A GAN-based Reduced Order Model for Prediction, Data Assimilation and Uncertainty Quantification

Anonymous Authors[1]

Abstract

We propose a new method in which a generative adversarial network (GAN) within a reduced-order model (ROM) framework is used for uncertainty quantification of a numerical physical simulation, considering the presence of measurements. The GAN is trained using only unconditional simulations of the discretised partial differential equation (PDE) model. We compare the proposed method with the golden standard Markov chain Monte Carlo. Additionally, we suggest the use of regularization to improve already known ROM approaches and the new proposed uncertainty quantification method. We also present a new way of evaluating the GAN training, taking advantage of the real/generated sample structure. We apply the proposed approaches to a spatio-temporal compartmental model in epidemiology. The results show that the proposed GAN-based ROM can efficiently quantify uncertainty and accurately match the measurements and the golden standard, using only a few unconditional simulations of the high-fidelity numerical PDE model.

1. Introduction

Complex physics and engineering systems are usually described in terms of partial differential equations (PDE) that, for most problems of practical interest, cannot be solved analytically. Then it is necessary to use numerical methods to solve the governing equations. The discretization of these equations is usually performed using finite difference, finite volume, finite element or a combination of these procedures (Golub et al., 1992; Ames, 2014). Nonetheless, these methods need a large number of degrees of freedom to solve the PDEs accurately. This fact can generate prohibitively expensive simulations in terms of computational time and memory demand. Furthermore, these models are built on limited information, which makes their predictions uncertain. Therefore, it is necessary to assimilate observed data (calibrate model states and parameters in order to generate results that match the measurements) and formally propagate the uncertainties through the forward numerical simulator. In this context, reduced-order models (ROM) are computationally appealing and have been attracting significant attention in the last decades (Rozza et al., 2008; Cardoso et al., 2009; Hesthaven & Ubbiali, 2018; Xiao et al., 2019). The aim of a ROM is to reduce the computational burden of numerical simulations by creating a low-dimensional representation of a high-dimensional model or discretized system.

In order to quantify the uncertainty of numerical PDE simulations, one requires multiple random models that match or are conditional to the measurements. After simulating the conditional models, an empirical distribution of the variables of interest can be obtained (Liu et al., 2003). The validity of the uncertainty quantification depends on the quality of the generated conditional simulations. Nonetheless, it is often difficult and computationally expensive to generate a single conditional model (that honours the measurements), suggesting that the task of quantifying uncertainty must be even more difficult (Liu et al., 2003; Sudret et al., 2017; Cacuci, 2019). It is unfeasible to use methods such as rejection sampling (RS) and Markov chain Monte Carlo (MCMC) to propagate uncertainty through most practical PDE simulations due to their computational cost (Oliver et al., 1997; Liu et al., 2003; Oliver & Chen, 2011; Stordal & Nævdal, 2018). Therefore, approximate methods need to be used. Among them, Liu et al. (2003) showed that the randomized maximum likelihood (RML) (Kitanidis, 1995; Oliver et al., 1996), also called randomize-then-optimize (RTO) (Bardsley et al., 2014), performed better than other approximate methods.

In this paper, we propose a new method inspired by the RML (or RTO) in which a generative adversarial network (GAN) within a ROM framework is used to quantify the uncertainty of a numerical PDE simulation in the presence of measurements. The GAN is trained using only unconditional simulations of the high-fidelity numerical model. After training, the GAN-based ROM can be used to predict the evolution of the spatial distribution of the simulation states and observed

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

data can be assimilated. No additional simulations of the high-fidelity numerical model are required. We apply these methods to quantify the uncertainty of a spatio-temporal compartmental model in epidemiology, that was constructed in order to represent the spread of COVID-19 in an idealized town.

We list the contributions of our work as follows:

- We propose a new method to quantify uncertainty of numerical PDE simulations in the presence of observed data. We compare it with the golden standard of the MCMC method, and show that it can generate predictions and uncertainties that reasonably match the golden standard, but with orders of magnitude less computational time.
- We demonstrate how regularization can be used to improve already known ROM methods, and the new proposed uncertainty quantification method.
- We present a new way of evaluating the GAN training, using the fact that for numerical PDE simulations the real/generated samples represent a spatio-temporal map with known structure.

The source code, data, hardware configuration, used in this work are available at `https://github.com/vlssanonymous/uq-predgan`.

2. Related work

Machine learning for solving PDEs:

Several works have used machine learning to solve PDEs, such as Lagaris et al. (1998); Lopez et al. (2008); Guo et al. (2016); Raissi et al. (2019); Long et al. (2018); De Bézenac et al. (2019); Sanchez-Gonzalez et al. (2020); Pfaff et al. (2020); Wang et al. (2020); Fuks & Tchelepi (2020); Chen et al. (2021). Among them, generative adversarial networks (GAN) have been demonstrating promising results. GANs have been used to predict spatio-temporal solutions for super-resolution fluid flow (Xie et al., 2018), carbon capture (Zhong et al., 2019), incoming waves from Hokkaido tsunami (Cheng et al., 2020), the spread of COVID-19 (Silva et al., 2023; Quilodrán-Casas et al., 2022), and short-timescale weather forecast (Ravuri et al., 2021). All of the previous works tackled the problem of forward simulations in time and/or space. In addition to predicting in time, for many practical applications, the ability to assimilate observed data (solve a inverse problem) and generate the corresponding uncertainties is also highly desirable.

Reduced order modelling:

Whilst retaining much of the accuracy of high-fidelity models, ROMs have been used successfully in several fields to reduce the computation time. Non-intrusive ROMs (NIROMs) are particularly flexible, as they avoid the need to make modifications to the original source code (Xiao et al., 2015; Hesthaven & Ubbiali, 2018; Quilodrán Casas et al., 2020b). To produce a NIROM, first, dimensionality reduction techniques are applied to the solutions from the original high-fidelity model (known as snapshots) in order to find a low-dimensional representation. Principal component analysis (PCA) (Holmes et al., 2012), autoencoders, or a combination of both (Quilodrán Casas et al., 2020a; Phillips et al., 2021) are commonly used for this purpose. The second stage involves interpolating the reduced variables in order to approximate the evolution of the model. Recently, a variety of machine learning methods have been used for this task, including multi-layer perceptrons (Hesthaven & Ubbiali, 2018), cluster analysis (Kaiser et al., 2014), LSTMs (Wang et al., 2018; Ahmed et al., 2019; Kherad et al., 2021), Gaussian Process Regression (Guo & Hesthaven, 2018), adversarial networks (Heaney et al., 2022) and autoencoders (Heaney et al., 2023).

Data assimilation and uncertainty quantification:

Data assimilation is an inverse problem with the aim of calibrating uncertain model parameters and states in order to generate results that match observed data to within some tolerance. (Tarantola, 2005; Oliver & Chen, 2011; D'Amore et al., 2014; Silva et al., 2017). The process of assimilating observed data into numerical models has evolved from finding a single "best" solution to finding multiple solutions that take uncertainty into account. Some researchers have used GANs to tackle the problem of generating conditional numerical simulations and their uncertainties. (Mosser et al., 2019; Kang & Choe, 2020; Razak & Jafarpour, 2020; Canchumuni et al., 2021). Even though in these works they still need to simulate the high-fidelity numerical model to predict forward in time.

3. Background

In this section, we demonstrate how a GAN within a ROM framework can be used to generate time series predictions. Then we describe how the prediction is extended to account for observed data (measurements).

3.1. Reduced order model

The underlying assumption of a ROM is that the solution of the forward model can be accomplished by using considerably fewer degrees of freedom. One of the most widely used model reduction methods is the proper orthogonal decomposition (POD) approach (also known as PCA). In this work, we use a NIROM with PCA as a compression method. We have also tested autoencoders to compress the data; however, the result using PCA was slightly better and it does not need training or hyperparameter optimisation.

After the first stage (the compression), the second stage of the NIROM (the evolution in time of the solutions) is accomplished using a GAN (PredGAN algorithm). GANs

have received much attention recently, after achieving excellent results for their generation of realistic-looking images (Chu et al., 2017; Frid-Adar et al., 2018; Liu et al., 2018; Karras et al., 2019; 2020). Although Long short-term memory (LSTM) networks are widely recognised as one of the most effective sequential models (Goodfellow et al., 2016) for times series predictions, Quilodrán-Casas et al. (2022) compared the performance of the LSTM and GAN as a NIROM. The GAN was able to learn better the underlying data distribution and reduce the forecast divergence, especially when predicting further in time. In the next section, we explain how we train the GAN with the numerical PDE solutions. Following the training, we describe how the generator can be used to evolve the compressed snapshots of the numerical PDE simulation in time. After that, we show how to modify the prediction process to assimilate observed data.

3.2. GAN training

Proposed by Goodfellow et al. (2014), GANs are unsupervised learning algorithms capable of learning dense representations of the input data and are intended to be used as a generative model. Here, the generator network G directly produces time-sequences of a numerical PDE simulation from a random distribution as input (latent vector $\mathbf{z}$). The discriminator network D attempts to distinguish between samples drawn from the training data and samples drawn from the generator, considered as fake.

In this work, the GANs are trained using the non-saturating loss (Goodfellow, 2016) for the DCGAN (Radford et al., 2015), and the Wasserstein loss with gradient penalty for the WGAN-GP (Gulrajani et al., 2017). During the training process the latent space $\mathbf{z}$ is generated as a Gaussian random noise $\mathcal{N}(0, I_L)$, where I_L is an identity matrix of size L. The loss function of the discriminator takes as inputs: a time sequence of compressed states and model parameters from the numerical PDE simulation ("real" sample), and a time sequence of compressed states and model parameters generated by the generator ("fake" sample). The loss function of the generator only takes the "fake" samples as input. Please see Appendix A for more details about the GAN training losses and Appendix B for the GAN architectures. After training, the discriminator can be discarded since only the generator is used during the prediction process.

