MDPI

Article

Multi-Output Regression with Generative Adversarial Networks (MOR-GANs)

Toby R.F. Phillips [1,*], Claire E. Heaney [1,2], Ellyess Benmoufok [3], Qingyang Li[3], Lily Hua[4], Alexandra E. Porter[5], Kian Fan Chung[6] and Christopher C. Pain [1,2]

1 Applied Modelling and Computation Group, Department of Earth Science and Engineering, Imperial College London, London SW7 2AZ, U.K.; c.heaney@imperial.ac.uk (C.E.H.); c.pain@imperial.ac.uk (C.C.P.)
2 Centre for AI-Physics Modelling, Imperial-X, Imperial College London, London, W12 7SL, UK
3 Department of Earth Science and Engineering, Imperial College London, London, SW7 2AZ United Kingdom; ellyess.benmoufok19@imperial.ac.uk (E.B.),qingyang.li19@imperial.ac.uk (Q.L.)
4 Department of Chemistry, Imperial College London, London, SW7 2AZ United Kingdom; l.hua20@imperial.ac.uk (L.H.)
5 Department of Materials, Imperial College London, London, SW7 2AZ United Kingdom; a.porter@imperial.ac.uk (A.E.P.)
6 Faculty of Medicine, National Heart & Lung Institute, Imperial College London, London, SW3 6LY United Kingdom; f.chung@imperial.ac.uk (K.F.C.)
* Correspondence: t.phillips18@imperial.ac.uk (T.R.F.P.)

Abstract: Regression modelling has always been a key process in unlocking the relationships between independent and dependent variables that are held within data. In recent years, machine learning has uncovered new insights in many fields, providing predictions to previously unsolved problems. Moreover, Generative Adversarial Networks (GANs) have been widely applied to image processing, producing good results. However, these methods have not often been applied to non-image data in small dimensional spaces. Seeing the powerful generative capabilities of the GANs, we explore their use, here, as a regression method. In particular, we explore the use of the Wasserstein GAN (WGAN) as a multi-output regression method. The resulting method we call Multi-Output Regression GANs (MOR-GANs) and its performance is compared to a Gaussian Process Regression method (GPR) - a commonly used non-parametric regression method that has been well tested on small dimensional datasets with noisy responses. The WGAN regression model performs well for all types of datasets and exhibits substantial improvements over the performance of the GPR for certain types of datasets, demonstrating the flexibility of the GAN as a model for regression.

Keywords: Generative Adversarial Networks; Wasserstein GAN; Regression; Multi-output regression; Multi-modal distributions

Citation: Phillips, T.R.F.; Heaney, C.E.; Benmoufok, E.; Qingyang, L.; Hua, L.; Porter, A.E.;Chung, K.F.; Pain, C.C. Multi-Output Regression with Generative Adversarial Networks (MOR-GANs). *Appl. Sci.* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Regression is a statistical technique which aims to find and describe relationships that exist between inputs (the independent variables also known as predictors, covariates, features) and outputs (dependent variables also known as responses, targets, outcomes). An abundance of data has enabled machine learning techniques to be successfully applied to regression modelling. Data from observations or experiments often comes from complex nonlinear systems that are challenging to model, therefore, a regression model that is able to model uni- or multi-modal distributions, single or multi-output regression problems and quantify uncertainty is highly desirable. Borchani *et al* [1] highlight two challenges for regression: (1) modelling uncertainty, both handling the uncertainty in the data itself, but also in quantifying the uncertainty in the responses; and (2) identifying co-dependencies between response variables (for multi-output regression problems). Two approaches are commonly used for multi-output regression problems: transforming the problem and applying single-output methods, and developing extensions to single-output regression

methods (such as kernel methods, regression trees and support vector regression) so they are capable of analysing multi-output distributions [2]. Although the former is more straightforward, the latter, when possible, gives better results. In this paper, we propose a generative model for performing regression. This model is flexible as it can be applied (without modification other than hyperparameter tuning) to uni- and multi-modal data; multiple regression problems; single- and multi-output regression tasks (including co-varying responses); and data with uncertainty or noise. It can also be used to calculate the uncertainty associated with a prediction. We compare this method to Gaussian Process Regression (GPR) which has performed well for regression problems. GPR is a machine learning technique based on Gaussian Processes introduced by Rasmussen and Williams in 1996 [3]. A probability distribution is defined, rather than a single-valued function, which can be applied to data where a range of responses can come from a single point in the regression phase space. Feed-forward neural networks give a single response for a given input, whereas, both GPR and the method proposed here can give multiple responses for a single input enabling the uncertainty in the response to be quantified. This is a highly desirable feature for a regression method.

Generative models were originally developed with the aim of creating a network that could generate realistic examples, that is examples that appear to be drawn from the distribution which was used to train the model. A powerful generative model is the Generative Adversarial Network (GAN) introduced in 2014 by Ian Goodfellow [4]. GANs have quickly become one of the most popular generative models and are widely used in image processing [5]. Instead of learning a mapping between an input and output determined by training data, these models attempt to learn the distribution underlying the training data (in fact, they learn a mapping from a simple distribution to the more complex distribution which describes the training data). This property is desirable as we would like to avoid extrapolating because it can lead to unreliable results. GANs are well known for generating images that are capable of tricking the human eye into believing that it is seeing genuine data [6]. A GAN consists of two neural networks, a generator and a discriminator that are trained simultaneously according to a min-max game. The generator and discriminator adopt the structure of popular neural networks [7–9]. Although many studies have explored the idea of using GANs when manipulating or identifying images, little research currently exists around implementing GANs to generate non-image data with a targeted distributions. One exception is Jolaade *et al* [10] who apply GANs to the time series prediction of fluid flow. Furthermore, GANs have shown to be able to perform well even with small samples of data [11], making them a reliable technique and suitable for regression in these circumstances.

Due to the structure of GANs, the independent and dependent variables appear in the output of the generator (whereas for feedforward networks, the independent variables would be more likely to appear in the output, and the dependent variables in the output). The input of a GAN is a set of random variables, and it generates a realistic sample from these random variables. For regression problems, although sampling the latent space will give a good idea of the distribution learned by the generator, it can also be desirable to be able to obtain a response at a particular value of the independent variable. In order to do this, we propose a prediction algorithm which involves minimising the difference between the output of the GAN for the independent variable and its desired value. This prediction algorithm has been used previously to enable a GAN to make time series predictions [10,12]. It is somewhat similar to an algorithm presented by Wang *et al* [13], which searches the latent space in order to match a given image with an image produced by the generator. The necessity for these algorithms comes about because the output of the GAN contains both the independent and dependent variables.

In this paper, we develop a new regression method based on GANs and show how it compares to a state-of-the-art GPR regression method by testing them on a range of datasets. The regression method we develop is based on a Wasserstein GAN (WGAN) [14]. WGAN methods are not widely used for regression in the literature with the exception of

Aggarwal *et al* [15] who use a Conditional GAN (CGAN) and McDermott *et al* [16] who use a semi-supervised Cycle Wasserstein Regression GANs (CWR-GAN).

The contributions of this article are the use of a WGAN to perform regression; the ability to apply this model to multi-modal data and multi-output regression (MOR-GAN) tasks with no modifications required; the presentation of a prediction algorithm to be used with the trained WGAN in order to predict a response for a given independent variable; the exploitation of the WGAN's critic to provide a confidence level for the predictions made by the WGAN's generator.

The remainder of the paper is organised as follows: Section 2 describes the methods used in this paper, Section 3 presents results from the synthetic example problems and Section 4 shows results from an *in vitro* study and conclusions are drawn in the final section.

2. Methods

2.1. Data Generation

We investigate the performance of Gaussian Process Regression (GPR) and Wasserstein GAN (WGAN) models for regression using a number of datasets. Simple functions were used to generate all but one of the datasets, which have different properties, including with and without additive Gaussian noise (which here, represents uncertainty in the data); one- or two-dimensional examples; uni- and multi-modal distributions; single or multi-output regression; and, for the WGAN model, we explore both random inputs and constrained inputs (where input refers to the independent variable or input of the regression problem not the input of the WGAN). The final dataset is taken from an *in vitro* study and explores the influence of silver nanoparticles on cells taken from the lungs. Following standard practice, preprocessing was applied to all the datasets to ensure that no bias is introduced due to different variables having different ranges of values. This was done by applying a linear mapping to normalise the values so they were in the range $[-1, 1]$.

2.2. Gaussian Process Regression

GPR is a machine learning technique, based on Bayesian theory and statistical learning which has wide applicability to complex regression problems with multiple dimensions and non-linearity [17]. The basic theory of prediction with Gaussian processes dates back to the 1940s [18,19], and, since then, there have been many developments and insights gained into using Gaussian Processes as a regression technique. For example, Sacks *et al* [20] introduced GPR for computer experiments and used parameter optimisation in the covariance function and also applied it to experimental design, i.e. the choice of input that provides the most information. Moreover, Rasmussen *et al* [17] described GPR in a machine learning context, and expressed the optimisation of the GPR parameters in terms of co-variance functions.

GPR has become an effective, non-parametric Bayesian approach that can be applied to regression problems and can be utilised in exploration and exploitation scenarios [21]. Instead of inferring the distribution of parameters, non-parametric methods can directly predict the distribution of functions. Gaussian Process Regression starts with a set of prior functions based on a specified kernel. After incorporating some known function values (from the training dataset), a posterior distribution is obtained. The posterior can then be evaluated at points of interest (from the test dataset) [17].

A python library `GPy` was used to perform the GPR [22]. Important to the performance of the GPR is the choice of kernel. Here we use a radial basis function (RBF) kernel which has three hyperparameters; length, kernel variance, and the standard deviation of the Gaussian noise. These hyperparameters are automatically tuned via `GPy`.

