g., forecasting) is often laborious and adds overhead complexity to the ABM (e.g., incorporating digital exposure data to ABMs [34]).

In this paper, we introduce GradABM to alleviate these concerns and enhance the potential of ABMs for practical decision making in epidemiology (Fig. 1). Our key insight is to make simulations differentiable which enables gradient-based learning with automatic differentiation. First, in contrast to conventional ABMs, GradABM follows a tensorized design [25] where agent states are represented as vectors and their interaction networks as (sparse) adjacency matrices. As a result, forward simulation of agent interaction and disease progression can be run rapidly, in a highly parallelized manner (millions-size networked populations in a few seconds on commodity hardware). Second, ABMs are stochastic simulators which routinely require sampling from discrete distributions for execution (e.g., all interactions with an infected agent may
not result in a new infection; COVID-19 tests may return false positives/negatives with some probability). Ensuring differentiability requires gradient estimation through such discrete stochasticity. To achieve this, GradABM reparameterizes discrete distributions (e.g., Bernoulli) with continuous relaxations using the gumbel-softmax gradient estimator [39, 41]. This allows GradABM to benefit from gradient-based optimization and tools for automatic differentiation.

Using GradABM’s scalable and differentiable design, we merge it with deep neural networks (DNNs) to help with calibration, forecasting and policy evaluation. In particular, we demonstrate how our method can be coupled with a calibration DNN allowing seamless integration of novel datasets from heterogeneous sources—several of which were difficult to incorporate in the calibration of previous ABM designs. To illustrate the effectiveness of our proposed framework, we run several experiments using real world datasets. In particular, we treat COVID-19 and Influenza as case studies, highlighting GradABM’s benefits with respect to robust, time-space personalized forecasting, and evaluation of policy interventions.

Our Contributions: 1) Present GradABM, which is a scalable and differentiable ABM design amenable for gradient-based learning with auto-diff. 2) Demonstrate that GradABM allows quickly simulating realistic population sizes, integrating with deep neural networks and ingesting heterogeneous data sources which provides an array of practical benefits for calibration, forecasting, and policy evaluation. 3) Validate effectiveness of GradABM through experiments using real-world datasets of COVID-19 and Influenza.

2 BACKGROUND AND RELATED WORK

Before describing GradABM, we first review relevant works in agent-based models (ABMs) and with relevance to epidemiology. Then, we present our work in context of broad research in differentiable simulators and also briefly mention an insight we find useful when designing GradABM (permutation invariance). We refer the reader to [28, 33, 46, 59] for a more extensive discussion.

2.1 Agent-based Modeling

ABMs represent systems [19, 37] as collections of agents which can act and interact with each other within a computational world. Owing to their flexibility, practitioners have applied ABMs in a wide variety of problem domains such as modeling cells in a tumor micro-environment to diagnose cancers [50], humans in a physical environment to study economic policies [72], infectious diseases [6, 61] and avatars in a digital world to study misinformation [20].

ABMs in Epidemiology. In the context of epidemiology, ABMs are used to understand how disease spreads and evaluate efficacy of health interventions [48]. They simulate transmission of pathogens within multiple contact networks with an epidemiological model [12, 34] used to describe between-host transmission and within-host progression of infection. [29, 52] survey some methods and assumptions for designing such simulators. Our work introduces a tensorized and differentiable epidemiological simulator that can generalize across multiple infections and model specifications.

Scalability of ABMs. It is a key consideration since modeling granular population details is computationally expensive. Conventional frameworks like Mesa [53] follow an object-oriented design which while ease to use (Python-API) is prohibitively slow to scale (require few hours for a single iteration). There have been attempts to reduce this burden and simulate realistic scale with distributed HPC systems and GPU-optimized implementations. EpiFast [18] demonstrated simulating large contact networks in a few minutes on distributed systems. However, the required compute for such execution is expensive and not widely available. OpenABM [34] presents an optimized C++/Cuda API to build an epidemiological ABM for fast execution on commodity hardware but it is non-trivial for epidemiological practitioners to extend, whilst not being amenable to gradient-based learning. Motivated by recent work in molecular dynamics [62], GradABM is implemented using highly optimized sparse tensor APIs of auto-diff packages (e.g. PyTorch/JAX) to accelerate and scale simulations while preserving ease of use.

