Cite this article

Mikhailova A, Adams NM, Hallsworth CA, Lau FDH and Jones DN (2019)
Unsupervised deep-learning-powered anomaly detection for instrumented infrastructure.
Proceedings of the Institution of Civil Engineers – Smart Infrastructure and Construction
172(4): 135–147,
<https://doi.org/10.1680/jsmic.19.00022>

Research Article

Paper 1900022
Received 15/12/2019; Accepted 14/09/2020
Published online 12/10/2020

Keywords: bridges/mathematical
modelling/statistical analysis

ICE Publishing: All rights reserved

Unsupervised deep-learning-powered anomaly detection for instrumented infrastructure

Aleksandra Mikhailova

PhD student, Department of Mathematics, Imperial College of Science, Technology and Medicine, London, UK (Orcid:0000-0002-6064-8031)
(corresponding author: a.mikhailova18@imperial.ac.uk)

Niall M. Adams

Professor of Statistics, Department of Mathematics, Imperial College of Science, Technology and Medicine, London, UK; Co-Director, Data Science Institute, Imperial College of Science, Technology and Medicine, London, UK (Orcid:0000-0003-0342-0513)

Christopher A. Hallsworth

Senior Teaching Fellow in Statistics, Department of Mathematics, Imperial College of Science, Technology and Medicine, London, UK

F. Din-Houn Lau

Lecturer in Statistics, Department of Mathematics, Imperial College of Science, Technology and Medicine, London, UK; Research Fellow, Lloyd's Register Foundation Programme on Data-Centric Engineering, The Alan Turing Institute, London, UK (Orcid:0000-0003-1065-828X)

Daniel N. Jones

Visiting Research Fellow, Mathematical Institute, University of Oxford, Oxford, UK

Deep learning methods have recently shown great success in numerous fields, including finance, healthcare, linguistics, robotics and even cybersports. Unsupervised learning methods identify the dominant patterns of variability that shape a data set. Such patterns may correspond to well-understood processes, previously unknown clusters or anomalies. This paper presents a case study where a state-of-the-art family of unsupervised deep learning models called variational autoencoder (VAE) is applied to data accrued from a network of fibre-optic sensors installed within a composite steel-concrete half-through railway bridge. The goals were (a) to characterise automatically the behaviour of the bridge based on sensor measurements and, (b) based on this characterisation, to determine when a train passes across a bridge. Based on the VAE model, an algorithm is presented to identify automatically the 'train event' points in an unsupervised setting. Two architectures for the VAE model are compared with commonly used baselines. The architecture tailored for modelling sequential data is shown to outperform other methods considered, on both seen and unseen data. No special hyperparameter optimisation is required. This study illustrates how state-of-the-art deep learning methods can be applied to a civil infrastructure engineering problem without directly modelling the physics of the objects or performing tedious hyperparameter optimisation.

Notation

C^t one of the hidden states of a recurrent neural network at step t

c_i vector of bias parameters of the i th layer in a multilayer perceptron

$D_{KL}(\cdot||\cdot)$ Kullback–Leibler divergence between two distributions

$\mathbb{E}_q[\cdot]$ mathematical expectation with respect to the distribution q

$f_i(\cdot)$ function computing the i th layer of a deep neural network

$f_{\dots}(\cdot, \cdot)$ mappings inside a recurrent neural network

$g(\cdot)$ non-linear activation function of a deep neural network

$g(\cdot, \cdot)$ mapping inside a recurrent neural network

H control limit parameter in a cumulative sum control chart (Cusum)

h^t one of the hidden states of a recurrent neural network at step t

k limit on the magnitude of changes in the process in Cusum

$\mathcal{L}(\cdot; \cdot, \cdot)$ evidence lower bound

$N(\cdot; \mu, \Sigma)$ probability density function of a Gaussian distribution with parameters μ and Σ

o^t output of a recurrent neural network at step t

$p(\cdot)$ probability density function

$p(\cdot, \cdot)$ probability density function of a joint distribution

$p(\cdot|\cdot)$ probability density function of a conditional distribution

$q_\phi(\cdot)$ distribution approximating the true posterior distribution

S_t^{high} higher cumulative sum in Cusum at step t

S_t^{low} lower cumulative sum in Cusum at step t

W_i matrix of weight parameters of the i th layer in a multilayer perceptron

x input vector to a deep neural network

x^t input to a recurrent neural network at step t

x_t strain at time t

y_t wavelength at time t

z vector of the latent values; more specifically, a realisation of the latent random variable Z

α threshold for the reconstruction probability

θ vector of the true parameters of the model

$\hat{\mu}$ estimated mean of the process in Cusum

$\mu_x^{(l)}$ l th output of the variational autoencoder (VAE) decoder (mean of the multivariate Gaussian in the input space)

μ_z	output of the VAE encoder (mean of the multivariate Gaussian in latent space)
ρ	photoelastic constant of the fibre-optic cable
$\Sigma_x^{(l)}$	l th output of the VAE decoder (covariance matrix of the multivariate Gaussian in the input space)
Σ_z	output of the VAE encoder (covariance matrix of the multivariate Gaussian in latent space)
$\hat{\sigma}$	estimated standard deviation of the process in Cusum
ϕ	vector of the parameters of model approximators

1. Introduction

The purpose of this paper is to investigate the applicability of a modern machine learning machinery to a real-world instrumented infrastructure problem. Smart sensors are being integrated into infrastructure: as an example, several bridges in the UK have been recently instrumented with fibre-optic sensors (Butler *et al.*, 2016). Analysis of measurements from such sensors can bring understanding of how bridges behave at rest, react to events and recover afterwards, which is helpful in monitoring bridge health and detecting possible damage, an activity that is generically referred to as structural health monitoring (SHM) (Chang *et al.*, 2003). These measurements record the response of multiple sensors over time and reflect both the behaviours of the bridge and the sensor network itself. These spatio-temporal data are high dimensional, high frequency and hard to interpret. Statistical methods are being developed to model such sensor data (Lau *et al.*, 2018a). The work presented in this paper demonstrates how a state-of-the-art unsupervised deep learning technique called variational autoencoder (VAE) (Kingma and Welling, 2013) can be leveraged to analyse these sensor measurements. This paper is an expanded version of the case study by Mikhailova *et al.* (2019), which considers only the ‘classic’ architecture of the VAE proposed by Kingma and Welling (2013) and compares it with a single commonly used benchmark. This extended version of the paper proposes another VAE architecture for this case study – one specifically tailored for sequential data types. In addition, two new benchmarks have been added, and the evaluation process has been expanded also to consist of two stages and thus has been made more robust and reliable. The engineering background and the connection of this work to SHM are also discussed in greater detail.

This paper investigates the applicability of VAE to the task of determining when a train passes across a sensor-instrumented bridge (‘a train event’). VAE is used to unravel the hidden structure in the sensor measurement data to identify these events, and the performance of this procedure is assessed. Thus, the case study is formulated as an anomaly detection task. The original purpose of these sensors is to provide SHM engineers with data to understand the behaviour of the bridge and to look for possible indicators of bridge degradation over time (Butler *et al.*, 2016; Lau *et al.*, 2018a). Bridge reaction to passing trains can be one of those indicators. In this case, automatically detecting when there is a train on the bridge is a proxy task that opens opportunities for further research on anomalous behaviour. However, the authors stress that the primary purpose of this paper is to show a proof of

concept for an anomaly detection method using modern unsupervised deep learning with an engineering data set.

