
Proceedings Paper:
Tsialiamanis, G., Chatzi, E., Dervilis, N. orcid.org/0000-0002-5712-7323 et al. (2 more authors) (2020) An application of generative adversarial networks in structural health monitoring. In: Papadrakakis, M., Fragiadakis, M. and Papadimitriou, C., (eds.) EURODYN 2020: Proceedings of the XI International Conference on Structural Dynamics. EURODYN 2020: XI International Conference on Structural Dynamics, 23-26 Nov 2020, Athens, Greece. European Association for Structural Dynamics (EASD) , pp. 3816-3831. ISBN 9786188507227

10.47964/1120.9312.19021

EURODYN 2020
XI International Conference on Structural Dynamics
M. Papadrakakis, M. Fragiadakis, C. Papadimitriou (eds.)
Athens, Greece, 22—24 June 2020

AN APPLICATION OF GENERATIVE ADVERSARIAL NETWORKS IN STRUCTURAL HEALTH MONITORING

G. Tsialiamanis[1], E. Chatzi[2], N. Dervilis[1], D. J. Wagg[1], K. Worden[1]

[1]Dynamics Research Group, Department of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD

[2] ETH Zurich, Institute of Structural Engineering
Stefano-Franscini-Platz 5, 8093 Zurich

Keywords: Structural health monitoring, machine learning, neural networks, generative adversarial networks, GANs

Abstract. *In the current work, the use of generative adversarial networks (GANs) in a simulated structural health monitoring (SHM) application is studied. A specific type of GAN is considered, aiming at a disentangled representation of underlying features and clusters of data through some latent variables. This idea could prove useful in SHM, since explanation of how damage mechanisms or environmental conditions affect a structure may be exploited in order to monitor structures more effectively. In a simulated mass-spring example, different damage cases are introduced by reducing the stiffness of specific springs and different damage levels by applying different extents of stiffness reduction. The GAN implementation proves able to capture different damage cases through its categorical latent variables, as well as the damage extent within its continuous latent variables. The results demonstrate that the latent variables are indeed capturing the effect of damage in the structure and can be exploited for the purpose of condition assessment.*

1 Introduction

Modern societies extensively rely on structural elements. Everyday activities depend on structures like bridges, engines, power generators etc. Malfunction or damage of these critical systems may lead to delays in transportation, power shutdowns or even injury or death of individuals. Avoiding such failures and guaranteeing safe operation and efficient performance is therefore a matter of critical importance. *Structural Health Monitoring* (SHM) offers a tool to this end. SHM is employed to ensure operability and safety of structures and to avoid consequences of failure. Many different approaches have been proposed in the context of SHM, but all of them can be categorised according to the hierarchical structure proposed by Rytter [1] and extended in [2]:

1. Is there damage (*existence*)?
2. Where is the damage in the system (*location*)?
3. What kind of damage is present (*type/classification*)?
4. How severe is the damage (*extent/severity*)?
5. How much useful (safe) life remains (*prognosis*)?

Ascending in Rytter's hierarchy makes the task at hand more difficult than the one in the previous step. Detecting if damage exists requires definition of the normal condition of a structure and divergence from that is an indication of damage and has to be examined. The second step of localising damage, requires further knowledge about the behaviour of a structure and how damage in various areas may affect the structural response. Moreover, defining the type of damage is a task that requires further understanding of the structural behaviour and of the manner in which different types of damage affect observed features of the structure. Finally, the two last steps in the hierarchy, demand understanding of the damage mechanism in order to define its severity and to further predict the useful life of the structure under the specific damage case. Although some data driven methods would promise to do this without really understanding the mechanism as long as sufficient instances of failure are recorded.

It is straightforward that dealing with these tasks involves acquiring and processing data in order to infer results about the condition of a structure. Taking into account the progress made in the disciplines of data analysis and *machine learning* (ML), diverse methods from these fields have been exploited for the purposes of SHM [2]. Regarding the first step in the aforementioned hierarchy, ML has been used to perform outlier detection using autoencoders to explain data corresponding to the normal condition of a structure [3]. Autoencoders were chosen because they are able to explain data belonging to an arbitrary manifold. For damage classification or localisation, ML classifiers have been used [4], yielding quite good accuracy. The main drawback of these methods is that data from damage states is required to perform their tasks. Oftentimes however, data from damaged states are absent, or only limited samples are available. Furthermore, to perform tasks further up in Rytter's hierarchy, understanding of the underlying physics of structures and damage mechanisms is needed.

Trying to understand the underlying physics within such a context, one may attempt to study a black box model that is performing well on a task related to the underlying mechanism of the problem. A representative example of such a case is a model trained on patient data [5] that was able to predict schizophrenia with adequate precision. It is clear then that it may be worth

spending some time studying the model in an attempt to understand better how schizophrenia works and which symptoms indicate its existence. The same scheme may be applied on structures with models trained on identifying damage location/type. A black-box model like a neural network [6] may be exploited to understand the way some type of damage affects a structure.

A specific type of black-box model that is targeted to capturing the physics of a specific problem, is the generative model. More specifically, Generative Adversarial Networks (GANs) [7] and Variational Autoencoders (VAEs) [8] are neural networks that learn how to generate data that look like reality. It is believed that, by studying these types of neural networks and the way they produce data, further understanding of the underlying problems may be achieved. In the current work, a specific type of GAN is used, the infoGAN [9]. This specific type of network achieves a disentanglement in the latent feature space of the data which may have great benefits in an SHM application. The use of such networks is considered in the task of classifying data from structures and also in defining latent variables that explain the extent of damage in the data or the variation of a parameter affecting the behaviour of the structure.