3.3. PredGAN for time series prediction

After training a GAN to produce data (compressed simulation states and parameters) at a sequence of $m+1$ time steps, i.e. given a latent vector $\mathbf{z}$, the output of the generator $G(\mathbf{z})$ will be data at time steps $n-m$ to n, no matter which point in time n represents, we can use the generator to make predictions in time in a recurrent way, using a ROM named Predictive GAN (PredGAN) as described in (Silva et al., 2023; Quilodrán-Casas et al., 2022).

Given known solutions at m consecutive time steps, we can perform an optimisation to match the first m time levels in the output of the generator with the known solutions. After convergence, the last time step, $m+1$, in the output of the generator is the prediction. We now can use this last time level $m+1$ as a known solution and perform another optimisation to predict the time step $m+2$. The process continues until we predict all time steps. Figure 1 illustrates how the PredGAN works.

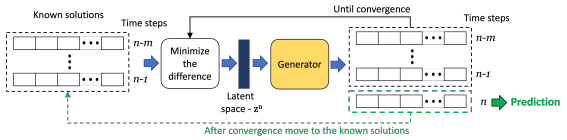


Figure 1. Overview of the PredGAN process.

In our case, after training, the output of the generator $G(\mathbf{z})$ is made up of $m+1$ consecutive time steps of compressed grid variables $\boldsymbol{\alpha}$ (outputs of the numerical PDE simulation), and model parameters $\boldsymbol{\mu}$ (inputs of the numerical PDE simulation). The compressed variables are principal component analysis (PCA) coefficients, but could also be latent variables from an autoencoder. For a GAN that has been trained with $m+1$ time levels, $G(\mathbf{z})$ takes the following form

$$G(\mathbf{z}) = \begin{bmatrix} (\boldsymbol{\alpha}^{n-m})^T, (\boldsymbol{\mu}^{n-m})^T \\ \vdots \\ (\boldsymbol{\alpha}^{n-1})^T, (\boldsymbol{\mu}^{n-1})^T \\ (\boldsymbol{\alpha}^{n})^T, (\boldsymbol{\mu}^{n})^T \end{bmatrix}_{(m+1) \text{ by } (N_{\text{PCA}}+N_\mu)} \tag{1}$$

where the compressed grid variables are defined as $(\boldsymbol{\alpha}^n)^T = [\alpha_1^n, \alpha_2^n, \cdots, \alpha_{N_{\text{PCA}}}^n]$. N_{PCA} is the number of principal components, and α_i^n represents the ith PCA coefficient at time level n. The model parameters are represented as $(\boldsymbol{\mu}^n)^T = [\mu_1^n, \mu_2^n, \cdots, \mu_{N_\mu}^n]$. N_μ is the number of parameters, and μ_i^n represents the ith parameter at time level n.

In each iteration of the PredGAN, one new time step is predicted. Assuming the GAN has already been trained, and we have solutions at time levels from $n-m$ to $n-1$ for the PCA coefficients, denoted by $\{\tilde{\boldsymbol{\alpha}}^k\}_{k=n-m}^{n-1}$, and also have the model parameters over all time steps $\tilde{\boldsymbol{\mu}}^k$, then to predict the solution at time level n we perform an optimisation defined as $\mathbf{z}^n = \text{argmin}_{\mathbf{z}^n} \mathcal{L}_p(\mathbf{z}^n)$, using

$$\begin{aligned} \mathcal{L}_p(\mathbf{z}^n) = & \sum_{k=n-m}^{n-1} \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right)^T \boldsymbol{W}_\alpha \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right) \\ & + \sum_{k=n-m}^{n-1} \zeta_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right)^T \boldsymbol{W}_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right), \end{aligned} \tag{2}$$

where $\boldsymbol{W}_\alpha$ is a square matrix of size N_{PCA} whose diagonal values are equal to the weights that govern the relative importance of the PCA coefficients, all other entries being zero. $\boldsymbol{W}_\mu$ is a square matrix of size N_μ whose diagonal values are equal to the model parameter weights, and the scalar ζ_μ controls how much importance is given to the model parameters compared to the compressed variables. The values for all the weighting terms are the same as in Silva et al. (2023). It is worth noticing that other genenative models (other than GANs) would also work with the PredGAN algorithm, since it only needs a generator and a way to optimise its output to match known solutions.

Figure 2 shows one example of prediction using the PredGAN. The results represent the evolution, in one cell of the grid (one point in space), of the number of people in each group (home, mobile) and compartment (S, E, I, R) over time. Each cycle in the graphs corresponds to a period of one day, when mobile people leave their homes during the day and return at night. After the first nine time iterations the PredGAN does not see any data from the high-fidelity numerical simulation, and relies completely on the predictions from PredGAN. Data from the high-fidelity numerical simulation is only required as a starting point.

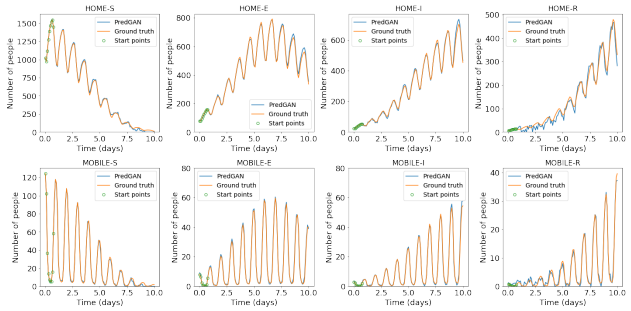


Figure 2. Prediction using the PredGAN. The green circles are the known solutions used to start the prediction process. The blue and orange lines are the prediction and the ground truth, respectively.

3.4. DA-PredGAN for data assimilation

Data assimilation is a type of inverse problem that aims to incorporate observed data into mathematical models. To perform data assimilation using the PredGAN process, Silva et al. (2023) proposed the Data Assimilation Predictive GAN (DA-PredGAN) that involves three changes to the PredGAN. (i) One additional term is included in the loss function in Eq. (15) to take account of the data mismatch between the observed data and the generated values. (ii) The aim of the data assimilation is to match the observed data and to determine the model parameters $\boldsymbol{\mu}^k$ (inputs of the numerical PDE simulation). Therefore, they are not known a priori, as in the prediction. (iii) The forward marching in time is now replaced by forward and backward marching.

The loss function for the optimisation at each iteration n of the forward and backward marches is given by

$$\begin{aligned}\mathcal{L}_{da}(\mathbf{z}^n) = &\sum_k \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right)^T \boldsymbol{W}_\alpha \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right) \\ &+ \sum_k \zeta_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right)^T \boldsymbol{W}_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right) \\ &+ \sum_k \zeta_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{obs}\right)^T \boldsymbol{W}^k_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{obs}\right), \quad (3)\end{aligned}$$

where $k \in \{n-m, n-m+1, \cdots, n-1\}$ and $k \in \{n+m, n+m-1, \cdots, n+1\}$, for the forward and backward marches respectively. The observed data at each time step k is stored in the vector $\boldsymbol{d}^k_{obs}$ of size N_{obs}. $\boldsymbol{d}^k$ is the generated data calculated based on the output of the generator at time step k. In our case, it represents data at some points in the grid (high dimensional states) and it is calculated through the PCA coefficients $\boldsymbol{\alpha}^k$ and stored eigenvectors. $\boldsymbol{W}^k_{obs}$ is a square matrix of size N_{obs} whose diagonal values are equal to the observed data weights, and the scalar ζ_{obs} direct controls how much importance is given to the data mismatch. The values in the diagonal of $\boldsymbol{W}^k_{obs}$ are set to zero where we have no observation. After convergence, the newly predicted time level n is added to the known solutions $\tilde{\boldsymbol{\alpha}}^n = \boldsymbol{\alpha}^n$, and different from the prediction, we also update the model parameters using the newly predicted time step $\tilde{\boldsymbol{\mu}}^n = \boldsymbol{\mu}^n$. After the forward march (going through all time steps), the process continues with a backward march.

After performing a forward and backward march using Eqs. (3), the average of the data mismatch (last term on the right of Eqs. (3)) at the end of all iterations n is calculated. If the average mismatch has not converged or the maximum number of iterations is not reached, the process continues with a new forward and backward march. A relaxation factor is also introduced to stabilize the process of marching forward and backward in time as in Silva et al. (2023).

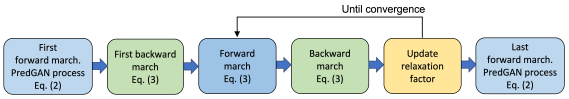


Figure 3. Overview of the DA-PredGAN. Each march represents going through all time steps.

Figure 3 shows an overview of the DA-PredGAN process and Figure 4 shows two data assimilation results (top and bottom plots). The results show the evolution in time of the simulation states in one cell of the grid (one point in space). We generate observed data from a high-fidelity numerical simulation that was not included in the training set of the GAN. We also added 5% noise to the chosen data. We note that for both cases (top and bottom plots) the observed data is honoured, although their results are slightly different. We

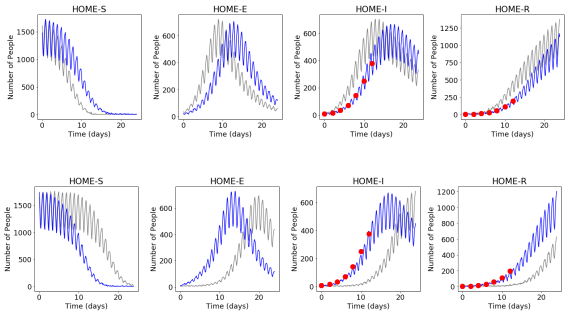


Figure 4. Two different data assimilations (top and bottom) using the DA-PredGAN. The circles in red are the observed data (the same in both cases), the lines in grey the initial conditions (first forward march), and the blue lines the final results (last forward march).

only show here the home group in order to save space, the results for the mobile group are similar.

4. Method

In this section, we present a method to quantify uncertainty of numerical PDE simulations in the presence of observed data. After that, we propose the use of regularization for the PredGAN, DA-PredGAN and the new proposed uncertainty quantification method. Then we present a way of evaluating the GAN training, using the fact that the real/generated samples represent a spatio-temporal map with known structure.

