On generating parametrised structural data using conditional generative adversarial networks

G. Tsialiamanis, D.J. Wagg, N. Dervilis & K. Worden
Dynamics Research Group, Department of Mechanical Engineering, University of Sheffield
Mappin Street, Sheffield S1 3JD, UK

Abstract

A powerful approach, and one of the most common ones in structural health monitoring (SHM), is to use data-driven models to make predictions and inferences about structures and their condition. Such methods almost exclusively rely on the quality of the data. Within the SHM discipline, data do not always suffice to build models with satisfactory accuracy for given tasks. Even worse, data may be completely missing from one's dataset, regarding the behaviour of a structure under different environmental conditions. In the current work, with a view to confronting such issues, the generation of artificial data using a variation of the generative adversarial network (GAN) algorithm, is used. The aforementioned variation is that of the *conditional* GAN or cGAN. The algorithm is not only used to generate artificial data, but also to learn transformations of manifolds according to some known parameters. Assuming that the structure's response is represented by points in a manifold, part of the space will be formed due to variations in external conditions affecting the structure. This idea proves efficient in SHM, as it is exploited to generate structural data for specific values of environmental coefficients. The scheme is applied here on a simulated structure which operates under different temperature and humidity conditions. The cGAN is trained on data for some discrete values of the temperature within some range, and is able to generate data for every temperature in this range with satisfactory accuracy. The novelty, compared to classic regression in similar problems, is that the cGAN allows unknown environmental parameters to affect the structure and can generate whole manifolds of data for every value of the known parameters, while the unknown ones vary within the generated manifolds.

Key words: Structural health monitoring (SHM), machine learning, conditional generative adversarial networks (cGANs), environmental conditions, artificial data.

1   Introduction

Structures are an irremovable part of modern society. Every part of modern life is connected to some structure. From buildings serving as houses and workplaces to vehicles facilitating human transport, structures prove to be essential elements of life. However, in order to deliver their function effectively, they should be monitored and their efficiency should be maintained. For this purpose, *structural health monitoring* (SHM) has been developed. SHM is the discipline of inferring whether damage is present in a structure, what type of damage and where it is present, how severe it is and how much useful life remains for the structure [1]. The most preferred way of performing such actions is through data-driven methods [2]. Following such an approach, the results rely largely on the data. In some cases, trying to compensate for the lack of data, knowledge is induced into such *black-box* models, forcing the model to learn in accordance with rules, that according to the user's knowledge are defined by the underlying physics problem. Combining knowledge and data in such a way leads to creation of *grey-box* models.

A significant factor that affects structures, their behaviour and the potential appearance of damage is the environment within which they exist and interact. Most loads that a structure is excited by are because of the environment it operates in. Because of variations in their environmental conditions, the response of structures also varies. The behaviour of a structure may change in accordance with environmental factors such as temperature, humidity, direction of the load, magnitude of the load, etc. All these conditions change the way that a structure responds to excitation and sometimes such a change in the response may even be mistaken for existence of damage by algorithms that detect novelties in the behaviour of structures.

A quite simple data-driven approach to detecting existence of damage within a structure, is that of calculating novelty indices [3]. The index is a distance metric between data acquired during the normal condition operation of the structure and a state of the structure that one wants to test for damage. Methods like this are efficient in some cases and able to indicate whether the structure behaves in a different way than normal. However, they are often vulnerable to false positive alerts regarding the existence of damage. The normal condition data are often represented only via their mean value and covariance matrix and the algorithm would be inefficient if extensive variations of normal condition structural behaviour existed.

Given a structure that operates within varying temperatures, its behaviour may change significantly as a function of the temperature. Lower temperatures can cause increases in stiffness of structural members and higher temperatures may have the opposite effect. Data can be acquired from a structure during a period in which its environment has a specific temperature. Following the novelty index approach would indicate damage existence if, during the period of the test state, a different temperature prevailed. But, even if a secondary and unknown environmental parameter variation appeared, such us change in humidity in specific parts of the structure, the novelty index approaches could also indicate existence of damage.

In the case of nonlinear structures, variance in normal condition features is also observed even as a result of variations in the characteristics of the loads. Linear structures would behave similarly under harmonic loads with different magnitudes; their *frequency response function* (FRF) is not a function of the load but only of the structure's parameters (stiffness, damping, mass etc.). A structure with nonlinear elements, such as a *Duffing oscillator*, has variations in behaviour, in terms of the FRF, according to the signal characteristics of the excitation force applied. Structures with much more complicated nonlinearities than a Duffing oscillator exist. Different kinds of nonlinearities (both geometric and in material behaviour) make the definition of normal condition characteristics a complicated task.

