Deep Learning-based Spatially Explicit Emulation of an Agent-Based Simulator for Pandemic in a City

Varun Madhavan [1] **Adway Mitra** [1] **Partha Pratim Chakrabarti** [2]

Abstract

Agent-Based Models are very useful for simulation of physical or social processes, such as the spreading of a pandemic in a city. Such models proceed by specifying the behavior of individuals (agents) and their interactions, and parameterizing the process of infection based on such interactions based on the geography and demography of the city. However, such models are computationally very expensive, and the complexity is often linear in the total number of agents. This seriously limits the usage of such models for simulations, which often have to be run hundreds of times for policy planning and even model parameter estimation. In this paper, we propose a Deep Learning model, based on the Dilated Convolutional Neural Network, that can emulate such an Agent-Based Model with high accuracy. We show that use of this model instead of the original Agent-Based Model provides us major gains in the speed of simulations, allowing much quicker calibration to observations, and more extensive scenario analysis. The models we consider are spatially explicit, as the locations of the infected individuals are simulated instead of the gross counts. Another aspect of our emulation framework is its divide-and-conquer approach that divides the city into several small overlapping blocks and carries out the emulation in them parallelly, after which these results are merged together. This ensures that the same emulator can work for a city of any size, and also provides significant improvement of time complexity of the emulator, compared to the original simulator.

[*]Equal contribution [1]Indian Institute of Technology Kharagpur [2]Indian Institute of Technology Kharagpur. Correspondence to: Adway Mitra <adway@cai.iitkgp.ac.in>, Varun Madhavan <varun.m.iitkgp@gmail.com>.

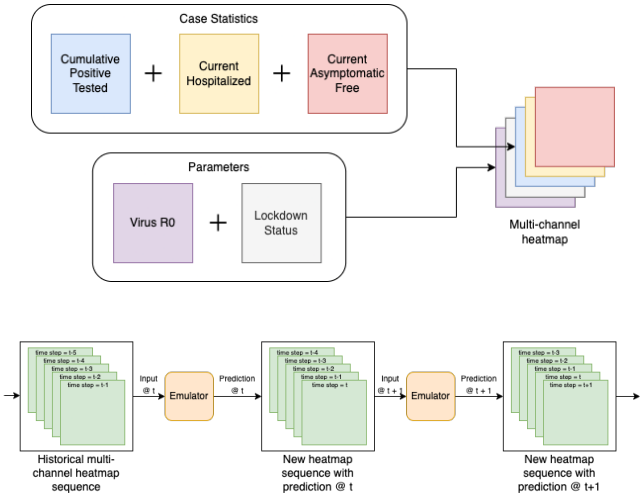

Figure 1. The multi-channel heatmap based emulation framework. The figure above shows how spatially-varying statistics and parameters are merged into a single multi-channel heatmap, and the figure below shows the input-output flow of the emulator.

1. Introduction

During the Covid-19 pandemic throughout 2020-2021, Epidemiological Models gained considerable attention worldwide, as policymakers scrambled to predict the impacts of Non-Pharmaceutical Interventions (NPIs) like lockdown orders, weekend curfews, containment zones etc. It is necessary to make a delicate trade-off between curbing disease spread and socio-economic disruption. To enable such a trade-off, it is necessary to have an estimate of the possible consequences of different policies, such as a *what-if* analysis. Carrying out such an analysis is difficult because there is no closed-form analytical formula to express the possible results of a particular policy, at least not for the pandemic spread. The way a pandemic may spread through a city is related to its demography, geography, public spaces and the interaction between these. It becomes necessary to develop simulators that may represent how the infections can spread out over space and time, under different characteristics of the virus (basic reproductive number R_0), the policies in place (eg. lockdown in whole or parts of the city)

and people's compliance with them. One way to build such simulators is by using *differential equation-based compartmental models*, or by *Agent-Based Models*. The former are simpler and more efficient to run, but are unable to incorporate individual-level behavior in response to NPIs and spatial nature of the infection spread, while these pros and cons are reversed for Agent-Based Models.

In this paper, we look to develop machine learning-based emulators for Agent-Based Models to get the best of both worlds, being able to produce spatially explicit simulations taking individual behavior into account like Agent-Based Models, but in a significantly less time. For this purpose, we use a neural network based on Dilated Convolutional Neural Network that is trained on spatial heatmaps of daily infections, hospitalizations, recoveries, deaths as simulated by Agent-Based Models as a response to different parameters like R_0 and lockdown policies, and is able to produce such spatial heatmaps by sequential predictions starting from specified initial conditions. We show through detailed experiments that the spatial heatmaps produced by the emulator very closely resemble those generated by the Agent-Based Model, while requiring only a small fraction of time apart from a one-time training phase. For 14 simulation runs, the emulator (including training) takes equal time as the ABM, for 29 runs the emulator requires only half the time required by the ABM, and for 58 runs the time required by emulator is only a quarter of ABM. To the best of our knowledge, this is the first work that can successfully emulate a full-scale Agent-Based simulate at spatial scale.

The rest of the paper is organized as follows - in section 2 we describe prior work on epidemiological modelling, Agent-Based Models and the emulation of ABMs. In section 3 we formally state the problem, followed by a description of working of the ABM used for epidemic modelling in section 4. In section 5 we describe the main Emulation framework, and discuss experiments and results of this approach in section 6. Finally, we discuss some of the possible applications of this framework in section 7 and conclude in section 8.

2. Related Works

In this section, we review relevant literature on the three aspects of this work- Epidemiological models, Agent-Based Models and Machine Learning based surrogate modeling.

2.1. Epidemiological Models

Compartmental models have commonly been used for epidemic simulations, where the total number of individuals in different health states like Susceptible, Infected and Recovered are tracked, and the time-evolution of these numbers modeled by differential equations. In the context of Covid-19, these well-known compartments are insufficient due to the presence of large number of *asymptomatic* cases. In these models, the impacts of NPIs is handled by varying the parameters (eg. rate of infection and recovery) over time. Silva et al. (2020) proposes a COVID-ABS model to predict the impact of NPIs on public health and urban economics by simulating the contained environment of an urban city. The framework implements a compartmental SEIR model and takes into account the demographic and economic distributions of the population. Census data for Brazil is used for the parameters of this SEIR model, but the framework can be extended for any demographic. Hoertel et al. (2020) perform a similar analysis for France. They use detailed attributes pertaining to the specific demographics and social contact structure in France and implement a statistical model for city spaces and transport. Agrawal et al. (2021) came up with the SUTRA compartmental model which includes more compartments like asymptomatic cases.