2 Generative Adversarial Networks (GANs)

2.1 Vanilla GANs

The traditional scheme followed in ML is the training of a model to perform classification [6] or regression [10]. To extend this to images, convolutional neural networks were developed [11] yielding superior performance in the two mentioned tasks. Recently, a new type of neural network has emerged, the *Generative Adversarial Network* [7]. The goal of this new scheme is to generate images that resemble reality, which is achieved via use of *two* neural networks. The first one is termed the *generator* and produces "fake" images given a latent vector. The second network is the *discriminator*, which tries to identify whether an image is fake (generated by the generator) or real (coming from the available dataset). Through training both of these networks, improve in reaching their objective and finally, the generator, provided with some latent vector, can generate images that appear to be real. More intuitively, this means that the generator maps a latent vector distribution into a distribution or a manifold of the real data. The layout of the basic (vanilla) GAN can be seen in Figure 1.

The generator is a multi-layer perceptron (MLP) that takes as input a latent noise vector $\mathbf{z}$ coming from a probability distribution $p_z(\mathbf{z})$ and maps it into a vector (or an image) $G(\mathbf{z})$ of dimension equal to the dimension of the training samples. The discriminator is another MLP that takes as inputs, vectors (or images) $\mathbf{x}$, and outputs the probability of the sample being real, $P(\mathbf{x} = real) = D(\mathbf{x})$. The training of D is carried out by maximising the probability that it assigns the correct label ("real" or "fake") to the samples. At the same time, the training of the generator, G, is accomplished by trying to minimise the probability that the discriminator classifies the generated samples as fake, i.e. minimisation of $log(1 - D(G(z)))$. Following from [7], the objective function can be interpreted as a two player minmax game given by,

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

Training of such a network is performed in two steps per epoch. During the first step, random samples are created by the generator and they are concatenated with a batch of real samples from the dataset. The resulting training batch is used to train the discriminator for one epoch by

back-propagating the error of the output. The target label for the real samples is 1 and for the generated ones is 0. The right-hand side of equation (1) is set in this step as the objective function and its **maximisation** is attempted. Consequently, the two networks are clipped together, as in Fig 1, and random samples of the latent vector are generated in order to create random generated samples. These samples are fed into the whole GAN and the target outputs are labels of 1. The weights of the discriminator's connections are considered as constants during the second training phase and the error is back-propagated in order to train the generator. This latter time, the objective function is comprised exclusively from the second term of the right-hand side of equation (1) and its **minimisation** is sought. Following this training scheme, during the first step the discriminator learns to distinguish between real and generated images and the generator to generate images that the discriminator classifies as real.

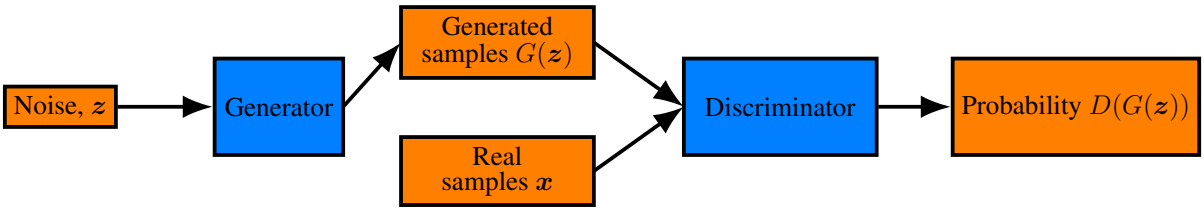


Figure 1: Vanilla GAN layout.

The most straightforward application of GANs is to generate artificial data to augment a dataset. Training neural networks is highly dependent on the size of the available dataset. The rule-of-thumb for training neural networks that generalise well [12], specifies that for each trainable weight of the neural network, 10 training samples are needed. Acquiring data is difficult and some times even expensive. Labelled images are hard to be obtained and their manual labeling costs. In cases of image datasets, augmentation can also be achieved by rotation of the pictures or colour change etc. In SHM though, the securing of data from structures in different damage cases or under different environmental conditions is expensive or even impossible; the samples are usually limited and augmentation is not that easy. For this purpose, GANs can be used to increase the size of a dataset and may even serve as interpolators. Especially for deep networks and even more for deep convolutional neural networks, where the number of trainable parameters is huge, augmentation of available dataset size could yield an efficient way to increase the generalisation performance of models [13].

Another use of GANs is found in filling gaps in pictures [14, 15]. In this case, GANs trained on a dataset are able to use pictures with missing pixels and generate a complete picture by replacing missing pixels with something that fits well in the gap. This is a great example of how this type of model may assist in repairing corrupted datasets. At the same time, these results are encouraging the perspective that GANs achieve a better linkage of data to the driving physics (or semantics) than traditional neural networks, since they are able repair corrupted data that they have never seen before.

Furthermore, GANs are applied in image processing [16, 17, 18]. This application is also an impressive usage of GANs, since they are able to capture through their latent variables, specific features of the data. Afterwards, the users are able to manipulate the latent vectors and use the generator to produce images, whose characteristics fit custom needs. A similar approach in the current work is followed and an attempt is performed to capture in the latent code the extent and type of damage in a structure. The ability to describe such features through a latent vector

and generate data corresponding to specific damage cases or to the extent of damage would be beneficial for the purposes of SHM, since further progress related to the fourth and fifth steps in Rytter's hierarchy could be achieved.