2.3. Generative Adversarial Networks

A Generative Adversarial Network (GAN) consists of two neural networks: a generative model or generator, G, and a discriminative model or discriminator, D. The models are trained simultaneously resulting in a generator that can produce samples which appear to be taken from the same distribution as the training data. During training, the generator

tries to fool the discriminator that it is generating real data, see [4]. For each data point the following combined loss function is defined for G and D:

$$L = \min_G \max_D [\log(D(\boldsymbol{x})) + \log(1 - D(G(\boldsymbol{z})))] \tag{1}$$

where x is a sample from the real data, $x \in \mathbb{P}_r$ and z represents the latent variables. The generator and discriminator are essentially playing a two-player min-max game through the corresponding function $V(G, D)$ [4]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2}$$

GANs are notoriously difficult to train, often reported to suffer from mode collapse and the vanishing gradient problem [23]. Mode collapse occurs when the generator G produces only one solution, or a limited set of solutions, which is/are able to fool the discriminator, and the vanishing gradient problem is described below (Section 2.4).

2.4. Wasserstein Generative Adversarial Networks

The WGAN [14] was developed in order to alleviate the issue of the vanishing gradient problem. To measure the distance between probability distributions, rather than use the Jensen-Shannon (JS) divergence (expressed by Equation (1)) as in the GAN, Arjovsky *et al* proposed the Earth-Mover (EM) or Wasserstein-1 distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|], \tag{3}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ (real data) and $\mathbb{P}_g$ (generated data) [14]. The Wasserstein-1 distance is able to provide a similarity measure between two probability distributions, even when the two probability distributions have no overlap, making it a more sensible cost function. The discriminative model is renamed the **critic** in the WGAN, as it is not explicitly attempting to classify inputs as real or fake, but rather to determine how real an input is. The WGAN value function is constructed via the Kantorovich-Rubinstein duality as Equation (3) is computationally intractable [14]:

$$\min_G \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] \tag{4}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions. To enforce the Lipschitz constraint, weight clipping was originally used by Arjovsky *et al* [14], who stated that this method of enforcement was terrible, despite it working well for the examples shown in their paper and was, at least, simple. Gulrajani *et al* [24] introduced an improvement to weight clipping, by enforcing the Lipschitz constraint with a **Gradient Penalty (GP)** method. By enforcing a soft version of the constraint with a penalty, the new loss function becomes:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{loss of the critic}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{gradient penalty}}. \tag{5}$$

Throughout this study we enforce the Lipschitz constraint by using the GP method in our WGAN models.

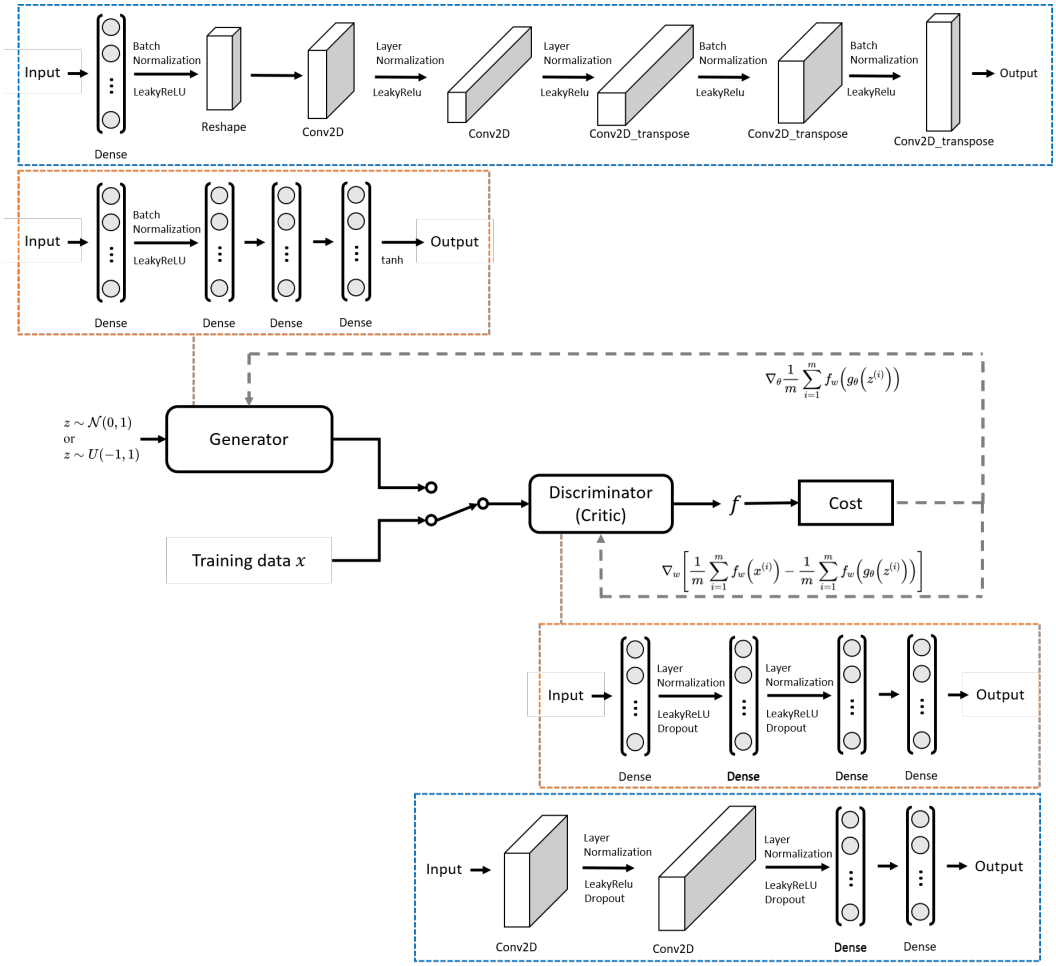


Figure 1. WGAN Structure. The architectures of the Generator and Critic are shown above and below the WGAN structure respectively. The equations displayed are the losses used to update each component. The orange-boxed structure used for the **single-output regression** problems is a multilayer perceptron network. The blue-boxed structure used for **multi-output regression** has convolutional layers.

2.5. Regression with WGAN

GANs are a type of generative network, the aim of which is to generate realistic looking samples that appear to have been drawn from the same distribution as the training data. The input to a WGAN (and GANs in general) is a set of random numbers (not related to the data) and the output contains the generated sample. Consider simple regression, where a relationship is sought between an independent variable x (also known as covariate or feature) and a dependent variable y (also known as a response). The WGAN is trained to produce both the independent and dependent variables of the regression problem (x, y) as its output. In contrast, for feed-forward networks, the input of the network often takes the independent variable and the output, the dependent variable. Therefore, in order to specify a particular value for the independent variable, a prediction algorithm is introduced to the WGAN.

Suppose we have trained the WGAN and wish to use it to predict the output at a given value of the independent variable x_p. First, the latent variables (z) are set to random numbers. The generator is evaluated at these values, $G(z)$, producing a pair of values (x, y) — the input and output or response in our regression problem. The difference between x and x_p is then minimised with respect to the latent variables. Once this is done, we assume that the output of the generator closely approximates (x_p, y_p). The minimisation can be done efficiently by using the same software libraries that are used for back-propagation

during training. This procedure means that we can generate multiple outputs for one input, x_p, by starting from different random states for the latent variables, and we can therefore produce a distribution of values which reflect the uncertainty in the output y_p. This procedure is detailed in Algorithm 1 and introduces a projection operator, Proj, which projects the output of the WGAN onto a space that contains only the variables that are to be constrained. For the example described in this paragraph, the projection operator would be represented by the matrix $[1\ 0]$ for an output of the generator in the form $(x, y)^T$.

Simple regression (for a single independent variable), multiple regression (for more than one independent variable) and multi-output or multi-variate regression (for more than one dependent variable) can all be performed by the WGAN and demonstrated by the results in this paper. Due to the generative nature of the WGAN, both independent and dependent variables are contained in the output of the generator which means that, when randomly sampling the latent space of the generator to produce an output, we have no control over the particular value of independent variable. In order to specify particular values of independent variable(s), a prediction algorithm is used, described in Algorithm 1. So there are two ways of using the WGAN, either with random inputs (for the independent variable) or constrained values:

- **Random input:** Random variables are assigned to the latent space from which the generator of the WGAN yields a realistic output of a n-tuple of the independent and dependent variables associated with regression problem. By sampling the generator many times, this can be used to assess the probability density function learned by the generator. The value of the independent variable(s) cannot be controlled, however, as they are an output of the generator. Although random inputs allow us to see the distribution learned by the generator, having the facility to constrain the independent variables is an important feature.
- **Constrained input:** An algorithm is used in conjunction with the (trained) WGAN to find predictions for given value(s) of the dependent variable(s). This results in a property similar to a GPR, where, for example, the independent variables are inputs of the GPR (and can be prescribed) and the outputs are dependent variables.

Algorithm 1 Prediction Function. Built to be used in conjunction with the trained WGAN, to constrain the independent variable of the regression problem.

Require: The desired value of the independent variable $\boldsymbol{x}_p$, initial values of the latent variables $\boldsymbol{z}^{(1)} \sim N(0, 1)$, trained generator G, number of iterations N.
for i=1, ..., N **do**
  $\tilde{\boldsymbol{x}}^i = G(\boldsymbol{z}^{(i)})$ ▷ Output of GAN from latent space of iteration i
  $\epsilon = \text{Proj}(\tilde{\boldsymbol{x}}^i - \boldsymbol{x}_p)$ ▷ Work out error between GAN output and the desired value
  $\boldsymbol{z}^{i+1} \leftarrow BackPropagation(\boldsymbol{z}^i, \epsilon)$ ▷ Adjust latent space by backpropagating the error
end for

2.6. WGAN architectures

The WGAN models were constructed using `Keras` [25]. The generator is a four-layered network, as displayed in the top orange box in Figure 1, which takes Gaussian distributed noise from the latent space as an input, and outputs the x and y coordinates in a 1D regression problem. The first dense layer employs batch normalisation and the leaky rectified linear activation functions (LeakyReLU) followed by fully-connected dense layers. The last layer applies the non-linear *tanh* activation function. The structure of the critic is also a four-layered network, with a reduced number of neurons. Its input is data from the training set and data generated from the generator. To reduce the likelihood of overfitting, dropout with a probability of 0.2 is applied to the critic. Layer normalisation is employed for the critic as opposed to batch normalisation, as the latter inhibits the performance of the gradient penalty term in Equation (5).

