Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

MODELING OF AGENT DECISIONS USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

Martin Bicher[1], Dominik Brunmeir[1], and Niki Popper[1,2]

[1]Institute for Information Systems Engineering, TU Wien, Vienna, AUSTRIA
[2]dwh simulation services, dwh GmbH, Vienna, AUSTRIA

ABSTRACT

In this paper, we investigate the use of Generative Adversarial Networks (GAN) to model agent behavior in agent-based models. We hereby focus on use cases in which an agent's decision-making process may only be modeled from data, but it is infeasible to be modeled causally. In these situations, meta-models are often the only way to quantitatively parameterize the agent-based model. However, methods that capture not only deterministic relationships but also stochastic uncertainty are rare. Since GANs are well known for their property to generate pseudo-random-numbers for complex distributions, we explore pros and cons of this strategy for modeling a delay-process in a large-scale agent-based SARS-CoV-2 simulation model.

1 INTRODUCTION

Agent-based simulation is a modeling paradigm that is based on defining rules, causalities and behaviors primarily not at the global level but at the individual level. The precise modeling of human behavior requires highly complex model processes and a large number of parameters, which require feasible values, many of which must be estimated or calibrated. The validation or calibration reference of an agent-based model (ABM) at the macro level is usually significantly lower-dimensional, but this often leads to unavoidable overfitting during the parametrization or calibration process. As a result it is crucial in such situations, to reduce the structural complexity of the micro-level and to include as much micro-data as possible to reduce the number of free parameters. In this work we want to present a concept which is, under the right prerequisites, capable of performing both tasks at the same time.

From the modeling perspective we would interpret an agent decision as a mapping

$$f : (\vec{u}, X) :\mapsto f(\vec{u}, X) =: y \tag{1}$$

which maps a vector of observations or states of the agent $\vec{u} = (u_1, \ldots, u_n)^T$ and an additional source of uncertainty X onto a resulting decision y. Note, that in our computer scientific interpretation we consider any model process on the agent level as a "decision" independent of whether the modeled real-system process is an actual decision in the social scientific sense. This choice is reasonable in this case since the methodology for simulation is identical.

Moreover, we assume that there exists a sample S of data about this decision process in form of N individual records:

$$S := (\vec{u}_k, y_k)_{k=1}^{N} . \tag{2}$$

In a strictly causal approach, this mapping would be modeled explicitly and every contributing component $u_1 \ldots, u_n$ is included in the model. Valid causal hypotheses for the relations between the components and the uncertainty source must be defined and quantified in the course of an input-modeling step (Biller and Gunes 2010) so one is able to compute the output y. However, it is often the case that corresponding processes are not directly responsible for the fundamental dynamics of the model, not within the focus of the research question, or simply not well enough understood. Thus, the modeling of the causal relationships

moves into the background, while the correct mapping of the input-output relationships $(\vec{u}, X) \mapsto y$ comes to the fore. We argue that in such circumstances, using a meta-model becomes a valid alternative.

There are various approaches to implement meta-models for certain models parts. Most typical approaches like regression or interpolation models (Pietzsch et al. 2020) do not include randomness and aim to improve performance e.g. for optimization (Barton 2009). Alternatives exist with respect to classification combined with bootstrapping, Markov-Chain Monte Carlo, or kernel-density estimation, but to our knowledge, there is surprisingly little research done for this specific application of meta-models.

In this work we propose a meta-model in form of a classic multi-layer perception (MLP), which is trained as the generator in a generative-adversarial-network (GAN) framework – to be specific, conditional GAN (condGAN, see Mirza and Osindero (2014)) – using suitable micro-data. GANs have a rich history of posing as pseudo random number generators (PRNGs) for various distributions (De Bernardi et al. 2019; Oak et al. 2019; Cai et al. 2021). In the condGAN strategy the input noise of the generator is supplemented by a feature vector, which acts as conditions to the probability distribution.

The general idea of using neural networks (NNs) in deterministic agent-based decision processes is not new and was introduced in Jäger (2019), and van der Hoog (2019). In Ghaffarzadegan et al. (2024), authors even introduced the term "Generative ABM" for ABMs using generative AI. Yet, GANs have, to the author's knowledge, never been used for our proposed use-case. In Vallecorsa et al. (2019) and Sarrut et al. (2019) a GAN is trained in an agent-based context, but in the role of replacing the simulation models as a whole.

Using a NN for sampling of stochastic decision processes in simulations comes with the following benefits:

- Like classic input-modeling, fitting and evaluation are two independent processes. Network training is only done once in a data pre-processing step.
- While the training process is time intensive, evaluation of the NN is fast and does not require a lot of memory. This perfectly supplements use in agent-based simulations which often suffer from long computation times themselves.
- The modeler has complete freedom to decide, which of the observables $u_1, \ldots, u_n$ should be included in the model, as long as they are depicted in the micro data set. Causal knowledge about the influences, though, decides about the training success.

We found the method to be well suited for our problem as the convergence of the GAN game is not based on some complicated statistical error function, but is founded solely on the output of the critic network.

2 METHODS

2.1 GAN Setup

We use a condGAN with a generator and a critic network of sufficient size and structure. Key for the strategy is to specify input and outputs of the networks properly to make it useful in the simulation model (see Figure 1).

The generator network aims to imitate function f. The $n+1$ dimensional input layer is fed a one-dimensional random-variable and the observation vector $\vec{u} := u_1, \ldots, u_n$. As our outcome variable y, and therefore also the output layer of the generator, henceforth $G(\vec{u}, X)$, is scalar, a one-dimensional random-variable is sufficient for our purpose. For optimal performance the noise is generated from a uniform distribution $U(0,1)$.

The critic network is trained to properly score generated data low and real data high. The input-layer is also $n+1$-dimensional, as are the data-records. The synthetic data to compare it with is given by $(\vec{u}, G(\vec{u}, X))$, whereas $\vec{u}$ is taken from the given observations.