The original data set does not contain any labels indicating when, exactly, there is a train on the bridge. Indeed, there is not a standard way to generate such a labelling based purely on the sensor data; hence, the authors frame train event detection as an unsupervised problem. The performance of the method is assessed in the manner of a supervised learning problem as common practice in anomaly detection studies (An and Cho, 2015). There is an unexpected latent structure in these sensor data (Lau *et al.*, 2018b), and the authors explore the extent to which the suggested unsupervised deep learning methods can identify this structure.

This paper provides necessary SHM and deep learning background, presents two different VAE models and defines a VAE-based train event detection algorithm based on the work of An and Cho (2015). The two VAE models presented in this paper are the ‘classic’ VAE model designed by Kingma and Welling (2013) and a VAE model based on a special kind of deep learning method tailored for sequential data. The data have pronounced temporal drifts partially due to environmental factors such as temperature. The authors propose a simple method for data transformation that removes temporal variability and ensures the stability of the VAE training procedure. Finally, the authors’ two VAE-based models are assessed using various classification performance metrics such as precision, recall and F_1 measure and compared with two variations of a widely used quality control method. It is demonstrated that both VAE models outperform the baselines, while the second VAE model, which is tailored for sequential data, has the highest quality of all.

2. Related work

This section reviews relevant literature in both SHM and deep learning domains and provides necessary deep learning background and definitions.

2.1 Structural health monitoring

Developments in instrumented infrastructure in recent years have enabled active research in the domain of SHM (Chang *et al.*, 2003). Automated collection and processing of high-quality measurement data help detect the signs of structural degradation and dramatically reduce costs associated with irreversible damage. As the volume and dimensionality of such data sets are typically high, manual data processing is often infeasible, while machine learning methods look particularly promising.

Machine learning distinguishes between supervised and unsupervised learning methods. Supervised methods require ground truth labels from which they learn. Examples of supervised learning tasks are classification and regression. Classification learns a discrete mapping from a data point to a class, based on the class labels present in the data. The goal of regression is to learn a continuous mapping from a point to a real-valued scalar or vector. Unsupervised learning methods do not

require any labels and at their essence, map high-dimensional input data to a numerical space of a lower dimensionality.

Supervised machine learning techniques are widespread in the domain of SHM. Two typical tasks where supervised methods are commonly used are damage type classification and damage localisation. Kim and Philen (2011) employ Adaboost classification algorithm (Freund and Schapire, 1997) to distinguish between two types of damage in metallic structures: cracks and corrosions. Tibaduiza *et al.* (2018) use supervised deep-learning-based detection and classification of damage in two different types of structures. Yizhou *et al.* (2017) propose a solution to the structural damage localisation task using deep learning models, formulating damage localisation as a supervised classification task.

However, supervised methods depend on the availability of the ground truth data. In their notable review paper, Farrar and Worden (2007) discuss the tasks and challenges of SHM. They argue that the problem of damage detection is a statistical learning task that often must be performed in an unsupervised setting due to the lack of the labels corresponding to the data entries representing damage. A similar point is made by Yuan *et al.* (2020): due to the possible lack of different damage scenarios in the training data, damage identification is often presented as an outlier detection task. Nick *et al.* (2015) follow this paradigm and employ unsupervised learning techniques for locating the damage but use supervised learning for identifying damage type and severity. Hernandez-Garcia and Masri (2014) develop fully unsupervised statistical monitoring methods using latent variable techniques for the detection of sensor faults characterised by abnormal sensor readings.

This study is facing a similar challenge, as the true train event labels do not exist in the data that the authors are concerned with, and there is not a commonly accepted way to obtain these labels; hence, the authors resort to unsupervised techniques. Following Farrar and Worden (2007) and Yuan *et al.* (2020), the authors present a proof of concept for a state-of-the-art unsupervised deep learning method that is based on the idea of anomaly detection, using a real-world engineering data set. The authors show that the suggested method can capture hidden structure in these data and discuss the practical steps needed for the method to work.

2.2 Deep learning

With recent improvements in computational capacity, deep learning methods are becoming increasingly popular and are already being adopted in numerous fields. Such fields include speech recognition (Graves *et al.*, 2013), natural language processing (Sutskever *et al.*, 2014), computer vision (Krizhevsky *et al.*, 2012), finance (Heaton and Polson, 2016), robotics (Levine *et al.*, 2018), playing board games (Silver *et al.*, 2017) and even solving fundamental scientific problems such as modelling protein structures (Evans *et al.*, 2018). Deep learning excels at analysing high-dimensional data and discovering complex hidden dependencies.

VAE is a deep-learning-based dimensionality reduction method. Unsupervised dimensionality reduction techniques are vital for numerous tasks, such as data compression and visualisation, generation of realistic synthetic data, feature extraction and data density modelling. The most widely used techniques include principal component analysis (PCA) (Pearson, 1901) and its non-linear generalisation kernel PCA (Schölkopf *et al.*, 1998). The popular *t*-distributed stochastic neighbour embedding method (van der Maaten and Hinton, 2008) projects data points onto a low-dimensional (two-dimensional (2D), three-dimensional) space and hence, is well suited for visualisation. In modern machine learning, neural-network-based data mappings have been most popular in recent years. A notable example is the family of word2vec models (Mikolov *et al.*, 2013) that embed words into a numerical vector space so that words often appearing together in texts become vectors close to each other. Google's trained word2vec model was trained on about 100 billion words taken from the Google News data set (word2vec, 2013), which illustrates the efficiency and capacity of neural networks.

Autoencoders are a family of techniques that learn both embedding ('encoder') and inverse mapping ('decoder', or 'reconstruction') simultaneously (Goodfellow *et al.*, 2016a). VAE adds a probabilistic flavour to the autoencoder framework by operating with distributions over embeddings and reconstructions rather than single-point estimates. This section describes the building blocks of the VAE in more detail.

2.2.1 Deep neural networks

As briefly mentioned earlier, an autoencoder learns its encoder and decoder mappings jointly. Modern autoencoder models, including VAE, typically use deep neural networks (DNNs) as their encoders and decoders. DNNs are commonly used to construct complex non-linear representations suitable for challenging data sets.

The DNN is the core building block of deep learning algorithms. There are numerous DNN architectures, each tailored for certain tasks and types of data. This paper considers the architecture called multilayer perceptron (MLP). An MLP is a composition of at least three functions f_n ($f_{n-1}(\dots f_1(x))$), where each component ('layer') $f_i(\cdot)$, except for the last, consists of a non-linear function, called an activation function, applied to a linear transformation of the input (Goodfellow *et al.*, 2016b):

$$1. \quad f_i(x) = g(W_i^T x + c_i)$$

Here, W_1 and c_1 are parameters to be determined, which are typically estimated by way of gradient descent optimisation of some objective function. The layers $f_2(\cdot)$, ..., $f_{n-1}(\cdot)$ are called hidden layers. DNNs are known to be extremely good and scalable approximations of complex functions with high-dimensional inputs.