The MOR-GAN used in Section 3.4 for the co-varying spiral dataset follows the same architecture described in the previous paragraph but with certain fully-connected layers

replaced by convolutional layers. In fully-connected or dense layers, every neuron in the input is connected to every neuron in the output. Instead, convolutional layers apply filters to the input where only neurons close to each other are connected to the output.

2.7. Visualisation

To compare the models, the predictions of a test set are visualised. Assuming we have a regression problem in which x is on the horizontal axis and y on the vertical axis and a point on the graph is represented by (x, y). The GPR outputs are the randomly sampled x values and the associated y values from the GPR posterior distribution. On the other hand the WGAN outputs a prediction of both the x and y values. After training, the generator will produce an output of (x, y) when given a value(s) of the latent variable(s) z.

3. Results from Synthetic Datasets

In this section we present results for the performance of the GPR and WGAN approaches for regression on a number of synthetic datasets. We would like our model to perform well on different types of dataset, so datasets with different properties are used here, including uni- and multi-modal distributions, one- or two-dimensional inputs (or independent variables), single- or multi-output. The WGAN can also be used in two ways: with a random input or a constrained input. These combinations are given in Table 1. For the datasets with noise, 500 samples are taken. All the models here follow the general architecture in the orange box of Figure 1 with slight variations on the number of nodes in each dense layer.

		distribution of dataset			**input type**	**Section**
	dimension	noise	type	output		
sine wave	1D	✓	uni-modal	single output	random	3.2.1
heteroscedastic	1D	✓	uni-modal	single output	random	3.2.1
circle	1D	✓	multi-modal	single output	random	3.2.2
sine wave with lines	1D	✓	multi-modal	single output	random	3.2.2
distance	2D	✗	uni-modal	single output	random	3.2.4
helix	2D	✓	multi-modal	single output	random	3.2.4
sine wave	1D	✓	uni-modal	single output	constrained	3.3
heteroscedastic	1D	✓	uni-modal	single output	constrained	3.3
circle	1D	✓	multi-modal	single output	constrained	3.3
eye	1D	✗	multi-modal	multi-output	constrained	3.4.1
spiral	2D	✗	uni-modal	multi-output	constrained	3.4.2

Table 1. Properties of the synthetic datasets

3.1. Training

The training process of the WGAN is described in Algorithm 2. Training a WGAN can be easier than training a GAN, due to the former's removal of the issues associated with mode collapse and weight clipping. Nonetheless, there are still many factors (neural network architecture and training hyperparameters) that can be optimised during training. See Table 2 for the set of hyperparameters that we use for WGAN training. Some values were found by hyperparameter optimisation; others were informed by the literature. For example, $\lambda = 10$ and $n_{\text{critic}} = 5$ are commonly used settings and have been shown to work well across a range of datasets and architectures [14,26].

Algorithm 2 WGAN with gradient penalty and sample-wise optimisation. All experiments in the paper used the default values $\lambda = 10, \alpha = 0.0001, n_{\text{critic}} = 5, \beta_1 = 0.5, \beta_2 = 0.9$. This algorithm is a modified version of the one displayed in the paper by Gulrajani *et al* [24].

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters ω_0, initial generator parameters θ_0.
  while θ has not converged **do**
    for $t = 1, \ldots, n_{\text{critic}}$ **do**
      for i=1, $\ldots, m$ **do**
        Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0,1]$.
        $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
        $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$
        $L^{(i)} \leftarrow D_\omega(\tilde{\boldsymbol{x}}) - D_\omega(\boldsymbol{x}) + \lambda(||\nabla_{\hat{\boldsymbol{x}}} D_\omega(\hat{\boldsymbol{x}})||_2 - 1)^2$
      end for
      $\omega \leftarrow \text{Adam}(\nabla_\omega \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \omega, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_\omega(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
  end while

Hyperparameters	**Single-output**	**Multi-output**
Learning rate	10^{-3}	10^{-4}
Number of Critic iterations per Generator iterations	5	5
Batch size	100	32
Latent Space Dimension	3	3 (3, 6, 12 used for spiral problem)
Adam optimiser hyperparameters (decay rates of moving averages)	0.5 & 0.9	0.5 & 0.9
Gradient penalty hyperparameter λ	10	10

Table 2. Hyperparameters used in the construction and training of our WGANs for both the single-output and multi-output distributions.

3.2. Single-Output Regression with Random Input Values

In this section, we sample the posterior of the GPR at random points. For the WGAN, we randomly sample points in the latent space which leads to outputs of n-tuples which are the inputs and responses of the regression problem. We do not control which values are taken by the independent variables(s) or inputs when using regression with randomly generated inputs. The test or sample data is generated by evaluating the functions used to create the training data with randomly generated independent variables. Therefore, the three sets of results have different values of the independent variable(s).

3.2.1. 1D Uni-Modal Examples

To generate a sinusoidal dataset with uncertainty, we use the function

$$y = \sin(x) + \eta\phi \quad \text{where} \quad \phi \sim N(\mu, \sigma) \tag{6}$$

where N is a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. The uncertainty is represented by Gaussian noise through the term ϕ and its magnitude is adjusted by a scalar $\eta \in [0, 1]$. We can see from Figure 2 that the random sampling from WGAN and GPR both match well to the test data. For the sinusoidal dataset, the WGAN structure shown in Figure 3 is used, and for the remaining problems in this section, we increase the number of neurons, see Figure 4.

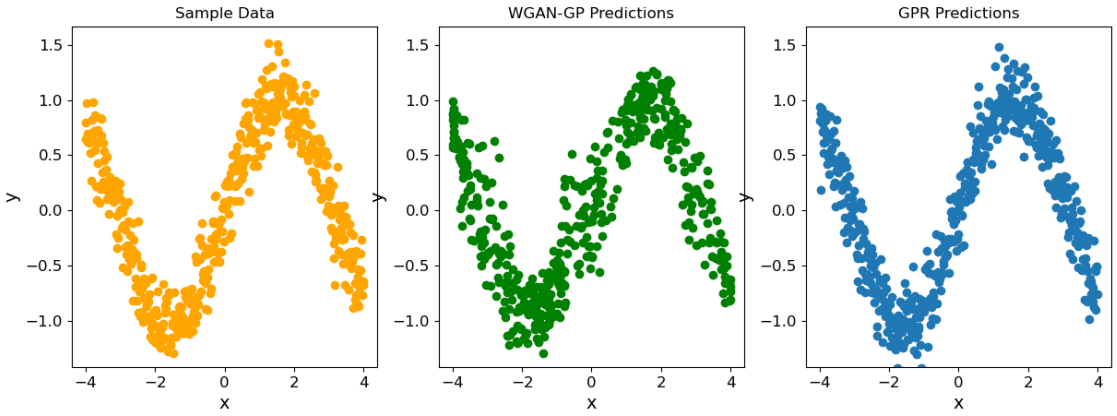


Figure 2. Sinusoidal dataset with added noise ($\eta = 0.2$, see Equation (6)). The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right

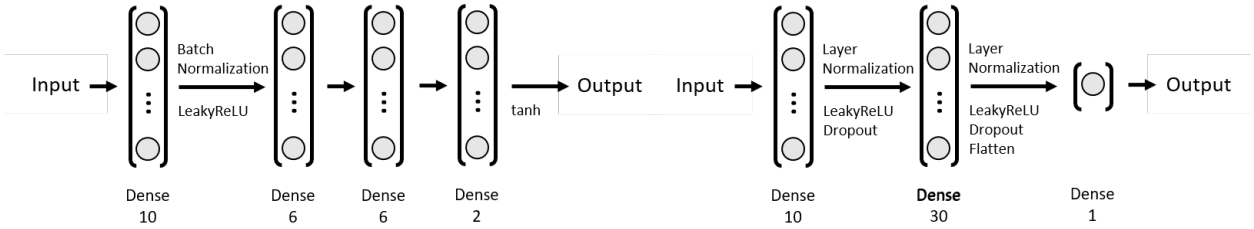


Figure 3. The structure of the generator (left) and critic (right) for the sinusoidal datasets.

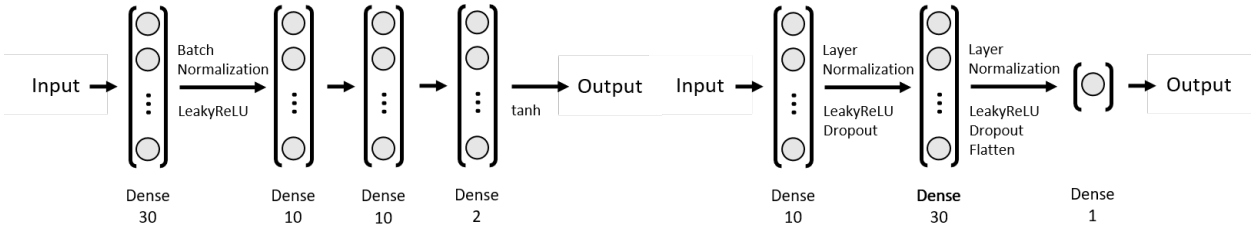


Figure 4. The structure of the generator (left) and critic (right) for the majority of the problems in this section.

The previous example modelled uncertainty by using noise that was independent of x. To test the WGAN model more thoroughly, a heteroscedastic dataset is used where the noise increases with increasing x. Figure 5 shows that the WGAN model is capable of modelling the variation in noise accurately, whereas the GPR, with a single kernel size representing the probability density function, is unable to do so. We note that there is a variant of GPR called Heteroscedastic GPR [27], which has been designed to handle intricate changes in noise. Implementing this method would result in a better performance of the GPR. However, here we aim to avoid tailoring methods to different datasets, so that we can demonstrate the flexibility of the single WGAN model.