Approaches to deal with such issues have been developed. An example is presented in [4], where an auto-associative Network (AAN) [5] is used as a tool to learn the normal condition manifold of a structure. The network is trained using points that represent normal condition characteristics of the structure. The AAN learns from such points, how to project them in a lower dimensional space and then return them to the original space. Having been trained in such a way, a novelty index is defined as the reconstruction error of an input point. The error is defined as the Euclidean distance between the input and the output of the network. Points that lie on the manifold of normal condition operation, should have a small reconstruction error, since the AAN has learnt how to project onto this specific manifold. At the same time, points that diverge from this manifold would have a higher reconstruction error and this would indicate that damage may exist in the structure.

Following the aforementioned approach, a description of the normal condition manifold is achieved and anything that does not belong to that, is considered as a potential damage existence sign. The most trivial problem with the method, as with any fully data-driven method, is that the model, is only as good as the available data. Since prior physical knowledge is not induced by the user into the model, any state of the structure that is not represented somehow by the normal condition data, is considered as a potential damaged state. This fact may lead to false inference and start an unneeded maintenance procedure that would result in economic loss.

As an attempt to deal with such a problem, in the current work, an algorithm that learns transformations of manifolds is considered. The algorithm is that of the *conditional generative adversarial network* (cGAN). The main difference of this algorithm, in contrast to the traditional *(vanilla) GAN*, is that knowledge is introduced in the algorithm through a part of the input vector controlled by the user. By knowledge of environmental conditions, it is expected that, using the cGAN algorithm, a more concrete manifold of normal condition characteristics of the structure will be created. This opportunity may be exploited in order to build more robust SHM models but also in order to study and understand better the underlying physics of a problem in hand.

2 Generative adversarial networks

2.1 Introduction

One of many frameworks for machine learning algorithms is that of training *generative* models. Such models are able to generate samples that belong to the same probability density distribution as some given data and in the case of images this also translates into generating images that look real. A novel architecture for such models is the generative adversarial network [6]. The novelty of the scheme is that of using two different neural networks in order to achieve the result of generating images that resemble reality. The two networks concerned are the *generator* and

the *discriminator*. In the original work, the algorithm was focussed on images but throughout the current work, the algorithm shall be focussed on vector samples, since they are of more use in a dynamics or SHM project

Each of the two networks serves a different purpose throughout training, and their interaction yields the desired results. The discriminator on the one hand is optimised with a view to identifying whether a sample is real and comes from the original dataset or if it is fake/generated. On the other hand, the generator's training is focussed on "fooling" the discriminator by generating samples that the later will consider as real. Following such a scheme, throughout training, both networks perform their functionality better and the generator, eventually, is a network that will be generating real-like samples. The discriminator eventually performs in a similar way to the novelty detection algorithm presented in [4], learning the manifold of the data and indicating samples outside it as outliers.

As a general case, both networks can be multi-layer perceptron (MLP) networks. The generator G takes as input a latent noise vector z from a probability distribution $p_z(z)$ and transforms it into a generated sample (or image) $G(z)$. The discriminator D takes as inputs vectors (or images), and outputs the probability of them being real, i.e. $P_{x \sim p_{data}} = D(x)$. The discriminator is optimised by setting as target values for the input samples their labels ("real" or "fake") while the generator is optimised to make the discriminator classify the generated images as real, i.e. minimisation of $\log(1 - D(G(z)))$. Therefore, the objective function of the described min-max game is given by,

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

In practice, training is performed in two steps per epoch. During the first step, the discriminator is trained. Samples from the latent distribution $p(z)$ are sampled, and using them, the generator outputs samples. A second batch of samples is picked randomly from the original dataset and concatenated with the generated ones. Using the two batches, the discriminator is trained having as target labels, 1 for the real samples and 0 for the generated. During this phase, only the first term of the right hand side of equation (1) is used and its maximisation is sought. During the second phase, only samples from the latter distribution are sampled and this time the two networks are clipped together as shown in Figure 1. This time, the trainable parameters of the discriminator are considered constants and they are not optimised. The target label set for training is 1 for all samples, meaning "real".

The error is back-propagated through the weights and biases of the generator. During this step, the second term of the right hand side of equation (1) is used and the objective is its minimisation. By the competitive procedure described, both networks get better at their respective objectives. In most cases, only the generator is utilised, with a view to generating samples, while the discriminator is considered an auxiliary network used only for training.
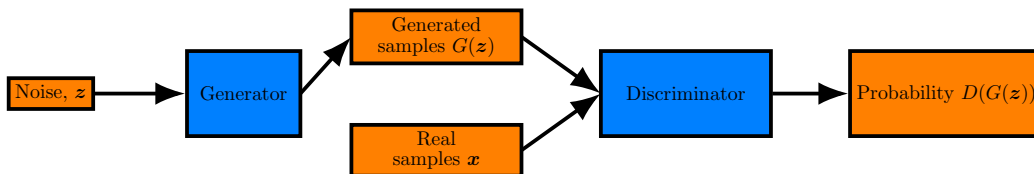

Figure 1: Vanilla GAN layout.