2.2.2 Recurrent neural networks

Optimisation of the parameters of MLPs is typically fast in practice, and there are also known improvements to it that help it achieve greater efficiency (Verma, 1997). The procedure usually quickly converges to high-quality solutions for many data sets of different kinds. However, MLPs do not account for any sequential structure in the input data. Moreover, many implementations of MLPs shuffle the training data before and even during the optimisation procedure to make the training process more robust. This makes it challenging for MLPs to process data adequately with sequential dependencies in it, such as text written in natural language. To capture such serial relationships in the input data, recurrent neural networks (RNNs) were designed.

RNNs originate from the work of Rumelhart *et al.* (1986). There are many different RNN designs, but they are all built on the idea of having loops inside the neural network to make the signal coming from the training data persist. This is achieved by performing repeated applications of one function with the same parameters to each element of the data sequence in turn while saving some information in the unit called 'hidden state' after each of these applications (Goodfellow *et al.*, 2016c). The parameters of the function being applied to the members of the sequence stay the same for all applications; hence, all members of the input sequence share the same function. The hidden state, however, changes after each function call. This state 'remembers' information seen in the previous data in the sequence and allows processing of further members within some already known context. This is particularly beneficial for data types such as text and numerical time series. However, it is worth noting that RNNs take much longer to optimise than MLPs, as they require processing of all entries in the training data set one by one, so there is a trade-off between the accuracy of the final model and the optimisation time.

2.2.3 Autoencoders

Classic autoencoders use two separate MLPs as encoders and decoders. For sequential input types, encoder and decoder MLPs can be replaced with RNNs for better quality of data processing. In training, the objective of an autoencoder is to minimise the metric called reconstruction loss, which is typically the mean squared error between the input and its reconstructed code. However, classic autoencoders are deterministic in nature, except for random initialisation of weights, and do not provide any measure of uncertainty either for the inferred code (i.e. the latent representation) or for the reconstruction. For complex non-linear models, it is usually necessary to perform approximate inference in order to quantify such uncertainty. VAE performs efficient variational inference, as discussed in Section 2.2.4, under certain assumptions about the model while keeping its latent space close to a standard Gaussian distribution and aiming at partitioning the latent space so that similar inputs are mapped to close codes.

2.2.4 Variational autoencoder

Consider a generative latent variable model that consists of a random variable X driven by a latent random variable Z . The joint probability density function can be written as follows:

$$2. \quad p_{\theta}(x, z) = p_{\theta}(x|z)p_{\theta}(z)$$

where θ denotes the true parameters of the model. Note that the authors are following the notation similar to that used by Kingma and Welling (2013). The authors would like to point out that various notations are used in the literature in this area, and they often differ from statistical conventions. Given a set of independent and identically distributed samples from $X|Z$, the goal is to estimate the true parameters θ and to sample from the posterior distribution of $Z|X$. This generally implies computing the marginal likelihood $p_{\theta}(x)$ and the posterior $p_{\theta}(z|x)$ shown in Equations 3 and 4, respectively, which are often intractable or infeasible to calculate and, hence, require approximation.

$$3. \quad p_{\theta}(x) = \int_z p_{\theta}(x, z) dz$$

$$4. \quad p_{\theta}(z|x) = \frac{p_{\theta}(x, z)}{p_{\theta}(x)}$$

The two most common approaches to posterior approximation are Markov chain Monte Carlo methods (Gelfand and Smith, 1990; Hastings, 1970) and variational inference (Jordan *et al.*, 1999; Wainwright and Jordan, 2008). According to Blei *et al.* (2017), the latter is often faster and scales better. VAE employs variational inference for its training procedure. In variational inference, the true posterior $p_{\theta}(z|x)$ is approximated by a simpler distribution $q_{\phi}(z)$ whose parameters ϕ are learned jointly with the parameters θ of the generative model. The maximisation objective is the evidence lower bound (Elbo), which is a lower bound of the true marginal likelihood $p_{\theta}(x)$:

$$5. \quad \mathcal{L}(q; \theta, \phi) = \mathbb{E}_{q_{\phi}(z)}[\log p_{\theta}(x|z)] - D_{\text{KL}}(q_{\phi}(z) \parallel p_{\theta}(z))$$

where $D_{\text{KL}}(q_{\phi}(z) \parallel p_{\theta}(z))$ denotes the Kullback–Leibler (KL) divergence. The KL divergence concept and notation are clearly explained in the book by Cover and Thomas (1991), and $\mathbb{E}_{q_{\phi}(z)}[\log p_{\theta}(x|z)]$ is the expected value of the log-likelihood function $\log p_{\theta}(x|z)$ with respect to the authors' approximate posterior distribution, $q_{\phi}(z)$ (Wasserman, 2004). The derivation of Equation 5 is given in the paper by Blei *et al.* (2017). Intuitively, by optimising Equation 5, the likelihood of the decoder is optimised (i.e. converting codes into realistic data) while keeping the latent space of the codes close to the prior over Z .

Under certain assumptions made in the VAE framework, $\mathcal{L}(q; \theta, \phi)$ is differentiable almost everywhere with respect to both θ and ϕ and can be optimised by way of gradient ascent with the stochastic gradient variational Bayes algorithm (Kingma and Welling, 2013).

More specifically, the authors consider the case when $q_\phi(z)$ and $p_\theta(x|z)$ are assumed to be Gaussian distributions whose parameters are estimated by encoder and decoder DNNs. All covariance matrices are assumed diagonal for simplicity, rather than a valid modelling assumption in the problem context. It is possible to extend the model to non-diagonal covariance matrices. The prior over the latent variables Z is set to the standard normal distribution, $\mathcal{N}(0, \mathbf{I})$. With these assumptions, the probability density function $p_\theta(x|z)$ is differentiable almost everywhere with respect to θ , and the KL divergence term in Equation 5 can be computed and differentiated in closed form as shown by Kingma and Welling (2013).

2.2.5 Long short-term memory-based VAE

As mentioned earlier, in the classic VAE model proposed by Kingma and Welling (2013), the encoder and decoder are MLP networks. In this paper, two VAE architectures are considered: a classic MLP-based VAE and a VAE using a special kind of RNNs as its encoder and decoder, called long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997).

Due to repeated applications of the same function to the input sequence in simple RNNs described in Section 2.2.2, it is often the case during gradient-based optimisation that the computed gradients either approach zero and lose predicting power or become numerically too large, causing instability and failure of the training procedure. This is known as the problem of vanishing and exploding gradients (Bengio *et al.*, 1994), which happens when the input sequences are sufficiently long, thus making it challenging to learn long-term dependencies with this simplest RNN architecture. There are several existing solutions for this problem, such as gradient norm clipping and constraint softening (Pascanu *et al.*, 2013). LSTMs were also designed partially to overcome this problem and to learn efficiently the long-term dependencies in the data. The core idea of LSTMs lies in filtering ('forgetting') some of the information coming from the newly seen data and learning which parts of the information can be filtered ('forgotten'). LSTMs effectively learn to store what is important while aiming to filter out the noise and thus make the gradient descent-based optimisation procedure more robust. LSTMs have since become a standard model choice for handling sequential data sets. They are widely used for processing complex data and have been shown to excel at modelling text (Sutskever *et al.*, 2014), music (Eck and Schmidhuber, 2002) and financial time series (Chen *et al.*, 2015).

