

ABSTRACT

A major problem of structural health monitoring (SHM) has been the prognosis of damage and the definition of the remaining useful life of a structure. Both tasks depend on many parameters, many of which are often uncertain. Many models have been developed for the aforementioned tasks but they have been either deterministic or stochastic with the ability to take into account only a restricted amount of past states of the structure. In the current work, a generative model is proposed in order to make predictions about the damage evolution of structures. The model is able to perform in a population-based SHM (PBSHM) framework, to take into account many past states of the damaged structure, to incorporate uncertainties in the modelling process and to generate potential damage evolution outcomes according to data acquired from a structure. The algorithm is tested on a simulated damage evolution example and the results reveal that it is able to provide quite confident predictions about the remaining useful life of structures within a population.

1. INTRODUCTION

Structures are a major part of everyday life and activities and therefore their performance and safety should be monitored. Structural health monitoring (SHM) is the discipline of structural dynamics that aims in monitoring structures and maintaining their condition [1]. SHM is performed in various ways which can be summarised by Rytter's hierarchy [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*)?

The first step of Rytter's hierarchy has been widely addressed in many ways [3] and their application has been quite successful. The steps of localisation and classification have also been addressed in a similar data-driven manner [4]. However, the final two steps of the hierarchy are arguably the

George Tsialiamanis, Dynamics Research Group, The University of Sheffield, Sheffield

most difficult to deal with. In order to infer the extent of damage from measurements and estimate the remaining useful life of a structure, a thorough understanding of the existing damage mechanism is required as well as sufficiently accurate modelling of the environmental and loading conditions of the structure in the future [5]. The task is quite complicated for both physics-based and data-driven approaches. Especially, because of the uncertainty in the future environmental and loading conditions, the model, most probably, has to be stochastic. Approaches have been developed to perform such estimations following a particle filtering approach [6]. However, such approaches are based on a Markov chain assumption, i.e. the next state of the structure depends only on the current state (or a predefined number of previous states). This is a restriction, as the structure and the way that damage evolves might depend on states a few steps behind the previous measurements and the actual patent that affects the next step might even vary according to the current conditions.

In the current work, to deal with such issues, a data-driven approach is followed, which aims in creating a generative model that acts as a *Turing mirror* of a structure [7] with presence of evolving damage. A model is considered a Turing mirror if it can pass the Turing test [8]. The test has two participants, the *interrogator* and the *oracle*. The interrogator presents questions to the oracle and the latter answers. If the interrogator is not able to distinguish whether the oracle is a human or a machine, then the oracle is considered to pass the test [9]. The proposed algorithm is able to operate within a *population-based* SHM (PBSHM) framework. The algorithm to be used is a variation of the *generative adversarial networks* (GANs) [10]. The algorithm variation, which is designed to act on time-series of data is the *Time-series generative adversarial networks* (TimeGANs) [11] and is designed to generate artificial time-series that look real. It is also an attempt to create sequential models that are truly generative, since autoregressive approaches so far are useful in the context of forecasting but are not generative, they are deterministic [11]. The algorithm presented herein aims at providing the users with potential outcome scenarios to the evolution of damage in a structure.

2. TIME-SERIES GENERATIVE ADVERSARIAL NETWORKS (TIMEGAN)

Generative adversarial networks [10], where initially developed in order to generate images that look real to the human eye. The algorithm is based on the idea of adversarial training, which is a competition between two agents. The two agents in this case are two neural networks, the generator and the discriminator. The first is trying to transform random noise into image samples that look real and the second is trying to distinguish whether an image sample is indeed real or artificial. After training and because of the competition of the two, they both become better at their tasks. As a result,

the generator should be able to fool the discriminator into classifying artificial images as real. It is expected that the human eye will also be fooled by the generated images and in many cases it is proved that the quality and the resemblance to reality of the generated images is impressive [12, 13]. The discriminator is often an auxiliary network that is not used for some purpose, other than to train the generator.

The whole procedure can also be applied in order to generate common vector data, but has also been used in order to generate time-series data. Following the same scheme as with images and using the appropriate type of neural networks for time-series (recurrent neural networks), in [14] a GAN that generates music is presented. In [15], another approach is presented, which generates time-series for medical data according to some conditional variables, similar to conditional GANs [16]. However, for the current work, the TimeGAN [11] algorithm was chosen to be used in order to generate the desired time-series. The specific algorithm, as will be shown, is able to learn temporal characteristics of the time-series, as well as to generate potential outcomes of some incomplete time-series.