Properly trained GANs could prove quite useful for SHM purposes. In SHM acquiring enough data in order to train a machine learning model is sometimes hard and might be expensive. Therefore, using a GAN in order to augment a dataset with artificial data is a possibly valuable use of the algorithm. The GAN can also be used as a tool to implicitly enhance the performance of a neural network model, as described in [7], by exploiting unlabelled data. Most applications of GANs are found in image processing [8, 9, 10]. Some of them have the quite impressive result of manipulating image features through the latter variables of the GAN. In some cases, GANs are used in order to fill gaps in images [11, 12]. All these applications encourage the existing belief that generative models may be able to understand and recreate the physics of the underlying problems and should be exploited in order for humans to also gain further insight into problems in which they lack knowledge. Even an augmentation of data may assist in this task by providing more samples to study and understand underlying patterns and trends in the data.

2.2 Conditional generative adversarial networks

Conditional generative adversarial networks are an attempt to control the output of the generator by conditioning on some variables. In contrast to the traditional GAN layout (Figure 1), where the product of the generator is

completely controlled by random noise $\mathbf{z}$, the output of the generator is partially controlled by some vector $\mathbf{c}$ (called a *code* here-in, in parallel with the code introduced in the infoGAN [13]). This code may be a continuous variable or a discrete/categorical one. Since the output of the generator depends on the code, so shall be the output of the discriminator. Following such a scheme, the layout of the cGAN is shown in Figure 2.
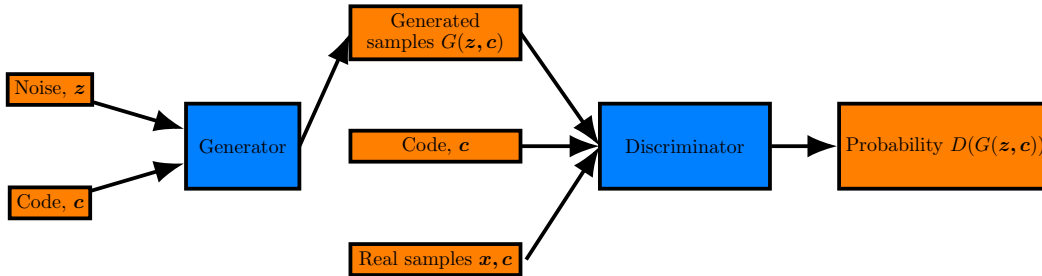

Figure 2: Layout of a cGAN.

According to this approach, the discriminator learns that for each different value of $\mathbf{c}$, a different decision boundary (or manifold) is defined. According to this realisation, the discriminator creates a different decision boundary in the feature space, as a function of $\mathbf{c}$, for the outputs of the generator, forcing the latter to create samples that correspond to the values of the code (also given as inputs to the generator). In cases of discrete or categorical variables, the result is that the generator learns to generate samples belonging to different categories. In [14], an illustration of this result is presented in the MNIST dataset; a dataset comprised of figures of hand-drawn digits between 0 and 9. By defining nine categorical variables and a one-hot encoding of the different classes, the generator is able to generate sample images in predefined classes, controlled by the code.

The use of continuous variables yields more interesting results. A continuous variable would force the decision boundary created by the discriminator, within the multidimensional feature space of the samples, to get gradually transformed according to the values of the code. The decision boundaries of the discriminator, in turn, force the generator to create samples within the region they define. Consequently the geometry of the generated manifolds is controlled by the code vector $\mathbf{c}$. Following this procedure, what the generator has eventually learnt is the transformation of the manifold of the generated points, as a function of the code.

The algorithm may be exploited in order to generate artificial data as a function of some code vector. In the current work, it is exploited in order to learn the transformations of the aforementioned manifolds, as functions of the code. The most straightforward way that the algorithm might be used would be to fully control the class of the generated samples (as in the case of the MNIST dataset) and create artificial datasets according to one's needs. For SHM purposes, a more convenient use is that of learning how the manifold of data transforms according to some environemental parameter.

As mentioned already, a common problem that arises in SHM is that the normal condition data are different for varying environmental conditions. Assuming that the normal condition data are not a single point but a manifold of points (as a result of noise or unknown environmental factors affecting the structure), their transformations as a function of known environmental parameters are desired. Learning such transformations would allow generating the normal condition manifold for unseen values of the code, understanding better the way that the structure behaves and creating a more complete dataset regarding its normal condition/undamaged state. In the following section, an application in a toy dataset is illustrated with a view to explaining the proposed algorithm and afterwards, an SHM application on simulated data is presented.