2.2. Agent-Based Modeling

Agent-Based Modeling emerged as late as the 1990s in the domain of computational social science (Epstein & Axtell, 1996; Bonabeau, 2002) for simulation of artificial societies and their underlying interactions and outcomes. In such models, the behaviors of individuals (called agents) and their mutual interactions are represented mathematically, to observe the outcomes of systems which are difficult to express mathematically. Agent-Based Modeling has also been used in the domains of economics (Tesfatsion, 2003), ecology (Grimm et al., 2006) and population biology (Brauer et al., 2012). Agent-Based Modeling has tremendous potential for modelling urban systems, as detailed in O'Sullivan & Haklay (2000) to simulate the interactions of all stakeholders with their urban environment. Batty (2008) and Bharath et al. (2016) explore temporal changes in urban sprawl and land-use compare ABMs with Cellular Automata-Based models. In these works, agents represent residents in a city and rule-based models are used to simulate their interactions (Heppenstall et al., 2016; Motieyan & Mesgari, 2018). Recent work on ABMs focuses on modelling specific unique characteristics/aspects of cities in greater detail. Works such as Fosset et al. (2016); Patel et al. (2018); Huynh et al. (2015); Hager et al. (2015); Kagho et al. (2020); Waddell et al. (2003) have focused on modeling different aspects of urban life, including urban population growth, pollution and vulnerability to climate change, mobility and traffic management, resource planning and so on. Several ABM-based software packages like MATSIM (Horni et al., 2016) and UrbanSim (Waddell, 2002) have also been developed.

In the context of Covid-19 pandemic simulation, several works have attempted to study the spread of infections as a function of social interactions among individuals of a city. For such modeling of social interactions, various aspects of a city such as the family structures, residences, workplaces,

spaces of social interactions etc can be represented at different levels of complexity or detail. Several works like Truszkowska et al. (2021); Hinch et al. (2020); Kerr et al. (2020) have developed such models for simulation of the pandemic.

This work uses the ABM developed in Suryawanshi et al. (2021) to compare the performance of the Neural Network-based emulators in predicting the outcomes of the counterfactual scenarios studied in the paper.

2.3. Emulating Agent-Based Models

Machine Learning and Deep Learning are increasingly being used for Agent-Based Modeling, primarily for parameter estimation using techniques like Approximate Bayesian Computation, or for developing surrogate models or emulators. This field is still quite nascent. Zhang et al. (2021) and Jäger (2021) explore how Agent-Based Models have been augmented using Machine Learning techniques. In (Zhang et al., 2018) Machine Learning techniques are used to emulate the real-time flow of city traffic as simulated by an Agent-Based Model. They use a shallow convolutional neural network (CNN), which is able to predict the traffic flow significantly faster than the original ABM while maintaining comparable accuracy. Angione et al. (2020) and Anirudh et al. (2020) also propose using Neural Networks to emulate the outputs of an ABM epidemic model. Angione et al. (2020) compares the performance of several candidate ML models as emulators and concludes that Neural Networks perform the best.

None of the emulators mentioned above are spatially explicit, i.e. they produce only a time-series of different quantities, not their spatial maps. Also, a comprehensive study comparing time benefits and accuracy of emulation is lacking. The current work aims to address these gaps.

3. Problem Definition

Let us formally define the problem that we aim to solve in this paper. We consider a city with N residents. The city is divided into K blocks, and they have populations $N_1, N_2, \ldots, N_K$ such that $\sum_{k=1}^{K} N_k = N$. Each resident i is provided with attributes such as age, family connections, health status and workplace, and mapped to a block S_i as residence. A pandemic strikes the city which has basic reproductive number $R_0(t)$ on day t (this index varies over time as the virus mutates and NPIs are put in place). At t=0, I_0 people are infected, who may belong to any block uniformly at random. Over the days, the infections spread, and on day t, I_{kt} people in block k are infected, R_{kt} persons recover there, D_{kt} persons pass away, etc. This number depends on the social interactions among the people and the NPIs imposed, like total or partial lockdown orders. However, an individual may follow orders with compliance rate γ.

We have an Agent-Based Model f which can simulate the multi-channel spatio-temporal sequence $X = \{X_{kti}\}_{k=1,t=1,i=1}^{K,T,L}$ as $X = f(N_0, R_0, \gamma)$. Index k refers to a block, t refers to a day and channel index i refer to different compartments of the pandemic, such as new positive cases, recoveries and hospitalizations on each day in a particular block. Our aim is to develop a neural network g, which can predict X_{kti} as $X_{kti} = g(\{X_{jt'l}\}_{j=1,t'=t-H,l=1}^{K,t-1,L}, \{R_0(t')\}_{t'=1}^{t-1}, \gamma)$. Here, H is a time horizon or lookback window. Using this neural network, we aim to predict the full sequence X of daily infections. In Section 4 we describe the Agent-Based Model f, and in Section 5 we discuss the emulator neural network g.

4. Agent-Based Simulator

In this section, we focus on the Agent-Based Model f that was discussed above to generate the spatio-temporal sequence of daily infections. This Agent-Based Model is largely based on the work by Suryawanshi et al. (2021). In this model, each individual in the city is considered as an *agent*. Each agent has a set of attributes - his home, family, workplace etc. The agent interacts with other agents in the city and, if infected, spreads the virus to those he comes in contact with probabilities based on the duration and nature of the interaction. The Agent-Based Model has two main components: the city component and the infection component.

4.1. City Component

The city model has specific modules for the city structure, economic activities, transportation, education, healthcare. The city module specifies many kinds of spaces - residential, workplaces, marketplaces etc and also different kinds of facilities and services. The movements and interactions of individuals in these places is simulated. The Economic Module is sub-divided into sectors, which are further divided into sub-sectors. Each sub-sector consists of several Workplaces. Workplaces are locations where agents go for work. Workplaces are mapped to a location within the city, and each employed agent is assigned to a specific Workplace, where they travel regularly and interact with co-workers. Public transport facilities are also represented for simulating the movements of individuals and their interactions in the process. The healthcare module consists of a network of Covid-19 Hospitals, Covid-19 Healthcare centres, and Covid Isolation Centres. Each of these facilities is associated with a location and has a fixed capacity. The education module deals with schools and colleges, each of which is associated with a location and capacity. Each agent representing an individual who is a student is assigned to an

educational institute (school or college), depending on age and residential area.