Training of GANs is highly dependent on the used loss functions for generator and critic to guarantee stability. Hereby, the concept of Wasserstein GANs (Arjovsky et al. 2017) is well suited. For this work

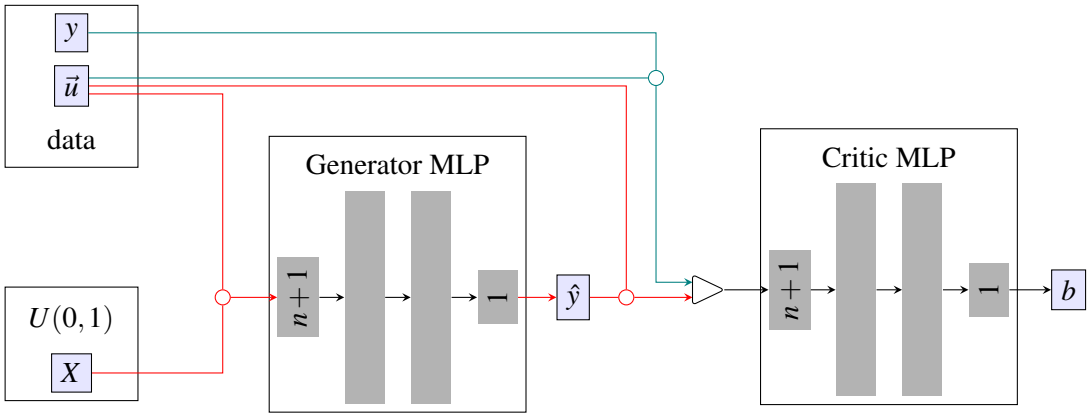

Figure 1: GAN structure. Generator and critic use an $n+1$-dimensional input: the generator is fed with the n observations and a 1 dimensional randomness source, the critic is fed with observation-value pairs to distinguish whether they come from data (green path) or are generated (red path). Circles indicate merging, triangles appending of two data stream.

we found that the gradient penalty method presented in Arjovsky et al. (2017) gives the best results. Summarizing, the used setup can be described as conditional Wasserstein GAN with gradient penalty (condWGAN-GP).

2.2 Convergence Tracking

One of the problems in any GAN game is to find out, when training can be stopped, since the loss functions only indicate, how balanced the game is, but not how good the players are. A different approach, based on the generated samples is needed.

Surprisingly, it is no simple statistical task to evaluate, whether two samples

$$S = (\vec{u}_i, y_i)_{i=1}^{N}, \quad y_i \sim D_{\vec{u}_i} \tag{3}$$

$$\hat{S} = (\vec{u}_i, \hat{y}_i)_{i=1}^{\hat{N}}, \quad \hat{y}_i \sim \hat{D}_{\vec{u}_i} \tag{4}$$

actually come from the same / similar distribution, i.e. $D \overset{d}{=} \hat{D}$. Approaches like Kolmogorov-Smirnoff test, T-test, etc. are not specified for a parametric distribution.

Suppose the expected values $E(\exp(ty_i) - \exp(t\hat{y}_i))$ for all t vanish, then

$$0 = E(\exp(ty_i) - \exp(t\hat{y}_i)) = E_{D_{\vec{u}_i}}(\exp(ty_i)) - E_{\hat{D}_{\vec{u}_i}}(\exp(t\hat{y}_i)),$$

which would imply that the moment generating functions are equivalent and furthermore that the distributions are equivalent.

Suppose the expected values $E(y_i^k - \hat{y}_i^k)$ for all k vanish, then

$$0 = E(y_i^k - \hat{y}_i^k) = E(y_i^k) - E(\hat{y}_i^k),$$

which would imply that the crude and in direct consequence also the centered moments (mean, variance, skewness,...) of the distributions match. While this does not prove the equivalence of the distributions (compare with the counterexample in Feller (1991), p227), it still would determine the similarity of the most important metrics of the distribution.

Since we aim, that $X \sim U(0,1) \Rightarrow G(\vec{u}_i, X) \sim D_{\vec{u}_i}$ we define the error

$$\varepsilon_k(G) := \frac{1}{N} \left| \sum_{i=1}^{N} y_i^k - G(\vec{u}_i, X_i)^k \right| \tag{5}$$

with N independent $U(0,1)$ samples X_i, which allows us to investigate the convergence progress of the k-th moment of the Generator's distribution. Since all moments should vanish equally, we may define the error

$$\varepsilon_{1,k}(G) := \sum_{j=1}^{k} \mu_j \varepsilon_j(G) \tag{6}$$

as a measure of convergence of the first k moments for positive weights μ_k. Note, that in case the output of the generator is normed to $[0,1]$, setting all weights $\mu_j = 1$ is justified, since all moments lie on the same scale.

2.3 Simulation Model

In our case study we integrate a generator network into an existing SEIR-type ABM (susceptible–exposed–infectious–recovered, see Brauer (2008)) for Austria. The model was first published in the supplemental material in Bicher et al. (2021), the version we apply in this work is described in the corresponding documentation *Covid19_Model-20230322.pdf* which can be found publicly on https://github.com/dwhGmbH/condgan-for-abm. The model was extensively used during the first years of the COVID-19 pandemic for decision support and various research problems including forecasts (Bicher et al. 2022), policy evaluation (Bicher et al. 2021), detection rate analysis (Bicher et al. 2022), and vaccine prioritization (Bicher et al. 2022).

In the model, every agent has various socio-demographic and spatial features and is assigned contact locations (households, school classes, workplaces, care homes) where it are able to meet other agents. In case of a contact between susceptible and infectious agents, an infection may occur, depending on epidemiological factors, such as virus strain, seasonality, location, shedding, and adherence. Infected agents then follow a disease progression pathway including relevant events from infection to immunity loss: start of infectiousness, symptom onset, recovery or death, start of immunity to immunity loss.

Focus of the present work is the duration between symptom onset and positive test result, furthermore called *reactionTime*. This time delay is highly relevant for the spread of the disease, since the agent is highly infectious but not yet identified and isolated during this period. Goal of the present work is to improve (in terms of validity) how an agent computes this time-span as soon as its symptom-onset event triggers. Although the *reactionTime* only partially refers to something which the modeled human can actually decide in the real system (adherence), we would still refer to this process as a decision in the computer scientific sense – the agent decides for how much time passes between symptom-onset and test result.