3 Applications

3.1 Toy dataset

Initially, in order to better illustrate the potential of the algorithm, a toy dataset was created. The data were two-dimensional and had x values sampled uniformly in the range $[-1, 1]$ and y values sampled from a normal distribution $y \sim \mathcal{N}(0, 0.03)$. The result is shown in Figure 3. Afterwards, in order to create manifolds whose transformation was a function of a known parameter, the samples, shown in Figure 3, were rotated using angles in the range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The result is partially shown in Figure 4.
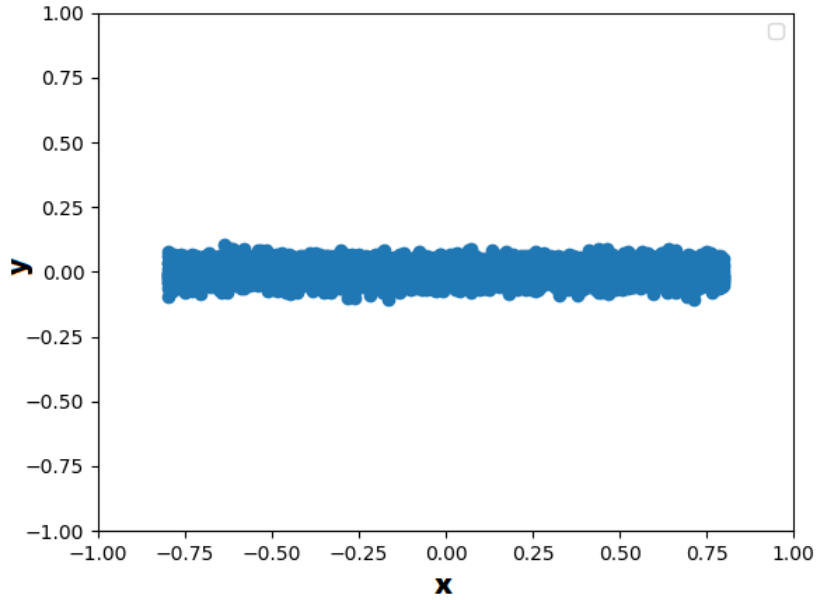
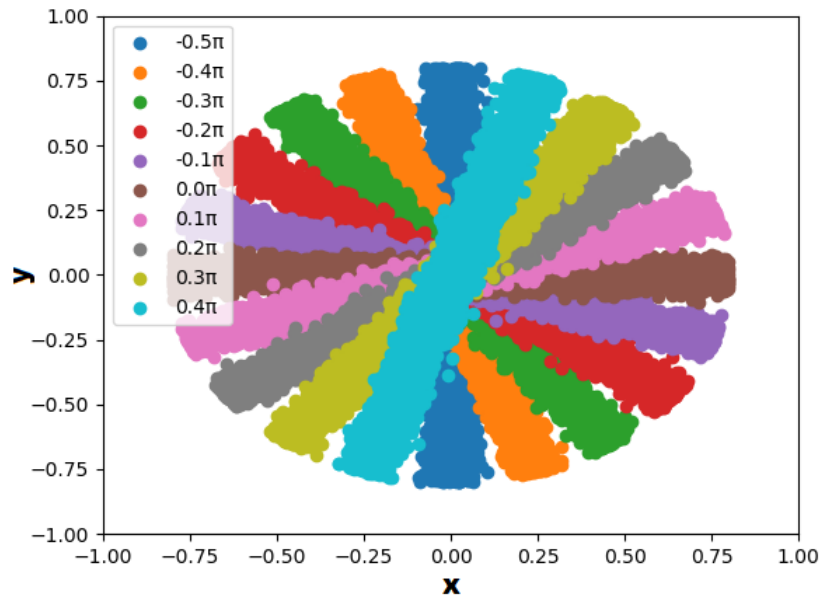Figure 3: Base data, corresponding to angle of 0.

Figure 4: Collection of manifolds of the toy dataset.

The transformation of the initial manifold in this case is simply a rotation of the points around the point $(0, 0)$. In order to test if the cGAN would be able to learn such a transformation, a traditional scheme followed in machine learning was applied. Three datasets were created, a training, a validation and a testing one. The training dataset was created by generating points for 11 values of the angle equally spaced in the interval $[-\frac{\pi}{2}, \frac{2\pi}{5}]$. The other two datasets were comprised of points for values of the angle also equally spaced in the same interval. Some of the manifolds within the created datasets are shown in Figure 5. The training dataset should contain manifolds corresponding to the two outer values of the interval $[-\frac{\pi}{2}, \frac{2\pi}{5}]$ so that the transformation task is an interpolation across values of the
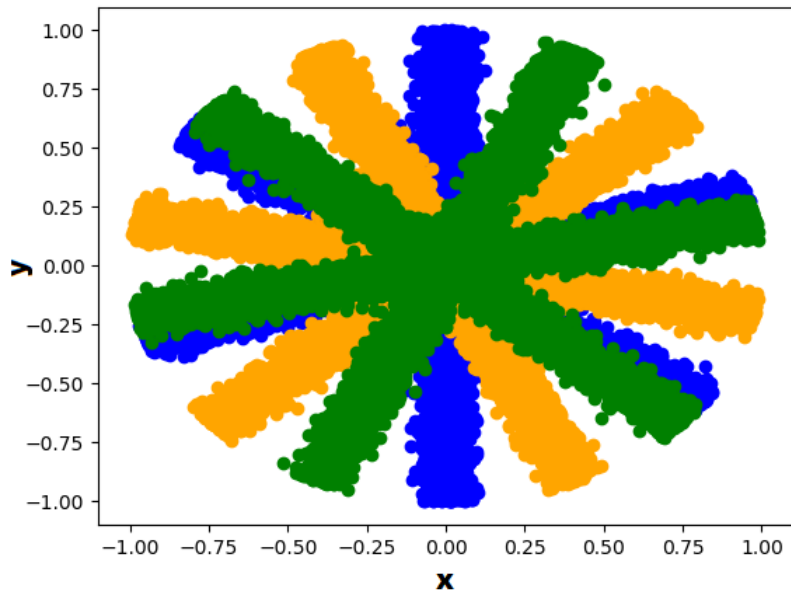
angle parameter.