The TimeGAN algorithm is trained in order to learn two types of probability density functions. Firstly, the algorithm learns to generate timeseries $\mathbf{x}_{1:t}$, which look real. This yields the first condition that the algorithm is trying to satisfy, which is given by,

$$\min_{\hat{p}} D(p(\mathbf{C}, \mathbf{x}_{1:T}) || \hat{p}(\mathbf{C}, \hat{\mathbf{x}}_{1:T})) \quad (1)$$

where \mathbf{C} is some input condition vector variable that partially controls the timeseries, $p(\mathbf{C}, \mathbf{x}_{1:T})$ is the real probability density function of the timeseries, $\hat{p}(\mathbf{C}, \hat{\mathbf{x}}_{1:T})$ is the probability density function of the artificial timeseries, $\hat{\mathbf{x}}_{1:T}$ are the artificial timeseries and D is a distance metric of the probability density functions. By satisfying this condition, the timeseries, if considered as a whole, look real by some critic, in this case the discriminator, and, after completion of training, some human critic.

The second condition that the algorithm tries to satisfy is to learn temporal characteristics of the timeseries. This condition is based on the physics that define the potential next steps, given a set of existing steps of the timeseries. This condition shall prove quite useful, when one wants to generate potential outcomes of the timeseries, conditioned on some recorded values up to timestep t . The second condition is given by,

$$\min_{\hat{p}} D(p(\mathbf{x}_t | \mathbf{C}_t, \mathbf{x}_{1:t-1}) || \hat{p}(\mathbf{x}_t | \mathbf{C}_t, \mathbf{x}_{1:t-1})) \quad (2)$$

where $p(\mathbf{x}_t | \mathbf{C}_t, \mathbf{x}_{1:t-1})$ is the probability density function of the value of the timeseries for timestep t , given the timeseries so far $\mathbf{x}_{1:t-1}$ and the value of the condition variable \mathbf{C}_t , $\hat{p}(\mathbf{x}_t | \mathbf{C}_t, \mathbf{x}_{1:t-1})$ is the probability density function of the potential next steps of the timeseries, as it is generated by the algorithm and D is some appropriate distance measure of the probability density functions.

The framework followed in order to achieve a generator model that satisfies the above conditions is slightly different to the one followed in classic

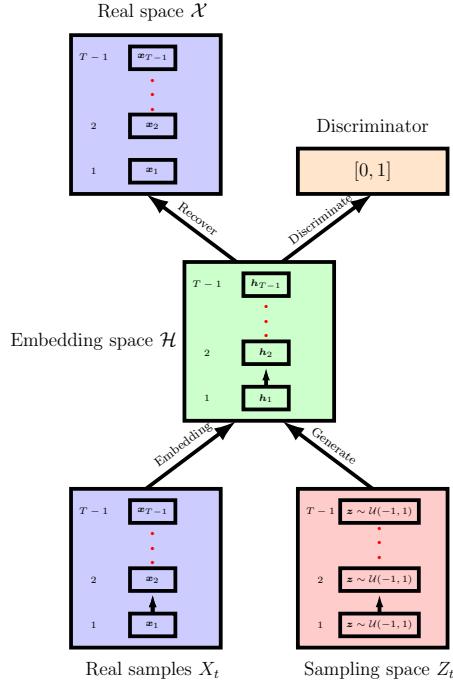


Figure 1: General framework followed in order to train the TimeGAN algorithm.