4.2. Infection Component

This component is based on *health states* of individuals, and how these states can change due to infection by the virus. Each agent consists of a state belonging to a finite state machine, which is a vector of dimension 2. The first dimension represents the virus-related state i.e Healthy (H), Infected (IF) Recovered (R) and Dead (D). The IF state has two sub-parts namely Symptomatic (S) and Asymptomatic (A). We call them Virus-State (S_v). The second dimension represents mobility-related states which are Free (F), Out-of-City (O), Quarantined (Q), Isolated (I) and Hospitalized (HP). We call them Mobility-State (S_m) These two kinds of states are independent of each other, and the final state consists of a concatenated vector of these two.

Regarding the Virus-state, initially, every person is assumed to be healthy (H). When a person in state H comes in contact with a person in state IF, the former gets infected with a certain probability. The viral load (VL) of a newly infected person is assumed to follow a Beta distribution, scaled appropriately. If this load is below a threshold, the person remains asymptomatic. Otherwise, the person undergoes a fixed incubation period after infection, after which they start showing symptoms. The peak infection period is sampled from a Gaussian distribution[1], after which the person may die or start recovering, depending on age and co-morbidities. The time taken to recover is also considered to follow a Gaussian distribution. Infection can spread from one person to another with a certain probability whenever they share the same physical space. This probability is a function of the contact duration and physical distance between them. This is why we have considered the number of work hours and physical gap as attributes of workplaces.

Additionally, we define Compliance Rate γ as the proportion of the population actually following the policy in place. Higher compliance rates mean no two agents meet beyond their daily schedule (family, school, workplace).

4.3. Simulation Methodology

The simulation proceeds by initializing the system, which includes setting up all the agents and their attributes, workplaces, medical facilities etc. This is done by sampling from probability distributions, such that all the attributes are consistent with each other. A small random subset N_0 of the population is infected with the disease as initialization. After that, we begin the simulation for the specified period. At each time step, we sample the movements of each agent according to a stochastic process based on their daily routine, whose distributions are based on the agent's workplace, residence and travel preferences. We track the interactions between pairs of individuals (when they come spatially close during their movements), and in such cases, the infection may take place stochastically as explained above.

The Agent-Based Model produces a spatio-temporal sequence, producing the number of infections, hospitalizations, deaths etc in each block of the city on each day. We can represent the effect of spatial policies like block-wise lockdowns, containment zones, spatial distribution of resources, the effect of competing virus variants, etc. Using the spatial blocks as a grid structure, this spatio-temporal sequence may be visualized as a time-series of multi-channel heatmaps, with the pixel intensity value in each channel representing the value of a statistic (e.g. number of positive tested individuals, number of hospitalized individuals, etc.) or parameter (e.g. the population of the block. One grid heatmap is created after each day of simulation.

5. Emulation Framework

In this section, we discuss the Deep Learning based model to emulate the simulation results generated by the Agent-Based Model. The model takes as input at each time step 1) the simulation parameters (initial number of infections (at $t = 0$), compliance rate γ, and Basic Reproductive Number $R_0(t)$), and 2) the past spatio-temporal heatmap sequence (as described in 4.3). Using this input, the model predicts the subsequent heatmap sequence over the next time steps. The model is trained to minimize the Mean Squared Error (MSE) between the predicted heatmaps (i.e. model output) and the actual heatmaps (i.e. ABM predictions). Using this approach at each time step, a complete prediction of the heatmap sequence is generated by the emulator.

After each time step the model is fed its own predictions back as inputs at the next time step, not the actual values (i.e. ABM outputs). This is important because in the end use-case we will need the emulator to make complete predictions over a long period without having intermediate access to ground-truths to correct previous mistakes.

Here we note that while the ABM was stochastic in nature, the emulator is deterministic. While a stochastic model has its benefits, in this work we only explore if Neural Networks can capture the main signal without the random effects. Although the ABM is stochastic by design, the simulation results we use for emulation are obtained by averaging over several simulation runs, thus smoothing out the random effects.

In this way, the emulator serves as a surrogate of the ABM (see Figure 2).

[1] For Gaussian sampling, we use min(0, Gaussian sample) to ensure the sampled value is positive, and round to the nearest integer for parameters that take only integral values (e.g. population).

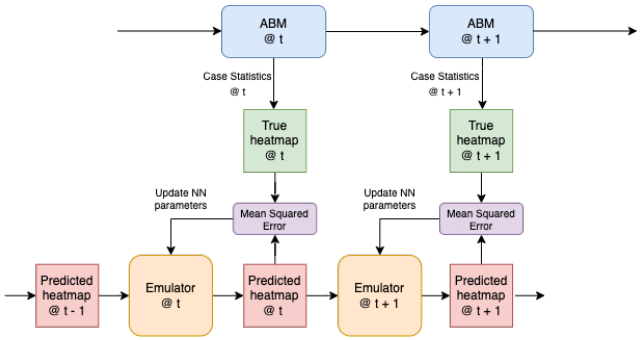

Figure 2. The training procedure of the emulator

5.1. Emulator Architecture

We experimented with several network architectures for the emulator, particularly ones that could leverage both spatial and temporal information given the nature of the heatmap prediction problem. Through our experiments, we demonstrated that a 2D version of the Dilated CNN presented in (Borovykh et al., 2019) performs the best. This architecture, inspired by DeepMind's WaveNet (van den Oord et al., 2016), uses stacks of convolutional layers with increasing dilations that allow it to access data from past time steps. Causal padding is used to ensure no leakage of future data. The following visualization from DeepMind captures this architecture well (3). Other architectures we tried included the original LSTM (Hochreiter & Schmidhuber, 1997), Convolutional LSTM (Shi et al., 2015) and the 3D-CNN (Ge et al., 2017). Complete details about the Dilated CNN architecture and training methodology can be found in Appendix A.2.

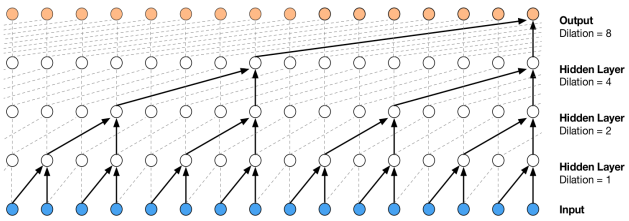

Figure 3. Stacked Dilated CNN (figure from DeepMind)

5.2. Divide-and-Conquer Framework

One of the key benefits of Deep Neural Networks is that modern GPU architectures allow for rapid inference by parallelizing matrix computations in batches. We use this ability to produce spatio-temporal trajectories for large cities using small amounts of training data. As we will demonstrate in 6, the ABM scales linearly with population and the number of city blocks. Instead of training separate emulators for different block sizes, we can use the same emulator by dividing the city into smaller overlapping regions, carrying out the emulation in each of them, and combining them by averaging over the overlapping regions. The emulations of these smaller regions can be done in parallel, and this gives massive time advantage. We call this as the *Divide-and-Conquer* framework for emulation. As an example, we demonstrate how we can simulate a 20x20 *Block* city using a 10x10 emulator in Algorithms 5.2 and 5.2. This procedure allows us to generate spatio-temporal trajectories for a 20x20 city using a 10x10 emulator that is significantly faster than an equivalent ABM, because we can generate all 10x10 predicted spatio-temporal heatmap sequences in parallel (rather than doing them one-by-one as we would have to do if we trained a 20x20 emulator instead).