2.2 Information Maximising Generative Adversarial Nets

According to the classic formulation of GANs described, the latent variables used as inputs in the generator do not encapsulate any specific features from the data. This is because no restriction is imposed during training that every variable would describe a specific feature, resulting in latent noise variables that correspond to entangled features or combinations of features. In order to improve performance regarding the representation of disentangled features by the latent variables, a slightly different architecture and training scheme is followed in [9], called *Information Maximising Generative Adversarial Nets* (InfoGAN).

The procedure followed to enforce this disentanglement will be described below. At first, the latent vector is divided into two parts, the latent noise $\mathbf{z}$ and the latent code $\mathbf{c}$. The noise part $\mathbf{z}$ is used to model noise that may be present in the data, while the latent code $\mathbf{c}$ is used to represent interpretable features of the data. Following [9], the output of the generator becomes $G(\mathbf{z}, \mathbf{c})$. In order for the latent code $\mathbf{c}$ to have a meaning, there should be high mutual information between the code $\mathbf{c}$ and the generator distribution $G(\mathbf{z}, \mathbf{c})$, i.e. $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$. The term $I(c; G(\mathbf{z}, \mathbf{c}))$ represents the amount of information learnt from knowledge of $G(\mathbf{z}, \mathbf{c})$ about $\mathbf{c}$ and can be calculated using two entropy terms from,

$$I(\mathbf{c}; G(\mathbf{z}, \mathbf{c})) = H(\mathbf{c}) - H(\mathbf{c}|G(\mathbf{z}, \mathbf{c})) = H(G(\mathbf{z}, \mathbf{c})) - H(G(\mathbf{z}, \mathbf{c})|\mathbf{c}) \tag{2}$$

where $H(\mathbf{x})$ and $H(\mathbf{x}|\mathbf{y})$ are the entropy of the random variable $\mathbf{x}$ and the conditional entropy of $\mathbf{x}$ given $\mathbf{y}$, respectively.

Trying to get further insight into equation (2), one realizes the reduction of uncertainty about $\mathbf{c}$ when $G(\mathbf{z}, \mathbf{c})$ is observed. If the two variables are completely independent, then the term is equal to zero. So for the maximisation of (2), given $\mathbf{x} \sim P_G(x)$, $P_G(\mathbf{c}|\mathbf{x})$ should have small entropy. Trying to maximise this term, enforces a deterministic relationship between the two variables. To include this quantity in the optimisation process, the following modification is made in the cost functions from equation (1),

$$\min_G \max_D V_1(D, G) = \min_G \max_D V(D, G) - \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c})) \tag{3}$$

In practice though, maximisation of $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ is non-trivial, as one has no access to the

posterior $P_G(\mathbf{c}|\mathbf{x})$. To avoid maximising it directly, an auxiliary distribution $Q(\mathbf{c}|\mathbf{x})$ is defined,

$$
\begin{aligned}
I(\mathbf{c}; G(\mathbf{z}, \mathbf{c})) &= H(\mathbf{c}) - H(\mathbf{c}|G(\mathbf{z}, \mathbf{c})) = \\
&\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log P(\mathbf{c}|\mathbf{x})]] + H(\mathbf{c}) = \\
&\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log \frac{P(\mathbf{c}|\mathbf{x})}{Q(\mathbf{c}|\mathbf{x})} Q(\mathbf{c}|\mathbf{x}))]] + H(\mathbf{c}) = \\
&\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log \frac{P(\mathbf{c}|\mathbf{x})}{Q(\mathbf{c}|\mathbf{x})}] - \mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log Q(\mathbf{c}|\mathbf{x}))]] \\
&+ H(\mathbf{c}) = \\
&\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\underbrace{D_{KL}(P(\cdot|\mathbf{x})||Q(\cdot|\mathbf{c}))}_{\geq 0} - \mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log Q(\mathbf{c}|\mathbf{x}))]] \\
&+ H(\mathbf{c}) \\
&\geq \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log Q(\mathbf{c}|\mathbf{x})]] + H(\mathbf{c})
\end{aligned}
\tag{4}
$$

The technique of maximising the lower bound of the mutual information is called Variational Information Maximisation [19]. In equation (4) the quantity to be maximised is the first term of the final formula, since $H(\mathbf{c})$ is a constant and has an analytical form and a common distribution is chosen for $\mathbf{c}$. To avoid sampling from the posterior in the inner expectation of the expression in equation (4), the **Lemma 5.1** of [9] is used; according to which, for random variables X, Y and functions $f(x, y)$ under suitable regularity conditions it applies that: $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, x\sim X|y}[f(x,y)]$. Following this result a variational lower bound, $L_1(G, Q)$, of the mutual information $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ is defined,

$$
\begin{aligned}
L_1(G, Q) &= \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}|\mathbf{x})}[\log Q(\mathbf{c}|\mathbf{x})]] = \\
&\mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}), \mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\log Q(\mathbf{c}|\mathbf{x})] + H(\mathbf{c}) \leq I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))
\end{aligned}
\tag{5}
$$

In equation (5), the quantity to be maximised is the log-probability of the auxiliary distribution Q. In practice Q is modeled using a neural network. The newly-defined neural network is part of the discriminator network of the vanilla GAN. By defining the neural network Q, maximisation of the quantity in equation (5) is performed by back-propagating its error.