GANs. The main difference is that according to the TimeGAN framework, both a latent and noise space are used. The general framework is schematically shown in Figure 1. At first, timeseries from the real space $\mathbf{x}_{1:t}$ are converted into corresponding timeseries in the embedding space $\mathbf{h}_{1:t} \in \mathcal{H}$ using an embedding function e given by,

$$\mathbf{h}_t = e(\mathbf{C}_t, \mathbf{h}_{1:t-1}, \mathbf{x}_{1:t}) \quad (3)$$

where \mathbf{C}_t are the input variables that partially control the timeseries. As described by the above equation, the embedding depends on the embeddings up to the previous timestep and on the values of all the timeseries steps so far. This forces the algorithm to learn the embeddings of the timeseries according to patterns that can even be as long as the timeseries. Together with the embedding function, a function that restores the embedded timeseries back to the real space \mathcal{X} . The restoration function is given by,

$$\hat{\mathbf{x}}_{1:t} = r(\mathbf{h}_{1:t}) \quad (4)$$

where $\hat{\mathbf{x}}_{1:t}$ is the approximation of the reconstruction of the embedded timeseries $\mathbf{h}_{1:t}$. In practical applications, both functions are recurrent neural networks [17] and more specifically recurrent neural networks with *long-short term memory* (LSTM) units [18] or *gated recurrent units* (GRU) [19]. The specific types of neural networks can efficiently learn dependencies between distant timesteps of the timeseries.

In a similar manner, the generator and the discriminator of the algorithm are defined. The generator transforms timeseries from the noise space \mathcal{Z} to

timeseries in the embedding space \mathcal{H} . The equation of the generator is given by,

$$\hat{\mathbf{h}}_t = g(\mathbf{C}_t, \hat{\mathbf{h}}_{1:t-1}, \mathbf{z}_{1:t}) \quad (5)$$

where t is the current timestep, g is the generator network, $\hat{\mathbf{h}}_{1:t-1}$ is the generated timeseries in the embedding space up to timestep $t - 1$ and \mathbf{z}_t is the noise vector which is sampled from a pre-defined distribution (in the current work, the uniform distribution is used, i.e. $\mathbf{z}_t \sim \mathcal{U}(-1, 1)$).

Similarly, the discriminator tries to identify whether a timeseries is real or artificial. This is done by taking as input the whole embedded timeseries $\mathbf{h}_{1:t}$ and the potential input variables $\mathbf{C}_{1:t}$. The equation of the discriminator is given by,

$$y = d(\mathbf{C}_{1:t}, \mathbf{h}_{1:t}) \quad (6)$$

where d is the discriminator network. The prediction y of the discriminator is one for timeseries that it considers to be real and zero for fake or artificial timeseries. The discriminator and the generator are also recurrent neural networks with LSTM or GRU units.

For the purposes of satisfying the two conditions of equations (1) and (2), three different loss functions are used during training. The first one is defined for the training of the embedding and reconstruction of the timeseries. The reconstruction loss \mathcal{L}_R is given by,

$$\mathcal{L}_R = \mathbb{E}[\sum_t \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2] \quad (7)$$

which is essentially the mean square error between the original and the reconstructed values of the timeseries. This loss function is similar to when training an autoencoder [20].

The second loss function is for the adversarial training part of the algorithm. It is considered an unsupervised training and its goal is to force the discriminator and the generator to perform their tasks. The unsupervised loss function \mathcal{L}_U is given by,

$$\mathcal{L}_U = \mathbb{E}_{\mathbf{x}_{1:T} \sim p}[\sum_n \log y_n] + \mathbb{E}_{\mathbf{x}_{1:T} \sim \hat{p}}[\sum_n \log(1 - \hat{y}_n)] \quad (8)$$

where p is the probability density function of the real timeseries and \hat{p} is the probability density function of the generated timeseries, which should after training match the one of the original samples.

The third loss function used is targeted to imposing the condition of equation (2). The algorithm should be able to provide a range of potential next steps, given a set of observations up to some timestep. In order to impose this ability to the algorithm, the supervised loss function \mathcal{L}_S is used and is given by,

$$\mathcal{L}_S = \mathbb{E}_{\mathbf{C}, \mathbf{x}_{1:t} \sim p}[\sum_t \|\mathbf{h}_t - g(\mathbf{C}, \mathbf{h}_{1:t-1}, \mathbf{z}_t)\|_2] \quad (9)$$

where \mathbf{h}_t is the embedding of the values of the timeseries at timestep t . This latter loss function enforces the understanding of the physics of the timeseries to the algorithm. Generated steps are random because of the random input variables \mathbf{z}_t to the generator, but are also informed by the underlying physics of the timeseries up to step $t - 1$, since the generator is trained using the supervised loss function \mathcal{L}_S .