Algorithm 1 Scaled ABM emulator Training

1: Generate training data using ABM simulations for a big city with blocks arranged as grids (size = (20, 20))
2: Split this big city into regions (sub-grid size = (10, 10), stride = (2, 2))
3: Train a single emulator Model (block size = (10, 10)) to predict the spatio-temporal sequence of heatmaps of each region independently

Algorithm 2 Divide-and-Conquer ABM emulator Inference

1: Use the Simulator to predict the case statistics for the first $H = 5$ days
2: Divide the initial city into regions (sub-grid size = (h, w) = (10, 10), stride = (sh, sw) = (2, 2))
3: Use the learned emulator to predict the spatio-temporal heatmap sequence for the rest of the days for each region parallelly
4: Combine the predictions thus obtained for these regions to recreate the entire city's heatmap predictions. The prediction for each block on any day is the prediction for that day by all regions that overlap with that block

Hence, this Divide-and-Conquer framework can be used to emulate a large city as smaller grids in parallel, allowing for faster inference. Additionally, once an emulator with a particular resolution (e.g. 10x10) has been trained, it can be used to emulate any larger city grid using this framework, e.g. 20x20, 30x30, 40x40, etc. It can also be used on a coarser grid, by coarsening the emulation results. But for a grid of finer resolution, we need to retrain the model.

6. Experimental Validation

Having described the models in detail, we now come to the experimental evaluation. The first step is to generate data by carrying out simulations using the Agent-Based Model. The next step is to use this data to train the deep learning-based emulator model, and test its performance.

6.1. Data Generation

The training data is generated by running the ABM repeatedly. Parameters like the virus *Reproduction Number* (R_0), population N_0, lockdown duration and compliance ration γ are varied to generate distinct series. Since the ABM has a stochastic component, the simulation outputs for a particular parameter configuration vary slightly (about 5% from the mean). To help the emulator learn the true signals and ignore the noise, we generate multiple series for each set of parameter configurations.

We create a city with a population of $N = 100000$ residents, distributed over a 10x10 gridded block structure. The population of each grid is sampled from a normal distribution with a mean of 1000 and a standard deviation of mean/6. To test the ability of the emulator to predict the case trajectory for different parameter values, we vary the R_0 between 1 and 4 in steps of 0.1. For each parameter value we generate several series of length $T = 100$ days. All other parameters of the ABM are kept constant for this experiment. Of all the case-statistics tracked by the ABM, we retain 3 most descriptive ones - *Cumulative Positive Tested*, *Current Hospitalizations* and *Current Asymptomatic Free*. Each heatmap is thus of size (10x10x3), and the length of the heatmap sequence is 100. The total number of such sequences generated for training is 512.

We split these sequences into training, validation and testing sets in the ratio 80:10:10 (410 + 51 + 51), uniformly across all parameter values. Each channel in X is normalized by dividing it by the mean of the channel (mean calculated using only the training set). In order to train the emulator, we rearrange the training data as into (predictor, prediction) tuples, where each predictor is a sequence of H consecutive heatmaps (10x10x3) and the parameters (γ, R_0), and the prediction is the heatmap in the next timestep. We set the lookback window as $H = 5$. The loss function is the Mean Squared Error between predicted and actual heatmaps.

6.2. Accuracy of Emulation

First of all, we compare the time-series of all the statistics as generated by the Agent-Based Model against those predicted by the emulator. Figure 4 shows the plot of the predicted (by emulator) values of the *Cumulative True Cases* (red) compared to the ABM-simulated values (blue) for 4 random values of the *Reproductive Number* (R_0), and we can see the close similarity between the curves in all cases. We obtain similar plots for daily *Hospitalizations* and *Deaths* too (not shown for space constraints). Next, we also examine the spatio-temporal heatmap sequences generated by the ABM and compare them against the predictions by the emulator. In Figure 5 we show the spatio-temporal heatmaps of the daily number of new infections detected in each block of the city, according to the emulator. These heatmaps closely match the heatmaps obtained from the simulator (not shown due to lack of space).

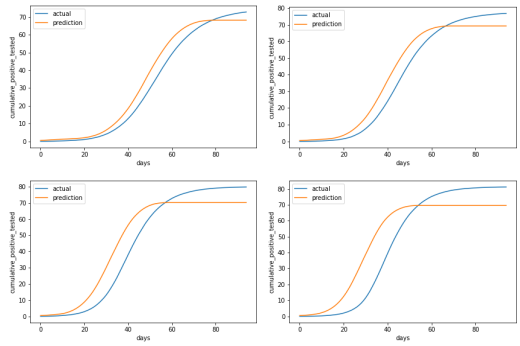

Figure 4. Time-series of Cumulative Daily Infections for different R_0 values: clockwise from top-left: 1.8, 2.0, 2.2, 2.4. Blue: ABM, Red: emulator

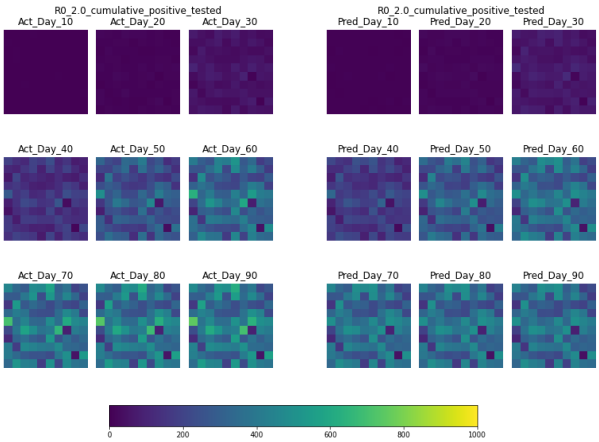

Figure 5. Cumulative Positive Tested heatmaps of the city grid at 10-day intervals for $R_0 = 2$. Left: ABM (Act_Day_X), Right: emulator (Pred_Day_X)

6.3. Generalization

We also wish to explore how the model performs when dealing with unseen parameter values. Figure 9 in the appendix shows the performance of emulating the ABM for unseen values of R_0 in-between the values it was trained on (the model was trained on values of R_0 between 1 and 4, in steps of 0.2). We observe that, as expected, it is able to perform well for these values of R_0. On the other hand, Figure 10 in the appendix shows that the model is also able to emulate the ABM for unseen values of R_0 outside the range of R_0 values it was trained on. We observe that while it is able to perform well for larger values of R_0, the emulator struggles for smaller values. This might be explained by the unstable behaviour of the ABM for smaller values of R_0, since for very low R_0, the infection may subside very quickly.