4.1. UQ-PredGAN for Uncertainty Quantification

The computation of a single model that matches the observed data is usually insufficient to quantify risks and uncertainties. Data assimilation is generally an ill-posed inverse problem (Tarantola, 2005; Oliver et al., 2008), hence several models can match the observed data, within some tolerance (as in Figure 4). In order to quantify uncertainty, we propose in this paper a ROM named Uncertainty Quantification Predictive GAN (UQ-PredGAN). This method is inspired by the RML (or RTO) as a way of sampling a posterior distribution conditioned to observed data. In the RML/RTO, the numerical simulation is used to predict forward, and for each sample, an optimisation (data assimilation) is performed to condition the models to the observed data. The challenge is usually to perform the optimisation, since the high-fidelity numerical simulation needs to be run several times and usually adjoints are not present. In this work, the proposed method UQ-PredGAN can compute uncertainties relying just on a set of unconditioned numerical simulations. The prediction, data assimilation and uncertainty quantification are performed using the inherent adjoint capability present within a GAN, and no additional high-fidelity numerical simulations, other than those used for training the GAN, are required.

The idea is to generate several models that match the observed data and can quantify the uncertainty in the model states (outputs) and model parameters (inputs). To this end, we perform several data assimilations using the DA-PredGAN algorithm with the modified loss functions

$$\begin{aligned}\mathcal{L}_{uq,j}(\mathbf{z}^n) = &\sum_k \left(\tilde{\boldsymbol{\alpha}}_j^k - \boldsymbol{\alpha}^k\right)^T \boldsymbol{W}_\alpha \left(\tilde{\boldsymbol{\alpha}}_j^k - \boldsymbol{\alpha}^k\right) \\ &+ \sum_k \zeta_\mu \left(\tilde{\boldsymbol{\mu}}_j^k - \boldsymbol{\mu}^k\right)^T \boldsymbol{W}_\mu \left(\tilde{\boldsymbol{\mu}}_j^k - \boldsymbol{\mu}^k\right) \\ &+ \sum_k \zeta_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}_{obs}^k + \boldsymbol{\varepsilon}_j^k\right)^T \boldsymbol{W}_{obs}^k \left(\boldsymbol{d}^k - \boldsymbol{d}_{obs}^k + \boldsymbol{\varepsilon}_j^k\right), \quad (4)\end{aligned}$$

where for the forward march $k \in \{n-m, n-m+1, \cdots, n-1\}$ and for the backward march $k \in \{n+m, n+m-1, \cdots, n+1\}$. Considering N_s as the number of data assimilations to be performed, then $j = 1, ..., N_s$. In this work, $N_s = 200$. This value was chosen based on previous experience using the RML/RTO. The observed data error is represented by the random vector $\boldsymbol{\varepsilon}$, and we consider that all measurement errors are uncorrelated, thus they are sampled from a normal distribution with zero mean and standard deviation equal to 5% of the corresponding observed data. For each data assimilation j, we use a different prior $\tilde{\boldsymbol{\mu}}_j^k$ with the corresponding initial condition $\{\tilde{\boldsymbol{\alpha}}_j^k\}_{k=0}^m$, and a different perturbation on the observed data $\boldsymbol{\varepsilon}_j^k$.

The UQ-PredGAN is proposed as follows:

1. Sample the model parameters $\tilde{\boldsymbol{\mu}}_j$ from a normal distribution $\mathcal{N}(\overline{\boldsymbol{\mu}}, \mathbf{C}_\mu)$, where $\mathbf{C}_\mu$ is the covariance matrix of the model parameters, and $\overline{\boldsymbol{\mu}}$ is the model parameter mean vector.
2. Sample the measurement error $\boldsymbol{\varepsilon}_j$ from a normal distribution $\mathcal{N}(0, \mathbf{C}_d)$, where $\mathbf{C}_d$ is the covariance matrix of the measurement error.
3. Assimilate data using the DA-PredGAN process with the loss in Eq. (4).
4. Repeat the process until the final number of data assimilation samples N_s is reached.

After performing N_s steps of the UQ-PredGAN, accept all realizations that obtained an acceptable level of data mismatch. It is worth mentioning that for the RML/RTO, when the case is linear and normal distributions are used to sample the model parameters and measurement error, the RML/RTO samples the corrected posterior distribution (Oliver et al., 1996; Bardsley et al., 2014; Stordal & Nævdal, 2018). In this work, we also use normal distributions to sample the model parameters and measurement error; however, the test case is nonlinear, there is an additional term in the loss function compared to the RML/RTO, and the weighting terms are seen as tuning parameters (as in Stordal & Nævdal (2018)). Thus, the results are an approximate sample of the posterior distribution.

4.2. Regularization

One limitation of the (DA/UQ)-PredGAN is that during the optimisation process of each time step, the latent vector $\mathbf{z}$ can end up in a region in the latent space not well populated. This can jeopardize subsequent iterations since they use the previous latent values as a initial condition. To solve this problem, we propose here to use a regularization of the latent vector during the time stepping process of the (DA/UQ)-PredGAN.

The new PredGAN and (DA/UQ)-PredGAN loss functions are now defined as,

$$\mathcal{L}_p^{reg}(\mathbf{z}^n) = \mathcal{L}_p(\mathbf{z}^n) + \frac{\lambda_p}{N_z}\|\mathbf{z}^n\|_2^2, \tag{5}$$

$$\mathcal{L}_{da}^{reg}(\mathbf{z}^n) = \mathcal{L}_{da}(\mathbf{z}^n) + \frac{\lambda_{da}}{N_z}\|\mathbf{z}^n\|_2^2, \tag{6}$$

$$\mathcal{L}_{uq,j}^{reg}(\mathbf{z}^n) = \mathcal{L}_{uq,j}(\mathbf{z}^n) + \frac{\lambda_{uq}}{N_z}\|\mathbf{z}^n\|_2^2, \tag{7}$$

where N_z is the size of the latent vector, and in this work we use $\lambda_p = \lambda_{da} = \lambda_{uq} = 0.01$. We noticed that by using the regularization we improve the prediction, data assimilation and uncertainty quantification. Considering the latter, after running the UQ-PredGAN for the 200 sampled model parameters $\mathcal{R}_{0h}$ (see Section 5.3), without using regularization, 104 realizations reached an acceptable level of data mismatch, while using regularization, 121 reached an acceptable level. Further details about these results can be found in Appendix F. For the remainder of this paper regularization of the latent space will be used in all the experiments.

4.3. GAN evaluation

GANs are notoriously difficult to evaluate while training as the loss paths of the generator and discriminator behave in non-predictable ways (Grnarova et al., 2019). However, such evaluation is of particular interest as it aids in assessing our model's convergence state relative to other models even before the model is run on the testing set for predictions, data assimilation and uncertainty quantification. In addition, it allows for picking the model with the best weights from the last learning iterations, and for stopping the training when no improvement is observed.

To evaluate the learning and convergence abilities of our GAN models while we train them, we propose an evaluation metric based on the Bhattacharyya coefficient (BC) (or the Hellinger distance (H), since $1-BC = H^2$) (Le Cam et al., 2000). It can be calulated as

$$H^2(P,Q) = 1 - \sum_{i=1}^{k}\sqrt{p_i q_i}, \qquad H^2 \in [0,1], \tag{8}$$

where $P = (p_1,...,p_k)$ and $Q = (q_1,...,q_k)$ are two discrete probability distributions, the first coming from the generated samples and the second from the training dataset. We use the fact that the output of the generator is a spatio-temporal map (Eq. (1)) that have semantically different axis with one being the time dimension, and the other being the PCA coefficients (or autoencoder latent values) and the model parameters. We could expect that if we take the mean and standard deviation from each PCA coefficient (one column) from one spatio-temporal map, the distributions of these means and standard deviations over all training spatio-temporal maps will have distinct values representing the characteristics of each PCA coefficient. Therefore, we calculate the evaluation metric (H^2) for all PCA coefficients mean and standard deviation, then we average them to form the final score. A comparison between the distributions of the real and generated data for different GANs can be found in the Appendix G.

Other metrics for comparing discrete distributions could also be used (Gibbs & Su, 2002). We chose the BC coefficient (or H^2) due to its relative simplicity when applied to a large number of samples. This is important as it would be undesirable to "waste" significant amount of time evaluating the model rather than training the model in practice.

5. Experiments

5.1. Test case description

The test case used here is the spatio-temporal variation of a virus infection in an idealized town. The extended SEIRS model used in this work (Silva et al., 2023; Quilodrán-Casas et al., 2022) extends the traditional theory of the dynamics of infectious diseases (Anderson et al., 1992; Bjørnstad, 2018; Bjørnstad et al., 2020) to account for variations not only in time but also in space.

5.1.1. Extended SEIRS model and problem set up

The extended SEIRS model used in this work consists of four compartments (Susceptible - Exposed - Infections - Recovered) and two people groups (Home - Mobile). Figure 5 shows the diagram of how individuals move between compartments and groups. The model starts with some individuals in the infectious compartments (Home-I/Mobile-I). The members of these compartment will spread the pathogen to the susceptible compartments (Home-S/Mobile-S). Upon being infected, the members of the susceptible compartments are moved to the exposed compartments (Home-E/Mobile-E) and remain there until they become infectious. Infectious individuals remain in the infectious compartment until they become recovered (Home-R/Mobile-R). Recovered people can also become susceptible again due to the loss of immunity. The equations describing the evolution in space and time of the number of people in each compartment and group can be found in (Silva et al., 2023; Quilodrán-Casas et al., 2022).

One important factor in dynamics of infectious diseases is

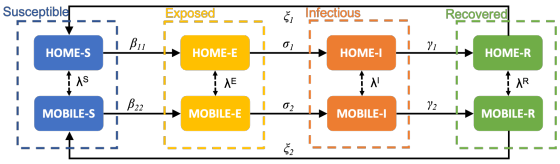


Figure 5. Diagram of the extended SEIRS model. The diagram shows how people move between groups and compartments at a given point in space (or one cell in the grid). The vital dynamics and the transport via diffusion are not displayed here.

the basic reproduction number ($\mathcal{R}_0$), which represents the expected number of new cases caused by a single infectious member in a completely susceptible population (Dietz, 1993; Hethcote, 2000). The $\mathcal{R}_0$ controls how rapidly the disease could spread. Here, we have two $\mathcal{R}_0$, one for each group of people. $\mathcal{R}_{01}$ representing the basic reproduction number of people at home, and $\mathcal{R}_{02}$ representing the basic reproduction number of people that are mobile and outside their homes therefore. For this case, we can also calculate an Effective $\mathcal{R}_0$ representing the $\mathcal{R}_0$ seen by the whole population at a specific time.