For the purposes of the current work, a fourth loss function is used. This last loss function does not affect the training procedure of the algorithm, but is used to train a model k that embeds real timeseries $\mathbf{x}_{1:t}$ into the noise space \mathcal{Z} . The embedding loss function \mathcal{L}_e is given by,

$$\mathcal{L}_e = \mathbb{E} \|\mathbf{x}_{1:t} - r(g(k(\mathbf{z}_{1:t})))\|^2 \quad (10)$$

where r is the recovery model, g is the generator and k is the embedding model. During training using this loss function, only the model k is trained, which is also a recurrent neural network. The parameters of every other model are considered constant for this later loss function. The model k will be particularly useful in order to embed timeseries in the noise space and generate potential next steps for them.

By using the specific algorithm on data that reflect the evolution of damage, the models are expected to learn the physics of the damage progress. At the same time, they are expected to incorporate the uncertainty of the process, since they are generative models and are able to generate potential outcomes. The algorithm should be able to match the probability density function of the next steps of a timeseries conditioned on the observed steps, according to the real uncertainty of the procedure, as it is reflected by the sample timeseries of the dataset, which will be used to train the TimeGAN.

3. TIMEGAN FOR REMAINING USEFUL LIFE ESTIMATION

Approaches have been developed in order to model damage evolution under uncertainty. In [21] an example is given about calculating the remaining useful life of structures with cracks in them. The approach is taking into account uncertainty, but yields quite wide intervals about the potential remaining useful life and is constrained to a specific type of damage, cracks, which should also be measurable. In [6], a first-order Markov chain assumption is made in order to define a model of the damage evolution. Although such approaches for some applications suffice a common drawback is that the first-order assumption may restrict the potential of the algorithm to locate dependencies between distant timesteps.

In order to address such issues, in the current work, the TimeGAN model is studied in order to model the evolution of damage. Such a model is considered herein to be a Turing mirror of a structure [7]. A Turing mirror is a model that behaves so similarly to the real structure, that an *interrogator* cannot distinguish whether the data come from real structures or from the Turing mirror model. In the case of the TimeGAN (or when training a

GAN model in general) the discriminator or some human plays the role of the interrogator. Using LSTM units for the various models that are included in the TimeGAN framework, it is expected that they will be able to learn long dependencies in the data and efficiently generate potential outcomes for damage scenarios.

The algorithm is even more appealing within a PBSHM framework [22, 23, 24, 25]. Such a framework allows exploiting data from different structures in order to perform inference about some structure that sufficient data might not be present. For damage prognosis and remaining useful life estimation, this might even be the only way to perform these tasks in a data-driven manner. Data from deployment until failure of a structure are definitely not available if the structure itself has not failed. Therefore, it is necessary to exploit data from similar structures in order to perform inference without including some physics in the inference procedure.

Following such a PBSHM framework, the form of a dataset D that can be used is given by,

$$D = \{(S_i, \mathbf{x}_{1:T_i}^i) \quad i = 1, 2, \dots, N, T_i \sim p_T\} \quad (11)$$

where S_i is the i th structure, $\mathbf{x}_{1:T_i}^i$ is the timeseries of some damage-sensitive feature monitored, T_i is the total lifetime of the i th structure, p_T is the probability density function of the total lifetime of all structures in the population and N is the number of structures in the population.

A naive approach to defining the remaining useful lifetime of the structures within the population would be to define a trivial probability density function according to the lifetimes of the structures in the dataset D . By doing so, one defines a form [22], which describes a general characteristic of the population. For a more sophisticated approach, such a dataset can be used in order to train the TimeGAN, which can be considered a Turing mirror of some damage type evolution process of the population. In order to use the model for some structure under evaluation, the sequence of data of the structure up to the current timestep t_c is required. To find the noise sequence $\mathbf{z}_{1:t_c}$, for which the generator g generates the testing timeseries $\mathbf{x}_{1:t_c}$, a recurrent neural network is trained, using as a loss function the embedding loss function of equation (10). Therefore, the embedding for some testing timeseries is given by,

$$\mathbf{z}_{1:t_c} = k(\mathbf{x}_{1:t_c}) \quad (12)$$

where k is the embedding model.