In the originally stated version of the model, this time duration is sampled as a Weibull distributed random number. Scale λ and shape k parameters of the distribution are fixed for predefined time-intervals (see below, Experiment 0). While this strategy might fit the data quite well, it does not regard any influence of other factors besides time such as the the current number of cases or the associated (over-)utilization on the testing and reporting system. This poses a major inaccuracy for what-if-scenarios.

2.4 Data

We will use data from the Austrian Epidemiological Reporting System (EMS) as the basis for all scenarios. In the period between 2020 and 2023, in which COVID-19 had mandatory registration in Austria, every positive case tested was entered in this register. For a significant number of cases, between 40 and 80%, depending on the period, there is also an estimated date of illness (onset of symptoms) in addition to the official reporting date of the case. The collection process of this estimate is rather opaque and we may assume that the corresponding data quality is comparably low and biased.

Causal knowledge about the system supported by correlation tests (Pearson) indicate, that the current date and the current incidence are the most important co-variates for the *reactionTime* in the EMS data-set. We define:

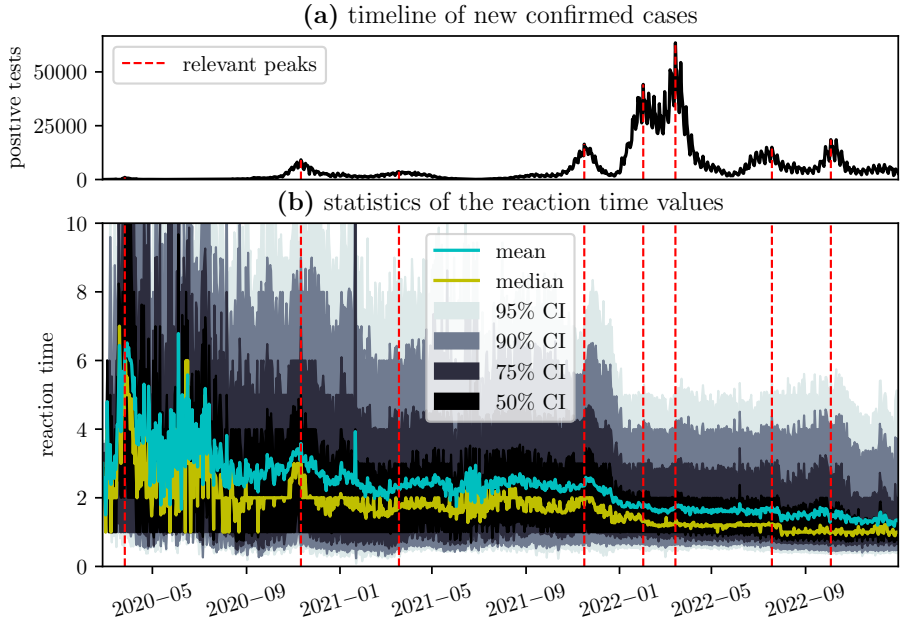

Figure 2: Figures (a) and (b) put the *cases* and the *date* variables in contrast with the *reactionTime*. While the overall trend goes towards smaller reaction times in the course of the pandemic, there are noticeable increases in times of large waves, in particular in Spring 2020, Autumn 2020 and Autumn 2021.

- *reactionTime* [days]: Duration between the time of estimated symptom-onset and registration time in the reporting system.
- *cases* [persons]: Total number of new confirmed positive cases on the day before the case was registered in the reporting system. This one-day difference is relevant to make the variable measurable in the simulation model.
- *date* [days]: Duration between the registration time in the reporting system and a fixed reference point in the past, furthermore specified by 2020-01-01. This strategy is necessary to make the registration time a numerical variable.

With these definitions our reference data-set is given by $N = 3,990,925$ entries with

$$S_{ems} = ((cases_i, date_i), reactionTime_i)_{i=1}^{N} \tag{7}$$

Hereby the sample corresponds to the total number of positive SARS-CoV-2 tests with registration date between 2020-01-01 and 2022-12-31, a non-empty estimate for the time of symptom onset, and with a feasible value (maximum of 20 days) for the reaction time.

We find

$$cases \in [0,65000], \quad date \in [0,1200], \quad reactionTime \in [0,20]. \tag{8}$$

We henceforth denote the product space of the first two intervals as parameter-space and the last interval as value-space. The values are furthermore used for normalization. Note, that the maximum 1200 for *date* instead of 1096 was kept for historic reasons.

Figure 2 shows the development of the reaction time in the course of the pandemic (mean value and confidence intervals). As the reaction times became shorter, we may assume a systemic change which may be caused by either a better equipped testing system, a better functioning reporting system, a higher adherence and test-awareness, or all named reasons at once.

2.5 Experimental Setup

We will conduct four experiments. Experiment 0 will generate a reference for the simulation using the ABM without any modification. Experiments 1 and 2 will be used to check the validity of the proposed GAN concept using synthetic data with a known distribution. Finally, experiment 3 will show if the GAN concept can also be applied to the actual data. All experiments will found on the same simulation setup which we describe together with Experiment 0 below.

We will evaluate the experiments in two ways: First, we compare the a priori distributions to evaluate the differences between the different methods to generate random numbers. In a second step, we compare the results of the agent-based simulations to analyze the differences between the methods when used as a sampler for the decision process in the ABM.

Experiment 0: Simulation Status-Quo As baseline reference, we use the simulation with its original static reaction-time sampling and calibrated to the COVID-19 pandemic in Austria between 2020-01-01 and 2023-01-01. The corresponding parameters and calibration strategy are found publicly in the documentation *Calibration_Feb2023.pdf* on https://github.com/dwhGmbH/condgan-for-abm. It can be regarded as a full retrospective of the three years and is capable for what-would-have-been scenarios.