In the categorical case, in order to maximise $\log Q(\mathbf{c}|\mathbf{x})$ by training Q, the output is given by a softmax function and the categorical-crossentropy loss function gives the desired results. In the case of a continuous code $\mathbf{c}$, a maximum likelihood estimation (MLE) scheme has to be followed. The outputs of the neural network are two for each continuous code c_i; one for the mean value of the code and one for the variance. Afterwards, the loss function to be maximised is the mean log probability of all the outputs corresponding to the continuous code inputs. Assuming that $\mathbf{c}$ comes from a Gaussian distribution and that the variables c_i are independent, the loss function is

given by,

$$\log[Q(\mathbf{c}|\mathbf{x})] = \log[\prod_{i=1}^{n_c} P(c_i)] = \sum_{i=1}^{n_c} \log P(c_i) = \\ \sum_{i=1}^{n_c} [-\frac{1}{2}\log(2\pi\sigma_i^2) - \frac{(c_i - \mu_i)^2}{\sigma_i^2}] \propto \sum_{i=1}^{n_c} [-\log(\sigma_i) - \frac{(c_i - \mu_i)^2}{\sigma_i^2}] \tag{6}$$

In equation (6), c_i corresponds to the input value in the neural network and μ_i, σ_i are the mean and variance of each distribution of the code c_i. An overview of the network architecture of the infoGAN is shown in Fig 2. It becomes clear that the logic behind this architecture is that a sample is generated by the generator; it should look real, so that the discriminator classifies it as real and the parameters of the code **c** that were used to generate it should be as distinguishable as possible by the auxiliary neural network Q. Following this scheme, the infoGAN is expected to generate samples belonging to different classes (or manifolds) using different categorical codes $\mathbf{c}_{cat}$ and to explain the variability of features within the classes, using the continuous code $\mathbf{c}_{cont}$. Also, some random noise is added via **z** because as usual, noise in the data exists and should be modelled.

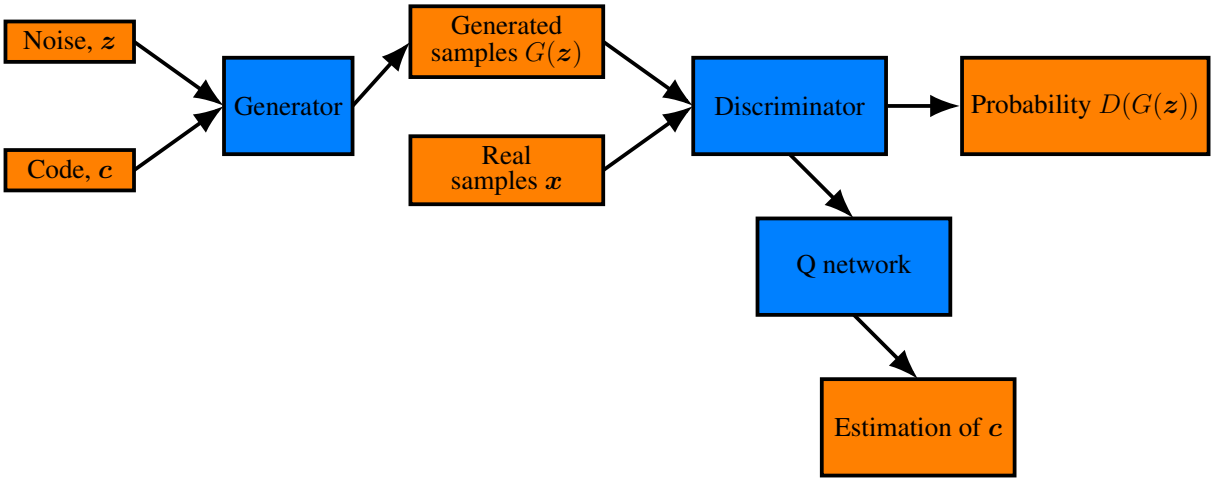


Figure 2: InfoGAN layout.

3 Application on simulated data

3.1 Simulation description

For the application of GANs to structural data, a simple simulated linear six-degree-of-freedom lumped mass system is examined. The masses of the system were all equal to one, the stifness of the undamaged states of the springs equal to 10^4 and a diagonal damping matrix was assumed with its elements equal to 25. The system was simulated using a fourth-order Runge-Kutta integration scheme. The excitation was a white noise signal applied on the first mass. In order to introduce different levels of damage in the structure, stiffness reductions were applied on springs 1-2, 2-3, 3-4 and 4-5. The reducted stifnesses came from normal distributions with mean values of 80% of the initial values of the system and variance equal to 4.5% of the initial stiffness of each spring.

In each simulation the *transmissibilities* between masses 1-2, 2-3, 3-4 and 4-5 were calculated, concatenated and considered as the samples representing each simulation. The normal condition concatenated transmissibilities are shown in Figure 4. Transmissibilities are a useful tool in monitoring real structures, where one has no knowledge about the input excitation to a dynamical system, in order to calculate the *frequency response function* (FRF). The sampling frequency of the simulation was $200Hz$; each transmissibility consists of 512 spectral lines and noise with variance equal to 5% of the transmissibility's variance was added to each one of them. In order to visualise the data, a *principal component analysis* (PCA) [20] was performed on the data and the first three principal components were plotted. In Figure 5 each batch/manifold of points corresponds to a different damage case. Moreover, through the gradual colour transition, the different damage levels are shown. A yellow colour represents points with low damage level (5-7.5% stiffness reduction) and dark red points correspond to higher damage levels (32-35%).