The idealized town and problem set up used in this work are the same as in Silva et al. (2023); Quilodrán-Casas et al. (2022). Further information about the discretization and solution methods of the high-fidelity numerical simulation can be also found in (Silva et al., 2023; Quilodrán-Casas et al., 2022)

5.2. Dataset and training process

For the training process 40 high-fidelity numerical simulations were performed in order to generate the training dataset. Each simulation consists of two different $\mathcal{R}_{0h}$, one for people at home and another for mobile people. The spatial domain of the numerical simulation is a regular grid of 10×10 (100 cells). Considering that each type of people (people at home and mobile) has the four quantities of the extended SEIRS model (Susceptible, Exposed, Infectious and Recovered), there will be eight variables for each cell in the grid per time step, which gives a total number of $100 \times 8 = 800$ variables. Principal component analysis is performed in the 800 variables, in order to work with a low dimensional space in the GAN-based ROM. The first 15 principal components were chosen and they capture $> 99.99\%$ of the variance held in the time snapshots. Hence the GAN is trained to generate the 15 PCA coefficients ($\boldsymbol{\alpha}^n$) and the two $\mathcal{R}_{0h}$ ($\boldsymbol{\mu}^n$) over a sequence of 10 time steps. We choose this time length because it represents a cycle (one day) in the results.

In this work, all the codes are implemented using Python and TensorFlow (Apache 2.0 license). We choose the size of the latent vector $\mathbf{z}$ to be 100. The networks receive/generate the 10 time levels as a two-dimensional array ("an image", Eq. (1)) with 10 rows and 17 columns. Each row represents a time level and each column comprises the 15 PCA coefficients and the two $\mathcal{R}_{0h}$. We choose this configuration, instead of a linear representation, to exploit the time dependence captured in the two-dimensional array. We also carried out initial tests using a linear representation of the time level outputs and a multi-layer perceptron as a generator and discriminator. However, it generated worse results than the two-dimensional representation.

We decided to test two well-known GANs in this work, the DCGAN (Radford et al., 2015) and the WGAN-GP (Gulrajani et al., 2017). The generator and discriminator architectures can be found in Appendix B. Figure 6 shows a comparison of both models over the training time using the proposed evaluation metric (H^2) (Section 4.3). In order to compare the proposed metric, we also calculate the Jensen–Shannon divergence. From Figure 6, we can see that DCGAN outperforms the WGAN-GP. Hence, the former was selected to continue the experiments. Further analysis of the evaluation metric can be found in Appendix G.

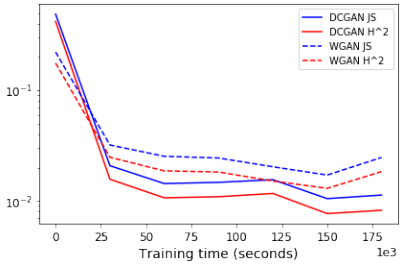


Figure 6. Comparison of the DCGAN and the WGAN-GP, using the proposed metric (H^2) and the Jensen–Shannon divergence (JS). Smaller values are better.

5.3. Results and discussion

In this section, we apply the UQ-PredGAN to quantify uncertainty in the extended SEIRS model considering the presence of measurements. The model represents the spread of COVID-19 in an idealized town (as in Quilodrán-Casas et al. (2022) and Silva et al. (2023)). We generate the observed data from a high-fidelity numerical simulation ($\mathcal{R}_{01} = 7.7$, $\mathcal{R}_{02} = 17.4$) that was not included in the training set. Observed data was collected at five points of the domain and the measurements are available every two days. We measured only infectious and recovered people, as in Silva et al. (2023). The $\mathcal{R}_{0h}$ are not used as observed data. For generating the priors (unconditional simulations) 200 model parameters $\mathcal{R}_{0h}$ were sampled from a normal distribution with a mean of 10 and standard deviation of 4. The mean and standard deviation were chosen based on Kochańczyk et al. (2020). The 200 model parameters and their corresponding initial conditions were used to start the UQ-PredGAN process. For each of the model parameters, one data assimilation was performed as described in Section 4.1. After the data assimilation, 121 realizations were accepted based on their data mismatch error.

It is worth noting that for the whole uncertainty quantification process using the UQ-PredGAN, we required only 40 high-fidelity numerical simulations (for training the GAN).

Figure 7 shows the UQ-PredGAN results for each group and compartment (model states) at one point in space. The priors (grey lines) are the first forward march of each data assimilation, and the posteriors (blue lines) are the last forward march of the accepted realizations. The posterior mean (black line) is also shown in the plots. We can see from these figures that the conditional simulations (posteriors) generated by the UQ-PredGAN match the observed data, within some tolerance (we considered a measurement error of 5%), and the uncertainty is propagated through the simulation time. The high frequency oscillation presented in the results corresponds to a daily cycle, when mobile people leave their homes during the day and return to them at night. Comparable results were observed at other points in domain, hence they are not presented here.

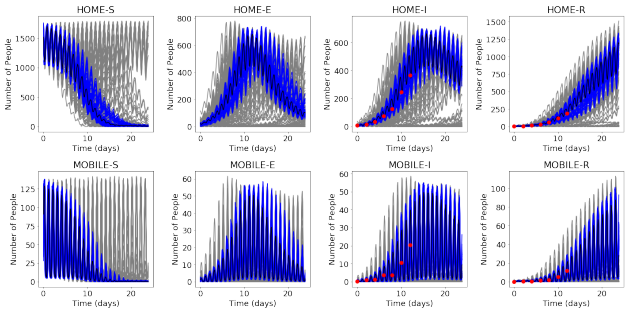


Figure 7. Results of the UQ-PredGAN for each group am compartment at one point in space. The red dots represent the observed data (measurements), the grey lines the unconditional simulations (before data assimilation), the blue lines the conditional simulations (after the data assimilation), and the black line the posterior mean.

In the plots from the first line of Figure 8, we show the probability density function (PDF) of the $\mathcal{R}_{0h}$ and Effective $\mathcal{R}_0$ (model parameters), for the priors and posteriors. The result shows that the UQ-PredGAN was able to reduce the uncertainty in the model parameters approaching the true values used to generate the observed data. Note that the data assimilation is an inverse and usually ill-posed problem, hence different values of $\mathcal{R}_{0h}$ could match the measurements within some tolerance. We have also run a comparison of the UQ-PredGAN with the golden standard of the MCMC method (using the Metropolis–Hastings algorithm). The results from the MCMC are in the plots from the second line of Figure 8. It is worth mentioning that to run the MCMC, it took more than a month on a dedicated workstation (more than 100,000 samples and we stopped because of time limitations, see Appendix H for the hardware configuration), while to run the UQ-PredGAN only some hours (two days if we add the GAN training). We can notice from Figure 8 that the UQ-PredGAN generates uncertainties that have a reasonable match with the golden standard, but with orders of magnitude less computational time. In practical terms, it is unaffordable to run the MCMC for most numerical PDE simulations.

We can notice that the distributions generated by the UQ-PredGAN have longer "tails" than the ones from the MCMC. We checked the results from the simulations in that portion of the distribution and they reasonably match the observed data. It may indicate that the MCMC has not converged yet, which means that we would need many more simulation samples (>100,000) for the MCMC to generate the correct tails, which makes it impractical. Figure 8 also shows that the UQ-PredGAN is able to sample the second small mode of the mobile group $\mathcal{R}_{02}$. It is where the true value of the $\mathcal{R}_{02}$ used to generate the observed data occurs. We can also see that for the mobile group, the UQ-PredGAN posterior PDF does not exactly match the MCMC results as well as for the home group. This could be because the number of mobile people is one order of magnitude smaller than the number of people at home, which gives the latter more importance during the data assimilation process, and the relative rate of change of the number of people in the mobile group is much greater than in the home group, thus small perturbations in the former can cause huge relative deviations.

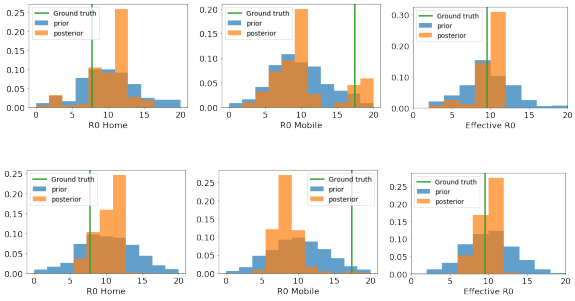
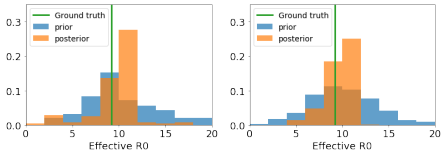


Figure 8. Probability density functions of the $\mathcal{R}_{0h}$ and Effective $\mathcal{R}_0$ at day 12. Comparison of UQ-PredGAN (fisrt line) with the golden standard MCMC (second line). The ground truth represents the values in the high fidelity numerical simulation used to generate the observed data.

Figure 9 shows the prediction of the Effective $\mathcal{R}_0$ at day 16, along with the ground truth, for the UQ-PredGAN and MCMC. Comparable results were observed for other points in time. The last observed data used in this experiment was at day 12. The results demonstrate that the UQ-PredGAN can generate predictions and uncertainties that accurately match the ground truth and have a reasonable match to the golden standard MCMC. Table 1 shows the posterior mean, mode and standard deviation (STD) for different days. The mode is calculated based on the histograms. We can notice that going further in time the posterior uncertainty (the STD) in the Effective $\mathcal{R}_0$ increases, in both methods. This is because the further away the predictions are from the last observed data, the greater the uncertainty will be in those predictions. The

difference in the standard deviation of the UQ-PredGAN and the MCMC is because it was impractical to run the amount of simulation samples required for the MCMC to represent the tails of the distributions, as already mentioned.