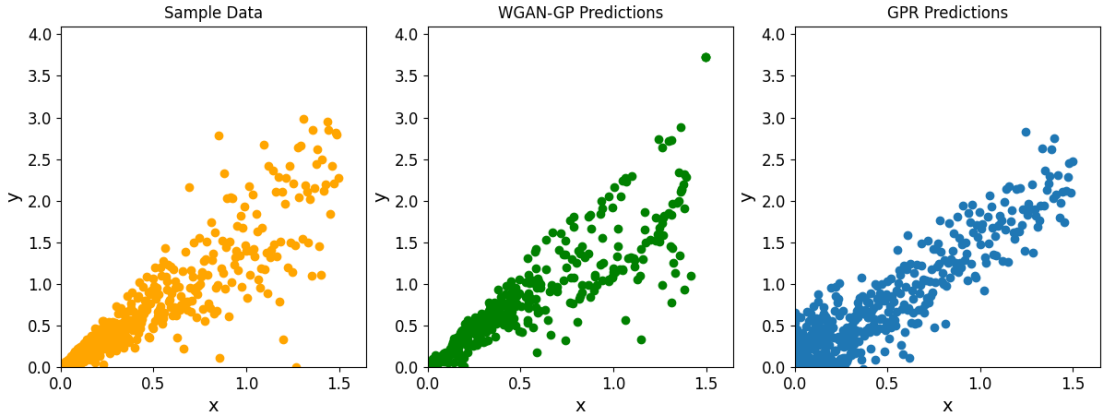


Figure 5. Heteroscedastic dataset. The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right.

3.2.2. 1D Multi-Modal Examples

Here we explore the use of WGAN and GPR to perform regression of multi-modal distributions. The WGAN models in this section use the architecture displayed in Figure 4. For the first multi-modal distribution, a uniform distribution of data points is generated within an annulus (i.e. between two concentric circles) as shown in Figure 6 (left). There is a significant difference in the performance of the GPR and WGAN. Whilst the WGAN captures the distribution very well (see Figure 6 (middle)), the GPR is unable to represent it (see Figure 6 (right)), predicting an almost uniform distribution of points.

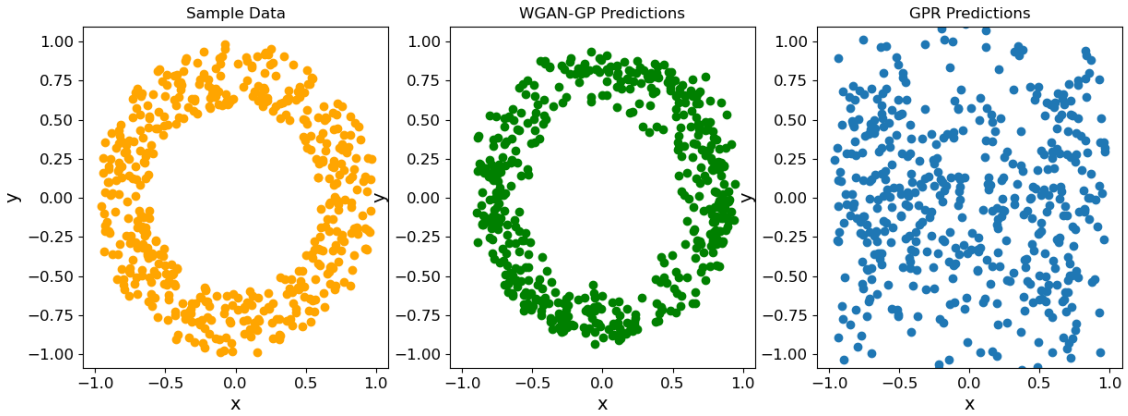


Figure 6. Annulus dataset. The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right.

The second multi-modal distribution is a sinusoidal wave with several intersecting lines. The same trends appear as seen when using the annulus dataset: the WGAN outperforms the GPR, which is unable to detect the gaps that exist in the dataset, see Figure 7. The overall profile of the data is visible, but within the bounds of the minimum and maximum y values there is no gap. Although GPR struggles with these complex functions, it has been used and built upon to work on clustering complex functions [28], so there is the capability of modelling these types of complex functions. However, we wish to compare the WGAN against one model, without tailoring it for different types of data.

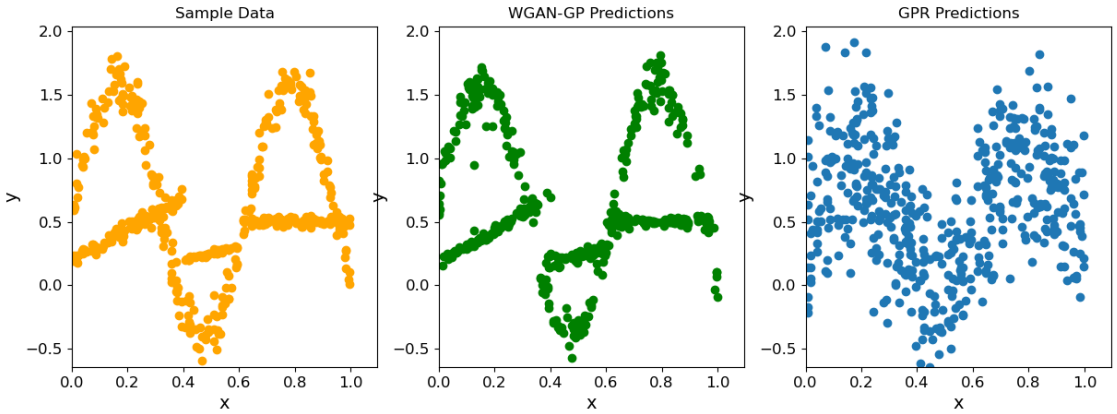
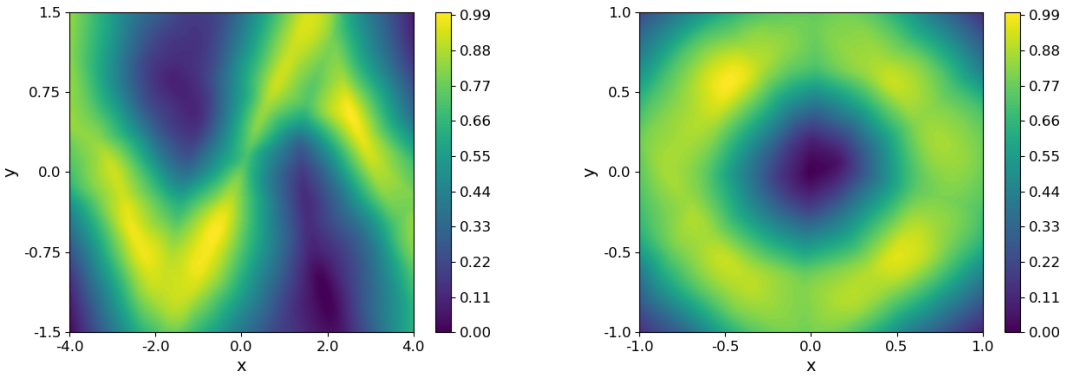


Figure 7. A sine wave intersected by several lines. The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right.

3.2.3. Confidence of solutions from the critic

Sections 3.2.1 and 3.2.2 show how sampled points produced by the generator of the WGAN match the distribution seen in the test data (or sample data). During the training of the WGAN, the critic learns to determine how real an sample is. This section demonstrates how the critic can be used to determine the confidence in a sample produced by the generator.



(a) The sinusoidal dataset. **(b)** The annulus dataset.

Figure 8. Contour plots showing the values of the two critics for the sinusoidal and annulus datasets. These indicate the 'confidence' in the predictions and also indicate where extra training data may be required.

Figure 8 shows the value taken by the critic for predictions or responses made throughout domain for both the sinusoidal and annulus datasets. These are produced by finding the value of the critic for each point on a 100×100 grid that covers the same domain as the original data. As previously stated, the critic of a WGAN does not explicitly determine whether a sample is real or fake, but instead, how real a sample is. Therefore the larger the value produced by the critic, the more confidence the model has in the prediction. The critic values shown here are normalised to be between 0 and 1.

Figure 8a shows the values of the critic produced for the sinusoidal dataset. It can be observed that the values are higher for the same coordinates the generator produces in its solutions in Figure 2. These values are higher for larger amounts of data, particularly around $x = -2.0$ and $x = 2.0$. Outside of where the sine wave is located the critic value sharply decreases, therefore confidence in any prediction made here is low.

Figure 8b shows the values of the critic produced by the annulus dataset. It can be observed that the values are higher for the same coordinates that the generator produces in its solutions in Figure 6. Here we can see that the critic produces lower values for coordinates outside of the annulus, meaning that the confidence in predictions or responses that occur here is low.

Figures 8a and 8b demonstrate how the critic can be used in conjunction with the generator to produce a confidence level in the predictions made by the generator. A lower critic value, and therefore a lower confidence, in a prediction made may indicate that extra training data is required there. A possible location requiring extra data for the sinusoidal dataset is $x = 0.5$ and for the annulus dataset is $x = 0.0$.

3.2.4. 2D Uni- and Multi-Modal Examples

Increasing the dimensions in the inputs of the regression problem means the need for a larger neural network, thus the following problems use the structure displayed in Figure 9.

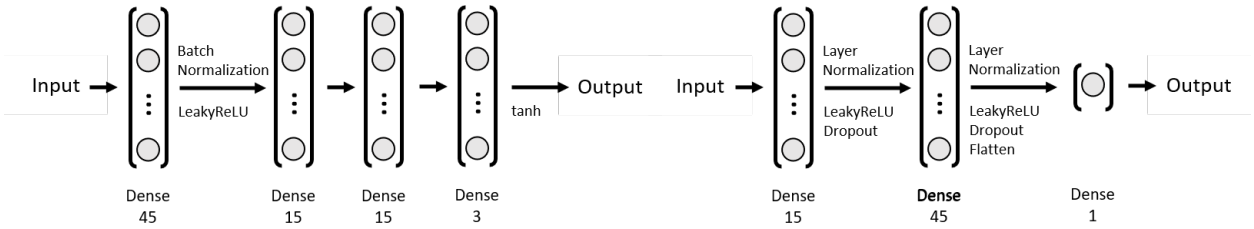


Figure 9. The structure of the generator (left) and critic (right) for the two-dimensional problems.