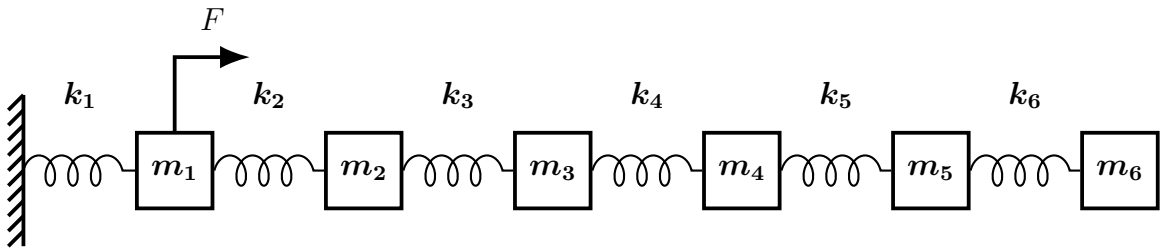


Figure 3: Mass-spring system.

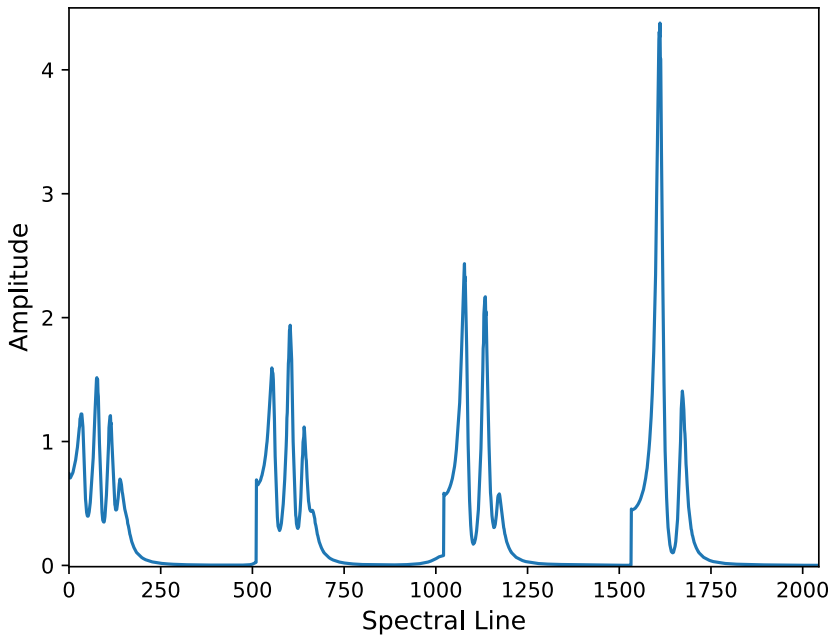


Figure 4: Normal condition concatenated transmissibilities.

Each damage case affects the transmissibilities in a different way. To be able to judge the results of the GANs to be trained consequently, an illustration of influence of that stiffness reduction on the transmissibilities is offered in Figure 6. Each different colour line corresponds to different

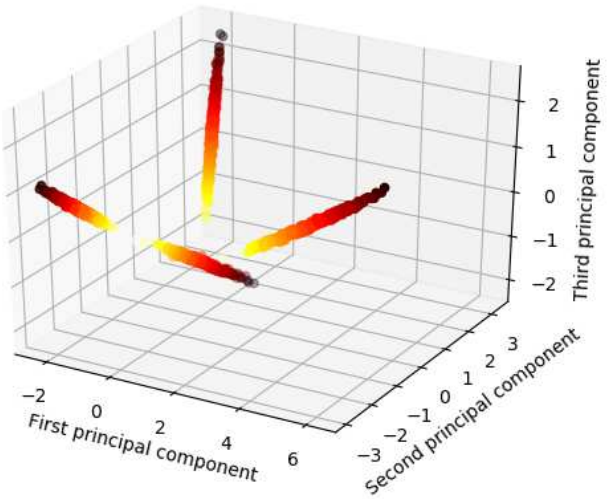


Figure 5: First three principal components of simulated transmissibilities. Gradual colour change from yellow to dark red represents lower to higher damage levels respectively.

damage level and the "movement" of the diagram as the damage extent increases is depicted. Damage induced in different springs causes similar effects to the one shown in Figure 6, but in different areas of the diagram.

3.2 InfoGAN training

The infoGAN was trained on the five first principal components of the data (explaining 99% of the variance) in order to reduce both training time and trainable parameters of the networks. Since the PCA transformation is linear, the infoGAN training is not considered to be assisted by it in any way other than the dimensionality reduction. In the input layer of the generator, two latent variables z_i sampled from a Gaussian distribution $z \sim N(0, 1)$ were used because noise is present in the data. Four categorical variables $c_{i,cat}$ sampled with equal probability were also used, since there are four different damage cases and one continuous c_{cont}, sampled also from a Gaussian distribution similar to the previous one. The expected result is that the generator should be able to generate samples corresponding to the different damage cases by altering the categorical variables and also corresponding to the damage extent by varying the continuous latent variable.