For the present work, the most important feature is that the reaction time is modeled using a Weibull distribution for which the parameters are simple step-functions of time:

$$reactionTime \sim \begin{cases} Weib(k=1.33, \lambda=5.66), & t \in [2020-01-01, 2020-06-01) \\ Weib(k=1.30, \lambda=2.90), & t \in [2020-06-01, 2021-09-01) \\ Weib(k=1.43, \lambda=2.59), & t \in [2021-09-01, 2021-11-01) \\ Weib(k=1.53, \lambda=1.99), & t \in [2021-11-01, 2023-01-01) \end{cases} \tag{9}$$

Experiment 1: Analytic Reference. In our first attempt to improve the model we sample *reactionTime* by drawing from a parameterized Weibull distribution for which the two parameters are a function of the current incidence and simulation time. We setup the probability model

$$Weib(k,\lambda) = Weib\left(\theta_4 + \theta_5 \cdot \frac{cases}{65000} + \theta_6 \cdot \frac{date}{1200}, \theta_1 + \theta_2 \cdot \frac{cases}{65000} + \theta_3 \cdot \frac{date}{1200}\right) =: Weib(\vec{\theta}) \tag{10}$$

for the *reactionTime* and fit the parameter vector $\vec{\theta}$ to the data in S_{ems} via Maximum Likelihood method and numeric optimization.

Experiment 2: GAN with Synthetic Data. Using the Weibull distribution from (10) with the parameter vector (12), we create a synthetic dataset

$$S_{weib} = ((cases_i, date_i), X_i)_{i=1}^{N}, \tag{11}$$

whereas $date_i, cases_i$ are taken from S_{ems} and $X_i \sim Weib(\vec{\theta_{opt}})$ are independent random samples drawn from the Weibull model with the fitted parameter vector $\vec{\theta}_{opt}$.

Furthermore, S_{weib} is used as training data for a GAN. The fully trained generator network G_{weib} is exported and used as a sampler in the simulation model.

In theory, the GAN should be indistinguishable from a Weibull PRNG, and Experiment 1 and 2 should lead to identically distributed simulation results. Moreover, the computation time difference will tell, how much slower the MLP evaluates compared to generation of a Weibull-distributed random number, which is usually very efficient using the inversion method.

Experiment 3: GAN with Real Data. Finally, we will use the actual microdata S_{ems} for training. We will find out, whether the GAN structure can cope with real data involving artefacts, biases and data errors. We will also evaluate whether the fitted GAN improves sampling of *reactionTime* values in the ABM compared to the Weibull-based approaches in Experiment 0 and 1.

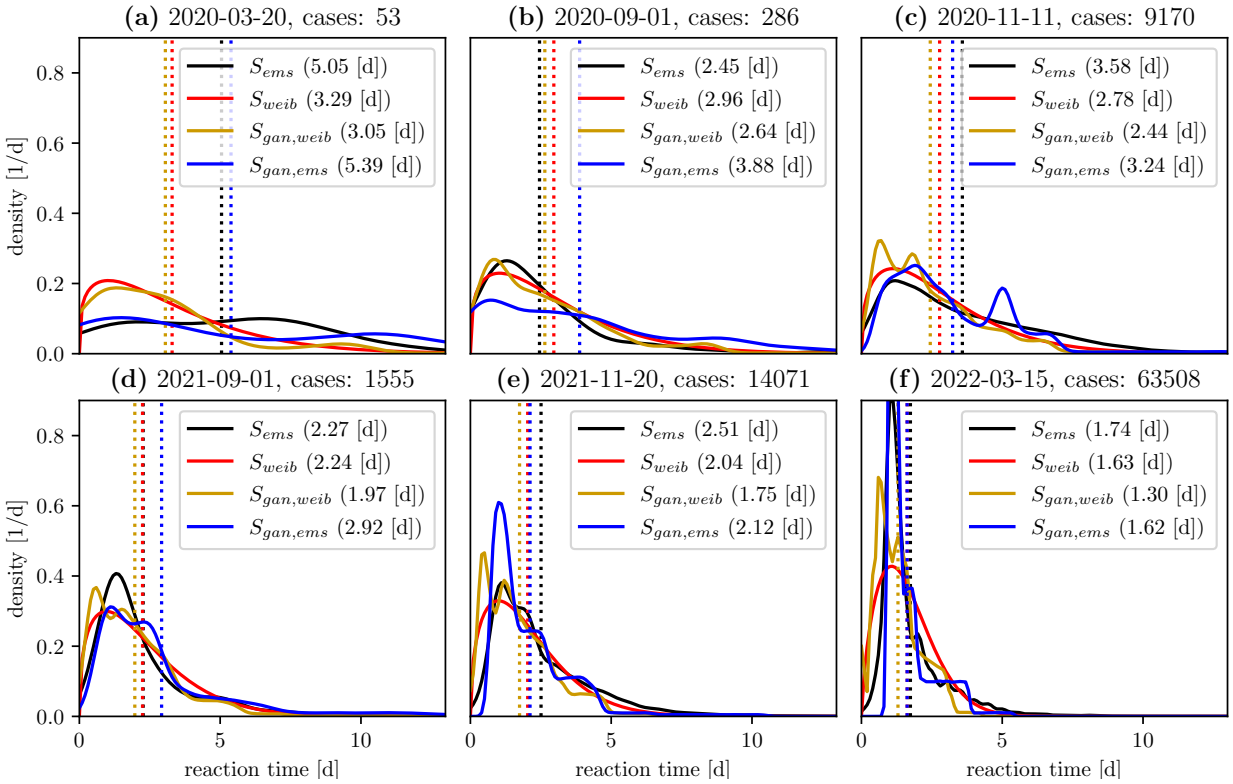

Figure 3: Comparison of S_{ems}, S_{weib} and the NN-created data $S_{gan,ems}, S_{gan,weib}$ for selected dates. The lines show a Gaussian Kernel Density estimation through the values in the corresponding dataset, the sample-mean is displayed with a dotted line.