Table 1. COMPUTATIONAL PERFORMANCE OF THE SIMULATOR AND THE EMULATOR IN SCALED INFERENCE. BOTH THE SIMULATION TIME AND THE EMULATOR INFERENCE TIMES ARE THE TIMES REQUIRED TO GENERATE COMPLETE TRAJECTORIES. ALL TIMES ARE MEASURED IN SECONDS.

TASK	SIMULATOR TIME
20X20 CITY-GRID SIMULATION TIME	38.0626 SEC
TASK	**EMULATOR**
EMULATOR TRAINING	503.93 SEC
INFERENCE (10X10)	0.75 SEC
SCALED INFERENCE (20X20)	0.76 SEC

6.4. Computational Benefits

The divide-and-conquer procedure allows us to generate spatio-temporal trajectories for a 20x20 city using a 10x10 emulator that is significantly faster than an equivalent ABM, because we can generate all 10x10 predicted spatio-temporal heatmap trajectories in parallel (rather than doing them one-by-one as we would have to do if we trained a 20x20 emulator instead). We show the time gain in Table 1, and the predicted spatio-temporal heatmap trajectory in the subsequent figures. Note that for n simulation runs on a 20×20 city, the Agent-Based Model will require $38n$ seconds, while the emulation will require $504 + 0.76n$ seconds, which gives a major gain especially for high values of n. This is illustrated in Figure 11 in the Appendix. Further, the Agent-Based Model also scales linearly in the number of agents, i.e. the city population N. In the next analysis, we vary N, keeping the grid size fixed at 10x10, and observe how the emulator compares with the simulator. Figure 6 shows that the emulator inference time remains nearly constant in all cases.

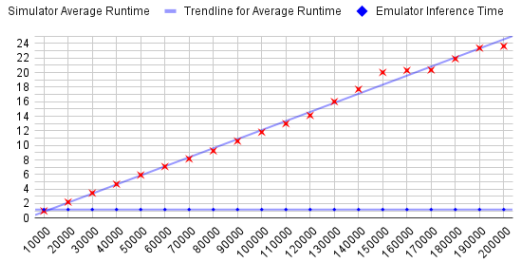

Figure 6. Time Analysis: Average Runtime vs. Population N for simulator (red) and emulator (blue)

7. Applications of Emulation

The computational benefits obtained by the process of emulation may be used in situations where hundreds of simulation runs are required. Carrying out such runs using the computationally-expensive Agent-Based Model is highly time-consuming. Although training the emulator model takes time, it is a one-time process. The inference, which is very fast, needs to be carried out repeatedly which gives a major computational benefit compared to the Agent-Based Model. Below, we consider two situations where multiple simulation runs are needed, and demonstrate how the emulator scores over the simulator in such situations.

Table 2. COMPARISON OF PERFORMANCES BY SIMULATOR AND EMULATOR TO CALIBRATE AGAINST OBSERVATIONS BY ESTIMATING R_0 VALUE THROUGH BAYESIAN OPTIMIZATION

	AVERAGE ERROR	AVERAGE TIME TAKEN
SIMULATOR	15.82%	881 SEC
EMULATOR	25.27%	33.9 SEC

7.1. Parameter Tuning

In practical applications of epidemic ABMs, we often have historical observations of some statistics, e.g. the number of new cases, hospitalizations, recoveries, deaths, etc., but we do have any information about others, e.g. the virus R_0 or γ. Suitable values of these statistics are usually chosen by calibrating them based on the available historical data, using parameter search techniques like grid search, Bayesian Optimization, rejection sampling, etc. In this experiment, we demonstrate how the emulator can be used to speed-up this parameter search.

We design an experiment where Bayesian Optimization is used to estimate the value of the virus R_0 from given historical data. A randomly sampled value of R_0 (uniformly sampled from the range [2.0, 3.0]) is used to produce one complete ABM simulation. Then, with this ground-truth value hidden, we use the emulator to line-search for this R_0 in a Bayesian Optimization paradigm. For each value of R_0 to be tested, a complete heatmap sequence is predicted by the emulator. The Mean Squared Error (MSE) of this predicted sequence vs. the actual heatmap sequence (i.e. the original ABM simulation) is used as the objective to be minimized by Bayesian Optimization. This process is then repeated using the ABM instead of the emulator to compare the time required.

This experiment is repeated for 20 distinct sampled values of R_0. Table 7.1 compares the results when we use the emulator vs. the ABM (i.e. the simulator). We find that the R_0 searched by the ABM is slightly more accurate than by the emulator on average, but the latter provides a very significant time gain. Further, the emulator can make a better estimate of the R_0 in 40% of the values tested.

7.2. Scenario Analysis

Another application in which a large number of simulation runs are usually needed is when we want to explore alternate intervention policies or counterfactual scenarios for *what-if* analysis. In the context of a pandemic, we can consider the possible impacts of alternate non-pharmaceutical interven-

tions like localized lock-downs and restricted use of public spaces, as discussed in Suryawanshi et al. (2021).

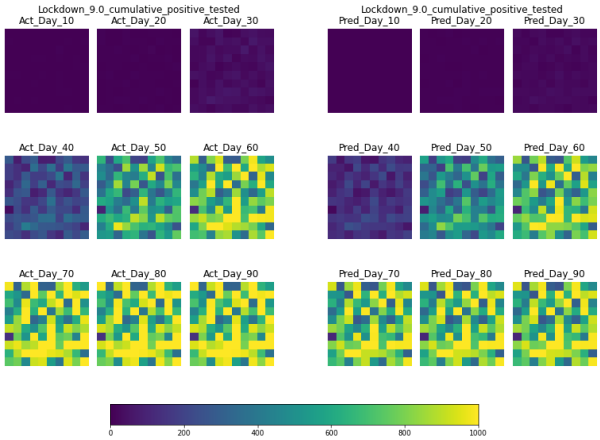

Figure 7. Scenario Analysis: Heatmaps for number of Cumulative Positive Tested persons, if there is a lockdown of 9 days from the 10th day. Left: observations from ABM simulator (actual values, Act_Day_X), Right: predictions by emulator (Pred_Day_X)

In this experiment, we demonstrate that our emulator can learn the effect of lockdowns on the spatio-temporal case trajectory. For this experiment, the lockdown status of the city (i.e., whether the city is currently in a state of lockdown or not) at each time-step is passed to the emulator as an additional parameter channel. We consider lockdowns of various lengths in the range [3, 45], starting at a randomly chosen time-step near the start of the simulation (day 10), and observe the predictions of the emulator. We find that the emulator is still able to predict the spatio-temporal case trajectories accurately. In Figure 7. It can be observed from the predicted trajectories that the emulator can learn the delay in rise of cases due to lockdowns quite accurately.