(a) UQ-PredGAN (b) MCMC

Figure 9. Predicted probability density functions of the Effective $\mathcal{R}_0$ at day 16 (the last observed data was at day 12). Comparison of UQ-PredGAN with the golden standard MCMC. . The ground truth represents the value in the high fidelity numerical simulation used to generate the observed data.

Table 1. Values of the posterior PDF of the Effective $\mathcal{R}_0$ over time. Comparison of UQ-PredGAN with the golden standard MCMC.

TIME (DAYS) =>	12	16	22
GROUND TRUTH	9.52	9.22	9.48
UQ-PREDGAN MEAN	9.86	9.70	9.91
MCMC MEAN	9.89	9.68	9.92
UQ-PREDGAN MODE	11.0	11.0	11.0
MCMC MODE	11.0	11.0	11.0
UQ-PREDGAN STD	1.73	2.44	2.70
MCMC STD	1.39	1.40	1.51

6. Conclusion

We proposed a novel use of a generative adversarial network as a reduced-order model (UQ-PredGAN), that is able to quantify uncertainty of numerical physical simulations, considering the presence of measurements. Additionally, we propose the use of regularization to improve the (DA/UQ)-PredGAN results, and we present a new way of evaluating the GAN training taking advantage of the known real/generated sample structure. We applied the proposed method to a spatio-temporal compartmental model in epidemiology. The results show that the UQ-PredGAN accurately matches the observed data and efficiently quantifies/reduces uncertainty in the model states (groups and compartments) and model parameters (basic reproduction numbers). We compare the UQ-PredGAN with the golden standard MCMC, and show that it can generate predictions and uncertainties that reasonably match the golden standard, but with orders of magnitude less computational time. The UQ-PredGAN is not limited to the underlying physics of this application: it is a general framework for quantifying uncertainties of numerical physical simulations.

References

Ahmed, S. E., Rahman, S. M., San, O., Rasheed, A., and Navon, I. M. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Physics of Fluids*, 31(12):126602, 2019. doi: 10.1063/1.5128374.

Ames, W. F. *Numerical methods for partial differential equations*. Academic press, 2014.

Anderson, R. M., Anderson, B., and May, R. M. *Infectious diseases of humans: dynamics and control*. Oxford University Press, 1992.

Bardsley, J. M., Solonen, A., Haario, H., and Laine, M. Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems. *SIAM Journal on Scientific Computing*, 36(4): A1895–A1910, 2014.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18 (1):5595–5637, 2017.

Bjørnstad, O., Shea, K., Krzywinski, M., and Altman, N. The SEIRS model for infectious disease dynamics. *Nature Methods*, 17(6):557–558, 2020.

Bjørnstad, O. N. *Epidemics: Models and data using R*. Springer, 2018.

Cacuci, D. G. *BERRU Predictive Modeling: Best Estimate Results with Reduced Uncertainties*. Springer, 2019.

Canchumuni, S. W., Castro, J. D., Potratz, J., Emerick, A. A., and Pacheco, M. A. C. Recent developments combining ensemble smoother and deep generative networks for facies history matching. *Computational Geosciences*, 25 (1):433–466, 2021.

Cardoso, M. A., Durlofsky, L. J., and Sarma, P. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International Journal for Numerical Methods in Engineering*, 77(9):1322–1350, 2009.

Chen, W., Wang, Q., Hesthaven, J. S., and Zhang, C. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*, 446:110666, 2021.

Cheng, M., Fang, F., Pain, C. C., and Navon, I. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. *Computer Methods in Applied Mechanics and Engineering*, 365:113000, 2020.

Chu, C., Zhmoginov, A., and Sandler, M. CycleGAN, a Master of Steganography. *arXiv preprint arXiv:1712.02950*, 2017.

De Bézenac, E., Pajot, A., and Gallinari, P. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124009, 2019.

Dietz, K. The estimation of the basic reproduction number for infectious diseases. *Statistical Methods in Medical Research*, 2(1):23–41, 1993.

D'Amore, L., Arcucci, R., Carracciuolo, L., and Murli, A. A scalable approach for variational data assimilation. *Journal of Scientific Computing*, 61(2):239–257, 2014.

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321: 321–331, 2018.

Fuks, O. and Tchelepi, H. A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

Gibbs, A. L. and Su, F. E. On choosing and bounding probability metrics. *International statistical review*, 70 (3):419–435, 2002.

Golub, G. H., Ortega, J. M., et al. *Scientific computing and differential equations: an introduction to numerical methods*. Academic press, 1992.

Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Grnarova, P., Levy, K. Y., Lucchi, A., Perraudin, N., Goodfellow, I., Hofmann, T., and Krause, A. A domain agnostic measure for monitoring and evaluating gans. *Advances in Neural Information Processing Systems*, 32, 2019.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. *Advances in neural information processing systems*, 30, 2017.

Guo, M. and Hesthaven, J. S. Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 341:807–826, 2018. doi: https://doi.org/10.1016/j.cma.2018.07.017.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Heaney, C. E., Liu, X., Go, H., Wolffs, Z., Salinas, P., Navon, I. M., and Pain, C. C. Extending the Capabilities of Data-Driven Reduced-Order Models to Make Predictions for Unseen Scenarios: Applied to Flow Around Buildings. *Frontiers in Physics*, 10:910381, 2022.

Heaney, C. E., Tang, J., Yan, J., Guo, D., Ipock, J., Kaluvakollu, S., Lin, Y., Shao, D., Chen, B., Mottet, L., Kumar, P., and Pain, C. C. Data Assimilation with Machine Learning for Dynamical Systems: Modelling Indoor Ventilation. *submitted*, 2023.

Hesthaven, J. S. and Ubbiali, S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

Hethcote, H. W. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

Holmes, P., Lumley, J. L., Berkooz, G., and Rowley, C. W. *Proper orthogonal decomposition*, pp. 68–105. Cambridge Monographs on Mechanics. Cambridge University Press, 2 edition, 2012. doi: 10.1017/CBO9780511919701.005.

Kaiser, E., Noack, B., Cordier, L., Spohn, A., Segond, M., Abel, M., Daviller, G., Östh, J., Krajnović, S., and Niven, R. Cluster-based reduced-order modelling of a mixing layer. *Journal of Fluid Mechanics*, 754:365–414, 2014. doi: 10.1017/jfm.2014.355.

Kang, B. and Choe, J. Uncertainty quantification of channel reservoirs assisted by cluster analysis and deep convolutional generative adversarial networks. *Journal of Petroleum Science and Engineering*, 187:106742, 2020.

Karras, T., Laine, S., and Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401–4410. 2019.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

Kherad, M., Moayyedi, M. K., and Fotouhi, F. Reduced order framework for convection dominant and pure diffusive problems based on combination of deep long short-term memory and proper orthogonal decomposition/dynamic mode decomposition methods. *International Journal for Numerical Methods in Fluids*, 93:853–873, 2021. doi: 10.1002/fld.4911.

Kitanidis, P. K. Quasi-linear geostatistical theory for inversing. *Water resources research*, 31(10):2411–2419, 1995.

Kochańczyk, M., Grabowski, F., and Lipniacki, T. Super-spreading events initiated the exponential growth phase of COVID-19 with $\mathcal{R}_0$ higher than initially estimated. *Royal Society Open Science*, 7(9):200786, 2020.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5): 987–1000, 1998.

Le Cam, L., LeCam, L. M., and Yang, G. L. *Asymptotics in statistics: some basic concepts*. Springer Science & Business Media, 2000.

Linnainmaa, S. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2): 146–160, 1976.

Liu, N., Oliver, D. S., et al. Evaluation of Monte Carlo methods for assessing uncertainty. *SPE Journal*, 8(02): 188–195, 2003.

Liu, Y., Qin, Z., Wan, T., and Luo, Z. Auto-painter: Cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks. *Neurocomputing*, 311:78–87, 2018.

Long, Z., Lu, Y., Ma, X., and Dong, B. PDE-Net: Learning PDEs from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018.

Lopez, R., Balsa-Canto, E., and Oñate, E. Neural networks for variational problems in engineering. *International Journal for Numerical Methods in Engineering*, 75(11): 1341–1360, 2008.

Mosser, L., Dubrule, O., and Blunt, M. J. Deepflow: history matching in the space of deep generative models. *arXiv preprint arXiv:1905.05749*, 2019.

Oliver, D. S. and Chen, Y. Recent progress on reservoir history matching: a review. *Computational Geosciences*, 15(1):185–221, 2011.

Oliver, D. S., He, N., and Reynolds, A. C. Conditioning permeability fields to pressure data. In *ECMOR V-5th European conference on the mathematics of oil recovery*, pp. cp–101. European Association of Geoscientists & Engineers, 1996.

Oliver, D. S., Cunha, L. B., and Reynolds, A. C. Markov chain monte carlo methods for conditioning a permeability field to pressure data. *Mathematical geology*, 29(1): 61–91, 1997.

Oliver, D. S., Reynolds, A. C., and Liu, N. *Inverse theory for petroleum reservoir characterization and history matching*. Cambridge University Press, 2008.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Phillips, T. R. F., Heaney, C. E., Smith, P. N., and Pain, C. C. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *International Journal for Numerical Methods in Engineering*, 122(15):3780–3811, 2021.

Quilodrán Casas, C., Arcucci, R., and Guo, Y. Urban air pollution forecasts generated from latent space representations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020a.

Quilodrán Casas, C., Arcucci, R., Wu, P., Pain, C., and Guo, Y.-K. A reduced order deep data assimilation model. *Physica D: Nonlinear Phenomena*, pp. 132615, 2020b.

Quilodrán-Casas, C., Silva, V. L., Arcucci, R., Heaney, C. E., Guo, Y., and Pain, C. C. Digital twins based on bidirectional LSTM and GAN for modelling the COVID-19 pandemic. *Neurocomputing*, 470:11–28, 2022.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., et al. Skillful precipitation nowcasting using deep generative models of radar. *Nature*, 597:672–677, 2021.