The recurrent structure of LSTMs is as follows. The LSTM has an internal state that stores dependencies within the input data. The internal state is updated at each time step, using a weighted combination of its previous value and values computed from the data at the current time. The coefficients of this linear combination are not constant, but rather values of functions that decide what is to be retained or thrown away at every time step. These coefficients have different values for all entries in the data set. The formal structure of the RNN is presented as follows (Hochreiter and Schmidhuber, 1997; Olah, 2015):

$$6. \quad \text{forget}^{(t)} = f_{\text{forget}}\left(h^{(t-1)}, \mathbf{x}^{(t)}\right)$$

$$7. \quad \text{retain}^{(t)} = f_{\text{retain}}\left(h^{(t-1)}, \mathbf{x}^{(t)}\right)$$

$$8. \quad C_{\text{new}}^{(t)} = f_C\left(h^{(t-1)}, \mathbf{x}^{(t)}\right)$$

$$9. \quad C^{(t)} = \text{forget}^{(t)} * C^{(t-1)} + \text{retain}^{(t)} * C_{\text{new}}^{(t)}$$

$$10. \quad \mathbf{o}_{\text{init}}^{(t)} = g\left(h^{(t-1)}, \mathbf{x}^{(t)}\right)$$

$$11. \quad h^{(t)} = \mathbf{o}^{(t)} = \mathbf{o}_{\text{new}}^{(t)} * \tan h\left(C^{(t)}\right)$$

where '*' denotes the element-wise product. This batch of equations characterises one time step of the LSTM and is called an 'LSTM cell'. Here, $f_{\text{forget}}(\cdot, \cdot)$, $f_{\text{retain}}(\cdot, \cdot)$, $f_C(\cdot, \cdot)$ and $g(\cdot, \cdot)$ are usually compositions of a linear mapping and a non-linear activation. The linear combination in Equation 9 allows the LSTM cell to control the amount of information taken from the previous time step and received from the new entry in the data sequence. As coefficients for this linear combination are computed by the functions whose parameters are optimised during the network training as seen in Equations 6 and 7, the rate of forgetting already seen examples depends on the given data set. As opposed to the simple RNN, LSTM maintains two hidden states instead of one: $C^{(t)}$ and $h^{(t)}$. Due to this greater number of mappings to be learned and hidden states to be stored, LSTM allows more flexibility.

In this case study, two LSTMs are employed as the encoder and decoder of the VAE, as the bridge sensor data are a time series with both spatial and temporal relationships in the data. LSTMs as encoders and decoders of both classic and variational autoencoders are widely used for sequence modelling (Bowman *et al.*, 2016; Gensler *et al.*, 2016).

3. Data

This analysis considers three distinct data sets of sensor measurements collected from one 26.8 m composite bridge located in Staffordshire, UK. Each of these three data sets is a result of up to 1 h worth of continuous monitoring of the sensor measurements. The three data sets were recorded at different times, when the weather conditions and other environmental factors (both unrecorded) might have been different. The data sets consist of 80-dimensional vectors representing measurements of 80 sensors taken at 250 Hz frequency. The sensor network consists of fibre-optic sensors with Bragg grating that were installed during bridge construction. This sensor network

was installed on this bridge to measure the vertical deflections, measured in strain, of the bridge. There are two separate rail tracks on the bridge for trains travelling in opposite directions. There are four lines of 20 equidistant sensors located in the main girders of the bridge. The details of installation of these sensors are described by Butler *et al.* (2016). The data sets contain 611 108, 908 238 and 803 594 observations, accounting for runs of approximately 2440, 3630 and 3210 s, respectively. Such large volumes of data observed at high frequency pose challenges for analysis.

The original data record wavelength measurements in nanometres. Following Lau *et al.* (2018b), the sensor measurements are converted to relative strains by way of the photoelastic constant as shown in Equation 12. This conversion of the data transforms the data into interpretable physical units and plays an important role in practical use of neural networks.

$$12. \quad x_t = \frac{1}{1 - \rho} \left(\frac{y_t - y_1}{y_1} \right) \times 10^6$$

In Equation 12, y_t and x_t are wavelength and strain (expressed in microstrains) at time t and $\rho = 0.22$ is the photoelastic constant of the fibre-optic cable.

As discussed earlier, the sensor data sets do not contain marked timestamps indicating when a train was on the bridge. The periods where trains were on the bridge were manually identified. Figure 1 shows 8 s of the sum of strains of all sensors marked by crosses and coloured in a darker shade where, according to the manual inspection, an event happens approximately.

The sensor data exhibit unpredictable temporal variation partially due to environmental factors such as temperature. Figure 2 shows a drift occurring in the authors' third data set in the strains of a sensor. Such variability in the data tends to negatively affect

Figure 1. Sum of sensor strains around a train event

machine learning methods that do not account for any sequential structure, or even a model aware of sequential structure that assumes stationarity. Therefore, in Section 5.1, a preprocessing technique for removing such drifts from the data is proposed. The spikes in the data that can be seen in Figures 2 and 3 likely correspond to the train events that the authors aim to detect automatically.

It is worth noting that temporal variation and train events manifest different features in individual sensors, subject to where the sensor is positioned on the bridge. Figure 3 shows strain values of a second example sensor from the same subset of data as that shown in Figure 2.

Part of the authors' interest in using VAEs is to capture automatically the rich structure present in these data.

Figure 2. Drift in the strains of an example sensor

Figure 3. Drift in the strains from a second example sensor

4. Model for anomaly detection

The train event detection problem is defined as an anomaly detection task: sensor measurements corresponding to train events are considered anomalies, and data points corresponding to when the bridge is at rest form the reference data distribution. An anomaly detection formulation is appropriate here as during manual inspection of the data discussed in Section 3, it was noticed that the suggested train events comprise only a small portion of the whole data set. It is also a commonly accepted formulation in the SHM literature, as presented in Section 2.1.

Two VAEs are used: the classic VAE where both the encoder and decoder are MLPs and a VAE with LSTMs as its encoder and decoder. The reconstruction probability ('recproba') metric calculated by way of Algorithm 1 as per An and Cho (2015) is employed to distinguish an 'event' from a 'no event'. More details on the architecture of the networks are given in Section 5.5. Notably, this reconstruction probability metric is based on a logarithm of the Gaussian probability density function (i.e. log-likelihood) and, despite its name, is not strictly a probability measure.

Algorithm 1. Calculation of reconstruction probability

> Let L be number of samples to compute recproba.
> Get $[\mu_z, \Sigma_z] = \text{encoder}(x)$.
> Draw $z^{(1)}, \dots, z^{(L)} \sim \mathcal{N}(\mu_z, \Sigma_z)$.
> Get $[\mu_x^{(l)}, \Sigma_x^{(l)}] = \text{decoder}(z^{(l)})$ for l in $1 \dots L$.
> Finally: $\text{recproba}(x) = \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x; \mu_x^{(l)}, \Sigma_x^{(l)})$.

Here, $\mathcal{N}(x; \mu_x^{(l)}, \Sigma_x^{(l)})$ is the probability density function for the normal distribution with the given parameters, evaluated at data point x . Essentially, the reconstruction probability is the Monte Carlo estimate of $\mathbb{E}_{q_\theta(z|x)}[\log p_\theta(x|z)]$, the left-hand term of the Elbo objective presented in Equation 5. For computational simplicity reasons, the number of samples L is set to 1 in this experiment.