The performance of the WGAN regression method for data with a single input has been shown to be very reliable. We now test the GPR and WGAN methods on two-dimensional data with a distance function $z = \sqrt{x^2 + y^2}$. The GPR performs exceptionally well, outputting predictions very close to the true model, see Figure 10. The WGAN also performs well, although some deviation from the distance function can be seen.

Having demonstrated that both models are capable of performing regression on datasets with multiple intputs, a more complicated problem is defined as a 2D multi-modal function in the form of a helix with additive Gaussian noise. Figure 11 shows that WGAN is capable of generating data similar to the true model, whereas the GPR struggles to recognise the variation in the z direction and fills the hole in the circle, looking at the $x - y$ plane.

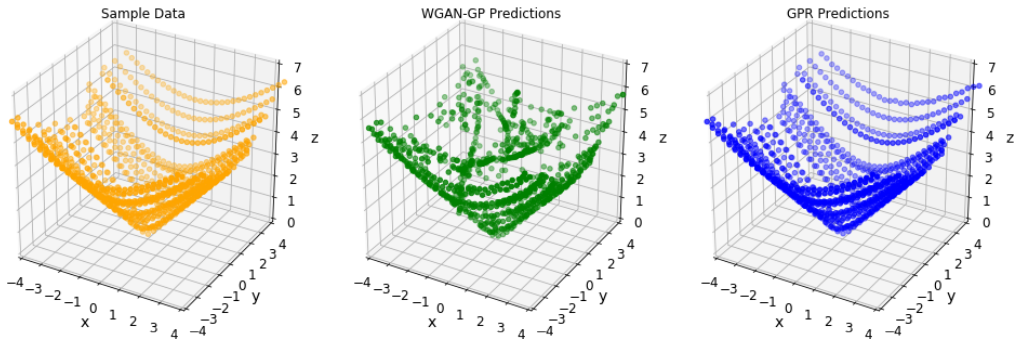


Figure 10. 2D distance function. The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right.

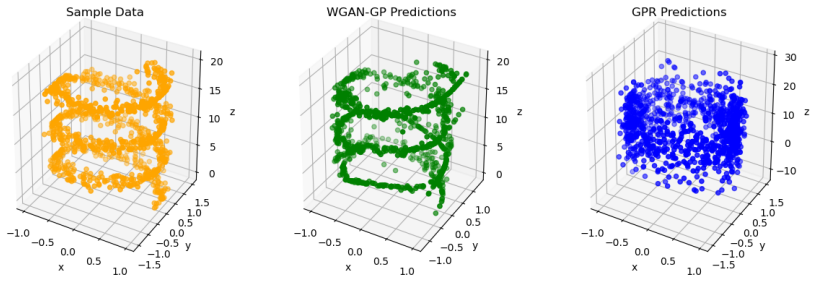


Figure 11. The helix dataset. The test data is shown on the left, sampled points from the WGAN are shown in the middle and sampled points from the GPR posterior are shown on the right.

3.3. Single-Output Regression with Constrained Input Values

A key benefit of using the WGAN for regression is its capability of producing a latent space that with a constrained input, can be optimised to produce multi-modal responses. In Figure 12 we can see the function displaying a few of the potential responses y, at differing fixed x. The WGANs used for the constrained input regression are the same ones used in Sections 3.2.1 and 3.2.2 for their respective datasets.

The way this optimisation is performed is to first randomly generate a latent input vector of the generator. Then from this initial condition point in latent space we apply our optimiser to minimise the least squares functional, see Algorithm 1, which aims to match the latent space with the specified x coordinate. We repeat this multiple times in order to obtain a probability density function for this fixed x coordinate but with differing initial latent space inputs.

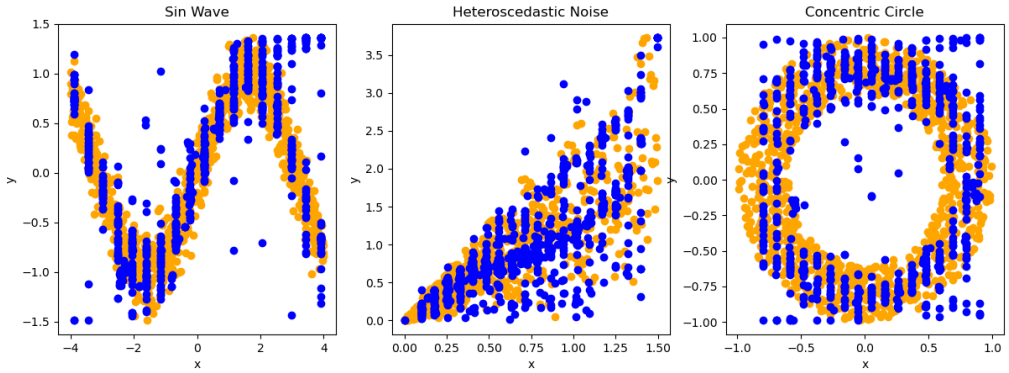


Figure 12. The sinusoidal wave dataset, heteroscedastic noise dataset and annulus dataset predicted at a given values of the x coordinate using the WGAN prediction method. Prediction displays potential responses at the given x coordinates.

3.4. Multi-Output Regression with MOR-GAN

3.4.1. 1D Eye Dataset with Covariance

By taking a digitised, hand-drawn eye and adding a second eye which is obtained by a rotation of 90° and a reflection of the first eye, we produce a distribution which is multi-modal and multi-output or multivariate, see Figure 15. This forms the dataset for the first multi-output regression test. The WGAN is trained to produce two pairs of coordinates, (x_1, y_1) and (x_2, y_2). The model in this section used the following structure:

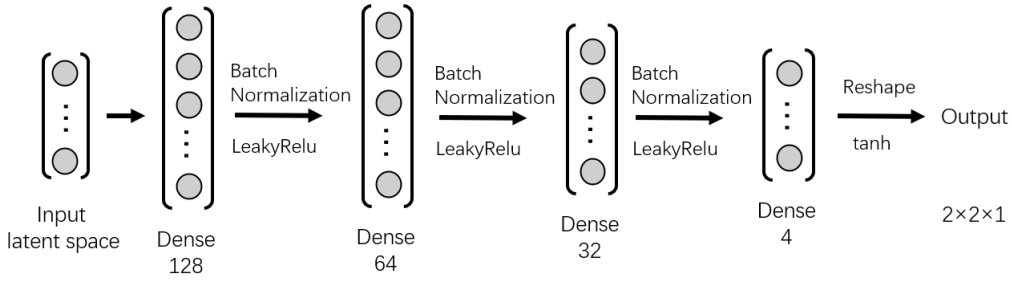


Figure 13. The structure of the generator for the eye dataset

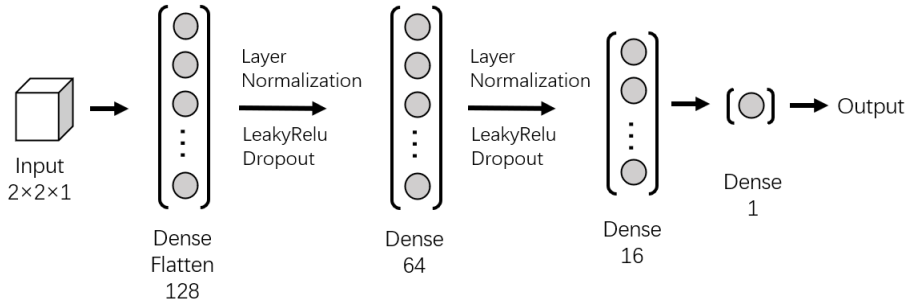


Figure 14. The structure of the discriminator for the eye dataset.

To provide a challenge for the algorithm which enables the WGAN to make predictions a particular values for the independent variable (Algorithm 1), we constrain the value of x_1 (for the non-rotated eye) and predict the corresponding values for y_1 (non-rotated eye), x_2 and y_2 (rotated eye). We repeat this process for every point in the eye dataset to form the image shown in Figure 16. Similarly, we constrain the value of x_2 (for the rotated eye) and predict the corresponding values for y_2 (rotated eye), x_1 and y_1 (non-rotated eye). This is done for every point in the dataset and the result can be seen in Figure 17. The predictions using the MOR-GAN method take into account the known or learned covariance information between the images, enabling the model to determine the second image from all the points in the first image and vice versa. The agreement between the real data and the predicted data using the constrained input is excellent.

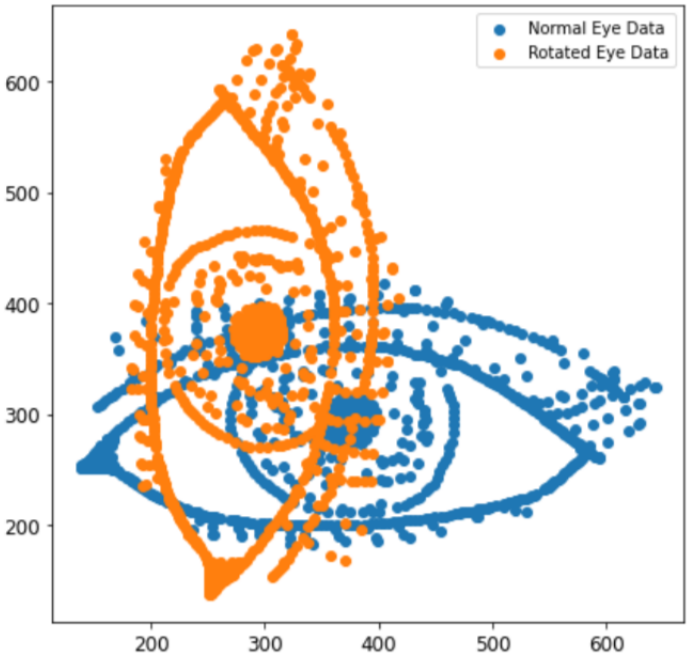


Figure 15. The eye dataset which contains two eyes. One eye is rotated and reflected to produce a second eye.