Figure 5: Points belonging to the training (blue), validation (orange) and testing (green) datasets.

Since the loss function of a GAN does not directly represent the quality of the model [6], another training scheme has to be followed in order to have a metric of the performance of the algorithm on the validation set. The strategy used here is to calculate the *Kullback–Leibler divergence* (KL divergence) [15] between the generated points for every validation code and the points in the validation dataset that correspond to each code. KL divergence is used to measure the distance between two probability distributions, so a probability distribution has to be defined for both the generated and the real points of each code. In order to define such distributions, kernel distributions [16] are fitted to both the real data and the generated. The divergence is then calculated between every pair of generated and real data for every code value. The model that has the lowest cumulative KL divergence in the validation set is considered the best one. The total KL divergence of a model is given by,

$$\sum_{n=1}^{n_{val}} D_{KL}(P_i||Q_i) = \sum_{n=1}^{n_{val}} \sum_{x \in X} P_i(x) \log(\frac{P_i(x)}{Q_i(x)}) \tag{2}$$

where the P_i and Q_i distributions are kernel density estimates fitted to the real and generated data, for the i-th code. The KL divergence is numerically computed on a grid in the feature space X for discrete values of $x \in X$. The bandwidth of the kernel density function can be considered as a hyperparameter of the algorithm. In the current work, a reasonable value for the bandwidth was considered to be 0.2, since all the data are normalised in the interval $[-1, 1]$ and a value equal to $\frac{1}{10}$ of the total range can sufficiently describe the distribution of data.

The generator was selected to be a three-layered neural network. The first layer was the input one with a total size of three (a two-dimensional noise **z** plus an one-dimensional code c), a hidden layer, for which different sizes in the set $\{10, 20, ...790, 800\}$ were tested, initialising each randomly ten times. Eventually, a 500-node sized one was found to yield the best results in terms of KL-divergence in the validation dataset. The output layer was a three-node layer. The activation function that performed best was *hyperbolic tangent*. In every case the discriminator had the same size hidden layer as the generator. Following such a procedure, a model was defined whose performance was also tested on the testing dataset. The result for two code values belonging to the testing dataset are presented in Figures 6 & 7.

It is revealed that the algorithm has indeed learnt how the manifolds transform/rotate according to the value of the code. The overlap between the generated and the real points for code values that the cGAN has never "seen" before is quite satisfactory as shown in Figures 6 & 7. Another advantage of a cGAN approach is revealed in the specific toy example. The manifolds share regions within the two-dimensional space. There are regions common to every

manifold, close to the center and manifolds with similar angle values share more points. If another variation of GAN were used, in order to achieve similar results, an extra axis would be needed in the data. This axis would be the variable that is known and that is induced as the code in the cGAN. Even doing so, the encoding of the new axis as a single latent variable of the GAN is not guaranteed.
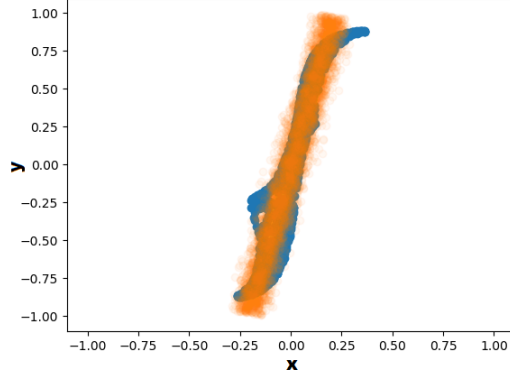

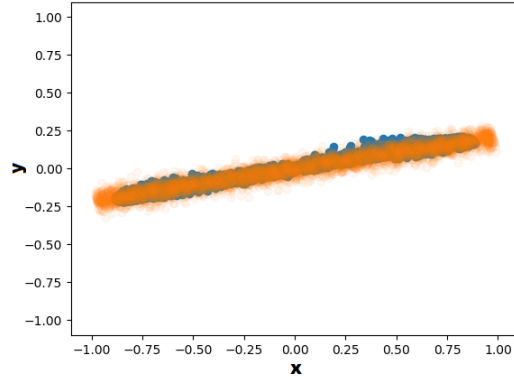Figure 6: Real (orange) and generated (blue) points for code value in the testing dataset.