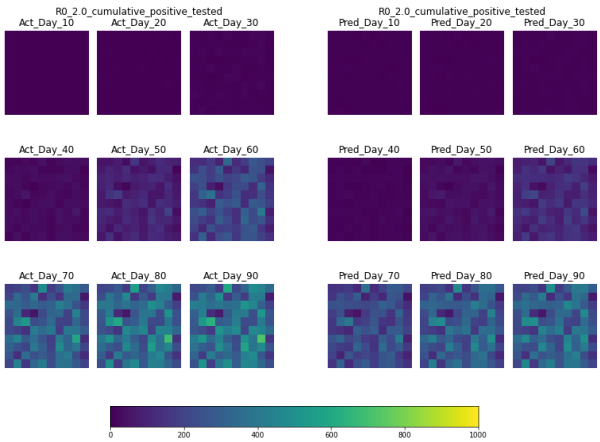

Figure 8. Effects of varying R_0 linearly from 1.0 till 2.0 and then back to 1.0. Left: observations from ABM simulator (actual values, Act_Day_X), Right: predictions by emulator (Pred_Day_X)

The basic reproductive number R_0 does not stay constant during a pandemic. With increasing levels of immunity, either due to vaccination or infection, R_0 tends to decrease, while it may increase in the presence of multiple variants of the virus as it mutates. We want to see if the emulator can adapt to such changes in the R_0 time-series. Towards this end, we vary the R_0 through the simulation and observe the predictions of the emulator. The value of R_0 on each time-step of the simulation is passed as a feature to the emulator as another parameter channel. We demonstrate a simple trajectory for R_0 here as a proof-of-concept; it starts at a low initial value of 1, then linearly increases to a maximum value near the mid-point of the simulation, then again linearly decreases to the initial value. In addition to varying R_0 throughout the simulation, we also vary the maximum value (and hence the slope on either side) of the R_0 trajectory to observe how well the emulator can learn these effects. From the results shown in Figure 8, we observe that the emulator can predict accurate trajectories for a wide range of maximum R0s.

8. Conclusion

This paper introduced a deep learning-based emulator for Agent-Based Models for pandemic spread in a city. The proposed model can almost perfectly reproduce the Agent-Based Model's outputs in a spatially explicit way, taking into account dynamically varying parameters, at only a small fraction of time. This approach is particularly suitable when hundreds of simulation runs are required.

References

Agrawal, M., Kanitkar, M., and Vidyasagar, M. Sutra: A novel approach to modelling pandemics with applications to covid-19. *arXiv preprint arXiv:2101.09158*, 2021.

Angione, C., Silverman, E., and Yaneske, E. Using machine learning to emulate agent-based simulations. *CoRR*, abs/2005.02077, 2020. URL https://arxiv.org/abs/2005.02077.

Angione, C., Silverman, E., and Yaneske, E. Using machine learning as a surrogate model for agent-based simulations. *PLOS ONE*, 17(2):1–24, 02 2022. doi: 10.1371/journal.pone.0263150. URL https://doi.org/10.1371/journal.pone.0263150.

Anirudh, R., Thiagarajan, J. J., Bremer, P., Germann, T. C., Valle, S. Y. D., and Streitz, F. H. Accurate calibration of agent-based epidemiological models with neural network surrogates. *CoRR*, abs/2010.06558, 2020. URL https://arxiv.org/abs/2010.06558.

Batty, M. Fifty years of urban modeling: Macro-statics

to micro-dynamics. In *The dynamics of complex urban systems*, pp. 1–20. Springer, 2008.

Bharath, A. H., Vinay, S., and Ramachandra, T. Agent based modelling urban dynamics of bhopal, india. *Journal of settlements and spatial planning*, 7(1):1–14, 2016.

Bonabeau, E. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl 3):7280–7287, 2002.

Borovykh, A., Bohte, S., and Oosterlee, C. Dilated convolutional neural networks for time series forecasting. *The Journal of Computational Finance*, 22(4):73–101, February 2019. ISSN 1460-1559. doi: 10.21314/JCF.2019.358.

Brauer, F., Castillo-Chavez, C., and Castillo-Chavez, C. *Mathematical models in population biology and epidemiology*, volume 2. Springer, 2012.

Epstein, J. M. and Axtell, R. *Growing artificial societies: social science from the bottom up*. Brookings Institution Press, 1996.

Fosset, P., Banos, A., Beck, E., Chardonnel, S., Lang, C., Marilleau, N., Piombini, A., Leysens, T., Conesa, A., Andre-Poyaud, I., et al. Exploring intra-urban accessibility and impacts of pollution policies with an agent-based simulation platform: Gamirod. *Systems*, 4(1):5, 2016.

Ge, L., Liang, H., Yuan, J., and Thalmann, D. 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S. K., Huse, G., et al. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*, 198(1-2):115–126, 2006.

Hager, K., Rauh, J., and Rid, W. Agent-based modeling of traffic behavior in growing metropolitan areas. *Transportation Research Procedia*, 10:306–315, 2015.

Heppenstall, A., Malleson, N., and Crooks, A. "space, the final frontier": How good are agent-based models at simulating individuals and space in cities? *Systems*, 4(1): 9, 2016.

Hinch, R., Probert, W. J., Nurtay, A., Kendall, M., Wymatt, C., Hall, M., Lythgoe, K., Cruz, A. B., Zhao, L., Stewart, A., et al. Openabm-covid19-an agent-based model for non-pharmaceutical interventions against covid-19 including contact tracing. *medRxiv*, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Hoertel, N., Blachier, M., Blanco, C., Olfson, M., Massetti, M., Rico, M. S., Limosin, F., and Leleu, H. A stochastic agent-based model of the sars-cov-2 epidemic in france. *Nature medicine*, 26(9):1417–1421, 2020.