Razak, S. M. and Jafarpour, B. History matching with generative adversarial networks. In *ECMOR XVII*, volume 2020, pp. 1–17. European Association of Geoscientists & Engineers, 2020.

Rozza, G., Huynh, D. B. P., and Patera, A. T. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Arch. Comput. Methods Eng*, 15:229–275, 2008.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Silva, V. L., Heaney, C. E., Li, Y., and Pain, C. C. Data Assimilation Predictive GAN (DA-PredGAN) Applied to a Spatio-Temporal Compartmental Model in Epidemiology. *Journal of Scientific Computing*, 94:25, 2023. doi: 10.1007/s10915-022-02078-1.

Silva, V. L. S., Emerick, A. A., Couto, P., and Alves, J. L. D. History matching and production optimization under uncertainties–application of closed-loop reservoir management. *Journal of Petroleum Science and Engineering*, 157:860–874, 2017.

Stordal, A. S. and Nævdal, G. A modified randomized maximum likelihood for improved Bayesian history matching. *Computational Geosciences*, 22(1):29–41, 2018.

Sudret, B., Marelli, S., and Wiart, J. Surrogate models for uncertainty quantification: An overview. In *2017 11th European conference on antennas and propagation (EUCAP)*, pp. 793–797. IEEE, 2017.

Tarantola, A. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020.

Wang, Z., Xiao, D., Fang, F., Govindan, R., Pain, C. C., and Guo, Y. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018. doi: 10.1002/fld.4416.

Wengert, R. E. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

Xiao, D., Fang, F., Buchan, A., Pain, C., Navon, I., and Muggeridge, A. Non-intrusive reduced order modelling of the Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:522–541, 2015.

Xiao, D., Heaney, C. E., Mottet, L., Fang, F., Lin, W., Navon, I. M., Guo, Y.-K., Matar, O. K., Robins, A. G., and Pain, C. C. A reduced order model for turbulent flows in the urban environment using machine learning. *Building and Environment*, 148:323–337, 2019.

Xie, Y., Franz, E., Chu, M., and Thuerey, N. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4): 1–15, 2018.

Zhong, Z., Sun, A. Y., and Jeong, H. Predicting CO_2 Plume Migration in Heterogeneous Formations using Conditional Deep Convolutional Generative Adversarial Network. *Water Resources Research*, 55(7):5830–5851, 2019.

A. GAN training losses

In this work, the GAN is trained using the non-saturating loss (Goodfellow, 2016) for the DCGAN (Radford et al., 2015), and the Wasserstein loss with gradient penalty for the WGAN-GP (Gulrajani et al., 2017). The generator network G directly produces time-sequences from a random distribution as input (latent vector $\mathbf{z}$):

$$G:\mathbf{z}\sim\mathcal{N}(0,I_L)\rightarrow\tilde{\mathbf{x}}\in\mathbb{R}^{m+1\times N} \tag{9}$$

where $\tilde{\mathbf{x}}$ is an array of $m+1$ time sequences with N dimensions, L is the size of the latent vector, and I_L is an identity matrix of size L. The discriminator network D attempts to distinguish between samples drawn from the training data $\mathbf{x}$ and samples drawn from the generator $\tilde{\mathbf{x}}$, considered as fake. The output of the discriminator $D(\mathbf{x})$ represents the probability that a sample came from the data rather than a "fake" sample from the generator. The output of the generator $G(\mathbf{z})$ is a sample from the distribution learned from the dataset.

Equations (10) and (11) show the loss functions of the discriminator and generator for the DCGAN training, respectively:

$$L_D=-\mathbb{E}_{x\sim p_{data}}[\log(D(\mathbf{x}))]-\mathbb{E}_{\tilde{x}\sim p_{gen}}[\log(1-D(\tilde{\mathbf{x}}))] \tag{10}$$

$$L_G=-\mathbb{E}_{\tilde{x}\sim p_{gen}}[\log(D(\tilde{\mathbf{x}}))] \tag{11}$$

Equations (12) and (13) show the loss functions of the discriminator and generator for the WGAN-GP training, respectively:

$$L_D=\mathbb{E}_{\tilde{x}\sim p_{gen}}[D(\tilde{\mathbf{x}})]-\mathbb{E}_{x\sim p_{data}}[D(\mathbf{x})]+\lambda\mathbb{E}_{\tilde{x}\sim p_{gen}}[(||\nabla_{\tilde{\mathbf{x}}}D(\tilde{\mathbf{x}})||_2-1)^2] \tag{12}$$

$$L_G=-\mathbb{E}_{\tilde{x}\sim p_{gen}}[D(\tilde{\mathbf{x}})] \tag{13}$$

B. GAN architectures

The generator and discriminator architectures for the DCGAN and WGAN-GP can be found in Figure 10.

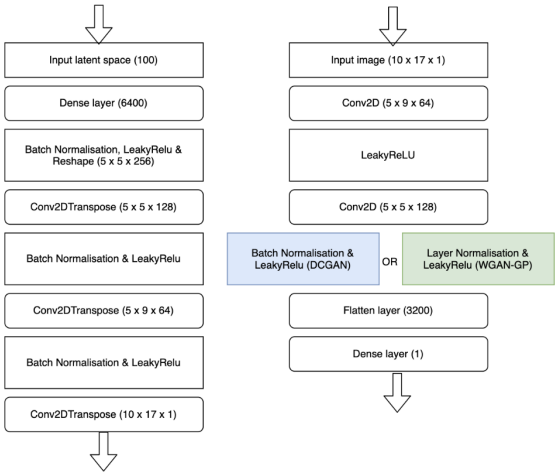


Figure 10. DCGAN and WGAN-GP architectures.

C. PredGAN algorithm

Assume we have the solutions at time levels up to and including m for the POD coefficients (compressed numerical PDE solutions), denoted by $\{\tilde{\boldsymbol{\alpha}}^k\}_{k=1}^m$, and consider model parameters known over the entire simulation time $\tilde{\boldsymbol{\mu}}^k$, then to predict future solutions:

1. a latent vector ${}^{(0)}\mathbf{z}^{m+1}$ is randomly generated in order to start the prediction of time level $m+1$. The superscript in brackets on the left of the latent vector is the optimisation iteration counter within a time step prediction;
2. time iteration counter is set to $n=m+1$;
3. optimisation iteration counter is set to $l=0$;
4. the generator of the GAN is evaluated at the current value of the latent variables, ${}^{(l)}\mathbf{z}^n$, yielding

$$G({}^{(l)}\mathbf{z}^n)=\begin{bmatrix}(\boldsymbol{\alpha}^{n-m})^T,(\boldsymbol{\mu}^{n-m})^T\\ \vdots\\ (\boldsymbol{\alpha}^{n-1})^T,(\boldsymbol{\mu}^{n-1})^T\\ (\boldsymbol{\alpha}^{n})^T,(\boldsymbol{\mu}^{n})^T\end{bmatrix} \tag{14}$$

5. the difference between the predicted values and the known values is calculated:

$$\mathcal{L}_p({}^{(l)}\mathbf{z}^n)=\sum_{k=n-m}^{n-1}\left(\tilde{\boldsymbol{\alpha}}^k-{}^{(l)}\boldsymbol{\alpha}^k\right)^T\boldsymbol{W}_\alpha\left(\tilde{\boldsymbol{\alpha}}^k-{}^{(l)}\boldsymbol{\alpha}^k\right)$$
$$+\sum_{k=n-m}^{n-1}\zeta_\mu\left(\tilde{\boldsymbol{\mu}}^k-{}^{(l)}\boldsymbol{\mu}^k\right)^T\boldsymbol{W}_\mu\left(\tilde{\boldsymbol{\mu}}^k-{}^{(l)}\boldsymbol{\mu}^k\right), \tag{15}$$

where $\boldsymbol{W}_\alpha$ is a square matrix of size N_{POD} whose diagonal values are equal to the weights that govern the relative importance of the POD coefficients. All other entries are zero. The weights could be based on the singular values if a POD method is used for compression, for example. $\boldsymbol{W}_\mu$ is a square matrix of size N_μ whose diagonal values are equal to the model parameter weights, and the scalar ζ_μ controls how much importance is given to the model parameters compared to the compressed variables. It is worth mentioning that the goal in each time iteration is to predict a new time level n, hence the POD coefficients $\boldsymbol{\alpha}^n$ and model parameters $\boldsymbol{\mu}^n$ of this time level are not in the loss function.

6. the gradient of the loss $\mathcal{L}_p$ is calculated with respect to the latent variables ${}^{(l)}\mathbf{z}^n$ (by automatic differentiation), and $\mathcal{L}_p$ is minimised in the gradient direction leading to an updated set of latent variables ${}^{(l+1)}\mathbf{z}^n$;

7. the optimisation iteration counter is incremented by one ($l \leftarrow l+1$);

8. steps 4 to 7 are repeated until convergence is reached;

9. the converged latent variables are saved as $\mathbf{z}^n$ (note, no optimisation iteration index) and used to initialise the latent variables at the next time level, ${}^{(0)}\mathbf{z}^{n+1} = \mathbf{z}^n$. The predicted time step n is added to the known solutions $\tilde{\boldsymbol{\alpha}}^n = \boldsymbol{\alpha}^n$;

10. the time iteration counter is incremented by one ($n \leftarrow n+1$);

11. go back to step 3 (until the final time level is reached).

It is worth mentioning that the gradient of Eq. (15) can be calculated by automatic differentiation (Wengert, 1964; Linnainmaa, 1976; Baydin et al., 2017). In other words, minimising the loss in Eq. (15) can be achieved simply by back-propagating the loss through the generator using the same methods that were employed when training the GAN.

D. DA-PredGAN algorithm

To assimilate data through time the following steps are performed:

1. March forward in time using the Eq. (15) and the prediction process in Section C, with guessed parameters $\{\tilde{\boldsymbol{\alpha}}^k\}_{k=1}^m$ for the first m time steps and $\tilde{\boldsymbol{\mu}}^k$ over all simulation time. This will results in an initial guess of $\boldsymbol{\alpha}^n$ at all time levels n.