2.6 Implementation

Numeric maximum likelihood fitting of (10) is done in Python3 using Scipy's *minimize*. Training of the GAN is done in Python3 as well using PyTorch (see Paszke et al. (2019)). As described in Bicher et al. (2021), the ABM is implemented in JAVA. For integration of the trained networks into the JAVA model, we utilize the TorchScript standard as interface format. The source code used for training the condGANs, the trained networks, and the synthetic dataset S_{weib} are available under https://github.com/dwhGmbH/condgan-for-abm. Note, that the original dataset S_{ems} is confidential and cannot be made available publicly.

3 RESULTS

We furthermore state the results starting with the model fitting and training procedure.

Fitting the Weibull model. The parameter vector for $Weib(\vec{\theta})$ introduced in (10) was fitted to

$$\vec{\theta}_{opt} = (3.708782991, -0.247241698, -2.457837151, 1.250395421, 0.286291613, 0.236835644)^T. \quad (12)$$

Most prominently, the scale parameter drastically decreases with time which is in line with Figure 2. For selected dates, Figure 3 compares, among others, the two data sets S_{ems} (black) and the $Weib(\vec{\theta}_{opt})$-generated synthetic S_{weib} (red). While the displayed Kernel Density Estimations indicate a feasible fit, the Weibull model does not properly factor the impact of the case numbers. While the sample-mean in S_{ems} increased from 2.45 to 3.58 days from 2020-09-01 (panel (b)) to 2020-11-11 (panel (c)), likely due to the high Delta wave, the mean in S_{weib} dropped by 0.2 days. Also chart (a) displays a comparably bad fit.

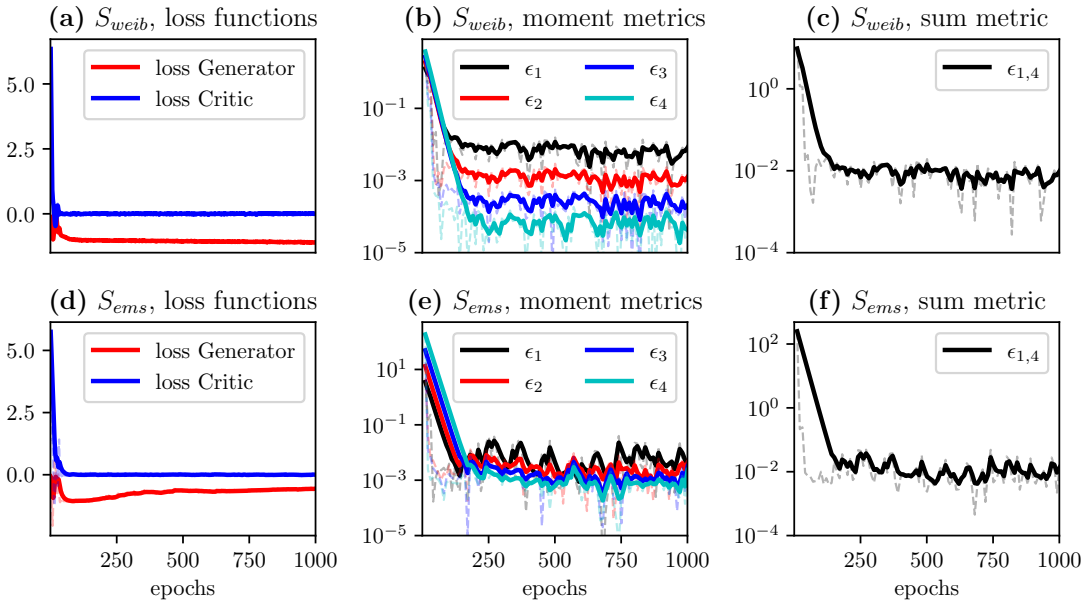

Figure 4: Training progress of the GAN for S_{weib} and S_{ems}. Part (a) displays the loss functions, (b) shows the defined convergence metrics ε_1 to ε_4, part (c) shows $\varepsilon_{1,4}$ as defined in (6) with weights equal to one. Thick lines show an exponential moving average with memory of 10 epochs.

Training of the GAN. Training of the GAN turned out highly sensitive with respect to network and hyper-parameters. The following setup leads to successful training and good distribution-similarity for Experiments 2 and 3:

- We used the values identified in (8) to norm all inputs to $[0,1]$.
- Both, for generator and critic, we use classic multi-layer-perceptrons (MLPs) with identical structure: The input layers have three nodes ($\vec{u}$ is 2-dimensional), two hidden layers with 128 nodes, and a scalar output layer. All use the ReLU activation functions without dropout.
- We use Wasserstein loss with gradient penalty factor of $\lambda = 10$.
- A learning rate of $2 \cdot 10^{-7}$ and a batch size of 512 was used.

Due to the comparably large training data set (about 4 Million entries) the learning rate turned out to be one of the most sensitive parameters and very small numbers had to be used. For training details and the trained networks we refer to the source code in https://github.com/dwhGmbH/condgan-for-abm. Training progress is shown in Figure 4. While the loss functions (panel (a) and (d)) have settled after about 100 epochs, the defined ε metrics (panels (b), (c), (e) and (f)) seem to stop improving after around 400-600 epochs and at least give a hint of the distribution similarity.