Horni, A., Nagel, K., and Axhausen, K. W. *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016.

Huynh, N., Perez, P., Berryman, M., and Barthélemy, J. Simulating transport and land use interdependencies for strategic urban planning—an agent based modelling approach. *Systems*, 3(4):177–210, 2015.

Jäger, G. Using neural networks for a universal framework for agent-based models. *Mathematical and Computer Modelling of Dynamical Systems*, 27(1):162–178, 2021. doi: 10.1080/13873954.2021.1889609. URL https://doi.org/10.1080/13873954.2021.1889609.

Kagho, G. O., Balac, M., and Axhausen, K. W. Agent-based models in transport planning: Current state, issues, and expectations. *Procedia Computer Science*, 170:726–732, 2020.

Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Hart, G., Rosenfeld, K., Selvaraj, P., Nunez, R. C., Hagedorn, B., George, L., et al. Covasim: an agent-based model of covid-19 dynamics and interventions. *medRxiv*, 2020.

Motieyan, H. and Mesgari, M. S. An agent-based modeling approach for sustainable urban planning from land use and public transit perspectives. *Cities*, 81:91–100, 2018.

O'Sullivan, D. and Haklay, M. Agent-based models and individualism: is the world agent-based? *Environment and Planning A*, 32(8):1409–1425, 2000.

Patel, A., Crooks, A., and Koizumi, N. Spatial agent-based modeling to explore slum formation dynamics in ahmedabad, india. In *GeoComputational analysis and modeling of regional systems*, pp. 121–141. Springer, 2018.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Silva, P. C., Batista, P. V., Lima, H. S., Alves, M. A., Guimarães, F. G., and Silva, R. C. Covid-abs: An agent-based model of covid-19 epidemic to simulate health and economic effects of social distancing interventions. *Chaos, Solitons & Fractals*, 139:110088, 2020.

Suryawanshi, G., Madhavan, V., Mitra, A., and Chakrabarti, P. P. City-scale simulation of covid-19 pandemic and intervention policies using agent-based modelling. *CoRR*, abs/2104.01650, 2021. URL https://arxiv.org/abs/2104.01650.

Tesfatsion, L. Agent-based computational economics: modeling economies as complex adaptive systems. *Information Sciences*, 149(4):262–268, 2003.

Truszkowska, A., Behring, B., Hasanyan, J., Zino, L., Butail, S., Caroppo, E., Jiang, Z.-P., Rizzo, A., and Porfiri, M. High-resolution agent-based modeling of covid-19 spreading in a small town. *Advanced Theory and Simulations*, 4(3):200277, 2021.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL http://arxiv.org/abs/1609.03499.

Waddell, P. Urbansim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American planning association*, 68(3):297–314, 2002.

Waddell, P., Borning, A., Noth, M., Freier, N., Becke, M., and Ulfarsson, G. Microsimulation of urban development and location choices: Design and implementation of urbansim. *Networks and spatial economics*, 3(1):43–67, 2003.

Zhang, W., Valencia, A., and Chang, N.-B. Synergistic integration between machine learning and agent-based modeling: A multidisciplinary review. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Zhang, Y., Grignard, A., Lyons, K., Aubuchon, A., and Larson, K. Real-time machine learning prediction of an agent-based model for urban decision-making (extended abstract). In *AAMAS*, 2018.

A. Appendix

A.1. Appendix A: Interpolation and Extrapolation

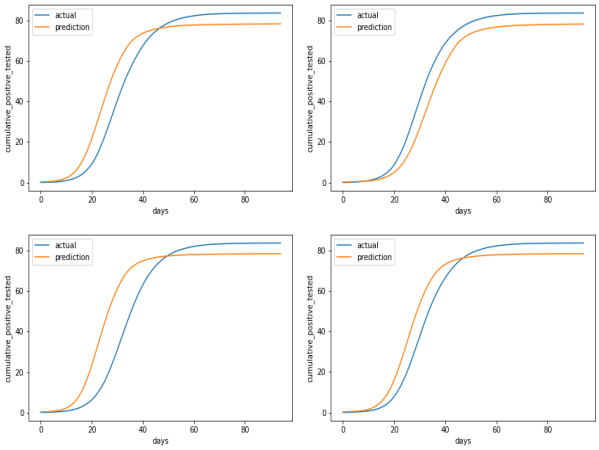

Figure 9. Temporal Interpolation: Cumulative True Cases vs. Days for different unseen R_0 values inside the training range: clockwise from top-left: 2.65, 2.75, 2.85, 2.95

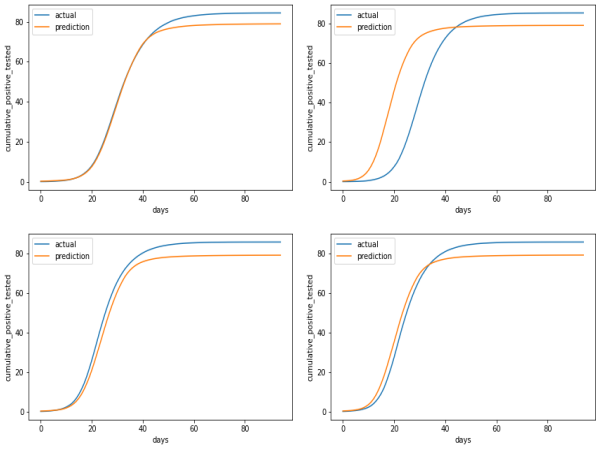

Figure 10. Temporal Extrapolation: Cumulative True Cases vs. Days for different unseen R_0 values outside the training range: clockwise from top-left: 3.05, 3.15, 3.25, 3.35

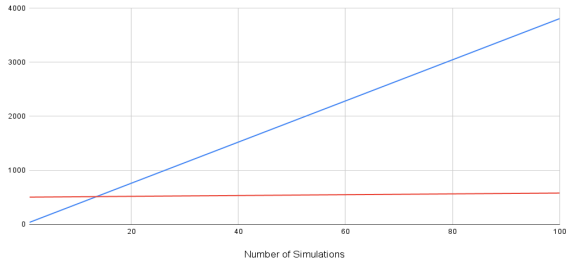

Figure 11. The time required in seconds (along Y-axis) for emulation (red), including the initial training time, increases negligibly with more simulation runs, and this makes it highly efficient compared to the agent-based simulator (blue)

A.2. Appendix B: Emulator Implementation

A.2.1. Generating Training Data

The training data is generated by running the ABM repeatedly. Parameters like the virus *Reproduction Number (R0)*, population, and lockdown duration are varied to generate distinct series. Since the ABM has an inherent stochastic component, the outputs for a particular parameter configuration may vary slightly (about 5% from the mean). To help the Emulator learn the true signals and ignore the noise, we generate multiple series for each set of parameter configurations as well.