To perform the event/no event classification, a threshold α is chosen for reconstruction probability. If the recproba value for a data point x is lower than this threshold, x is marked as an anomaly – that is, a train event. In this case study, α is set to -130 based on empirical observation of a subset of the data. Generally, the choice of α controls the false-positive and the false-negative rates of such binary classifiers (see Section 5.2 for details on the evaluation methods).

There are two major simplifications in this model. First, for computational convenience, both covariance matrices Σ_z and $\Sigma_x^{(l)}$ are diagonal; hence, correlation between sensors is not modelled directly. Second, temporal dependencies in the data are not captured in this method when MLPs are used as the encoder and decoder in VAE.

5. Demonstration

This section discusses data preparation for anomaly detection, describes the baselines used to compare the VAE models against and defines metrics for evaluation of model quality. Finally, the evaluation of the baselines and the VAEs is presented.

5.1 Data preprocessing

As shown in Figures 2 and 3 in Section 3, the individual sensor strain data are non-stationary and have unpredictable temporal variation. Such variation often interferes with the process of training neural networks in the VAE models. The highly complex and non-convex VAE loss function is sensitive to variability in the input data, and when drifts become more pronounced (e.g. like that shown in Figure 2), the loss explodes, and the training procedure does not converge. Hence, a data preprocessing procedure is introduced to remove the temporal variations in the data, which are largely due to changes in temperature. While this data preprocessing is crucial to ensure convergence of the training procedure of the authors' deep learning models, it is valid for preparation for any model, not just deep-learning-based approaches. The proposed data normalisation procedure has only one hyperparameter (desired frequency of the data), which is sufficiently easy to tune. The procedure is as follows.

The difference transformation is a common method for removing temporal variability from data (Cowpertwait and Metcalfe, 2009: p. 93). It is performed by subtracting the data point at the previous time step from the data point at the current time: $\Delta x_t = x_t - x_{t-1}$. However, as the sensor strain data were taken at a high frequency, differences between consecutive observations are small enough that the train events are no longer apparent in the data plots, in the time domain at least. This phenomenon is shown in Figure 4.

To resolve this issue, the data are first downsampled so that only one observation is retained at regular intervals. Intuitively, downsampling the data should make the differences larger when a train event starts to happen compared with when the bridge is at rest. The authors show empirically that lowering the frequency from 250 to 5 Hz is enough to remove visibly the temporal variability while preserving the signal of train events in the data.

Figure 4. Sum of sensor strains before and after the difference transformation

Figure 5 shows the same subset of data differenced at various frequencies: 250 Hz (original frequency), 50 Hz and 5 Hz.

5.2 Metrics and manual identification of train events

The performance of the authors' algorithms is evaluated by considering how accurately they detect train passage events. Therefore, in order to establish whether an algorithm has correctly detected a train passage event, some ground truth is required. As mentioned previously, the sensor data sets contain no marked timestamps indicating when a train passage event occurred. Therefore, manually identified train passage events are used as the ground truth. This is a standard evaluation practice for unsupervised learning methods when there is no gold standard model to compare against. Event and no-event labels needed for evaluation of the authors' algorithms were constructed manually by considering the sum of strains of all sensors and hand-picking events. The authors would like to stress that these labels are artificial, and they are used solely for evaluation in this paper. The event and no-event classes turned out to be heavily imbalanced: the event class accounts for only about 0.43% of all the observations (out of 2 322 940 data samples in total, 9934 were labelled as 'event'). This manual inspection is laborious and infeasible for larger data sets; hence, unsupervised automated methods are important.

For model evaluation, the standard metrics of the quality of binary classification are used: precision, recall and F_1 measure (Manning *et al.*, 2008: pp. 154–157). The definitions of these metrics are built on the concept of a confusion matrix for binary classification. A binary confusion matrix is a table that characterises the performance of a classification algorithm based on the known ground truth labels. Henceforth, the event class is referred to as 'positive' and the no-event class as 'negative'. The confusion matrix counts the number of times that the algorithm correctly identified instances of both classes ('true positive' and 'true negative') and the number of times that it mistook an

ordinary data point for an anomalous event ('false positive') and vice versa ('false negative') (Table 1).

In this context, precision is the ratio of correctly detected events to the total number of data points labelled as events, which measures the accuracy of an anomaly detector:

$$13. \text{ precision} = \frac{TP}{TP + FP}$$

Precision (Equation 13) alone is not enough to evaluate the quality of a classifier. Suppose that the classifier is tuned to be very conservative to the false-positive labels and classifies an entry as positive only in very few cases, hence missing a large amount of actual positive samples. In this setting, precision is going to be very high, but such a classifier is practically useless. Therefore, the number of samples labelled as positive needs to be assessed as well.

Recall (Equation 14) is the ratio between the number of items correctly labelled as positive by the classifier and the number of all items that have ground truth positive labels. Recall essentially characterises how many positive items that the classifier misses, the quantity that is particularly important to access when the classes are unbalanced (as in the authors' case):

$$14. \text{ recall} = \frac{TP}{TP + FN}$$

Similarly, recall can be high even if the classifier is poor: consider the classifier that labels every data point as positive.

Finally, the F_1 measure (' F_1 ') balances precision and recall and summarises them in one number. It is defined as the harmonic mean of precision and recall:

$$15. F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

These three metrics require ground truth event and no-event labels to be assigned to the data. Hence, manual identification of the train events in the sensor data was performed as mentioned earlier.

5.3 Baseline: cumulative sum control chart

The cumulative sum control chart ('Cusum') method initially proposed by Page (1954) is applied to the authors' event detection problem, and the VAE model is compared against it. As the

Figure 5. Sum of sensor strains before and after difference transform with a lowered frequency

Table 1. Layout of the binary confusion matrix

	Predicted yes	Predicted no
True class yes	True positive (TP)	False negative (FN)
True class no	False positive (FP)	True negative (TN)

sensor data are multidimensional and classic Cusum accepts only one-dimensional (1D) inputs, the vector data must be transformed before being passed to the Cusum mechanism. To achieve this, two dimensionality reduction techniques are considered. All sensor strain values are also transformed before they are summed, as described in Section 5.1.

The first data summarisation technique is summing up the strains of all 80 sensors at each time step. The second technique is computing the first principal component (Pearson, 1901) of the measurement vectors at every time step. The first principal component finds the direction of the largest variance. Using the sum or the first principal component of strain measurements within a Cusum is a simple and arbitrary choice. There are downsides in both these approaches. Using the sum makes the input signal noisy, while using the first principal component might lose quite a lot of significant information, as it is not always possible to compress the data to one dimension. Other 1D summaries can be used within the Cusum approach.

The Cusum method is used to detect points in a time series when the process goes out of control. It monitors changes in the mean of the process over time and signals when it detects a deviation from the expected mean that is outside of the allowed limits. This paper considers two standard Cusum charts for train event detection, where events are flagged using the same threshold on both charts.