The discriminator was chosen to be a neural network with two hidden layers comprising 60 and 30 nodes and an output layer. The generator was also defined as a neural network with two hidden layers of 100 and 15 nodes and an output layer with five nodes. The ideal size of the networks is not considered to be the object of the current work, therefore, choosing the best architecture is not examined and was simply performed by trying different architectures manually and selecting the ones that had satisfying results in terms of the value of the loss function. The networks were initialised several times and the one with the lowest value for the loss function was kept. In contrast to vanilla GANs, where the value of the objective function oscillates during training, the value of the regularisation term in the loss function of infoGANs (4) gets lower by training and selection of the network with the lowest value of the cost function is a legitimate

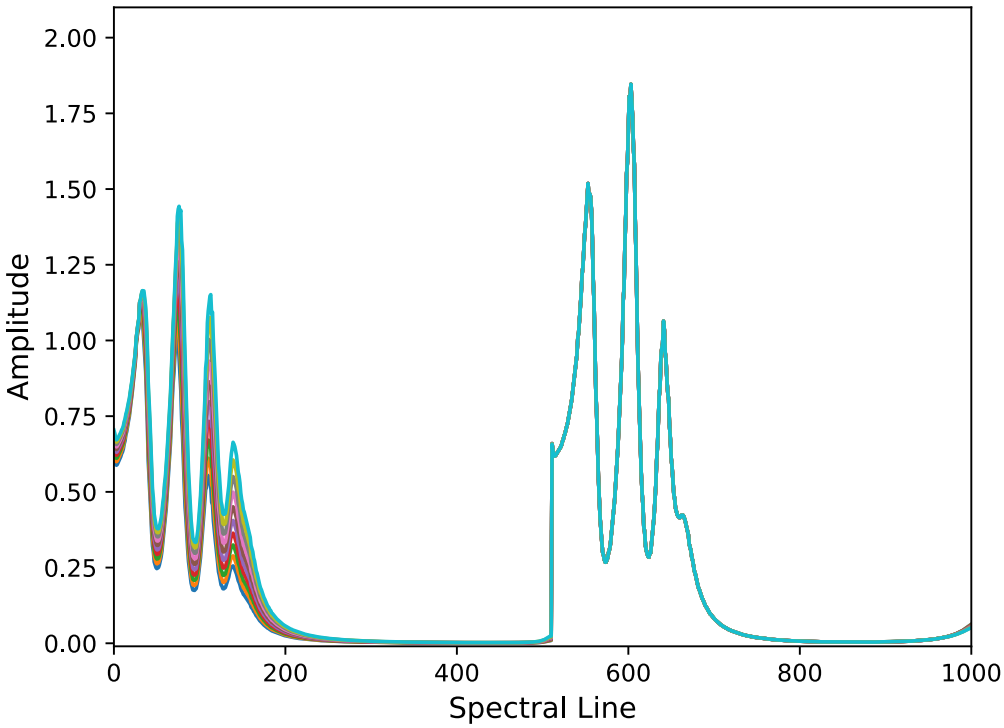


Figure 6: First damage case samples, focused on the first transmissibility, which is affected by the damage. Different colours reflect gradually increasing damage level, from cyan (no damage) to blue (highest damage level).

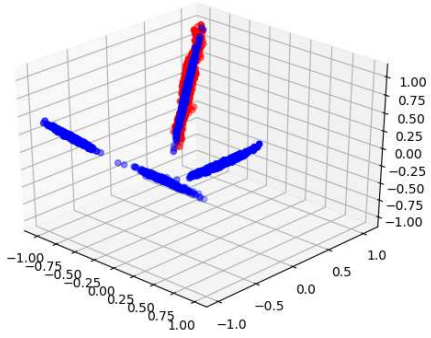
strategy.

Having trained the generator, random samples are generated and studied. For each categorical variable, 1000 samples were generated varying the continuous c_{cont} in the interval $[-2.0, 2.0]$ and generating random values for the latent noise vector $\mathbf{z}$. In Figure 7. It is clear that the generated points are quite close to the areas of the original points. It is also clear that each categorical variable forces the generated points to belong to different clusters of the data, as long as the manifolds of these clusters are not connected.

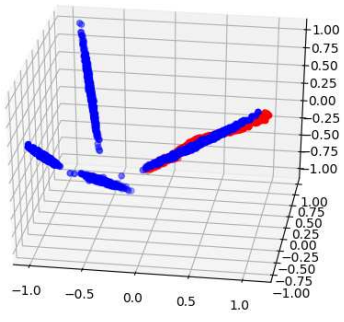
3.3 Classification and regression using GANs

In order to exploit the results of training a GAN, apart from generating artificial data, two more use cases are examined. The first one is classification of the data and the second one is a regression scheme; both in an unsupervised manner. The algorithm so far required minimal supervision for the results above. The only supervised part of the algorithm was the definition of the number of noise, categorical and continuous variables in the latent space.

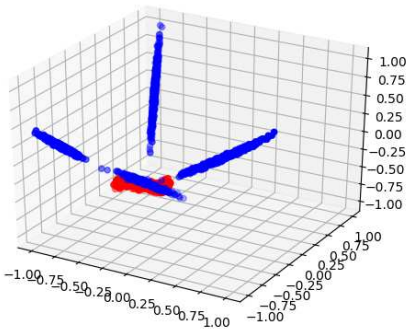
For the classification process, random samples are generated in the principal component space for each categorical variable separately. Subsequently, for each sample in the original dataset the closest generated point is sought in terms of the Euclidean distance between the points. The class assigned to the sample from the dataset is the class of the categorical variable that was used to generate the closest point to the sample. Following this procedure for every point in



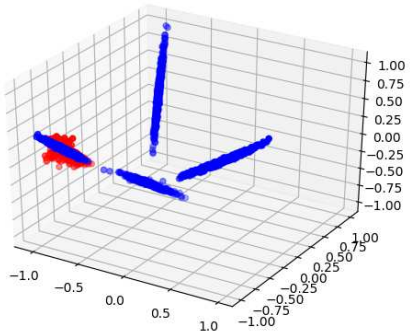
(a) First categorical variable.