Figure 7: Real (orange) and generated (blue) points for code value in the testing dataset.

3.2 Generation of artificial parametrised data

In order to test the potential of the algorithm in a different dataset, which is bound by some underlying physics, a simulated six-degree-of-freedom system was considered (Figure 8). In the system, two different variables were chosen to affect its behaviour and to be causing variations in the characteristics. The first variable, the one known by the user and passed into the cGAN as prior knowledge, is the temperature. The temperature affects the stiffness of all springs. Higher temperatures causes stiffness reductions, while lower ones increase it. As a second varying environmental parameter, humidity was selected. Humidity in such a system could potentially affect the damping parameters of the system. Higher humidity could lead to higher damping coefficient c_i and lower to the opposite. Temperature was selected to be varying in the interval $[20, 40]$ and humidity in the range $[80\%, 100\%]$.

The initial values of the structural parameters were $k_i = 10^4$, $c_i = 10$ and $m_i = 1$ ($i = 1, 2...6$). In both cases, the relationship between the parameters and the environmental conditions was linear. Stiffness (k_i) increases or decreases linearly according to $k_i = k_0 * (1 - \frac{T-30}{100})$, where k_0 is the initial value of the stiffness and T is the temperature. Thereinafter, 1000 simulations were performed for each one of 11 discrete values of the temperature using a white noise excitation on mass #1, as shown in Figure 8 and random values for humidity. Humidity was considered to affect the value of the damping parameters following the relationship $c = c_0 * (1 - \frac{h-90}{100})$, where h is the value of
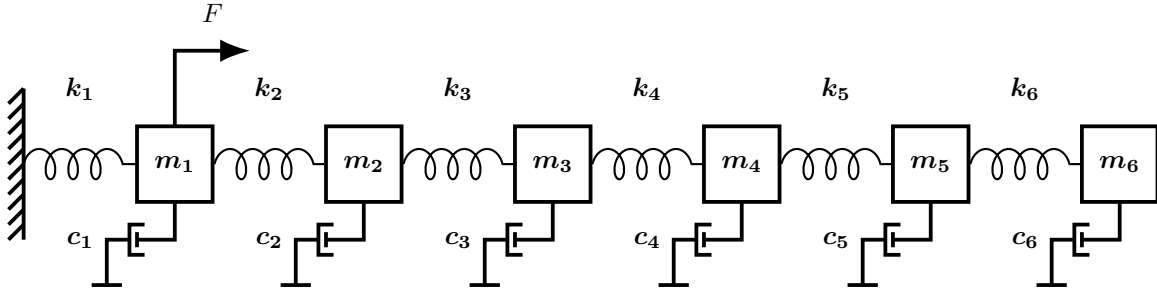
Figure 8: Mass-spring system.

the humidity and c_0 the initial damping parameter values. For every simulation, the transmissibilities between the masses 1-2 and 2-3 were recorded, polluted with Gaussian noise (r.m.s. equal to 5% of the signal's variance) and concatenated. A typical example of such a feature is shown in Figure 9.
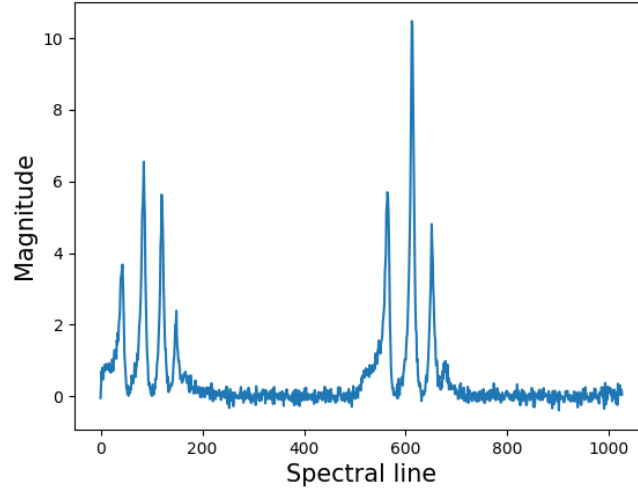

Figure 9: Example of simulated transmissibilities recorded.

After collecting the transmissibilities, a *principal component analysis* (PCA) [17] was performed, both in order to depict the data and to accelerate the training of the cGAN. The collection of generated points is shown in Figure 10. The movement of the data within the feature space, according to the temperature, is revealed by the gradual colour change. The variation caused by the humidity is also visible, since each batch of points having the same temperature is spanned along a line segment.