A priori Analysis of the Generator Networks. For the a priori analysis we evaluated the trained generator networks G_{ems} and G_{weib} ten times for the whole sample space to create two more synthetic data sets $S_{gan,weib}$ and $S_{gan,ems}$: For $Q \in \{weib, ems\}$,

$$S_{gan,Q} := \left((cases_{\lceil i/10 \rceil}, date_{\lceil i/10 \rceil}), G_Q(cases_{\lceil i/10 \rceil}, date_{\lceil i/10 \rceil}, X_i) \right)_{i=1}^{10N}, \tag{13}$$

with X_i being independent $U(0,1)$ random samples. Note, that we regard down-scaling of the inputs *cases* and *date* as well as up-scaling of the output as part of the evaluation of the network. In Figure 3, synthetic datasets $S_{gan,weib}$ and $S_{gan,ems}$ are depicted together with their corresponding training sets S_{weib} and S_{ems}

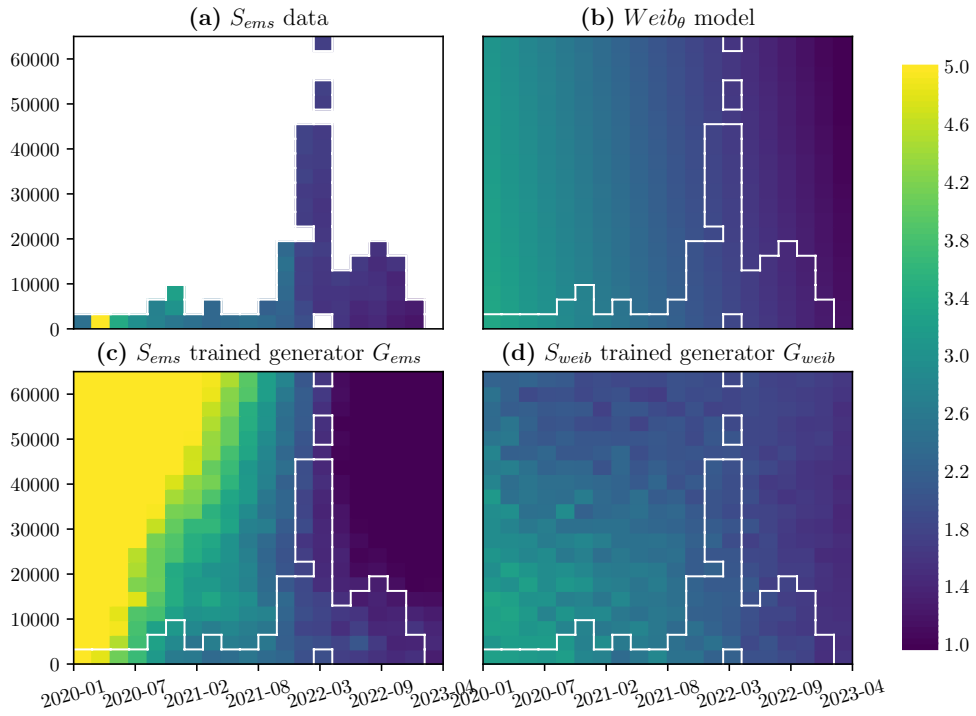

Figure 5: Heatmap of the average *reactionTime* for different points in the parameter space. White lines bound the region in the parameter space in which training-data is available.

via density estimation. Figure 5 shows how the different models extrapolate beyond the sample space provided by the S_{ems} data. For part (a) the average was collected over all data samples which lie within the investigated grid-cell. For part (b), the analytic mean $\lambda(\theta)\Gamma(1+1/k(\theta))$ of the $Weib_\theta$ model was computed for the midpoints of each cell. For parts (c) and (d) the arithmetic mean of 200 evaluations of the corresponding generators G_{ems} and G_{weib}, when using the midpoints of each cell and a uniformly distributed pseudo random number as input.

Agent-based Simulation Results. The ABM simulation results show a similar yet not fully equivalent picture for all four experiments, i.e. the simulated new-confirmed cases in all simulations match panel (a) in Figure 2. This indicates that the distribution of the *reaction time* is a relevant but not highly sensitive parameter for the model.

Since the results are similar enough, it is possible to analyze and compare the sampled *reactionTime* values and the simulated cases. Analogous to Figure 2, Figure 6 shows a comparison between the *reactionTime* values given in the S_{ems} data (a) and the ones simulated by the ABM epidemic model. Part (b) shows the sampled *reactionTime* values from Experiment 0 when using the classic approach, in which the Weibull parameters are a step function with time. Part (c) shows the results using the fitted Weibull model $Weib_\theta$ in which the parameters are a continuous function of time. Part (d) shows the results from Experiment 2, in which the synthetic dataset S_{weib} was used for training of the generator network G_{weib}. Finally, part (e) shows the analogous results from Experiment 3, in which G_{ems} was trained with the actual data.

With respect to computation time, the average simulation run with the unmodified sampler (Experiment 0) takes around 250 minutes, which is slightly longer than four hours. Using the Weibull PRNG from Experiment 0 and 1, generation of 6.6 Million pseudo-random *reaction time* values takes less than one second, using the MLP-based sampling (Experiment 2 and 3) it takes around 320 seconds. Although this increases the computational efforts for random *reaction time* generation more than 300-fold, the overall simulation time only increases by about 2%.