(b) Second categorical variable.



(c) Third categorical variable.

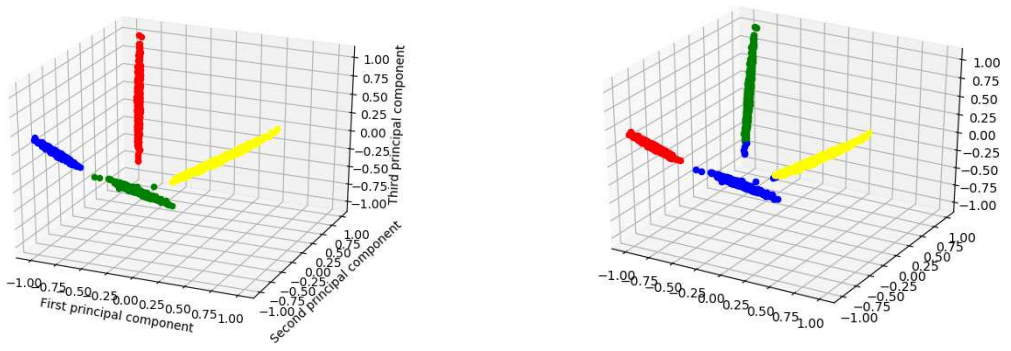


(d) Fourth categorical variable.

Figure 7: Generated points (red) and original points (blue) using different categorical variables.

the dataset, the result of the classification is shown in Figure 8a. Each colour corresponds to a different cluster assigned by the algorithm. The classification is perfect, as each point is classified correctly. In order to contrast performance against a more trivial unsupervised classification algorithm, a K-means clustering algorithm [21] is also applied on the same dataset and the results are shown in Figure 8b. It is clear that there are some misclassified points using the K-means algorithm.

Regarding the regression scheme, the captured variance from the continuous variable is examined. In order to illustrate what the continuous variable has captured, more samples were generated this time with the noise latent variables z considered constant and equal to zero. Again for each categorical variable, several samples were generated by using different values for the continuous variable in the interval $[-3.0, 3.0]$. The results are shown in Figure 10. The variable has captured some of the physics, explaining how the extent of damage affects the transmissibilities but the fit is not perfect. By generating samples in the principal component space and transforming them back to the original space, the effect of varying the continuous and the categorical code can be illustrated. In Figure 9 the transmissibilities generated the way mentioned above are shown for the categorical variable that corresponds to the first damage case. It is clear that the continuous variable has the same effect on the diagram as the same "movement" occurs as in the one with



(a) (b)

Figure 8: Results of the classification procedure using the GAN generator (left) and a K-means algorithm (right).

different damage extents in Figure 6. Similar behaviour is observed with the rest of the damage cases.

4 Conclusions

Summarising, this work demonstrates that there exist certain benefits in using GANs, and more specifically infoGANs, in SHM. Using simulated data, it was illustrated that an infoGAN with only prior knowledge of the number of different states of the structure is able to fit latent variables in the distribution of the real data. Being trained on the data, the algorithm could be used to classify the samples according to the categorical variables defined and at the same time to produce artificial samples in a desired class. Furthermore, the continuous variable was able to capture an underlying varying feature of the data, that of the extent of damage. As before, samples could be generated by altering the continuous variable, while keeping the categorical variable constant. This way, the effect of the level of damage was observed on the generated *transmissibilities*. It could be stated that the infoGAN is imitating the damage mechanism through the continuous code variables and such a scheme could be followed with a view to understanding how damage mechanisms affect specific structures.

Since the GAN is an artificial intelligence algorithm, the results it yields should be evaluated in this context. In artificial intelligence, the goal is for algorithms to perform as good as human intelligence would and if possible, outperform it. In the case of classifying points in a feature space, human-like behaviour is quite similar to the way infoGANs cluster data. A human examining this data, in order for two batches of points to belong in different clusters, their manifolds should be disjoint. The infoGAN exhibits a similar behaviour, as for it to generate points using different categorical variables, the real points should belong in two unconnected manifolds. Under this assumption, the algorithm's performance is quite good, since it classifies points the same way a human would and can moreover perform it in multidimensional spaces, which humans usually are not able to perceive and process.

Focusing on SHM and damage detection, if damage occurs gradually, the points representing the status of the structure smoothly moves in space as a function of the damage extent. In this case the infoGAN algorithm should be able to capture this change through a continuous latent