Calibration of ABMs. This involves identifying appropriate values for latent parameters and is essential to ensure reliability of results. Generally, a hybrid strategy is used where some parameters are sourced from control trials conducted offline by clinical experts and then others are calibrated in-silico by achieving a goodness-of-fit between emergent ABM output and real-world macro data.
(e.g. number of deaths). The conventional technique is to use grid search [22, 61], beam search [68] by running several forward simulations and finding the best-fit. This is often slow and difficult to scale to many parameters. Recent works have proposed new optimization methods for faster and more accurate calibration. For example, [57] uses an ML model to identify what parameters to calibrate, however, values are identified with random search. [70] use a gaussian process emulator which is computationally more efficient than the ABM. In a similar vein, [6] propose a surrogate strategy by first calibrating a compartmental SEIR-based model using standard ODE solvers, and supplying the resulting parameters to an ABM. However, such surrogate methods increase model miss-specification error as they cannot simulate stochasticity inherent in the ABM. Further, the SEIR-like models have few tunable parameters in comparison to ABMs, limiting the degree of calibration which can be achieved. In contrast, the scalable and differentiable design of GradABM allows efficiently calibrating large-scale ABMs with gradient-based learning (and using deep neural networks).

**DNNs with ABMs.** Integrating deep neural networks (DNNs) with ABMs is an active direction. One popular approach has focused on multi-agent reinforcement learning (MARL), wherein the ABM is treated as a reinforcement learning environment, and a DNN is used to learn policies for the agents. Such perspective has been employed by [72] to learn equitable economic policies, [64] to analyze societal segregation dynamics, and [55] to learn oil and gas macro strategies. An alternate strategy has focused on using DNNs to emulate agent-based models [9, 15, 57] by using the ABM to generate training dataset for DNNs. Recently, [15] used an epidemiological ABM to generate synthetic datasets and trained DNN models to predict infectiousness for contact tracing. [57] used a similar strategy (of DNN as surrogate) to identify what simulation parameters to calibrate. Both these directions (MARL, Surrogate) are constrained by non-differentiability of the ABM and hence cannot utilize automatic differentiation. In contrast, we introduce differentiable ABMs (GradABM) which allows joint training of DNNs with the ABM. This makes it possible to integrate heterogeneous data to infer latent micro variables and calibrate simulation parameters through hybrid DNN-ABM pipelines.

### 2.2 Automatic Differentiation for Simulations

Differentiable simulation is a powerful family of techniques that applies gradient-based methods to learning and control of physical systems. With progress in automatic differentiation, as computing gradients become easier, such simulators are emerging across wide range of systems. In combination (and joint training) with deep neural networks, such simulators are being used from computer graphics for looking through scattering media [23] and reconstructing human body pose [14]; to molecular dynamics for analysing the force field of ionic liquids [49, 62], to robotics for grasping and locomotion [32, 38]. Even in epidemiology, gradient-based optimization with epidemiological simulators has also been recently explored with SIR-like models to learn simulation parameters [10, 54] and for making neural models to learn epidemic dynamics from mechanistic models [58]. GradABM aims to realize differentiable simulations for stochastic agent-based models. In contrast to relevant recent work in [8, 11], GradABM obtains gradient estimates through stochasticity by efficiently reparameterizing with the gumbel-softmax gradient estimator ( unlike smoothing in [8] which has exponential cost scaling), shows scalability to million-size populations (unlike 25 agents in [11]) and also demonstrates ability to integrate with deep neural networks to calibrate large epidemiological ABMs.

### 2.3 Invariances, Computation and ABMs

As a result of leveraging the structure of the physical world, DNN architectures are effective in overcoming the curse of dimensionality [21]. This is key to efficient computation on grids with CNNs (translation invariance) and graphs with GNNs via neural message passing (permutation invariance). We posit that while useful for learning DNNs, utilizing these invariances can make computation tractable in physical systems, where they exist naturally. Specifically, we observe that epidemiological models [6, 25, 34, 61] also adhere to permutation invariance wherein the order of infectious interactions (pair-wise message passing) within a step does not matter when estimating the probability of infection from those interactions. Motivated by this observation, we implement the transmission model of GradABM as a differentiable message passing operation defined with physical equations instead of DNNs.

### 3 DIFFERENTIABLE AGENT-BASED MODELING FOR EPIDEMIOLOGY

In what follows, we present a framework for epidemiological modeling with differentiable ABMs. The pipeline is summarized in Fig. 2, where the inner loop is the differentiable epidemiological simulator (GradABM) and the outer loop is the gradient-based calibration procedure using the neural network CalibNN. The epidemiological simulator is described in Sec. 3.1, the calibration procedure in Sec. 3.2 and the use of the final model for forecasting and evaluating policy interventions in Sec. 3.3.

#### 3.1 GradABM: Epidemiological Simulator

Here, we describe the inner loop shown in Fig. 2. We consider a $K$-step discrete-event simulation with a population of $n$ interaction agents. The key contribution of this work is GradABM: an agent-based modeling design amenable for gradient-based learning with automatic differentiation. To achieve this, GradABM follows a tensorized implementation where each agent state is represented as a vector, the interaction networks as adjacency matrices and all discrete distributions (e.g., Bernoulli) are reparameterized with continuous approximations (Gumbel-Softmax [41]), for end-to-end differentiability. As we discuss later, this design allows scaling agent-based models to realistic population sizes and integrating them with deep neural networks, which provides an array of practical benefits. Before proceeding, we first describe the epidemiological model used in our implementation which follows from [25, 34, 61].