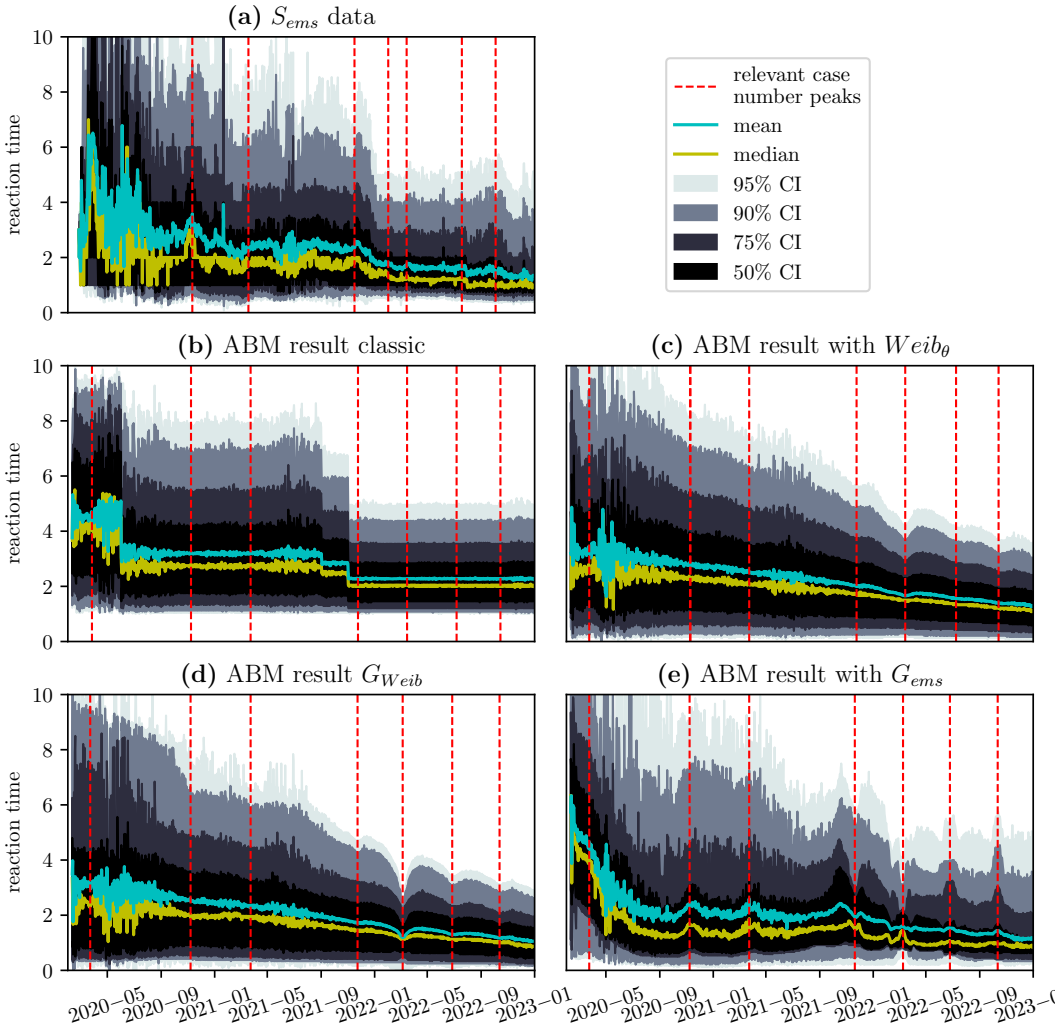

Figure 6: Comparison of given and simulated *reactionTime* values. Part (a) shows the given S_{ems} data, parts (b)-(e) show the simulated *reactionTime* values with the ABM as defined in Experiments 0, 1, 2, and 3, respectively.

4 DISCUSSION

In this work we introduced a concept to use a NN, trained in a condGAN setup, within an ABM to sample random decisions. In our experiments, a standard feed-forward MLP was trained to data about the time between symptom-onset and positive test result for SARS-CoV-2 positive persons. The trained MLP was used as a sampler in an agent-based epidemics model in which it created random reaction-times. The success of this concept depends on two features: the capabilities of a NN to learn a parametrized distributions in a condGAN game, and the usability of a NN for random decision sampling in ABMs.

First of all, we discuss the general capabilities of the condGAN to learn distributions from data. Comparing the trained networks with the corresponding training data, i.e. Figures 3 shows that a GAN can indeed be trained to a parameterized distribution but only to a certain level of precision. At this point, the similarity between S_{weib} and $S_{weib,gan}$ does not stand statistical distribution tests: Applying the cumulative distribution function of $Weib(\vec{\theta}_{opt})$ we transform the y-component of $S_{weib,gan}$ into a vector which we expect to be iid $U(0,1)$ distributed. Yet, a corresponding Kolmogoroff-Smirnoff test clearly suggested to refuse

this hypothesis (test statistics 0.129). The systematic errors learned by the GAN are subtle but simply too large compared with the huge sample size. We expect that similarity could be improved by hyper-parameter tuning of the GAN game.

With respect to the random numbers created in the ABM in Experiments 1 and 2, which are shown in parts (c) and (d) in Figure 6, we see high visual similarity. Mode and mean value follow the same linear trend including a similar disruption in Spring 2022. Within in the expected level of precision, the implemented Weibull sampler for Experiment 1 and the trained network for Experiment 2 are exchangeable for our use-case.

Finally, we take a closer look at the simulated reaction-time values in Experiment 3, shown in part (e) of Figure 6. Not only do the sampled values show high visual similarity with the original S_{ems} data, the also indicate a much better fit than the two Weibull approaches in Experiment 0 and 1. This implies, that the specified condGAN game also learns very subtle effects from complex and noisy real data. Partially the GAN even correctly learned a multi-modal distribution.

The original model in Experiment 0 does not incorporate case number dependencies. We have already labeled this as an important weakness of the approach. The Weibull model trained for Experiment 1 shows an even more counter-intuitive behavior: Parameter θ_2 was fitted to a negative value. So, the model learned a negative correlation between case number and the scale parameter. This contradicts our causal hypothesis and does not properly depict the behavior of the original data in other major disease waves (compare parts (a) and (c) in Figure 6 during February 2022).

As seen in parts (b) and (c) in the Figure 5 the NNs behave rather unexpected and partially chaotic beyond the scope of the training data. The comparatively long reaction times at the beginning of the observation interval are mapped much better with the network than with the Weibull model. Secondly, a positive correlation between reaction time and case numbers is clearly recognizable for the period up to 2022-01-01.

As a novel approach we also found some drawbacks during our research with this method:

- The model is prone to incorporate data errors directly. To mitigate this, a proper data-cleaning and quality assurance process is important.
- The training success is dependent on the chosen hyper-parameters. In our results, statistical tests could still clearly distinguish the distributions learned by the GANs from the original ones.
- The training process including parameter tuning is highly sensitive and time-intensive.
- Apart from the ε metrics (5) and (6) there are no suitable convergence indicators and no direct tests for divergence.
- The found model is a typical black-box model and it has to be tested thoroughly before use.

To emphasize the last point, we consider part (c) of Figure 5. It shows how G_{ems} extrapolated beyond its own training data. From a systemic point of view, the extrapolation is reasonable: It makes sense that especially in the period when there were limited test capacities in Austria, i.e. until autumn 2021, the case numbers had a strong influence on the *reactionTime* and that this influence disappeared afterwards, but without our expertise we could only guess about the behavior of the GAN.