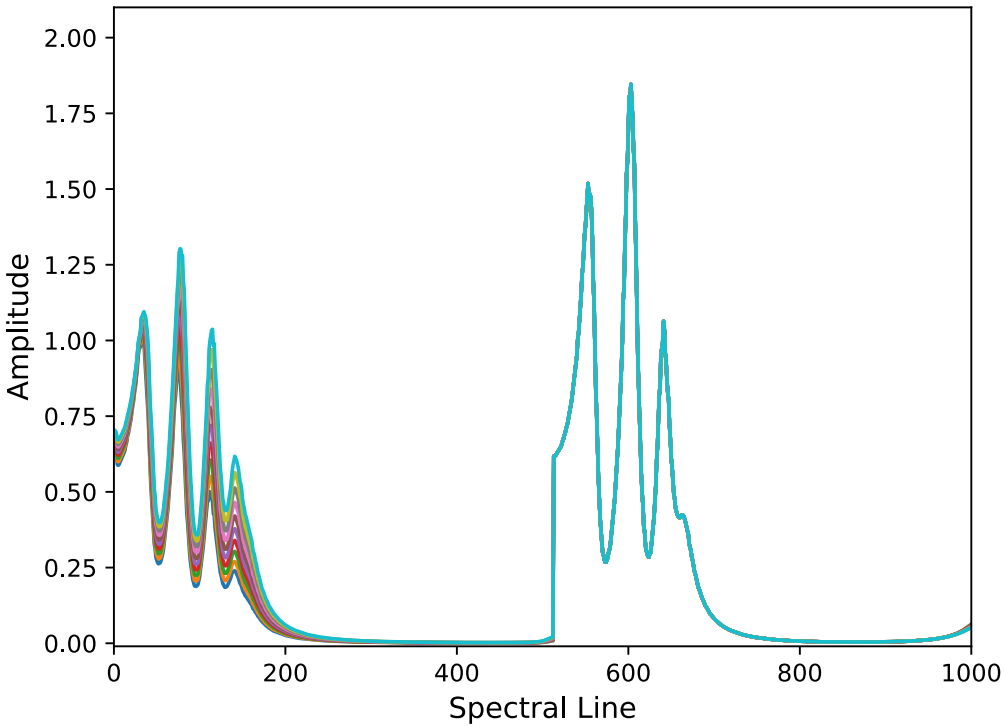
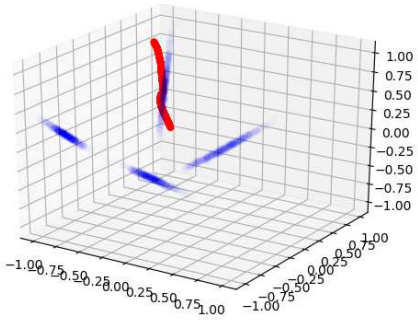


Figure 9: Generated samples, using a single categorical variable, corresponding to the first damage case and varying the continuous variable, focused on the first transmissibility, which is affected by the damage. Different colours reflect gradually increasing damage level, from cyan (lower damage level) to blue (higher damage level).

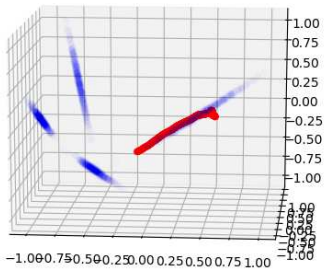
variable. If damage occurs suddenly, the infoGAN might be able to capture it as a different cluster through its categorical variables. At the same time, generation of artificial data for different clusters or values of underlying variables as damage extent is possible. In cases of continuous monitoring of structures, some varying environmental conditions in the acquired data may be captured. In this case, the infoGAN should also be able to capture the variation of the parameters in its continuous variables. This observation points out a potential use of infoGANs in large databases in order to define trends and different clusters of data caused either by damage or benign changing environmental conditions.

Acknowledgements

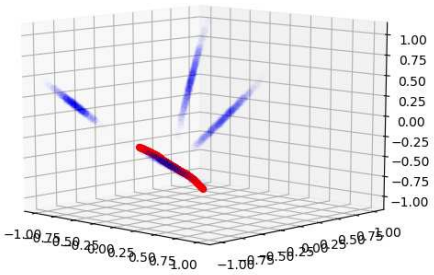
The authors would like to acknowledge the support of the European Union (EU). G.T is supported by funding from the EU's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement DyVirt (764547). The authors also gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) through grant references EP/R003645/1, EP/R004900/1 and EP/S001565/1.



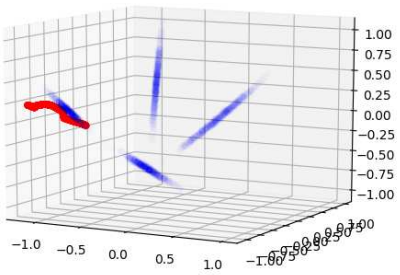
(a) First categorical variable.



(b) Second categorical variable.



(c) Third categorical variable.



(d) Fourth categorical variable.

Figure 10: Generated points (red) and original points (blue) using different categorical variables, constant noise variable equal to zero and varying the continuous code variable in the interval $[-3.0, 3.0]$.

REFERENCES

[1] A. Rytter, Vibrational based inspection of civil engineering structures, Ph.D. thesis (1993).

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

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

[4] G. Manson, K. Worden, D. Allman, Experimental validation of a structural health monitoring methodology: Part III. Damage location on an aircraft wing, Journal of Sound and Vibration 259 (2) (2003) 365–385.

[5] R. Miotto, L. Li, B. A. Kidd, J. T. Dudley, Deep patient: An unsupervised representation to predict the future of patients from the electronic health records, Scientific Reports 6 (1) (May 2016).

[6] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Inc., New York, NY, USA, 1995.

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

[8] D. P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Foundations and Trends in Machine Learning 12 (4) (2019) 307–392.

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

[10] D. F. Specht, A general regression neural network, IEEE transactions on neural networks 2 (6) (1991) 568–576.

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[12] L. Tarassenko, Guide to Neural Computing Applications, Elsevier, 1998.

[13] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, H. Greenspan, GAN-based synthetic medical image augmentation for increased cnn performance in liver lesion classification, Neurocomputing 321 (2018) 321–331.

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

[15] Y. Li, S. Liu, J. Yang, M.-H. Yang, Generative face completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3911–3919.

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

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

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

[19] D. Barber, F. V. Agakov, The IM algorithm: a variational approach to information maximization, in: Advances in Neural Information Processing Systems, 2003, p. None.

[20] K. Pearson, LIII. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11) (1901) 559–572.

[21] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.