The state of each agent $j \in \{1, \ldots, n\}$, at time step $t \in \{0, \ldots, K\}$, is denoted by a 3-D vector $X^j_t = \{a_j, d_j^t, e_j^t\}$ where $a_j \in \{0 \text{– } 10, 11 \text{– } 20, 21 \text{– } 30, \ldots, 71 \text{– } 80, 80+\}$ is the age; $d_j^t$ is the current disease stage; and $e_j^t \in \{-1, \ldots, t - 1\}$ denotes the time step of last exposure. For example, in COVID-19, $d_j^t$ can take values in $\{S, E, I, R, M\}$ which denote susceptible (S), exposed (E), infected (I), recovered (R)
where_update function is proportional to an aggregate rate of transmission actions between susceptible and infected (or exposed) agents. This is characterized with time-dependent parameters which govern the transmission function \(\text{Transmit}(t, \omega(t))\) of infection transmission noted in Sec. 2 (more details in appendix). Before proceeding, the probability \(q(t)\) is used to sample the agent’s new disease stage from a Bernoulli random variable as 
\[ q(t) = \lambda A \] 

and mortality (M) stages. Agent state may update due to change in disease stage resulting from new infectious interaction with an exposed or infected agent (captured by the transmission function) or the natural progression of a previously incubated infection (captured by the progression function). This simulator is parameterized with time-dependent parameters which govern the transmission function \(\omega(t) = [R_i^t, S_a^t, T_e^t, \omega_k^t]\) and progression function \(\psi(t) = [\tau_{EI}^t, \tau_{IR}^t, \tau_{AM}^t, m]\). Here \(R_i^t\) is the time-dependent disease reproduction number (measure of infection rate), \(S_a^t\) is the age-stratified susceptibility, \(T_e^t\) is the disease stage-stratified infection transmissibility, \(\omega_k^t\) is the percentage of infections at time \(t = 0\), \(\tau_{EI}^t, \tau_{IR}^t, \tau_{AM}^t\) are the pairwise stage transition times from E to I, I to R and I to M, respectively, and \(m\) is the mortality rate (fraction of expired agents amongst all agents in the \([R, M]\) disease stage). At each step \(t\), every agent \(i\) interacts with a set of neighbors \(j : j \in N_i\) from which she/he may accumulate or transmit infection. This disease stage evolution, at any step \(t\), is given by:

\[
d_{i,t+1}^{\text{Update}} = \text{Update}(X_i^t, N_i, (X_j^t)_{j \in N_i}, \omega_k^t, \psi(t)),
\]

where \(\text{Update}(X_i^t, N_i, (X_j^t)_{j \in N_i}, \omega_k^t, \psi(t))\) is

\[
\begin{cases} 
\text{Transmit}(X_i^t, N_i, (X_j^t)_{j \in N_i}, \omega_k^t), & \text{if } d_i^t = S, \\
\text{Progress}(X_i^t, \psi(t)), & \text{if } d_i^t \in [E, I].
\end{cases}
\]

The transmission function \(\text{Transmit}(X_i^t, N_i, (X_j^t)_{j \in N_i}, \omega(t))\) computes the probability of infection transmission as a result of interactions between susceptible and infected (or exposed) agents. This probability is proportional to an aggregate rate of transmission accumulated over the multiple interactions and is defined as

\[
q(t) = 1 - e^{-\lambda A}
\]

where \(\lambda A = \bigcup_{j \in N_i} \{(\lambda(R_i^t, S_i^t, T_j^t, \omega_j^t)\}.\) Here, \(\lambda\) denotes the rate of transmission from a single interaction and \(\bigcup\) is an aggregation function which accumulates transmission over multiple interactions. In our implementation, \(\lambda\) is a linear function as defined in [6] and \(\bigcup\) is the summation (\(\Sigma\)) function following the invariance of infection transmission noted in Sec. 2 (more details in appendix). Before proceeding, the probability \(q(t)\) is used to sample the agent’s new disease stage from a Bernoulli random variable as 
\[ Q = \text{Bernoulli}(q(t), 1 - q(t)). \] Once infected, the agent can now infect other agents on subsequent steps and also enters a hierarchy of disease stage progression which triggers subsequent changes in the state of the agent. The progression function \(\text{Progress}(X_i^t, \psi(t))\) updates the disease stage from \(E \rightarrow I\) or from \(I \rightarrow [R, M]\) at times determined by the stage transition time parameters, which may also be stochastic and require sampling from discrete distributions. This sequence of transmission and progression is repeated for \(K\) steps of the simulation. Finally, the aggregate cumulative deaths is determined as: 
\[ \hat{y} = \text{Aggregate}(d_i^k) = m \times (d_i^k \in [R, M])_{i=1,\ldots,n}, \] 
where \(m \in \theta_p\). Ensuring differentiability of GradABM requires gradient estimation of discrete stochasticity. For this we replace the non-differentiable sample from the Bernoulli distribution with the Gumbel-Softmax gradient estimator [39, 42].
3.2 Training and Calibration of GradABM