We conclude that it is possible and reasonable to use a condGAN trained NN as a meta-model for stochastic decision processes in agent-based models, if users are willing to invest time in hyper-parameter tuning and analysis of the outcomes of the network if it is used beyond the region of fit. More research on a proper probability model and more meaningful convergence metrics are needed.

ACKNOWLEDGMENTS

The authors would like to thank Gesundheit Österreich Forschungs- und Planungs GmbH for providing the data. This research was funded in whole, or in part, by the Austrian Science Fund (FWF) I 5908-G.

REFERENCES

Arjovsky, M., S. Chintala, and L. Bottou. 2017. "Wasserstein Generative Adversarial Networks". In *Proceedings of the 34th International Conference on Machine Learning*, Volume 70, 214–223: PMLR.

Barton, R. R. 2009. "Simulation Optimization Using Metamodels". In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 230–238 https://doi.org/10.1109/WSC.2009.5429328.

Bicher, M., C. Rippinger, G. Schneckenreither, N. Weibrecht, C. Urach, N. Popper *et al*. 2022. "Model Based Estimation of the SARS-CoV-2 Immunization Level in Austria and Consequences for Herd Immunity Effects". *Scientific Reports* 12(1):2872 https://doi.org/10.1038/s41598-022-06771-x.

Bicher, M., C. Rippinger, C. Urach, D. Brunmeir, U. Siebert and N. Popper. 2021. "Evaluation of Contact-Tracing Policies against the Spread of SARS-CoV-2 in Austria: An Agent-Based Simulation". *Medical Decision Making* 41(8):1017–1032 https://doi.org/10.1177/0272989X211013306.

Bicher, M., C. Rippinger, M. Zechmeister, B. Jahn, G. Sroczynski, N. Popper *et al*. 2022. "An Iterative Algorithm for Optimizing COVID-19 Vaccination Strategies Considering Unknown Supply". *PLOS ONE* 17(5):e0265957 https://doi.org/10.1371/journal.pone.0265957.

Bicher, M., M. Zuba, L. Rainer, F. Bachner, C. Rippinger, P. Klimek *et al*. 2022. "Supporting COVID-19 Policy-Making with a Predictive Epidemiological Multi-Model Warning System". *Communications Medicine* 2(1):157 https://doi.org/10.1038/s43856-022-00219-z.

Biller, B. and C. Gunes. 2010. "Introduction to Simulation Input Modeling". In *proceedings of the 2010 Winter Simulation Conference*, 49–58. IEEE https://doi.org/10.5555/2433508.2433517.

Brauer, F. 2008. *Compartmental Models in Epidemiology*, 19–79. Springer.

Cai, Z., Z. Xiong, H. Xu, P. Wang, W. Li and Y. Pan. 2021. "Generative Adversarial Networks: A Survey Toward Private and Secure Applications". *ACM Computing Surveys (CSUR)* 54(6):1–38 https://doi.org/10.1145/3459992.

De Bernardi, M., M. Khouzani, and P. Malacaria. 2019. "Pseudo-Random Number Generation using Generative Adversarial Networks". In *ECMLPKDD 2018 Proceedings*, 191–200. Springer https://doi.org/10.1007/978-3-030-13453-2_15.

Feller, W. 1991. *An Introduction to Probability Theory and its Applications, Volume 2*, Volume 81. John Wiley & Sons.

Ghaffarzadegan, N., A. Majumdar, R. Williams, and N. Hosseinichimeh. 2024. "Generative Agent-Based Modeling: an Introduction and Tutorial". *System Dynamics Review* 40(1):e1761 https://doi.org/10.1002/sdr.1761.

Jäger, G. 2019. "Replacing Rules by Neural Networks a Framework for Agent-Based Modelling". *Big Data and Cognitive Computing* 3(4):51.

Mirza, M. and S. Osindero. 2014. "Conditional Generative Adversarial Nets". *arXiv preprint* https://doi.org/10.48550/arXiv.1411.1784.

Oak, R., C. Rahalkar, and D. Gujar. 2019. "Poster: Using Generative Adversarial Networks for Secure Pseudorandom Number Generation". In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2597–2599 https://doi.org/10.1145/3319535.3363265.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, L. Antiga *et al*. 2019. "Pytorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in neural information processing systems* 32 https://doi.org/10.5555/3454287.3455008.

Pietzsch, B., S. Fiedler, K. G. Mertens, M. Richter, C. Scherer, K. Widyastuti, , *et al*. 2020. "Metamodels for Evaluating, Calibrating and Applying Agent-Based Models: a Review". *The Journal of academic social science studies* 23(2) https://doi.org/10.18564/jasss.427.

Sarrut, D., N. Krah, and J.-M. Létang. 2019. "Generative Adversarial Networks (GAN) for Compact Bbeam Source Modelling in Monte Carlo Simulations". *Physics in Medicine & Biology* 64(21):215004.

Vallecorsa, S., F. Carminati, and G. Khattak. 2019. "3D Convolutional GAN for Fast Simulation". In *EPJ Web of Conferences*, Volume 214, 02010. EDP Sciences.

van der Hoog, S. 2019. "Surrogate Modelling in (and of) Agent-Based Models: A Prospectus". *Computational Economics* 53(3):1245–1263 https://doi.org/10.1007/s10614-018-9802-0.

AUTHOR BIOGRAPHIES

MARTIN BICHER has a PhD in Technical Mathematics from TU Wien and is expert in agent-based modeling. He was head developer of the dwh/TU SARS-CoV-2 modeling team during 2020-2023 and performed most of the experiments shown in the paper. His email address is martin.bicher@tuwien.ac.at.

DOMINIK BRUNMEIR is PhD student at TU Wien. His research concerns the overlap between discrete event simulation and AI Methods. His email address is dominik.brunmeir@tuwien.ac.at.

NIKI POPPER is research associate at TU Wien and CSO of dwh GmbH. His research focus lies on comparison of different modeling techniques and the synergies of AI and simulation models. His email address is nikolas.popper@tuwien.ac.at.