2. Time march backwards in time optimising

$$\begin{aligned}\mathcal{L}_{da,b}(\mathbf{z}^n) = &\sum_{k=n+1}^{n+m} \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right)^T \boldsymbol{W}_\alpha \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right) \\ &+ \sum_{k=n+1}^{n+m} \zeta_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right)^T \boldsymbol{W}_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right) \\ &+ \sum_{k=n+1}^{n+m} \zeta_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{\text{obs}}\right)^T \boldsymbol{W}^k_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{\text{obs}}\right), \quad (16)\end{aligned}$$

starting from the m final time levels (obtained from the previous iteration). Use the same prediction process described in Section C, but change the functional to Eq. (16) and decrease the time iterations counter ($n \leftarrow n-1$), instead of increasing it. The subscript b of $\mathcal{L}_{da}$ indicates that this loss function applies to the backwards march. This tries to perform time stepping while attempting to match the observations using the observed data mismatch part of the functional (last term on the right of Eq. (16)). During the time march update the model parameters using the newly predicted time step $\tilde{\boldsymbol{\mu}}^n = \boldsymbol{\mu}^n$.

3. Time march forward in time optimising

$$\begin{aligned}\mathcal{L}_{da,f}(\mathbf{z}^n) = &\sum_{k=n-m}^{n-1} \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right)^T \boldsymbol{W}_\alpha \left(\tilde{\boldsymbol{\alpha}}^k - \boldsymbol{\alpha}^k\right) \\ &+ \sum_{k=n-m}^{n-1} \zeta_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right)^T \boldsymbol{W}_\mu \left(\tilde{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k\right) \\ &+ \sum_{k=n-m}^{n-1} \zeta_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{\text{obs}}\right)^T \boldsymbol{W}^k_{obs} \left(\boldsymbol{d}^k - \boldsymbol{d}^k_{\text{obs}}\right), \quad (17)\end{aligned}$$

starting from the first m time levels (obtained from the backward march). Use the same prediction process described in Section C, but changing the functional to Eq. (17). The subscript f of $\mathcal{L}_{da}$ indicates that this loss function applies to the forward march. This tries to perform time stepping while attempting to match the observations using the observed data mismatch part of the functional (last term on the right of Eq. (17)). During the time march update the model parameters using the newly predicted time step $\tilde{\boldsymbol{\mu}}^n = \boldsymbol{\mu}^n$.

4. Keep time stepping backwards till the start of time and then forwards to the end of time using Eqs. (16), (17), until the algorithm has converged.

5. After convergence, perform a last forward march using the Eq. (15) and the prediction process in Section C. Use the last calculated parameters $\{\tilde{\boldsymbol{\alpha}}^k\}_{k=1}^m$ and $\tilde{\boldsymbol{\mu}}^k$ for this. This last forward march is optional (most used for parametric problems).

D.1. Applying relaxation and convergence criteria

To stabilise the process of marching forwards and backwards, a relaxation parameter is used in the resulting latent vector $\mathbf{z}^n$ at each time iteration. After performing a forward and backward time marches using Eqs. (17) or (16), the resulting latent vector is relaxed by

$$\mathbf{z}^n = (1-r^j)\mathbf{z}^{n-1} + r^j \hat{\mathbf{z}}^n \quad (18)$$

where $\hat{\mathbf{z}}^n$ is the resulting latent vector generated in each time iteration by minimising Eqs. (17) or (16). j represents a iteration corresponding to a entire forward and backward march, and r^j is the relaxation factor used in the iteration j. Thus each relaxation parameter r^j is used in all time iterations n within the forward and backward marches.

The relaxation factor r^j starts the data assimilation process with the value one. If the average data mismatch at j is greater than at $j-1$, then $r^j = r^j/2$ and the forward and backward iteration j is repeated. On the other hand, if the average data mismatch at j is less than at $j-1$, the algorithm goes to the next iteration $j+1$ using $r^{j+1} = 1.5r^j$, also respecting the maximum value of one for the relaxation factor (if $r^{j+1} > 1$ then $r^{j+1} = 1$). We use as convergence criteria $r^j < 0.01$ to stop the data assimilation process.

D.2. Calculating the weighting terms

The weighting terms in the loss function of Eqs. (17) or (16) are calculated as

$$\zeta_{obs} = \hat{\zeta}_{obs}\left(\frac{\Delta\alpha}{\Delta d}\right)^2\left(\frac{m\sum_{i=1}^{N_{\text{POD}}}(w_\alpha)_{ii}}{\sum_k\sum_{i=1}^{N_c}(w_{obs})^k_{ii}}\right), \tag{19}$$

where $\hat{\zeta}_{obs}$ is a tuning parameter and in this work it is set to 10. $\Delta\alpha$ and Δd are the ranges of the compressed variables and the observed data, respectively. $(w_\alpha)_{ii}$ are the the terms on the diagonal of $\boldsymbol{W}_\alpha$, and $(w_{obs})^k_{ii}$ are the terms on the diagonal of $\boldsymbol{W}^k_{obs}$.

$$\zeta_\mu = \hat{\zeta}_\mu\left(\frac{\Delta\alpha}{\Delta\mu}\right)^2\left(\frac{\sum_{i=1}^{N_{\text{POD}}}(w_\alpha)_{ii}}{\sum_{i=1}^{N_\mu}(w_\mu)_{ii}}\right), \tag{20}$$

where $\Delta\mu$ represents the range of the model parameters, $(w_\mu)_{ii}$ are the the terms on the diagonal of $\boldsymbol{W}_\mu$, and $\hat{\zeta}_\mu$ is a tuning parameter. ζ_μ controls how quickly one lets the parameters $\boldsymbol{\mu}^k_j$ change within the data assimilation method. In order to let $\boldsymbol{\mu}^k$ change more rapidly at the beginning and more slowly when the process is near convergence, during the data assimilation, we choose to dynamically update ζ_μ. Therefore, we start with $\hat{\zeta}_\mu = 10^{-4}$ and increase it by a factor of 1.2 after each forward-backward iteration. For the prediction, we use $\hat{\zeta}_\mu = 10^{-2}$.

E. Extended SEIRS model

The extended SEIRS model used in this work consists of four compartments (Susceptible - Exposed - Infections - Recovered) and two people groups (Home - Mobile). Figure 5 shows the diagram of how individuals move between compartments and groups. The model starts with some individuals in the infectious compartments (Home-I/Mobile-I). The members of these compartment will spread the pathogen to the susceptible compartments (Home-S/Mobile-S). Upon being infected, the members of the susceptible compartments are moved to the exposed compartments (Home-E/Mobile-E) and remain there until they become infectious. Infectious individuals remain in the infectious compartment until they become recovered (Home-R/Mobile-R). Recovered people can also become susceptible again due to the loss of immunity. Modelling the movement of people is of the utmost importance in the spread of infectious diseases, such as COVID-19. Therefore, the goal of the extended SEIRS model is to reproduce the daily cycle of night and day, in which there is a pressure for mobile people to go to their homes at night, and there will be many people leaving their homes during the day and thus joining the mobile group. To this end, the extended SEIRS model uses a diffusion term and an interaction term to model this process:

$$\frac{\partial S_h}{\partial t} = \eta_h N_h - \frac{S_h\sum_{h'}(\beta_{hh'}I_{h'})}{N_h} + \xi_h R_h - \nu^S_h S_h - \sum_{h'=1}^{\mathcal{H}}\lambda^S_{hh'}S_{h'} + \nabla\cdot(k^S_h\nabla S_h), \tag{21a}$$

$$\frac{\partial E_h}{\partial t} = \frac{S_h\sum_{h'}(\beta_{hh'}I_{h'})}{N_h} - \sigma_h E_h - \nu^E_h E_h - \sum_{h'=1}^{\mathcal{H}}\lambda^E_{hh'}E_{h'} + \nabla\cdot(k^E_h\nabla E_h), \tag{21b}$$

$$\frac{\partial I_h}{\partial t} = \sigma_h E_h - \gamma_h I_h - \nu^I_h I_h - \sum_{h'=1}^{\mathcal{H}}\lambda^I_{hh'}I_{h'} + \nabla\cdot(k^I_h\nabla I_h), \tag{21c}$$

$$\frac{\partial R_h}{\partial t} = \gamma_h I_h - \xi_h R_h - \nu^R_h R_h - \sum_{h'=1}^{\mathcal{H}}\lambda^R_{hh'}R_{h'} + \nabla\cdot(k^R_h\nabla R_h), \tag{21d}$$

where $\mathcal{H}$ represents the number of groups. Here, we have two groups of people, hence $\mathcal{H} = 2$, one representing people at home $h = 1$, and the second representing people that are mobile $h = 2$ and outside their homes therefore. N_h represents the total number of individuals in each group, $\beta_{hh'}$ is the transmission rate between groups, σ_h is the rate of exposed individuals becoming infectious, γ_h is the recovered rate, and ξ_h is the rate recovered individuals return to the susceptible group due to loss of immunity. The vital dynamics are represented by η_h and ν_h, where η_h is the birth rate and ν_h is the death rate. The diffusion coefficient is represented by k_h and describes the movement of people around the domain. The interaction terms, $\lambda_{hh'}$, control how people move between groups, for example, how people that are in the mobile group move to the home group. When moving between groups people remain in the same compartment, and when moving between compartments, people remain in the same group.