Among the 11 temperature values, one of them (24°C), was considered to be the validation dataset and another one (34°C) to give the testing. The highest and lowest values were included in the training dataset. Following the same procedure as before, the cGAN was trained and the criterion to choose which one performed better was once again the KL divergence between the real and generated points of the validation code/temperature.
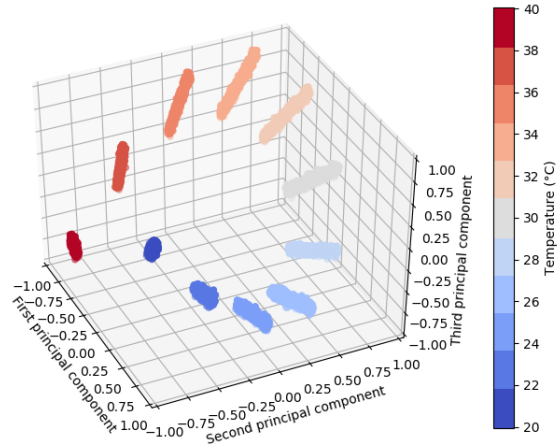
Figure 10: First three principal components of simulation data.

The performance on the testing set is shown in Figure 11. The two batches (real and generated) are, as before, quite close. In order to illustrate also the results in terms of the transmissibilities, in Figure 12 two tranmissibilities are shown, one generated using the testing code and one drawn from the testing dataset. For comparison, they are plotted along with a real transmissibility that corresponds to 20°C. It is clear that the cGAN has captured correctly the way that the diagram moves according to the temperature. A quite interesting application of such a model is that it may be used to generate the whole manifold of points for all intermediate temperatures that were not recorded for the structure. By generating points for many temperatures in the range $[20, 40]$ the resulting manifold is shown in Figure 13. Such a manifold provides a better description of the normal condition characteristics of the structure over many different temperatures. In the example presented here, the only unknown varying parameter was humidity but in any case there could be any number and the cGAN would be able to generate such manifolds for every value of the known code.
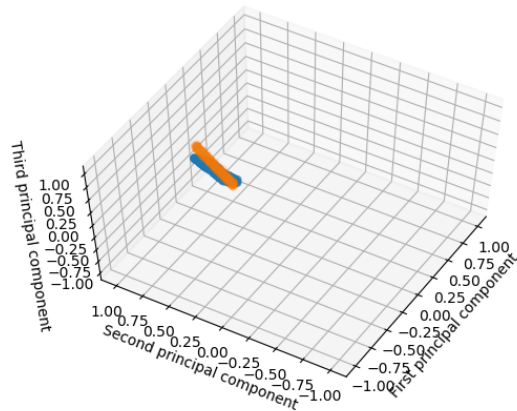

Figure 11: Real (orange) and generated (blue) points on the testing set for the temperature value of 34°C.
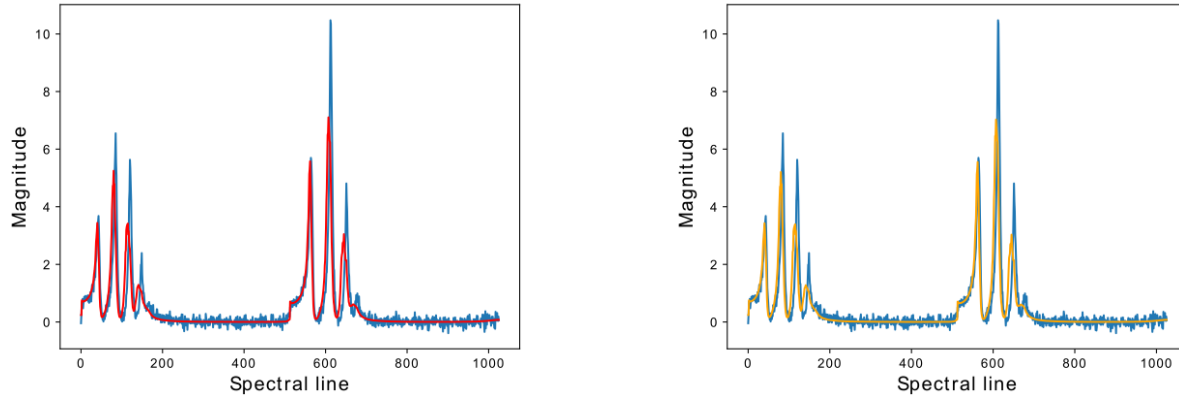
Figure 12: Transmissibility recorded from simulations with temperature 20°C (blue) compared to recorded transmissibility in 34°C (orange) and generated (red).