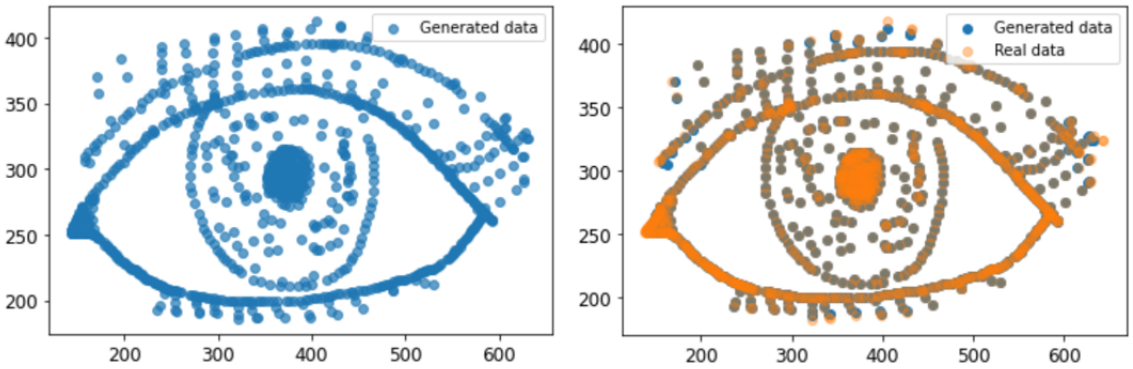


Figure 16. The eye generated by the WGAN (left) and the comparison between the real data and the generated data (right) using the constrained input method described in Algorithm 1.

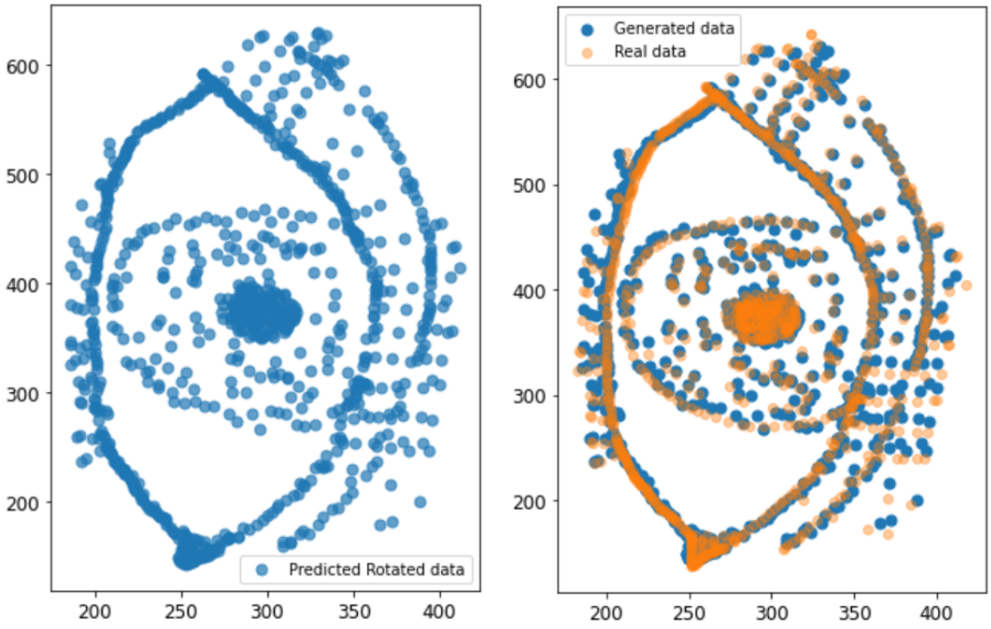


Figure 17. The rotated eye predicted using Algorithm 1 (left) and the comparison between the real rotated data and the predicted data (right).

3.4.2. Co-Varying Spiral Dataset

In the real-world, multiple variables have co-varying effects. In this work, we use a two dimensional spirals dataset as a benchmark to compare the capability of both GAN and WGAN. x and y are the variables that define the spiral at 20 different z levels which are equally spaced with $z \in [0, 4]$. Thus there are 20 pairs of x, y coordinates as the output of the MOR-GAN.

The structure of the model and the hyperparameters of each layer used in this section are displayed in Figures 18 and 19, and Table 3.

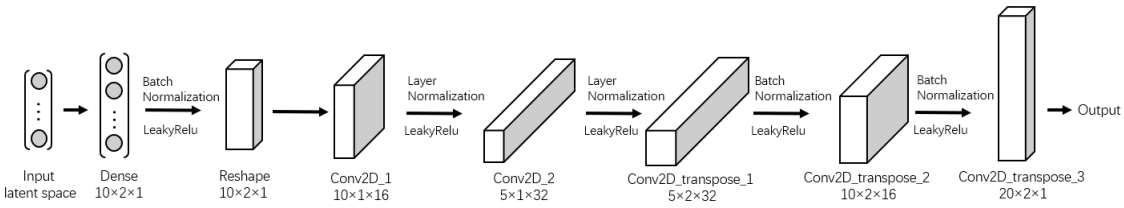


Figure 18. The structure of the generator for the Co-Varying Spiral problem.

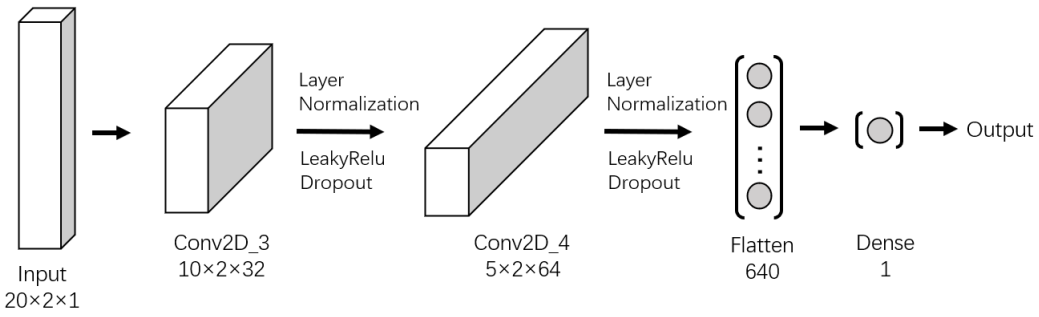


Figure 19. The structure of the discriminator for the Co-Varying Spiral problem.

Layer	Kernel size	Strides	Padding	Use bias
Conv2D_1	$(8, 2)$	$(1, 2)$	same	True
Conv2D_2	$(8, 2)$	$(2, 1)$	same	True
Conv2D_transpose_1	$(8, 2)$	$(1, 2)$	same	False
Conv2D_transpose_2	$(8, 2)$	$(2, 1)$	same	False
Conv2D_transpose_3	$(8, 2)$	$(2, 1)$	same	False
Conv2D_3	$(8, 2)$	$(2, 1)$	same	True
Conv2D_4	$(8, 2)$	$(2, 1)$	same	True

Table 3. Hyperparameters used in the construction of the convolutional neural network.

The three-dimensional spiral curves dataset is generated based on the equations below:

$$x = r \sin\theta, \tag{7}$$
$$y = r \cos\theta, \tag{8}$$
$$z = 4\left(\frac{\theta - a}{b - a}\right), \tag{9}$$

where $\theta \in [a, b]$, $a = 4\pi x_1 - 2\pi$ and $b = 4\pi x_2 + 2\pi$ for x_1, x_2 chosen randomly from the unit interval, and the radius r is chosen randomly from the interval $[0.6, 1]$. For each spiral, r, x_1 and x_2 are chosen at random, and 20 equally-spaced values for θ are chosen from the interval $[a, b]$ to generate the curves shown in Figure 20.

Figure 20 shows the predictions made by the MOR-GAN. The first 10 data points in the spiral, shown as solid blue dots, are used to predict the next 10 data points in the spiral, produced by constraining the output using Algorithm 1. The real spiral is given by the blue line and the spiral generated by the algorithm constraining the first 10 samples is given by the red line. Three different sizes of latent spaces are used and it can be observed that all latent spaces give reasonable reconstructions of the real spirals, therefore demonstrating that the reconstruction reliability of the shape of the curve does not vary much with the increasing dimension of latent space. Figure 20 also shows that the MOR-GAN can learn the structure of the input data and can recreate the shapes (which are spirals in this case) with approximate distributions (which are annular distributions representing the start of the spirals in this case).