In order to use GradABM for real-world applications, we need to be able to calibrate its parameters \((\theta_T^T, \theta_p^T)\) in such a way that the aggregate predicted quantities from the model such as cumulative deaths match the observed values from real-world data. Here, the differentiability of GradABM enables the possibility of utilizing gradient-based learning to calibrate the simulator by integrating with deep neural networks. Further, we note that the tensorized specification of GradABM is also critical since fast forward simulations are required to this iterative optimization where the simulator is executed at each optimization step. The calibration protocol is visualized in the outer loop in Fig. 2 and described below. This outer training and calibration loop runs for \(W\) optimization steps, with each step \(w \in \{1, ..., W\}\) comprising of the following four stages:

**Stage 1.** At the beginning of each outer loop, candidate parameters \((\theta_T^T, \theta_p^T)^w\) are generated by a calibration neural network (CalibNN), whose weights at time step \(w\) we denote by \(\phi^w\). To aid the generation of parameters, the calibration network is passed heterogeneous data as input. This may include data sources that were not considered during the initial construction of the underlying ABM being calibrated. For example, the calibration neural network may be passed social network data (e.g., self-reported symptom surveys conducted by Facebook) that was not integrated into the ABM, but that may still be of value when predicting disease prevalence. Intuitively, one may view this calibration network as a domain expert who integrates new data that the ABM designer was previously unaware of.

**Stage 2.** After parameters have been generated, they are passed to GradABM in order to perform a \(K\) step simulation, as described previously as the the inner loop in Fig. 2. This outputs the aggregate predictions from the simulator which includes cumulative infections, deaths etc.

**Stage 3.** The cumulative deaths across the \(K\) steps of simulation are then compared with corresponding ground truth values. More specifically, a mean-squared loss between the simulated and actual cumulative deaths is computed. This is defined as: \(L(\hat{y}^w, y; (\theta_T^T, \theta_p^T)^w) = \text{MSE}(\hat{y}^w, y)\), where \(y\) is the ground truth data representing cumulative deaths, \(\hat{y}^w\) the cumulative deaths predicted by GradABM at the \(w^{th}\) training step with input parameters \((\theta_T^T, \theta_p^T)^w\) and \(\text{MSE}\) denotes the mean-squared error function.

**Stage 4.** Since GradABM is differentiable (as established earlier), we may calculate the gradient of the computed loss with respect to each parameter of CalibNN, via backpropagation. Moreover, due to GradABM’s network-centric tensorized design, auto-diff packages from libraries such as PyTorch can be leveraged to compute these gradients efficiently in a highly parallelized manner. The final step of this gradient computation involves back-propagation through the weights of the CalibNN neural network \(\phi\) as well.

Once, the gradient of the loss is computed, we use the classical gradient descent algorithm to update the weights of the calibration neural network at the end of each iteration as follows:

\[
\phi^{w+1} = \phi^w - \alpha \frac{\partial L(\hat{y}^w, y; (\theta_T^T, \theta_p^T)^w)}{\partial \phi},
\]

where \(\alpha\) is the learning rate, \((\theta_T^T, \theta_p^T)^w = f(D; \phi^w)\) and \(\hat{y}^w\) is computed by calling \(\text{Update} K\) times with same parameters \((\theta_T^T, \theta_p^T)^w\). In essence, calibration here involves optimizing weights of CalibNN \(\phi\) which predicts the simulation parameters \((\theta_T, \theta_p)\) instead of the parameters directly. We are essentially training the calibration neural network to understand heterogeneous data as a means of producing good parameterizations for GradABM. We call this calibration pipeline involving CalibNN as deep calibrated GradABM (DC-GradABM).

Consider for instance the scenario with multiple simulators in an experiment. This is realistic since a distinct GradABM may be defined for each county in a state. We now have the opportunity to jointly calibrate parameters for all counties using a shared CalibNN. Given \(T\) counties, the weights of CalibNN \(\phi\) may be trained on the average of the loss for all counties (Stage 2). The same weights of CalibNN can be used to predict personalized parameters for each county-specific GradABM. We call such a multi-task calibration procedure as joint deep calibrated GradABM (JDC-GradABM).

Note that each step of gradient descent requires a full forward simulation of GradABM. As a result, GradABM’s network-centric design and tensorized implementation are key in ensuring that a sufficient number of gradient steps can be quickly computed to effectively train the calibration network. Additionally, observe that this optimization strategy is not possible with classical object-oriented ABMs, as they do not allow for scalable gradient computation.