The cGAN is called upon to learn such movements and rotations (transformations) of the manifolds that arise because of the effect of temperature on the data. The convenience of the algorithm lies in the fact that a clear separation between known and unknown environmental parameters is performed. One needs to specify the values of the known parameters, and the algorithm is able to generate a range of plausible values (shown in Figure 10 as points with the same colour/temperature) of the feature vectors used for training. The aforementioned range is caused by the effect of unknown underlying factors, i.e. the variation in humidity herein. Having further knowledge about the factors, measuring more environmental parameters and using them as inputs in the algorithm, the aforementioned range, potentially, becomes smaller and the algorithm yields more confident predictions. Therefore, the code and noise vectors can be thought of as respective representatives of the existing knowledge about a problem and the *epistemic uncertainty/lurking variables* affecting it.
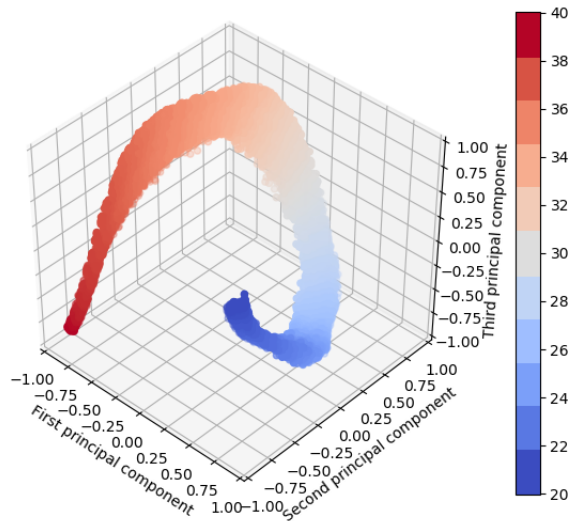

Figure 13: Manifold representing samples for all temperature values in the range $[20, 40]$.

4 Conclusions

The algorithm presented in the current work may be exploited in order to create data for unseen states of a structure. Data that correspond to discrete values of some parameter affecting the behaviour of the structure, are used to train the cGAN. Considering that the structure operates under known (or measured) and unknown (not measured or immeasurable) environmental parameters, the algorithm proposed can take both into account and generate artificial data for values of the known ones for which data are absent. This effect is achieved by learning transformations of manifolds according to the known environmental parameters. The method proposed in order to construct the whole manifold of potential states of a structure for varying environmental conditions provides a tool for understanding better the structural behaviour.

A major drawback of the described algorithm is the calculation of the KL divergence as an error metric in the validation dataset. The quantity of equation (2) is calculated on a grid over the $\mathbb{R}^n$, where n is the dimensionality of the data. Such a calculation is computationally inefficient. For example, having a grid with only 10 points per axis would require computing the probability for every validation code in 10^n points. In the presented applications, the computation was performed in a three-dimensional space and so it was feasible, but for larger spaces it would not be. Nevertheless, since in SHM the monitored quantities can be low dimensional (or projected in lower-dimensional spaces, as was done here using PCA), the method still constitutes a viable tool.

The whole manifold of potential states may prove useful in situations where a novelty detection system is to be set up for monitoring. Apart from such applications, manifolds characterising structures may be even more useful in knowledge transfer situations. As proposed in [18], a knowledge transfer scheme between structures of a population (even a heterogeneous one) can be applied in order to generate data for structures that one has not acquired. Under this framework, for each structure, the complete manifold of potential states, and not just parts of it, would make knowledge transfer much more applicable. In total, even the current method can be thought as a knowledge transfer method between states of the structure, parametrised by environmental conditions.

Finally, a useful and thought-invoking characteristic of the algorithm, is the way that knowledge and uncertainty are represented through the two parts of the input vector of the generator. Knowledge is represented through the code vector, which incorporates the known variables and parameters that affect the behaviour of the structure. Uncertainty, both epistemic and aleatory are encoded within the code vector. The cGAN generates data according to both, proving a convenient way of maybe propagating uncertainty in a fully data-driven framework.

5 Acknowledgement

References

[1] A. Rytter. *Vibrational Based Inspection of Civil Engineering Structures*. PhD thesis, 1993.

[2] C.R. Farrar and K. Worden. *Structural Health Monitoring.: A Machine Learning Perspective*. John Wiley & Sons, 2012.

[3] K. Worden, G. Manson, and D. Allman. Experimental validation of a structural health monitoring methodology: Part I. Novelty detection on a laboratory structure. *Journal of Sound and Vibration*, 259(2):323–343, 2003.

[4] K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85–101, 1997.

[5] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

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

[7] M. McDermott, T. Yan, T. Naumann, N. Hunt, H.S. Suresh, P. Szolovits, and M. Ghassemi. Semi-supervised biomedical translation with cycle Wasserstein regression GANs. 2018.

[8] G. Perarnau, J. Van De Weijer, B. Raducanu, and J.M Álvarez. Invertible conditional GANs for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[9] A. Brock, T. Lim, J.M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[10] H. Zhang, V. Sindagi, and V.M Patel. Image de-raining using a conditional generative adversarial network. *IEEE transactions on circuits and systems for video technology*, 2019.

[11] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A.A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[12] Y. Li, S. Liu, J. Yang, and M.H. Yang. Generative face completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3911–3919, 2017.

[13] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[14] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[15] S. Kullback. *Information Theory and Statistics*. Courier Corporation, 1997.

[16] B. W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological)*, 43(1):97–99, 1981.

[17] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[18] K. Worden. Towards population-based structural health monitoring, part VI: structures as geometry. In *Dynamics of Civil Structures, Volume 2*, pages 221–236. Springer, 2020.