Since the noise embedding up to the current timestep t_c is available, generation of potential outcomes of the current incomplete timeseries is needed. To do so, a set of noise codes D_z is defined following,

$$D_z = \{\mathbf{z}_{1:T_f}^k \mid \mathbf{z}_{1:t_c}^k = k(\mathbf{x}_{1:t_c}), \mathbf{z}_{t_i}^k \sim \mathcal{U}(-1, 1) \forall t_i = t_c + 1, t_c + 2, \dots, T_f\} \quad (13)$$

where t_c is the current timestep of the tested structure and T_f is a number of timesteps large enough in order for the structure to have certainly reached

its failure point and $k = 1, 2, \dots, N_a$ where N_a is the number of artificially generated samples. Subsequently, the codes are used as inputs to the generator and a dataset D_x^g with potential outcomes of the currently tested timeseries is given by,

$$D_x^g = \{\mathbf{x}_{1:T_f}^k = r(g(\mathbf{z}_{1:T_f}^k))\} \quad (14)$$

In the dataset defined above, every timeseries is identical up to the current timestep t_c . Now, one can study the potential outcomes in order to see how the structure might behave in the future, i.e. one can ask the oracle questions, instead investigating the available data from the population. Moreover, one can define a criterion based on experience or some understanding in order to define when each potential outcome reaches the end of the lifetime of the structure. By collecting all the timesteps at which each outcome reached the end of the life of the structure, a probability density function can be defined over the remaining useful lifetime.

4. APPLICATION EXAMPLE

4.1. Description of the simulated dataset

In order to evaluate how effectively can the proposed algorithm provide estimations of the remaining useful life of a structure, a simulated dataset is considered. The simulated system is shown in Figure 2. In this system, damage is simulated as stiffness reductions of springs 2 and 3. In this case the damage sensitive feature, which is considered to be monitored, is the *frequency response function* (FRF) of the acceleration of the second mass.

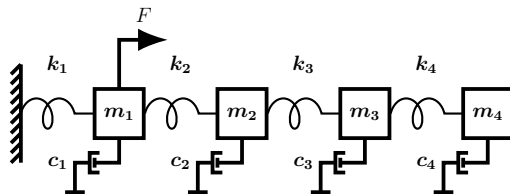


Figure 2: Mass-spring system.

In order to induce some physics into the problem, damage is considered to be increasing according to some rules. A first assumption is that the structures that are studied belong to a population (a homogeneous one) of 1000 structures. The initial and undamaged stiffness parameters of the springs are sampled from a Gaussian distribution. More specifically, $k_2, k_3 \sim \mathcal{N}(6000, 120)$. For the purposes of defining the damage evolution process, a nominal step k_n is uniformly sampled from the interval $[42, 90]$ for every structure. At every timestep of the damage evolution process, the new value of the stiffness of the second spring k_2 is defined by,

$$k_2^{t+1} = k_2^t - (k_n + \mathcal{N}(0, 0.1 \times k_n)) \quad (15)$$

where k_2^{t+1} is the stiffness of the next timestep and k_2^t is the stiffness of the current timestep. This damage evolution process was selected with a view to having multiple sources of uncertainty that the algorithm will have to learn. The first source of uncertainty is the selection of the nominal step, which however can be approximated as the mean value of the degradation steps when several timesteps are available. The second source of uncertainty is the last term in the right hand side in equation (15), which cannot be learnt somehow and the algorithm will have to model it as a random variable.

The limit state of the structures is considered to be when their third natural frequency reaches a specific point. That point here was the $119Hz$, chosen since most structures had higher third natural frequency than that after several damage evolution timesteps (it might even be considered a conservative limit). The selected criterion is based only on human examination of the data and is aimed at simulating a criterion derived from one’s experience. In Figure 3, FRFs of a structure as damage progresses are shown. Since the FRFs are high-dimensional features, in order to train the algorithm faster and be able to visualise the dataset, *principal component analysis* (PCA) [26] was performed on the timeseries of the FRFs and the first three principal components of a subset of timeseries are shown in Figure 4.