**Remark 1.** We note that the CalibNN based approach presented in this paper is fundamentally distinct from emulation or surrogate models. Emulation models take the same input as the ABM’s input and predict the output of the ABM, without actually simulating any agent behavior. In contrast, the output of CalibNN serves as the input to GradABM, which then simulates the behavior of the agents. Thus, CalibNN extends simulation pipeline, by enabling us to learn the correct inputs for GradABM.

3.3 Forecasting and policy evaluation

Once trained the hybrid framework we propose, consisting of CalibNN and GradABM, it is ready to make forecasts in new, previously unseen scenarios as well as help decide on intervention policies. These unseen scenarios may include predicting future the evolution of multiple infections for a county/region, or it may involve making predictions even when the data (inputs to CalibNN) for a county/region is noisy. In each of these scenarios, the data available for the county/region is fed to CalibNN, which then outputs personalized time-varying parameters \(\theta_T^t\) and \(\theta_p^t\) for \(t \in \{0, ..., K + H\}\), where \(H\) is the forecasting horizon. These parameters are used in GradABM, which is run for \(K + H\) steps, where the last \(H\) simulation steps will serve as our forecasts.

4 EXPERIMENTAL SETUP

To illustrate the effectiveness of our framework, we conduct experiments for COVID-19 and influenza on multiple counties of the state of Massachusetts, USA, learning personalized parameters for each county. Before presenting our analyses, we provide details regarding our experimental setup.
4.1 Heterogeneous Data Sources

We describe the data sources passed as input to CalibNN, used to construct the population for GradABM and parameterize disease transmission and progression. (i) To start, we outline the features used as input to CalibNN during our experiments. In the case of COVID-19, CalibNN receives 5 input signals 1 including insurance claims data, online symptoms surveys from Facebook, and line-list data. Since all of these datasets contain granular information regarding each specific county, the data points input to CalibNN varies based on the county being modeled. Meanwhile, for influenza, CalibNN receives 14 signals originating from the Google symptoms dataset [2]. This data is reported at the state level and hence we use the same data for all counties. (ii) In order to construct GradABM, several datasets are used. Demographic information from the US Census [3] is used to generate a synthetic agent population. Agent interaction graphs are created using census block level mobility data from Safegraph and leveraging demographic information (such as age) inside each census block. (iii) With respect to parameters in the transmission and progression models, we utilize data from reliable sources such as the CDC [4] and clinical papers [31, 34]. The target variable for COVID-19 is the COVID-associated mortality, whilst for influenza we use influenza-like-illness (ILI) counts collected by the CDC. Ground truth data for both target variables is obtained from the JHU and the CDC.

4.2 Baselines

To showcase the performance of our calibration procedure (the outer loop described in Fig. 2), we compare GradABM with numerous benchmarks inspired by popular approaches for calibrating an ABM. ExpertSearch-ABM: Following [34, 61], it uses parameters set by a combination of expert advice and randomized search. For COVID-19, we get $R_0$ and case-fatality rate from [5, 17] and $R_0$ for flu from [26]. SurrogateODE-ABM: Following [6], it calibrates parameters by using a compartmental ODE model to emulate the ABM. We use SEIRM and SIRS models for COVID-19 and influenza respectively. Lastly, we compare three versions using GradABM that leverage gradients during the calibration process, namely calibrated GradABM (C-GradABM), deep calibrated GradABM (DC-GradABM) and jointly deep calibrated GradABM (JDC-GradABM) respectively. C-GradABM: directly optimizes the simulation parameters via gradient descent, without a calibration neural network (CalibNN). To clarify, the gradient update (in Stage 4) is given as: $\theta_t^{(L)} = \theta_t^{(L)} - \alpha \frac{\partial L(y_t, y_t^{(L)}(\theta_t^{(L)}))}{\partial \theta_t^{(L)}}$. DC-GradABM: calibrates the parameters by optimizing weights of CalibNN using automatic differentiation, as described in Section 3.2. JDC-GradABM: jointly calibrates parameters for all counties using a shared CalibNN (multi-task learning), as in Section 3.2. Here, instead of a new CalibNN for each county, a single CalibNN is shared. Intuitively, one may hope that such a calibration network learns to transfer useful information about disease spread from one county when reasoning about another.

4.3 Metrics

In order to provide a rigorous evaluation for the aforementioned calibration procedures, we use several standard metrics for evaluating epidemic predictions [7, 67]. Specifically, we use normal deviation (ND), root mean squared error (RMSE) and mean absolute error (MAE). We provide further details in our appendix. Following CDC forecasting guidelines [16, 27], we make weekly predictions for 1 to 4 weeks ahead in the future. Our evaluation for both diseases is of at least 4 months in 10 counties. More details regarding the periods of evaluation and counties investigated are in the appendix.