The method uses the following parameters: (a) expected mean of the process, $\hat{\mu}$; (b) magnitude of a change in the process that is to be labelled as an anomaly, k ; and (c) control limit, H . The parameters k and H are set to $\hat{\sigma}$ and $3\hat{\sigma}$, respectively, where $\hat{\sigma}$ is the estimated standard deviation of the process. In this case study, $\hat{\mu}$ and $\hat{\sigma}$ are set manually based on empirical observation of a subset of the data. When the summing of all sensors is used as the summarisation technique, $\hat{\mu}$ and $\hat{\sigma}$ are set to 0 and 4, respectively. When the first principal component is used, these values are set to 0 and 15, respectively. For consistency, the data subset used here is the same as that chosen to compute the VAE reconstruction probability threshold mentioned in Section 4. The full Cusum-based anomaly detection method is presented in Algorithm 2.

Algorithm 2. Cusum-based event detection

- > Compute $\hat{\mu}$ and $\hat{\sigma}$.
- > Set $k := \hat{\sigma}$ and $H := 3\hat{\sigma}$.
- > Set $S_1^{low} := 0$ and $S_1^{high} := 0$.
- > For each data point (sum OR first principal component of sensor strains) x_t :
 - >> Set $S_t^{low} := \min(0, S_{t-1}^{low} + x_t - \mu + k)$.
 - >> Set $S_t^{high} := \max(0, S_{t-1}^{high} + x_t - \mu - k)$.
 - >> If $S_t^{low} \leq -H$ or $S_t^{high} \geq H$, label x_t as event.

5.4 Baseline: supervised classification with support vector machines

Support vector machines (SVMs) (Cortes and Vapnik, 1995) are a widely used classification method that aims to separate the data

classes with a hyperplane in its kernel space. In their review of supervised learning techniques for SHM, Nick *et al.* (2015) found that an SVM with a linear kernel performs well at the task of classifying types of structural damage. This paper uses SVM as an additional supervised learning baseline for this experiment; however, the authors want to stress that supervised learning methods will not be applicable in practice for this problem due to the lack of true labels in the data. A binary SVM classifier is trained using the artificial (manually constructed) ground truth described in Section 5.2. The inputs to SVM are preprocessed as described in Section 5.1, for a fairer comparison with the VAE models.

5.5 Technical specifications of the VAE models

The input to VAE is multidimensional; hence, vectors of sensor strains are passed to it directly after the preprocessing procedure described in Section 5.1 is performed.

The presented choice of the hyperparameters and the DNN properties (e.g. number and sizes of layers, types of activation functions) is a common standard choice for MLP and LSTM networks and the VAE framework. No special hyperparameter tuning was performed for this case study. It is technically possible to tune all these characteristics. However, it is remarkable that this somewhat arbitrary standard choice adopted already shows decent performance.

The data are normalised by computing the Z score before being passed to the VAE model, as such standardisation of inputs aids the convergence of the training of DNNs.

5.5.1 MLP-based VAE

Both the encoder and decoder are MLP networks with one hidden layer (see Section 2.2.1). The hidden layer size is set to 40, rectified linear unit activation (Nair and Hinton, 2010) is chosen as the non-linearity and L2 weight regularisation is used in both the encoder and decoder. The latent space (code) dimensionality is set to 2. All the neuron weights are initialised randomly from a Gaussian distribution with mean 0 and variance 0.01. The model is trained on the whole data set with mini-batches of size 64 for 50 epochs by way of stochastic gradient descent using the Adam optimiser (Kingma and Ba, 2014).

5.5.2 LSTM-based VAE

Both the encoder and decoder are RNNs with three LSTM layers (see Section 2.2.5), two of which are executed in parallel.

In the encoder, the size of the output of the first LSTM layer is set to 40, and its activation function is the hyperbolic tangent function. In the second and third LSTM layers, the output size is equal to the latent space dimensionality, which is set to 2. The second LSTM layer outputs the mean of the Gaussian distribution in the latent space, while the third layer outputs the (diagonal) covariance matrix of the distribution. This Gaussian distribution is the approximation of the true posterior distribution (see Section 2.2.4). No activation is used in the second and third LSTM layers, as their outputs represent the parameters of a Gaussian distribution, which can be unbounded.

The decoder employs a symmetrical RNN with three LSTM layers. The outputs of this recurrent network are the mean and the (diagonal) covariance matrix of the distribution over possible reconstructions. This distribution is the generative part of the authors' latent variable model represented by the VAE (see Section 2.2.4).

Similarly to the MLP-based VAE, the model is trained for 50 epochs with mini-batches of size 64 using the Adam optimiser (Kingma and Ba, 2014).

5.6 Evaluation

The evaluation of the methods is performed in two stages.

The first stage of the evaluation consists of measuring the quality of the VAE-based event detection models on the exact data set that the underlying VAEs were trained on (Section 5.6.1). It is important to note that the thresholds for the anomaly detector were not tuned on the same data set, and the VAE is a fully unsupervised framework that does not require any labels for training. Hence, no bias is introduced into this evaluation procedure by using already seen data.

In the second stage of the evaluation, the 'transferability' of the models is measured by testing each of the three anomaly detection models on the two data sets that were not used to train the model (Section 5.6.2). For instance, the anomaly detection model using the VAE trained on data set 1 is evaluated on data sets 2 and 3. The average of the performance results of the three models is reported for each of the standard metrics.

5.6.1 Evaluation on seen data

Table 2 gives the results of the evaluation of the two Cusum-based and the two VAE-based anomaly detection methods and one SVM-based binary classifier. Both Cusum- and VAE-based algorithms are sensitive to the choice of hyperparameters, mainly to the thresholds controlling the size of anomalies to be detected. As mentioned earlier, no special hyperparameter tuning was performed – reasonable standard choices were adopted instead. With the given choice of parameters, VAE noticeably outperforms Cusum and SVM, while the notable high precision and low recall of SVM are expected due to class imbalance in the data.

Figure 6 shows one of the events labelled by Cusum and VAE anomaly detectors, compared with the manually constructed ground truth. Notably, the sum-based Cusum produces a few false positives right after the end of the event and misclassifies several anomalous points in the middle of the event. The first-principal-

Figure 6. Predicted against ground truth labels. PC, principal component

component-based Cusum is the worst performer. The first principal component might not be the best choice for anomaly detection on data that consists of a mixture of two classes, as it aligns with the line connecting the centroids of the two clusters, and the first part of the train event might go parallel to the direction of the first principal component.

Note that as the frequency of the data was lowered during preprocessing, the number of points input to the algorithms and shown in Figure 6 is smaller than the original number of entries in this time interval. To bring the predicted labels to the original data, one can set all the original observations falling between the two labelled points to event or no-event based on these two predicted labels.

5.6.2 Evaluation on unseen data

Table 3 shows the results of the evaluation of the four anomaly detection methods and the classification method on the unseen data, meaning that only the data that were not shown to the models during training are used for evaluation. It is worth noting that the Cusum-based benchmarks do not require any training except for threshold tuning, which is performed separately. Hence, their results should not be sensitive to the data set.

The MLP-based VAE fails to perform on the data that it has not seen before. Its precision is low, while its recall is high, which means that it is calling a lot of false positives of the event class. The LSTM-based VAE, on the other side, has a good performance regardless of the data set.