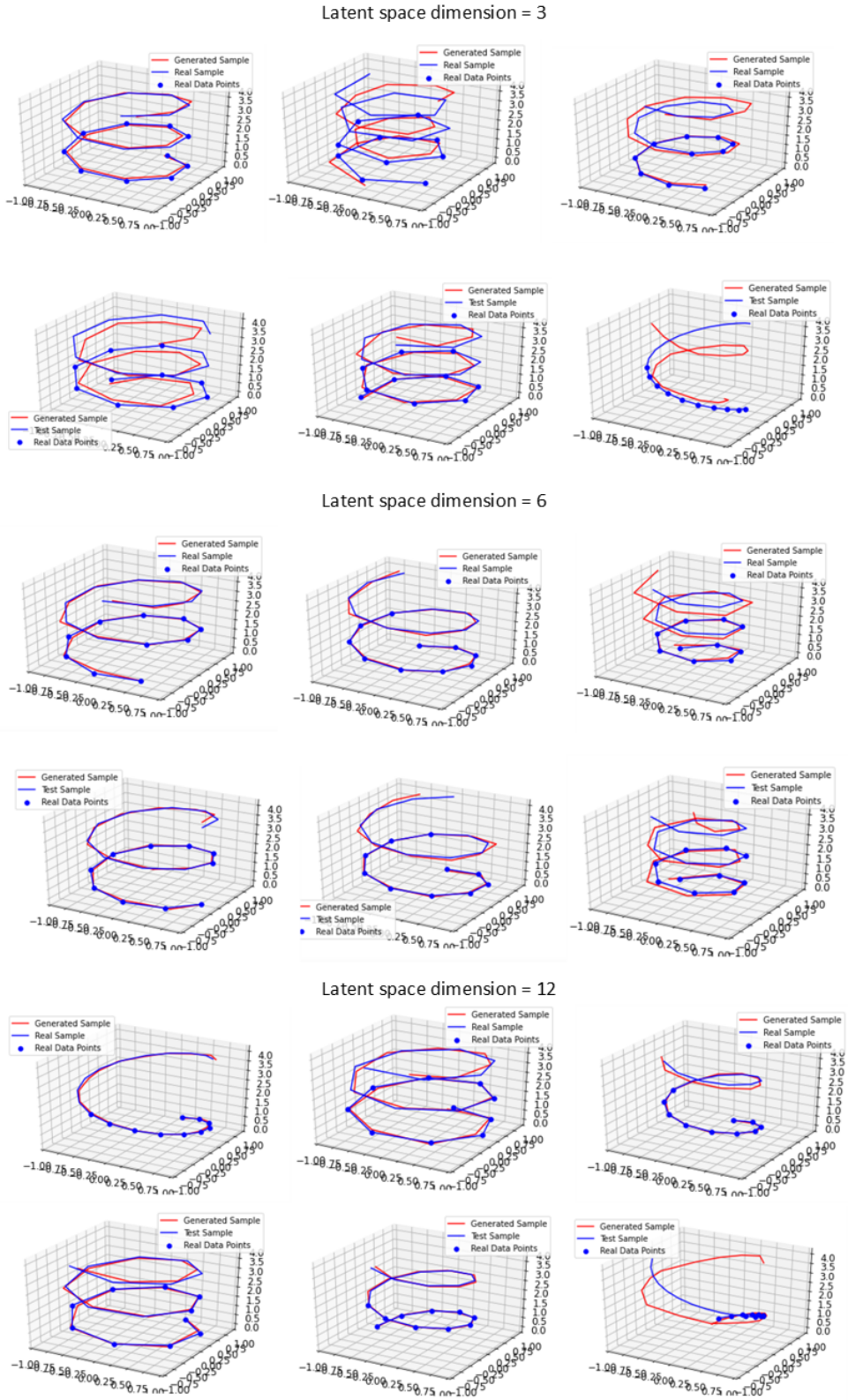


Figure 20. The figures above show the generated data using the prediction function on real samples (rows 1, 3, 5) and test samples (rows 2, 4, 6) when the size of the latent space is 3, 6 and 12 respectively. The blue lines indicate the real spiral, the solid blue dots show the 10 data points that are constrained using Algorithm 1 and the red lines show the spiral produced by the generator for these 10 constrained points.

4. Silver Nanoparticle Data

We now explore the application of WGAN to real-world data regression. Reference [29] explores the effects on cells from the lungs of four types of silver nanoparticles (AgNPs): silver nanospheres (AgNS) of diameter 20 nm and 50 nm; short silver nanowires (s-AgNWs) of length 1.5 µm and diameter 72 nm; and long silver nanowires (l-AgNWs) of length 10 µm and diameter 72 nm. Silver nanoparticles are increasingly used in consumer products and reports state that up to 14% of products containing AgNPs will release these nanoparticles into ambient air [30,31] where they can be inhaled into the lungs of workers and consumers. The work in [29] explores the influence of the nanoparticles on airway smooth muscle (ASM) cells, which are an important component of the airways in the lungs, being responsible for narrowing the airways in conditions such as asthma. Bronchi and tracheas from transplant donor lungs were dissected to obtain the cells. These cells were serum-starved overnight and then incubated with 20 nm or 50 nm AgNSs, or s-AgNWs (5 µg mL^{-1} or 25 µg mL^{-1}) or Ag^{+} ions (0.25 µg mL^{-1} or 25 µg mL^{-1}) for 24 or 72 hours. Change in cell viability assessed by a reduction assay and change in cell proliferation assessed by the rate of DNA synthesis were both measured, and the results are reproduced in Figure 21. Cell viability is defined as the number of live, healthy cells in a sample.

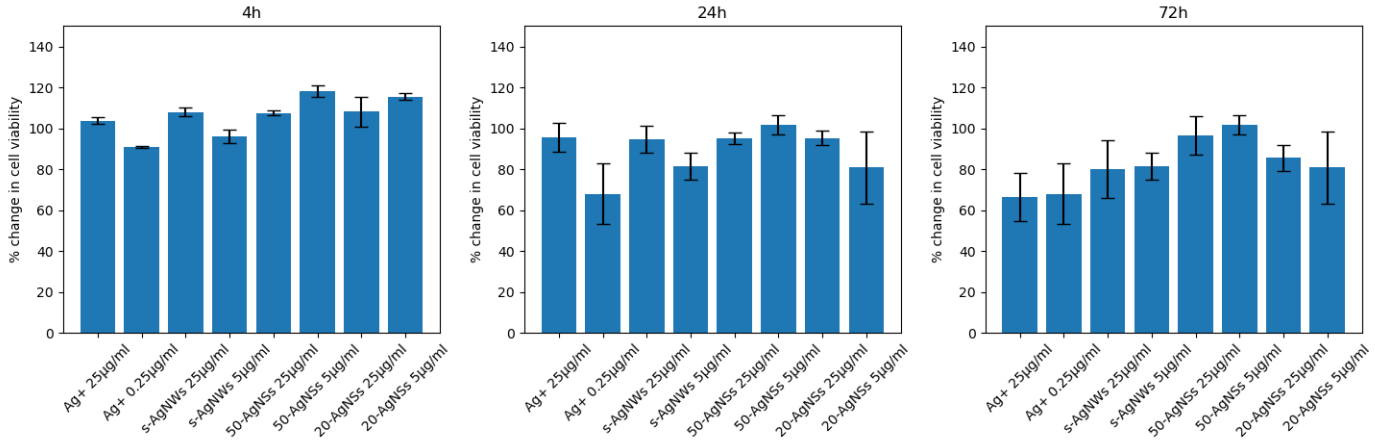


Figure 21. Concentration and time-dependent effect of AgNSs and AgNWs, and Ag^{+} ions on ASM cell viability after 4 hours, 24 hours and 72 hours. The bars represent mean values of 3 ASM cell donors and the whiskers indicate standard error of the mean (SEM). The data is expressed as percentage change with respect to the untreated control. This plot was reconstructed using data from [29].

The data from [29] contains four different molecules analysed at two concentrations analysed over three different times. The molecules were given numerical values based on their specific surface area, defined as the total surface area of a material per unit of mass. This can be seen in Table 4:

Molecule	Specific Surface Area $m^2\,g^{-1}$
Ag +	4.4
s-AgNWs	4.6
50nm AgNSs	6
20nm AgNSs	40.4

Table 4. Specific surface area of the particles formed from different molecules which form independent variables for the WGAN regression.

The generator part of the WGAN was trained to produce four outputs: the specific surface area of the particles containing a specific molecule, the concentration level, the time level (we sample the response at 3 time levels: 4 hours, 24 hours and 72 hours) and the response (change in cell viability). All four outputs were scaled to be between 0 and 1. The WGAN architecture can be seen in Figure 22.

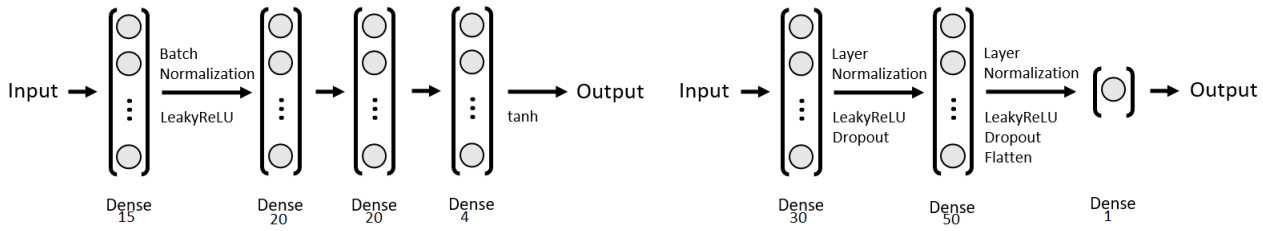


Figure 22. The structure of the generator (left) and critic (right) for modelling the silver data in this section.

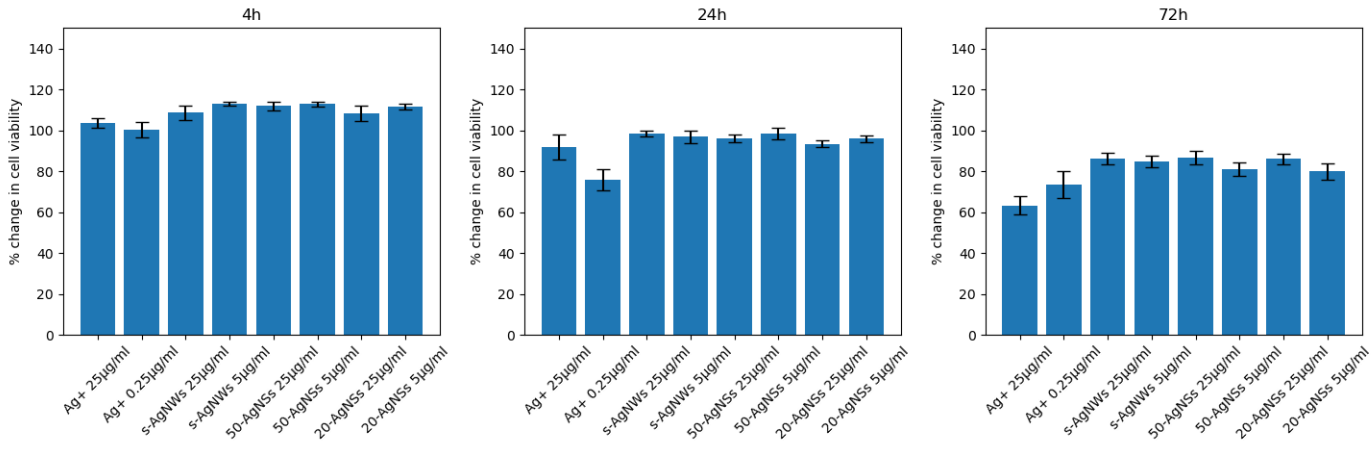


Figure 23. Concentration and time-dependent predictions of AgNSs and AgNWs, and Ag^+ ions on ASM cell viability after 4 hours, 24 hours and 72 hours. The bars represent mean values of 10 predictions made using a WGAN and the whiskers indicate standard error of the mean (SEM).