4.4 Implementation Details

In each of our experiments, we allow ABM parameters to change at each time step. That is, quantities such as the disease reproduction number and age-stratified susceptibilities are free to vary as the epidemic progresses. Consider the following intuitive argument for this design choice. It is reasonable to believe that individuals take greater precautions as an epidemic progresses. For example, many people began washing their hands more frequently during the COVID-19 pandemic, reducing the risk of disease transmission. Whilst GradABM does not explicitly model hand washing, parameters such as age-stratified susceptibility can be varied through time to account for its effect, especially if time series data related to sanitary practices is passed as input to CalibNN. Put differently, by allowing for ABM parameters to vary through time, we may account for environment dynamics or changes in agent behavior that aren’t explicitly modeled by the simulator. We stress that our calibration framework does not depend upon CalibNN’s neural architecture and that end-to-end training via a combination of backpropagation and gradient descent is possible regardless of CalibNN’s implementation. In our own experiments, we design CalibNN so that it may take advantage of the heterogeneous time series data it is passed as input. More specifically, we employ an encoder-decoder architecture, based on GRUs and self-attention [69]. More details regarding our choice of CalibNN architecture can be found in the appendix. 2

5 RESULTS

First, we show that calibrating with CalibNN can improve GradABM’s forecasting accuracy significantly. Moreover, we show that GradABM scales efficiently as the number of interactions between agents grows. Next, we also show that our calibration procedure is robust to noisy data, in the sense that GradABM can be well calibrated even with observational error in ground truth; and that GradABM can be used to conduct sensitivity analyses for evaluating the sensitivity of policy interventions.

5.1 Forecasting multiple infectious diseases

Table 1 displays the results of calibration for each of the benchmarks discussed in Section 4. Observe that JDC-GradABM outperforms other benchmarks across all metrics for both Influenza and COVID-19 (rows 1-3 in Table 1)3. For instance, JDC-GradABM improves ND to 0.97 from 8.75 in ExpertSearch-ABM and 2.21 in Emulator-ABM on COVID-19; and to 0.41 from 0.57 in ExpertSearch-ABM and

---

1Data links: delphi.cmu.edu, google/covid19symptomdataset, safegraph.com, coronavirus.jhu.edu/csse.com/ncov/surveillance/fluportaldashboard.html

2Our code and data is publicly available: https://github.com/AdityaLab/GradABM

3We performed the unpaired t-test test ($\alpha = 0.05$) over 5 runs to verify JDC-GradABM performance gains w.r.t other ABM calibration methods are statistically significant
Table 1: Forecasting results for COVID-19 and influenza over 5 runs. JDC-GradABM is the only one consistently among the best performing for all (lower error metrics is better).

| Model                  | COVID-19 |  |  | Influenza |  |  |
|------------------------|----------|---|---|-----------|---|---|
|                        |          | ND | RMSE | MAE       |   |   |
| ExpertSearch-ABM [61]  | 8.75     | 689.92 | 270.13 | 0.57 | 2.03 | 1.72 |
| SurrogateODE-ABM [6]   | 2.21 ± 1.36 | 121.87 ± 63.97 | 68.20 ± 41.84 | 0.59 ± 0.02 | 2.17 ± 0.05 | 1.77 ± 0.05 |
| JDC-GradABM            | 0.97 ± 0.18 | 50.99 ± 12.12 | 30.02 ± 5.60 | 0.41 ± 0.02 | 1.47 ± 0.06 | 1.22 ± 0.06 |
| DC-GradABM             | 1.15 ± 0.24 | 67.09 ± 23.89 | 35.50 ± 7.36 | 0.50 ± 0.19 | 1.78 ± 0.62 | 1.50 ± 0.57 |
| C-GradABM              | 2.39 ± 0.35 | 205.14 ± 42.56 | 73.66 ± 10.88 | 0.88 ± 0.14 | 2.97 ± 0.44 | 2.64 ± 0.43 |

Figure 3: GradABM run-time scales linearly with the number of interactions and is roughly 300x faster than prior-art. This benefit is due to the sparse-tensor calculus based design.

5.2 Robustness, Scalability and Decision Making

Next, we study the practical applicability of our framework by focusing on three key considerations: i) scalability to large populations, ii) robustness to observation error in data and iii) utility for practical decision making via sensitivity analyses of policy interventions.

5.2.1 Scalability of GradABM simulations. To investigate scalability, we examine how the simulation time of GradABM scales as the number of agent interactions increases. As is evident from Figure 3, the run-time for GradABM scales linearly with number of interactions in the population and executes very quickly (even as the adjacency matrix scales quadratically). For instance, GradABM executes a simulation with 800,000 agents (5 million interactions) over 133 steps in 4 seconds on a GPU (and 60 seconds on CPU). This is roughly 300x faster than the equivalent Mesa [53] implementation. This performance improvement can be attributed to GradABM’s sparse tensor-calculus implementation. Since forward simulation is efficient, sensitivity analysis of GradABM can be conducted rapidly via repeated simulations with different parameter settings. Moreover, iterative algorithms which perform a forward simulation at each step, such as gradient descent, can be used for calibration (as is demonstrated in our work).