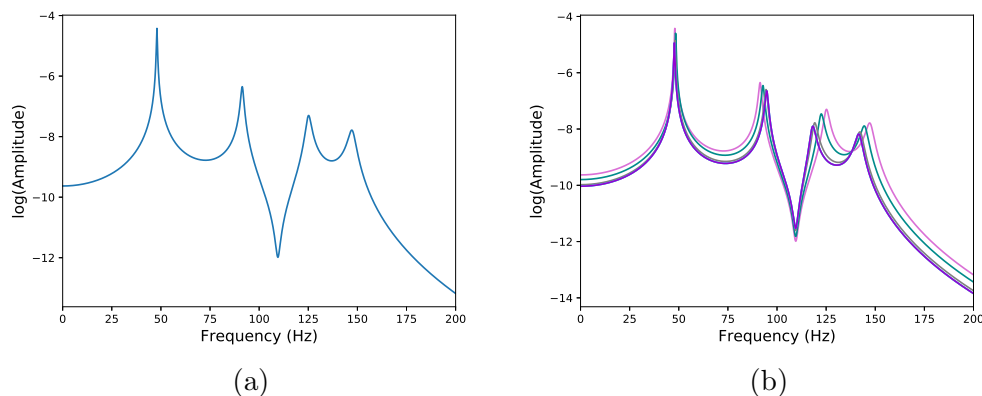


Figure 3: FRF sample on the beginning of the simulations (left) and FRFs with increasing damage (right), from low damage (pink) to higher stiffness reduction (purple).

4.2. Application of the TimeGAN on the simulated dataset

The TimeGAN model is considered as a model to define the remaining useful life of the structures. The neural networks of the model were all chosen to be LSTM neural networks with two LSTM units. Each LSTM network had neural networks with 128 neurons in their hidden layer and all the activation functions of the networks were hyperbolic tangent activation functions, except for the activation function of the discriminator, which was a sigmoid activation function. The sampling space \mathcal{Z} was selected to be a three-dimensional space and the embedding space \mathcal{H} a 128-dimensional space. The

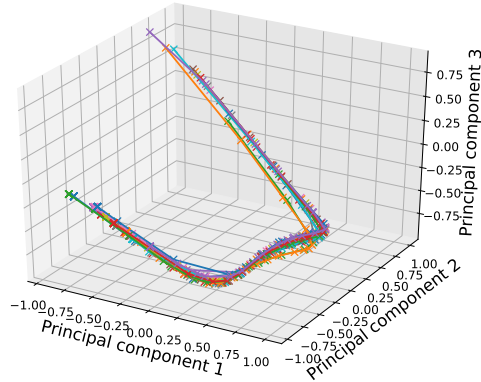


Figure 4: Principal components of paths from deployment until the failure of structures within the population; different colours correspond to different structures.

real space was three-dimensional, since a PCA was performed on the FRFs before the training of the algorithm and the three first principal components explained 96% of the variance of the data. In order to perform training, the Adam optimiser was used [27].

A standard and overall defined validation process for training generative adversarial networks has not been established. In contrast to traditional regression or classification tasks, where a cross-validation scheme can be followed, the performance of generative adversarial networks is often evaluated according to how real do images look like or how close are the generated data distributions to the distributions of the real data. In absence of such a validation scheme, in the current work, a metric chosen to evaluate whether the TimeGAN is properly trained was the distribution of the total lifetime of the timeseries. In Figure 5, a comparison between the PDF of the real samples and 1000 artificially generated by the TimeGAN algorithm samples is shown. The PDF was calculated using a kernel density estimation and the bandwidth was calculated using Silverman’s algorithm [28]. The KL divergence [29] between the two distributions is equal to 0.25, which for the purposes of the current work was considered to be low enough.

After training the model, the algorithm which was described in the previous section was applied. In order to evaluate the ability of the algorithm to provide estimations of the remaining useful life of the structures, a second testing population of structures is considered using the same characteristics as the population used for training. The testing population was generated using the same characteristics for the random distributions of the structural parameters, as well as the same procedure of random damage evolution. Afterwards, for every timestep of every structure before its failure, the algorithm was used to estimate the remaining life of the structures. In Figure 6, the evolution of the predicted PDF of the total lifetime of some structure

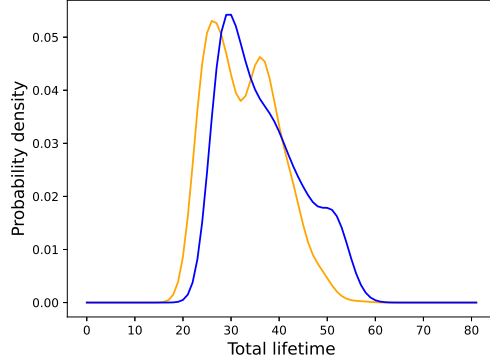


Figure 5: Probability density function of the total lifetime of the real samples (blue) and of the artificial samples (orange).

is shown, as the algorithm acquires more information at every timestep. In the same Figure, the red vertical line indicates the real total lifetime of the specific structure.