While the parameters for each experiment are different, for the baseline we create a city with a population of 100000 residents, distributed over a 10x10 city grid. The population of each grid is sampled from a normal distribution with a mean of 1000 and a standard deviation of mean/6. To test the ability of the Emulator to predict the case trajectory for different parameter values, we vary the R0 between 1 and 2.5 in steps of 0.1 (i.e. 16 distinct values). For each value of *R0*, we generate 32 distinct series of 100 days each. . All other parameters of the ABM are kept constant for this experiment. Of all the case-statistics tracked by the ABM, we retain 3 most descriptive ones - *Cumulative Positive Tested*, *Current Hospitalizations* and *Current Asymptomatic Free*. We additionally add $R0$ and block population as parameter channels to the heatmaps to serve as features for the Emulator's predictions, making for a total of 5 channels. The total number of heatmap sequences is hence 512 (16*32), each of size [100, 10, 10, 5].

A.2.2. Data Preprocessing

We split the development set of 512 sequences into training, validation and testing sets in the ratio 80:10:10 (410 + 51 + 51), uniformly across all parameter values. Each channel is normalized by dividing it by the mean of the channel (mean calculated using only the training set).

A.2.3. Emulator Architecture

The Dilated CNN architecture, inspired by DeepMind's WaveNet (van den Oord et al., 2016), uses stacks of convolutional layers with increasing dilations that allow it to access data from past time steps. Causal padding is used to ensure no leakage of future data. The figure below shows the model architecture, including the number of kernels, kernel size, feed-forward dimensions, etc.

A.2.4. Training the Emulator

In order to train the Emulator, we need to rearrange the series in a (lagged heatmaps, future heatmaps) format. The Emulator will be trained to take the lagged heatmaps as input and predict future heatmaps as close as possible to the actual future heatmaps (loss function - MSE between predicted and actual future heatmaps). From each simulated series, we create ordered tuples (lagged heatmaps, future heatmaps) of shape ([*lag*, 10, 10, 5], [*horizon*, 10, 10, 3]). Note that we do not require the Emulator to produce outputs for the parameter channels. For the baseline experiments, we set *lag* to 5 and *horizon* to 1. The emulator was implemented in Python using Tensorflow 2.0. All computation times were recorded on free instances of Google Colaboratory.

A.2.5. Generating Predictions using the Emulator

Once the Emulator is trained, we generate full length predictions as follows -

1. Start with an initial lag sequence of shape [*lag*, city grid rows, city grid cols, num case statistics + num parameters]

2. While generated sequence length ¡ simulation length, do

   (a) Pass lag sequence through Emulator to get an output of shape [*lag*, city grid rows, city grid cols, num case statistics], and append it to the prediction sequence

   (b) Rotate lag sequence to the left by *horizon* time steps

   (c) Replace the first *horizon* time steps from the right of the lag sequence with the output tensor, effectively replacing the earliest *horizon* days with the newest *horizon* days

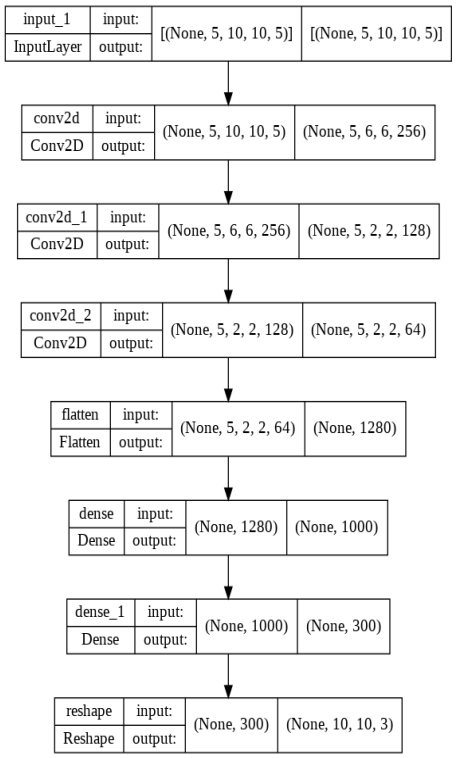

Figure 12. Stacked Dilated CNN Architecture

While the algorithm described above predicts a single series, in actual implementation we leverage the batching abilities of GPUs to predict several series at once.

A.3. Appendix C: Comparison against Emulator Baselines

In this section, we compare our emulator with Angione et al. (2022), another proposed NN-based surrogate for ABMs. Angione et al. (2022) uses a feed-forward network based approach, but lacks the spatial component in our paper. The network takes as input the historical values of the case statistics (aggregated across all blocks), and predicts the subsequent values. Like our emulator, it is trained to minimize the Mean Squared Error between the predictions and the actual case statistics (i.e. ABM outputs). However the predictions are temporal only, with no spatial distribution across blocks. In Figure 13, we observe that our emulator is better at predicting the case statistics temporally.

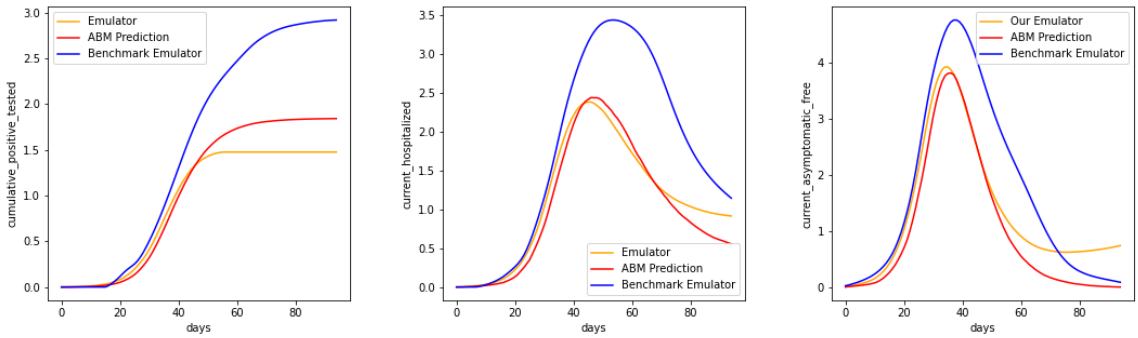

Figure 13. Comparison against baselines for cumulative positive tested, current hospitalized and current asymptomatic free respectively.