5.2.2 GradABM robustness to observational error. To investigate the robustness of our proposed calibration procedure, we run experiments using ground truth data distorted by gaussian noise. More specifically, we add gaussian noise to each ground truth target with mean $\mu = 0$ and varying scales of standard deviation $\sigma$. To set the standard deviation of noise for each county, we first compute the standard deviation of the ground truth data and multiply it by a $\lambda$ factor. In our experiments, we test four different values for $\lambda$. Results are presented in Figure 4. Even for a large degree of noise ($\lambda = 4$), we observe that JDC-ABM outperforms both Emulator-ABM and ExpertSearch-ABM on noiseless data. We attribute this to CalibNN. More specifically, we conjecture that CalibNN alleviates overfitting by representing parameters in high-dim space of the neural network (instead of scalars) and allows for integration of heterogeneous data that may assist in selecting appropriate parameters even when the ground truth is noisy.

5.2.3 Evaluating sensitivity of policy interventions with GradABM. Early in the COVID-19 pandemic, uncertainty in vaccine supplies...
We introduced CalibNN which allows encoding simulation parameters with while relative mortality greater than 1 implies that policy P1 is (more details in the appendix). Following the exact experimental opens up multiple directions of future work both in terms of utility for calibration, forecasting, and evaluating policy interventions. More specifically, we consider a scenario with two alternative policies - P1: second dose is administered under standard schedule and P2: second dose is administered with a delay (more details in the appendix). Following the exact experimental setup from clinical works on this question [61], we vary the protection of the first COVID-19 dose from 50% to 80% in ten percentile increments and run simulations for each of the four configurations under both policies. Then, we compare both policies by computing the ratio of cumulative deaths of P2 by P1, which we denote as relative mortality. Basically, if the relative mortality is less than 1, then policy P2 is better (can delay the second COVID-19 dose); while relative mortality greater than 1 implies that policy P1 is better (don’t delay the second COVID-19 dose). Results for Franklin County, MA are shown in the appendix with additional details. We observe that once the protection of the first COVID-19 vaccine dose is greater than 60%, GradABM recommends policy P2 (to delay the second dose). These results are consistent with past clinical works that evaluated this policy intervention [40, 61].

6 CONCLUSION AND FUTURE WORK

We introduced GradABM, a design for agent-based modeling that is amenable to gradient-based learning with automatic differentiation. GradABM achieves this via a tensorized implementation where each agent state is represented as a vector, the interaction networks as adjacency matrices and all discrete distributions (e.g., Bernoulli) are reparameterized for end-to-end differentiability. Experiments demonstrate that GradABM can quickly simulate realistic population sizes, integrate with deep neural networks and ingest heterogeneous data sources. This provides an array of practical benefits for calibration, forecasting, and evaluating policy interventions.

The compatibility of GradABM with automatic differentiation opens up multiple directions of future work both in terms of utility to epidemiology and for advancing the science of agent-based modeling. First, in our work, disease progression in an individual was modeled using a linear deterministic model, but future work could explore more complex and stochastic models (e.g., [61]). Future work could also explore how to use gradient-based explainability methods [63] to have a better understanding of the underlying drivers of epidemic predictions. In this work, calibration with CalibNN predicts the maximum likelihood estimates but the our design allows us to also estimate the posterior distribution of the parameters using techniques from generative DNNs such as normalizing flows [51]. In addition, ABMs may be mis-specified for the actual disease dynamics for which we may have to change the model specification. In this work, we focused on GradABM using the OpenABM epidemiological model [35] designed for the US. As we explain next, GradABM generalizes to other model specifications.

The JUNE epidemiological model [12] is a large-scale agent-based model for epidemiology designed to support the National Health Service in England. It was also adopted by the UN Global Pulse to study refugee settlements [13]. In the England setting, JUNE simulates the movement and interactions of 55 million agents and features significantly more complex contact graphs than OpenABM, since it is based on the highest resolution of English census data. We were able to successfully translate it to the fully tensorized design by converting the contact structure in a heterogeneous graph by representing agents and locations as different node types. In this new implementation, which we refer to as GradABM-JUNE, we achieve a 10,000x speed increase in simulation time. Due to the high computational cost of JUNE, the original model was calibrated using a surrogate model in the form of a Gaussian Process (GP) emulator [70]. The differentiability of the new implementation coupled with the CalibNN pipeline allows us to quickly find parameter sets (reducing calibration time from 100,000 CPU hours to 12 CPU hours) that match the observed data well (Fig. 5) and are comparable in quality to the fit of the GP emulator. This case study is an encouraging evidence on the utility of GradABM for creating real impact and also is a step towards our vision of making GradABM a domain-agnostic design for scalable and differentiable agent-based modeling.
AAMAS 2023, May 29–June 2, 2023, London, United Kingdom

[42] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparametrization with Gumble-Softmax. In International Conference on Learning Representations.