In order to holistically evaluate the algorithm, the mean probability assigned to the real remaining lifetime at every timestep is calculated from,

$$\mathcal{P} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} p(T_i) \quad (16)$$

where n_{test} is the total number of testing timesteps and $p(T_i)$ is the real lifetime of the testing structure i . This metric for the testing population was equal to 0.058. Following a naive approach and considering the PDF shown in blue in Figure 5 as a form [22] in order to assign a probability density to the remaining life of structures within the population, the metric of equation (16) is equal to 0.037. This means that the TimeGAN algorithm proposed provides predictions with higher confidence.

5. DISCUSSION

In the current work, a method to estimate the remaining useful life of structures within a PBSHM framework was presented. The algorithm is based on creating a Turing mirror model that is also a form of a homogeneous population. The model is about the evolution of damage that is observed on the structures. It is expected to learn the mechanism of the damage evolution and incorporate the uncertainty of the procedure. The core model used (TimeGAN) is a stochastic model that learns to generate artificial time-series according to rules learnt from a dataset incorporating any uncertainties that might be present in the procedure. The major advantage of the algorithm is that it learns to generate new points in the time-series taking into account previous states without a Markov assumption.

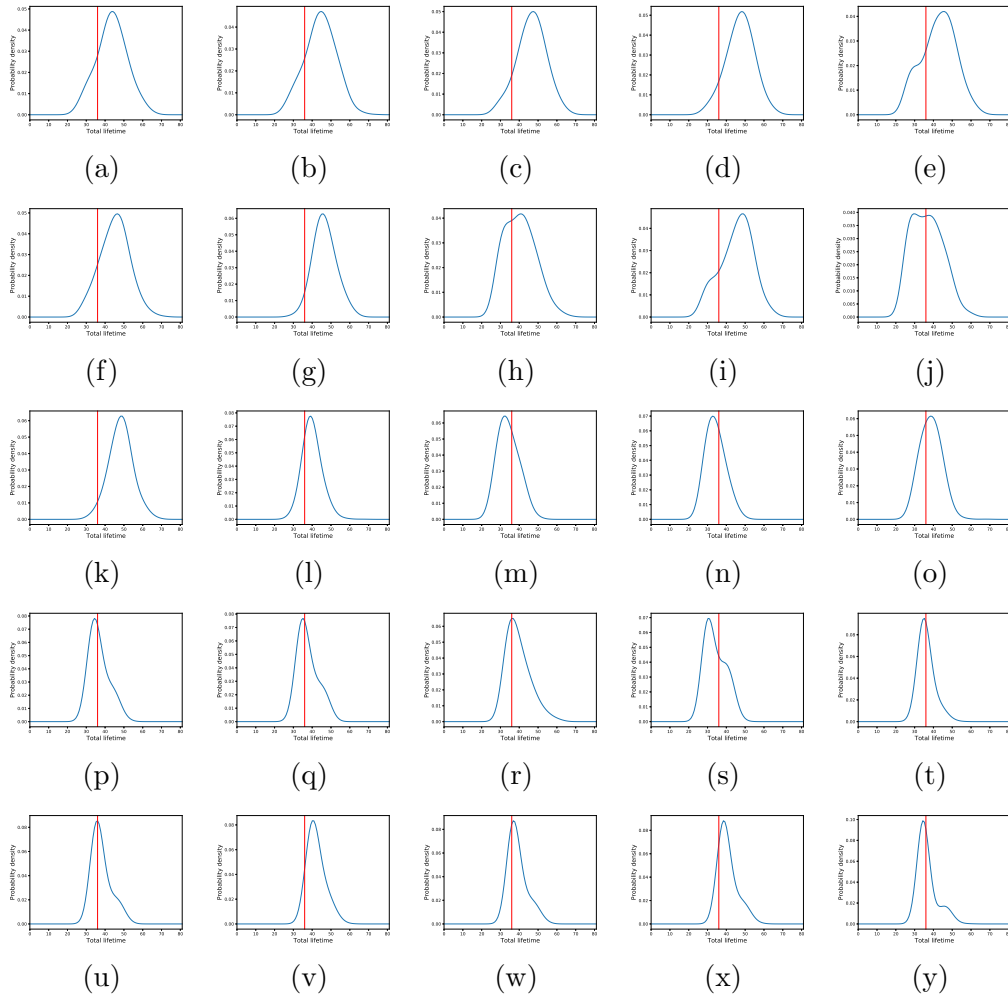


Figure 6: Real lifetime of tested structure (red vertical line) and evolution of predicted total life-time PDF using data from one more time-step in every figure; from left to right and from top to bottom.

In the simulated example presented, the algorithm yielded quite encouraging results. Even though the way the data were generated may not resemble some realistic situation, similar damage evolution mechanisms are expected to be observed in fatigue damage situations. The algorithm is expected to learn to model the uncertain quantities of the procedure, such as environmental conditions and random events. Moreover, the algorithm is able to provide the user with potential future timeseries. Therefore, one has the chance to study them according to his knowledge and make appropriate decisions about the future of the structure.

REFERENCES

- [1] Charles R Farrar and Keith Worden. *Structural Health Monitoring.: A Machine Learning Perspective*. John Wiley & Sons, 2012.

- [2] A. Rytter. *Vibrational based inspection of civil engineering structures*. PhD thesis, 1993.
- [3] K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85–101, 1997.
- [4] G. Manson, K. Worden, and 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):365–385, 2003.