Table 2. Evaluation results on the seen data

	Cusum (sum)	Cusum (first PC)	SVM	VAE (MLP)	VAE (LSTM)
Precision	0.579	0.282	1.000	0.840	0.931
Recall	0.905	0.623	0.186	1.000	0.955
F ₁ measure	0.706	0.388	0.314	0.913	0.943

Table 3. Evaluation results on the unseen data

	Cusum (sum)	Cusum (first PC)	SVM	VAE (MLP)	VAE (LSTM)
Precision	0.582	0.354	1.000	0.376	0.944
Recall	0.904	0.611	0.188	1.000	0.966
F ₁ measure	0.708	0.399	0.314	0.532	0.948

Figure 7. Sums of sensors in each of the three data sets

Figure 8. Latent space of the MLP-based VAE

Figure 7 shows the sums of all sensors for each of the three data sets used. Notably, the data sets are structurally different. However, the well-chosen RNN structure of the VAE model performs well on all data sets despite those differences.

5.6.3 Latent space of the VAE

Apart from event detection, another interesting property of the VAE is the representation of the sensor strain data in the latent space. Figure 8 shows the 2D latent space of the MLP-based VAE on a subset of the seen data. The data points are marked based on the manually constructed ground truth markers. One can see that the event and the no-event data are clearly separated in the latent space, with the no-event data distribution being close to a standard Gaussian.

6. Conclusion

The purpose of this case study is to show that state-of-the-art deep learning methods have potential for engineering problems and are

feasible to adopt. It is crucial that an unsupervised learning approach is adopted in this experiment due to the lack of true data labels. VAE is a fully unsupervised probabilistic deep learning technique that is commonly used to model data distributions. This paper demonstrates how VAE models can be applied to a real-world anomaly detection problem for instrumented infrastructure. The performance of the authors' VAE models is reasonable compared with that of the widely used Cusum benchmark, while VAE, unlike Cusum, can process vector data and map inputs to a low-dimensional space, separating the anomalous points there. The VAE approach does not involve any mathematical modelling of the bridge structure and behaviour. As this case study reveals, VAE can be treated as a black box as long as the input data do not have drifts and are standardised to expedite the convergence of the training procedure. The authors showed that a standard choice of hyperparameters provides good performance.

As stated earlier, the standard VAE model is not sequential in nature – that is, it does not consider temporal dependencies between subsequent data points. When training on or predicting for x_t , VAE does not take earlier observations $x_1, \dots, x_{t-1}$ into account and assumes the input process to be stationary. Due to this, the LSTM-based VAE was proposed for the authors' anomaly detection task, and it has been shown that the LSTM-based model is more robust and transferrable to other data sets, showing considerably better performance on both seen and unseen data. However, the authors observed that LSTMs are also sensitive to perturbations of input data. Hence, the proposed data preprocessing procedure is crucial for both architectures to ensure training convergence.

Finally, comparison with a widely used supervised benchmark (SVM) has been performed (purely for demonstration, as supervised methods cannot be applied to this problem due to the lack of the ground truth event and no-event labels), and the proposed VAE-based method has been shown to outperform it.

The full source code of the authors' case study is available online at the GitHub repository by Mikhailova *et al.* (2020).

Acknowledgements

The authors would like to acknowledge the Cambridge Centre for Smart Infrastructure and Construction, The Laing O'Rourke Centre at Cambridge and the Staffordshire Alliance (Network Rail, Volker Rail, Atkins and Laing O'Rourke) for providing the data set used in the paper. F. D.-H. Lau's work was supported by The Alan Turing Institute under the Engineering and Physical Sciences Research Council grant EP/N510129/1 and the Turing-Lloyd's Register Foundation Programme for Data-Centric Engineering.

REFERENCES

- An J and Cho S (2015) *Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability*. SNU Data Mining Center, Seoul, South Korea.
- Bengio Y, Simard P and Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* **5**(2): 157–166, <https://doi.org/10.1109/72.279181>.

- Blei DM, Kucukelbir A and McAuliffe JD (2017) Variational inference: a review for statisticians. *Journal of the American Statistical Association* **112**(518): 859–877, <https://doi.org/10.1080/01621459.2017.1285773>.
- Bowman SR, Vilnis L, Vinyals O et al. (2016) Generating sequences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)* (Riezler S and Goldberg Y (eds)). Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 10–21.
- Butler LJ, Gibbons N, Middleton C and Elshafie MZEB (2016) Integrated fibre-optic sensor networks as tools for monitoring strain development in bridges during construction. In *The 19th Congress of IABSE Proceedings, Stockholm, Sweden* (Elfgren L and Jonsson J (eds)). Curran Associates, Red Hook, NY, USA, pp. 1767–1775.
- Chang PC, Flatau A and Liu SC (2003) Review paper: Health monitoring of civil infrastructure. *Structural Health Monitoring* **2**(3): 257–267, <https://doi.org/10.1177/1475921703036169>.
- Chen K, Zhou Y and Dai F (2015) A LSTM-based method for stock returns prediction: a case study of China stock market. In *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*. Santa Clara, CA, USA, pp. 2823–2824.
- Cortes C and Vapnik V (1995) Support-vector networks. *Machine Learning* **20**(3): 273–297, <https://doi.org/10.1023/A:1022627411411>.
- Cover TM and Thomas JA (1991) Entropy, relative entropy and mutual information. In *Elements of Information Theory*. Wiley, New York, NY, USA, pp. 12–49.
- Cowpertwait PSP and Metcalfe AV (2009) *Introductory Time Series with R*. Springer, New York, NY, USA.
- Eck D and Schmidhuber J (2002) *A First Look at Music Composition Using LSTM Recurrent Neural Networks*. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Manno, Switzerland, Technical Report No. IDSIA-07-02.
- Evans R, Jumper J, Kirkpatrick K et al. (2018) De novo structure prediction with deep-learning based scoring. In *The 13th Critical Assessment of Techniques for Protein Structure Prediction (Abstracts), Riviera Maya, Mexico*, pp. 11–12.
- Farrar CR and Worden K (2007) An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **365**(1851): 303–315, <https://doi.org/10.1098/rsta.2006.1928>.
- Freund Y and Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1): 119–139, <https://doi.org/10.1006/jcss.1997.1504>.
- Gelfand AE and Smith AFM (1990) Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association* **85**(410): 398–409, <https://doi.org/10.2307/2289776>.
- Gensler A, Henze J, Sick B and Raabe N (2016) Deep learning for solar power forecasting – an approach using autoencoder and LSTM neural networks. *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016), Budapest, Hungary*, pp. 002858–002865.
- Goodfellow I, Bengio Y and Courville A (2016a) Autoencoders. In *Deep Learning*. MIT Press, Cambridge, MA, USA, pp. 499–523.
- Goodfellow I, Bengio Y and Courville A (2016b) Deep feedforward networks. In *Deep Learning*. MIT Press, Cambridge, MA, USA, pp. 164–223.
- Goodfellow I, Bengio Y and Courville A (2016c) Sequence modeling: recurrent and recursive nets. In *Deep Learning*. MIT Press, Cambridge, MA, USA, pp. 367–415.
- Graves A, Mohamed A and Hinton G (2013) Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada*, pp. 6645–6649.
- Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**(1): 97–109, <https://doi.org/10.2307/2334940>.
- Heaton JB and Polson NG (2016) Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry* **33**(1): 3–12, <https://doi.org/10.1002/asmb.2209>.
- Hernandez-Garcia MR and Masri SF (2014) Application of statistical monitoring using latent-variable techniques for detection of faults in sensor networks. *Journal of Intelligent Material Systems and Structures* **25**(2): 121–136, <https://doi.org/10.1177/1045389X13479182>.
- Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural Computation* **9**(8): 1735–1780, <https://doi.org/10.1162/neco.1997.9.8.1735>.
- Jordan MI, Ghahramani Z, Jaakkola TS and Saul LK (1999) An introduction to variational methods for graphical models. *Machine Learning* **37**(2): 183–233, <https://doi.org/10.1023/A:1007665907178>.
- Kim D and Philen M (2011) Damage classification using Adaboost machine learning for structural health monitoring. *Proceedings of SPIE* **7981**: article 79812A, <https://doi.org/10.1117/12.882016>.
- Kingma DP and Ba J (2014) Adam: a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)* (Bengio Y and LeCun Y (eds)). San Diego, CA, USA.
- Kingma DP and Welling M (2013) Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)* (Bengio Y and LeCun Y (eds)). Banff, AB, Canada.
- Krizhevsky A, Sutskever I and Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems* (Pereira F, Burges CJC, Bottou L and Weinberger KQ (eds)). Curran Associates, Red Hook, NY, USA, vol. 1, pp. 1097–1105.
- Lau FDH, Adams NM, Girolami MA, Butler LJ and Elshafie MZEB (2018a) The role of statistics in data-centric engineering. *Statistics & Probability Letters* **136**: 58–62, <https://doi.org/10.1016/j.spl.2018.02.035>.
- Lau FDH, Butler LJ, Adams NM, Mohammed MZEB and Girolami MA (2018b) Real-time statistical modelling of data generated from self-sensing bridges. *Proceedings of the Institution of Civil Engineers – Smart Infrastructure and Construction* **171**(1): 3–13, <https://doi.org/10.1680/jsmic.17.00023>.
- Levine S, Pastor P, Krizhevsky A, Ibarz J and Quillen D (2018) Learning hand–eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research* **37**(4–5): 421–436, <https://doi.org/10.1177/0278364917710318>.
- Manning CD, Raghavan P and Schützh H (2008) *Introduction to Information Retrieval, online edn*. Cambridge University Press, Cambridge, UK.
- Mikhailova A, Adams NM, Hallsworth CA, Lau FDH and Jones DN (2019) Unsupervised deep learning for instrumented infrastructure: a case study. In *International Conference on Smart Infrastructure and Construction 2019 (ICSIC 2019): Driving Data-informed Decision-making* (DeJong MJ, Schooling JM and Viggiani GMB (eds)). ICE Publishing, London, UK, pp. 395–402.
- Mikhailova A, Adams NM, Hallsworth CA, Lau FDH and Jones DN (2020) *ICSIC VAE for Bridges*. GitHub, San Francisco, CA, USA. See <https://github.com/red-cheese/icsic-vae-for-bridges> (accessed 29/09/2020).
- Mikolov T, Chen K, Corrado G and Dean J (2013) Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*. Scottsdale, AZ, USA, pp. 1301–1378.
- Nair V and Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In *ICML'10: Proceedings of the 27th International Conference on Machine Learning* (Fürnkranz J and Joachims T (eds)). Omnipress, Madison, WI, USA, pp. 807–814.
- Nick W, Shelton J, Asamene K and Esterline A (2015) A study of supervised machine learning techniques for structural health monitoring. In *Proceedings of the 26th Modern AI and Cognitive*