One important factor in dynamics of infectious diseases is the basic reproduction number ($\mathcal{R}_0$), which represents the expected number of new cases caused by a single infectious member in a completely susceptible population (Dietz, 1993; Hethcote, 2000). The $\mathcal{R}_0$ controls how rapidly the disease could spread and for each group it is define as

$$\mathcal{R}_{0h} = \frac{\sigma_h}{(\sigma_h+\nu_h)}\frac{\beta_{hh'}}{(\gamma_h+\nu_h)}, \tag{22}$$

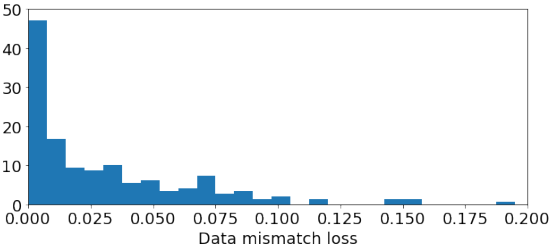
where we assume $\beta_{hh'} = 0$ when $h \neq h'$ because people in their homes never directly meet mobile people (who are outside their homes). For this case, we can also calculate an Effective $\mathcal{R}_0$ representing the $\mathcal{R}_0$ seen by the whole

population at a specific time. It can be calculated as

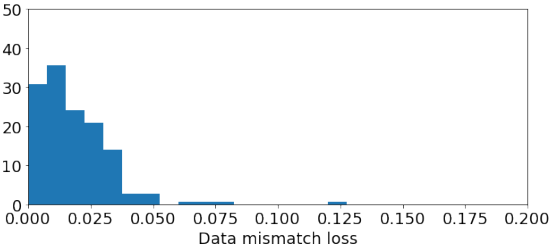
$$\text{Effective } \mathcal{R}_0 = \frac{\sum_h S_h \mathcal{R}_{0h}}{\sum_h S_h}, \tag{23}$$

F. Regularization

In this work, we use regularization to improve the prediction, data assimilation and uncertainty quantification. Considering the latter, after running the UQ-PredGAN for the 200 sampled model parameters $\mathcal{R}_{0h}$ (see Section 5.3), without using regularization, 104 realizations reached an acceptable level of data mismatch, while using regularization, 121 reached an acceptable level. Figure 11 shows the normalised histogram of the data mismatch (average of the last term on the right of Eq. (3) over all times) between the observed and simulated data. We can see that the cases with regularization reached a considerably lower data mismatch. We considered an acceptable level of data mismatch to be a loss function less than 0.02.



(a) No regularization



(b) With regularization

Figure 11. Histogram of the data mismatch between the observed and simulated data.

G. GAN evaluation

The proposed evaluation metric (Eq. (8)) uses the fact that the output of the generator is a spatio-temporal map (Eq. (1)) that has semantically different axes with one being the time dimension, and the other being the PCA coefficients (or autoencoder latent values) and the model parameters. We could expect that if we take the mean and standard deviation from each PCA coefficient (one column) from one spatio-temporal map, the distributions of these means and standard deviations over all training spatio-temporal maps will have distinct values representing the characteristics of each PCA coefficient. We can see this from Figures 12, 13, 14 and 15, where we show the distribution of the PCA coefficients mean and standard deviation for the real and generated data. We can see that during the training process the generated means and standard deviation distributions and the real means and standard deviations distributions become indistinguishable from each other. Therefore, we calculate the evaluation metric (H^2) for all PCA coefficients mean and standard deviation distributions, then we average them to form the final score.

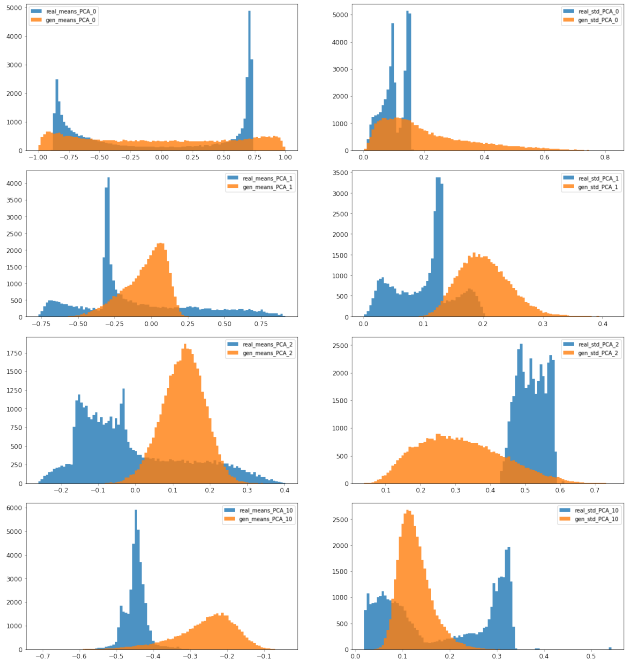


Figure 12. DCGAN first epoch: PCA coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different PCA coefficient.

The results from the Figures 12, 13, 14 and 15 correlate with the values of the proposed evaluation metric (H^2) shown in Figure 6. We can notice that after the first epoch the generated distributions from the WGAN-GP enclose better the real distributions than the generated distribution from the DCGAN. Hence the WGAN-GP starts with a higher evaluation metric (H^2) than the DCGAN, as we can see in Figure 6. However, the DCGAN outperforms the WGAN-GP when we look the final results in Figure 6. We can also see this in Figures 13 and 15 (the bottom left plot, for example). They show that the result from the WGAN-GP does not match the real distribution as well as the DCGAN.

We chose to generate the distributions of the mean and standard deviation of the PCA coefficients since in each sample we expect a strong correlation in time, but not among the coeffients (using PCA the directions are perpendicular

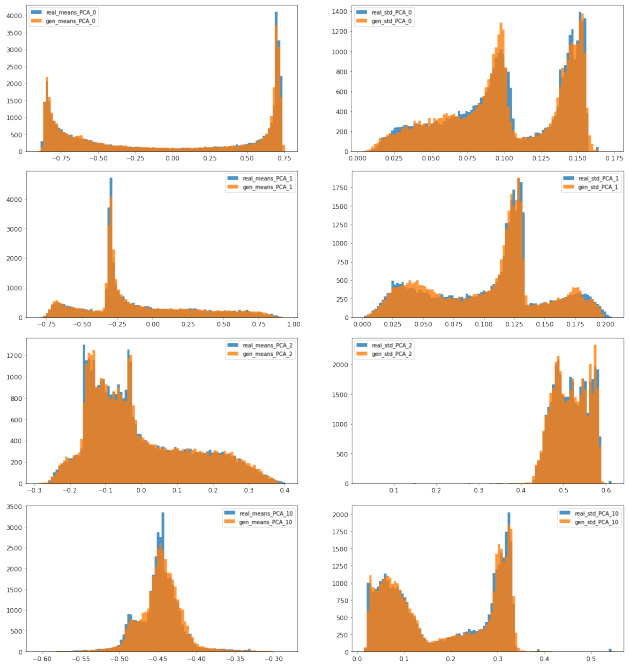


Figure 13. DCGAN final epoch: PCA coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different PCA coefficient.

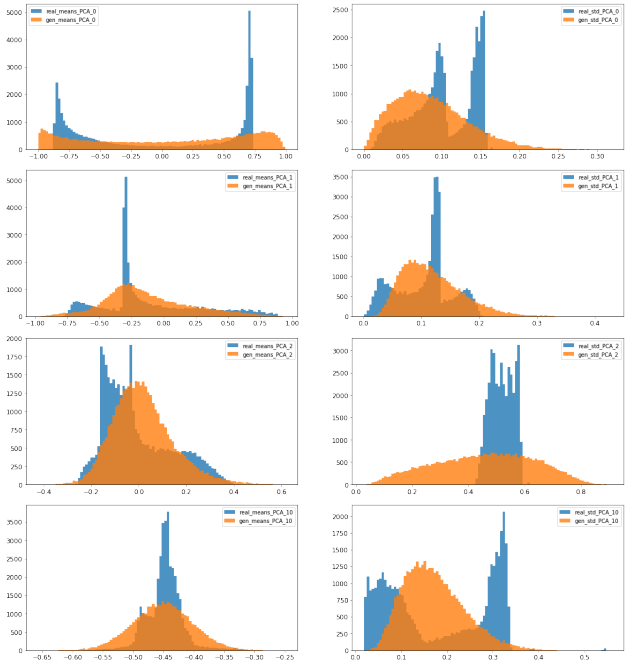


Figure 14. WGAN-GP first epoch: PCA coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different PCA coefficient.

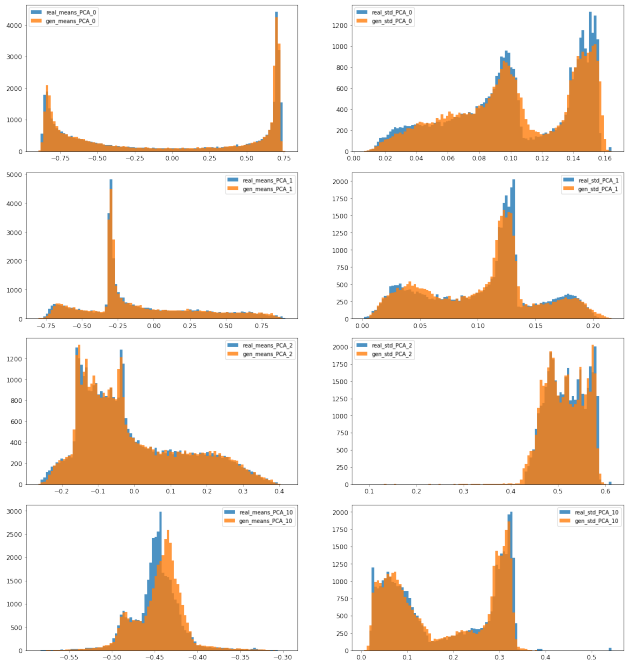


Figure 15. WGAN-GP final epoch: PCA coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different PCA coefficient.

to each other). Therefore, as each sample represents roughly a day, we are looking the average and the corresponding variation of the PCA coeffients on each day.

H. Hardware configuration

A Linux (Ubuntu 18.04.6 LTS) workstation was used to train the machine learning models and run all the numerical simulations. Table 2 shows the hardware configuration.

Table 2. Hardware configuration.

Description	Quantity
16GB DDR4 3200 MHz RAM Memory	16
Samsung 970 EVO PLUS 2TB SSD/Solid State Drive	1
Seagate IronWolf PRO 4TB SATA HDD/Hard Drive	3
Nvidia Quadro RTX 4000 Video Card	1
AMD 32 Core 2nd Gen EPYC 7452 CPU/Processor	2
AMD EPYC 7000 EATX Gigabit Server Motherboard	1