[43] Harshavardhan Kamathri, Linghai Kong, Alexander Rodrigues, Chao Zhang, and B Aditya Prakash. 2022. CAMul: Calibrated and i-V Accurate Multi-view Time-Series Forecasting. In Proceedings of the ACM Web Conference 2022. 3174–3185.

[44] Harshavardhan Kamathri, Alexander Rodrigues, and B Aditya Prakash. 2022. Back2Future: Leveraging Backfill Dynamics for Improving Real-time Predictions in Future. International Conference on Learning Representations (2022).

[45] Daniel B Larremore, Bryan Wilder, Evan Lester, Soraya Shehata, James M Burke, James A Hay, Milind Tambe, Michael J Mina, and Roy Parker. 2021. Test sensitivity is secondary to frequency and turnaround time for COVID-19 screening. Science advances 7, 1 (2021), eabd5395.

[46] Alexander Lavin, Hector Zenil, Brooks Paige, David Krakauer, Justin Gottschlich, Tim Mattson, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Yaman, et al. 2019. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the United States. Proceedings of the National Academy of Sciences 116, 8 (2019), 3146–3154.

[47] Joseph T Wu, Kathy Leung, and Gabriel M Leung. 2020. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. The Lancet 395, 10225 (2020), 689–697.

[48] Stephan Zheng, Alexander Trotz, Sumit Sinha, David C Parkes, and Richard Socher. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. Science Advances 8, 18 (2022), eabk2607.

[49] Alexander Rodriguez, Harshavardhan Kamathri, Pulak Agarwal, Javen Ho, Mira Patel, Suchet Sapre, and B Aditya Prakash. 2022. Data-Centric Epidemic Forecasting: A Survey. arXiv preprint arXiv:2207.09370 (2022).

[50] Alexander Rodriguez, Nidhi Muralidhar, Bijaya Adhikari, Anika Tabassum, Naren Ramakrishnan, and B Aditya Prakash. 2021. Steering a Historical Disease with reinforcement learning and agent based modeling. Nature Communications 12, 1 (2021), 88–101.

[51] Emanuel Sert, Yannakos Chatzakos, and Balaji Lakshminarayanan. 2021. Normalizing Flows for Probabilistic Modeling and Inference. J. Mach. Learn. Res. 22, 57 (2021), 1–64.

[52] Lorenzo Pellis, Frank Ball, Shweta Bansal, Ken Eames, Thomas House, Valerie Isham, and Pieter Trapman. 2015. Eight challenges for network epidemic models. Epidemics 10 (2015), 58–62.

[53] Shane Tomlinson, and Mark Workman. 2021. Revealing Robust Oil and Gas Company Macro-Strategies Using Deep Multi-Agent Reinforcement Learning, Available at SSRN 3933996 (2021).

[54] Nicholas G Berch, Logan C Brooks, Spencer J Fox, Sasikiran Kandula, Craig J McGowan, Evan Moore, Dave Osthus, Evan L Ray, Ablainav Tushar, Teresa K Yamana, et al. 2019. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the United States. Proceedings of the National Academy of Sciences 116, 8 (2019), 3146–3154.

[55] Thibault Risser, Monica Columbeau, Andrew Shattuck, Lydia Burgert, Thomas A Smith, Sarah Filippi, Ewan Cameron, and Melissa Aenny Penny. 2021. Machine learning approaches to calibrate individual-based infectious disease models. medRxiv (2021).

[56] Alexander Rodrigues, Jiaming Cui, Naren Ramakrishnan, Bijaya Adhikari, and B Aditya Prakash. 2022. EINNs: Epidemiologically-Informed Neural Networks. arXiv preprint arXiv:2202.1046 (2022).

[57] David B Larremore, Bryan Wilder, Evan Lester, Soraya Shehata, James M Burke, James A Hay, Milind Tambe, Michael J Mina, and Roy Parker. 2021. Test sensitivity is secondary to frequency and turnaround time for COVID-19 screening. Science advances 7, 1 (2021), 20210135.

[58] Samuele Schoenholz and Ekin Dogus Cubuk. 2020. Jax md: a framework for simulating multiatom systems. Proceedings of the IEEE international conference on computer vision. 618–626.

[59] Joseph T Wu, Kathy Leung, and Gabriel M Leung. 2020. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. The Lancet 395, 10225 (2020), 689–697.

[60] Stephan Zheng, Alexander Trotz, Sumit Sinha, David C Parkes, and Richard Socher. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. Science Advances 8, 18 (2022), eabk2607.