Figure 23 contains the predictions made by the WGAN for the same sets of parameters as the original study. For each combination of parameters, 10 predictions are made using the prediction Algorithm 1, minimising the error in the numerical value associated with a molecule, the concentration level and the time of interest. It can be observed that the mean of the predictions is close to the mean of the assessment.

5. Conclusions

In this paper, we demonstrate that Generative Adversarial Networks (GANs) can perform well for a number of regression tasks, sometimes outperforming a model based on state-of-the-art Gaussian Process Regression (GPR). The particular model used is a Wassterstein GAN (WGAN), which can be easier to train than a standard GAN. For simple regression and multiple regression tasks, both GAN and GPR perform well, although for the dataset which has variable uncertainty (modelled as heteroscedastic noise), the GPR fails to learn any variation in uncertainty, whereas the GAN captures this variation well. Also, for the more challenging problem of multi-modal distributions, the GPR struggles to learn the distribution whereas the GAN is able to reproduce the distribution very well. Furthermore, for multi-output regression, the WGAN also demonstrated good performance, showing that the GAN is able to capture the covariance information between all the output variables (which includes the independent and dependent variables of the regression problem).

Although the GPR can been modified for improved performance on specific types of data (such as heteroscedastic noise and multi-output regression), we wanted to show that the WGAN needs no modification for these problems: one single WGAN model can perform well for all the datasets we employed.

Novelties of the work include using a GAN for regression; being able to apply this model to multi-modal data and multi-output regression (MOR-GAN) tasks with no fundamental modifications; the presentation of a prediction algorithm to be used with the

trained GAN in order to predict a response for a given independent variable; using the critic to provide a confidence level of the predictions made by the generator, which could ultimately be used to help determine where more data is needed.

Author Contributions: Conceptualisation, C.C.P., A.E.P. and K.F.C.; methodology, C.C.P., T.R.F.P. and C.E.H.; software, T.R.F.P., E.B., Q.L. and L.H.; data curation, A.E.P.; writing—original draft preparation, T.R.F.P., E.B. and C.E.H; writing—review and editing, C.C.P., C.E.H. and K.F.C; funding acquisition, C.C.P. and K.F.C. All authors read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge the following EPSRC grants: INHALE, Health assessment across biological length scales (EP/T003189/1); RELIANT, Risk EvaLuatIon fAst iNtelligent Tool for COVID19 (EP/V036777/1); MUFFINS, MUltiphase Flow-induced Fluid-flexible structure InteractioN in Subsea applications (EP/P033180/1); the PREMIERE programme grant (EP/T000414/1) and MAGIC, Managing Air for Green Inner Cities (EP/N010221/1).

Conflicts of Interest: The authors declare no conflict of interest

1. Borchani, H.; Varando, G.; Bielza, C.; Larrañaga, P. A survey on multi-output regression. *WIREs Data Mining Knowl. Discov.* **2015**, *5*, 216–233. https://doi.org/10.1002/widm.1157.
2. Xu, D.; Shi, Y.; Tsang, I.W.; Ong, Y.S.; Gong, C.; Shen, X. Survey on Multi-Output Learning. *IEEE Transactions on Neural Networks and Learning Systems* **2020**, *31*, 2409–2429. https://doi.org/10.1109/TNNLS.2019.2945133.
3. Rasmussen, C.E. Gaussian Processes in machine learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2004**, *3176*, 63–71. https://doi.org/10.1007/978-3-540-28650-9_4.
4. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. Technical report, 2014, [1406.2661v1].
5. Kazeminia, S.; Baur, C.; Kuijper, A.; van Ginneken, B.; Navab, N.; Albarqouni, S.; Mukhopadhyay, A. GANs for Medical Image Analysis. *Artificial Intelligence in Medicine* **2020**, p. 101938, [1809.06222]. https://doi.org/10.1016/j.artmed.2020.101938.
6. Wang, K.; Gou, C.; Duan, Y.; Lin, Y.; Zheng, X.; Wang, F.Y. Generative adversarial networks: Introduction and outlook. In Proceedings of the IEEE/CAA Journal of Automatica Sinica. Institute of Electrical and Electronics Engineers Inc., 2017, Vol. 4, pp. 588–598. https://doi.org/10.1109/JAS.2017.7510583.
7. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, *1511.06434*. https://doi.org/10.48550/ARXIV.1511.06434.
8. Kunfeng, W.; Yue, L.; Yutong, W.; Fei-Yue, W. Parallel imaging: A unified theoretical framework for image generation. In Proceedings of the Proceedings - 2017 Chinese Automation Congress, CAC 2017. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 7687–7692. https://doi.org/10.1109/CAC.2017.8244169.
9. Zhang, K.; Kang, Q.; Wang, X.; Zhou, M.; Li, S. A visual domain adaptation method based on enhanced subspace distribution matching. In Proceedings of the ICNSC 2018 - 15th IEEE International Conference on Networking, Sensing and Control. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1–6. https://doi.org/10.1109/ICNSC.2018.8361269.
10. Jolaade, M.; Silva, V.L.; Heaney, C.E.; Pain, C.C. Generative Networks Applied to Model Fluid Flows. In Proceedings of the International Conference on Computational Science. Springer, 2022, pp. 742–755.
11. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. Technical report, 2017, [1606.03498v1].
12. Silva, V.L.; Heaney, C.E.; Li, Y.; Pain, C.C. Data Assimilation Predictive GAN (DA-PredGAN): applied to determine the spread of COVID-19. *arXiv preprint arXiv:2105.07729* **2021**.
13. Wang, S.; Tarroni, G.; Qin, C.; Mo, Y.; Dai, C.; Chen, C.; Glocker, B.; Guo, Y.; Rueckert, D.; Bai, W. Deep generative model-based quality control for cardiac MRI segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2020, pp. 88–97.
14. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. Technical report, 2017, [1701.07875v3].
15. Aggarwal, K.; Kirchmeyer, M.; Yadav, P.; Keerthi, S.S.; Gallinari, P. Regression with Conditional GAN. Technical report, 2019, [1905.12868v1].

16. Mcdermott, C.B.A.; Matthew, B.A. Semi-Supervised Biomedical Translation with Cycle Wasserstein Regression GANs. Technical report, 2018.
17. Rasmussen, C.; Williams, C. *Gaussian Process for Machine Learning*; MIT Press, 2006.
18. Kolmogorov, A.N. *Interpolation and extrapolation of stationary random sequences*; Izvestiya the Academy of Sciences of the USSR, 1941.
19. Wiener, N. *Extrapolation, Interpolation and Smoothing of Stationary Time Series*; MIT Press, 1949.
20. Jerome Sacks, William J. Welch, T.J.M.; Wynn, H.P. *Design and Analysis of Computer Experiments*; Institute of Mathematical Statistics, 1989. https://doi.org/10.2307/2245858.
21. Schulz, E.; Speekenbrink, M.; Krause, A. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology* **2018**, *85*, 1 – 16. https://doi.org/https://doi.org/10.1016/j.jmp.2018.03.001.
22. GPy. GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.
23. Barnett, S.A. Convergence Problems with Generative Adversarial Networks (GANs) A dissertation presented for CCD Dissertations on a Mathematical Topic. Technical report, 2018, [1806.11382v1].
24. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs. *arXiv* **2017**, p. 1704.00028. https://doi.org/10.48550/ARXIV.1704.00028.
25. Chollet, F.; et al. Keras. https://github.com/fchollet/keras, 2015.
26. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds.; Curran Associates, Inc., 2017; pp. 5767–5777.
27. Le, Q.V.; Smola, A.J.; Canu, S. Heteroscedastic Gaussian process regression. In Proceedings of the ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning; ACM Press: New York, New York, USA, 2005; pp. 489–496. https://doi.org/10.1145/1102351.1102413.
28. Kim, H.C.; Lee, J. Clustering based on Gaussian processes. *Neural Computation* **2007**, *19*, 3088–3107. https://doi.org/10.1162/neco.2007.19.11.3088.
29. Michaeloudes, C.; Seiffert, J.; Chen, S.; Ruenraroengsak, P.; Bey, L.; Theodorou, I.G.; Ryan, M.; Cui, X.; Zhang, J.; Shaffer, M.; et al. Effect of silver nanospheres and nanowires on human airway smooth muscle cells: role of sulfidatio. *Nanoscale Adv.* **2020**, *2*, 5635–5647. https://doi.org/10.1039/d0na00745e.
30. Quadros, M.E.; Marr, L.C. Silver nanoparticles and total aerosols emitted by nanotechnology-related consumer spray products. *Environmental Science and Technology* **2011**, *45*, 10713–10719. https://doi.org/10.1021/es202770m.
31. Benn, T.; Cavanagh, B.; Hristovski, K.; Posner, J.D.; Westerhoff, P. The Release of Nanosilver from Consumer Products Used in the Home. *Journal of Environmental Quality* **2010**, *39*, 1875–1882. https://doi.org/10.2134/jeq2009.0363.