- [5] C.R. Farrar and N.A.J. Lieven. Damage prognosis: the future of structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):623–632, 2007.
- [6] M. Corbetta, C. Sbarufatti, M. Giglio, and M.D. Todd. Optimization of nonlinear, non-gaussian bayesian filtering for diagnosis and prognosis of monotonic degradation processes. *Mechanical Systems and Signal Processing*, 104:305–322, 2018.
- [7] K. Worden, E.J. Cross, R.J. Barthorpe, D.J. Wagg, and P. Gardner. On digital twins, mirrors, and virtualizations: Frameworks for model verification and validation. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems Part B Mechanical Engineering*, 6(3), 2020.
- [8] A. M. Turing. Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer, 2009.
- [9] Ex machina, 2015.
- [10] 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.
- [11] J. Yoon, D. Jarrett, and M. van der Schaar. Time-series generative adversarial networks. 2019.
- [12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [13] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- [14] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

- [15] C. Esteban, S.L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.
- [16] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [17] L.R. Medsker and L.C. Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.
- [18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [20] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991.
- [21] K. Worden and G. Manson. Prognosis under uncertainty—an idealised computational case study. *Shock and Vibration*, 15(3, 4):231–243, 2008.
- [22] L.A. Bull, P.A. Gardner, J. Gosliga, T.J. Rogers, N. Dervilis, E.J. Cross, E. Papatheou, A.E. Maguire, C. Campos, and K. Worden. Foundations of population-based SHM, Part I: homogeneous populations and forms. *Submitted to Mechanical Systems and Signal Processing*, 2020.
- [23] J. Gosliga, P.A. Gardner, L.A. Bull, N. Dervilis, and K. Worden. Foundations of population-based structural health monitoring, part III: Heterogeneous populations – graphs, networks and communities. *Submitted to Mechanical Systems and Signal Processing*, 2020.
- [24] P.A. Gardner, L.A. Bull, J. Gosliga, N. Dervilis, and K. Worden. Foundations of population-based SHM, Part III: heterogeneous populations – transfer and mapping. *Submitted to Mechanical Systems and Signal Processing*, 2020.
- [25] G. Tsialiamanis, C. Mylonas, E. Chatzi, N. Dervilis, D.J. Wagg, and K. Worden. Foundations of population-based SHM, Part IV: The geometry of spaces of structures and their feature spaces. *Mechanical Systems and Signal Processing*, 157:107692, 2021.
- [26] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- [27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

- [28] B. W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological)*, 43(1):97–99, 1981.
- [29] S. Kullback. *Information Theory and Statistics*. Courier Corporation, 1997.