- Science Conference* (Glass M and Kim JH (eds)). Greensboro, NC, USA, vol. 1353, pp. 133–138.
- Olah C (2015) *Understanding LSTM Networks*. GitHub, San Francisco, CA, USA. See <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (accessed 10/12/2019).
- Page ES (1954) Continuous inspection schemes. *Biometrika* **41**(1–2): 100–115, <https://doi.org/10.1093/biomet/41.1-2.100>.
- Pascanu R, Mikolov T and Bengio Y (2013) On the difficulty of training recurrent neural networks. In *ICML'13: Proceedings of the 30th International Conference on Machine Learning* (Dasgupta S and McAllester D (eds)). JMLR.org, Atlanta, GA, USA, vol. 28, pp. III-1310–III-1318.
- Pearson K (1901) On lines and planes of closest fit to systems of points in space. *Philosophical Magazine* **2**(11): 559–572, <https://doi.org/10.1080/14786440109462720>.
- Rumelhart DE, Hinton GE and Williams RJ (1986) Learning representations by back-propagating errors. *Nature* **323**(6088): 533–536, <https://doi.org/10.1038/323533a0>.
- Schölkopf B, Smola A and Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10**(5): 1299–1319, <https://doi.org/10.1162/089976698300017467>.
- Silver D, Schrittwieser J, Simonyan K et al. (2017) Mastering the game of Go without human knowledge. *Nature* **550**(7676): 354–359, <https://doi.org/10.1038/nature24270>.
- Sutskever I, Vinyals O and Le QV (2014) Sequence to sequence learning with neural networks. In *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems* *Conference (NIPS 2014)*, Montreal, QC, Canada (Ghahramani Z, Welling M, Cortes C, Lawrence ND and Weinberger KQ (eds)). MIT Press, Cambridge, MA, USA, vol. 2, pp. 3104–3112.
- Tibaduiza D, Torres-Arredondo MÁ, Vitola J, Anaya M and Pozo F (2018) A damage classification approach for structural health monitoring using machine learning. *Complexity* **2018**: article 5081283, <https://doi.org/10.1155/2018/5081283>.
- van der Maaten L and Hinton G (2008) Visualizing data using t-SNE. *Journal of Machine Learning Research* **9**: 2579–2605.
- Verma B (1997) Fast training of multilayer perceptrons. *IEEE Transactions on Neural Networks* **8**(6): 1314–1320, <https://doi.org/10.1109/72.641454>.
- Wainwright MJ and Jordan MI (2008) Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* **1**(1–2): 1–305, <https://doi.org/10.1561/22000000001>.
- Wasserman LA (2004) Expectation. In *All of Statistics*. Springer, New York, NY, USA, pp. 48–61.
- word2vec (2013) *word2vec*. Google Code, Mountain View, CA, USA. See <https://code.google.com/archive/p/word2vec/> (accessed 28/12/2018).
- Yizhou L, Nie Z and Ma H (2017) Structural damage detection with automatic feature-extraction through deep learning. *Computer-aided Civil and Infrastructure Engineering* **32**(12): 1025–1046, <https://doi.org/10.1111/mice.12313>.
- Yuan FG, Zargar SA, Chen Q and Wang S (2020) Machine learning for structural health monitoring: challenges and opportunities. *Proceedings of SPIE* **11379**: article 1137903, <https://doi.org/10.1117/12.2561610>.

How can you contribute?

To discuss this paper, please email up to 500 words to the editor at journals@ice.org.uk. Your contribution will be forwarded to the author(s) for a reply and, if considered appropriate by the editorial board, it will be published as discussion in a future issue of the journal.

Proceedings journals rely entirely on contributions from the civil engineering profession (and allied disciplines). Information about how to submit your paper online is available at www.icevirtuallibrary.com/page/authors, where you will also find detailed author guidelines.