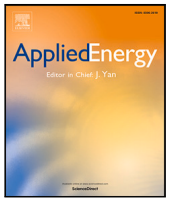
Applied Energy 304 (2021) 117695

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting ☆

Sumeyra Demir [a,*], Krystof Mincev [b], Koen Kok [a], Nikolaos G. Paterakis [a]

[a] Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands
[b] School of Computer Science, University of St Andrews, United Kingdom

ARTICLE INFO

Keywords:
Adversarial network
Augmentation
Autoencoder
Electricity price forecasting
Multivariate time series
Regression

ABSTRACT

A model's expected generalisation error is inversely proportional to its training set size. This relationship can pose a problem when modelling multivariate time series, because structural breaks, low sampling rates, and high data gathering costs can severely restrict training set sizes, increasing a model's expected generalisation error by spurring regression model overfitting. Artificially expanding the training set size, using data augmentation methods, can, however, counteract the restrictions imposed by small sample sizes: increasing a model's robustness to overfitting and boosting out-of-sample prediction accuracies. While existing time series augmentation methods have predominantly utilised feature space transformations to artificially expand training set sizes and boost prediction accuracies, we propose using autoencoders (AEs), variational autoencoders (VAEs) and Wasserstein generative adversarial networks with a gradient penalty (WGAN-GPs) for time series augmentation. To evaluate our proposed augmentors, as a case study we forecast Belgian and Dutch day-ahead electricity market prices using both autoregressive models and artificial neural networks. Overall, our results demonstrate that AEs, VAEs, and WGAN-GPs can significantly boost regression accuracies; on average decreasing benchmark model mean absolute errors by 2.23%, 2.73% and 2.97% respectively. Moreover, our results demonstrate that combining AE, VAE, and WGAN-GP generated time series can further boost regression accuracies; on average decreasing benchmark errors by 3.44%. As our proposed augmentors outperform existing augmentation methods, we strongly believe that both practitioners and researchers aiming to generate time series or reduce time series regression errors will find utility in our study.

1. Introduction

Sample sizes play a critical part in determining the optimal complexity of hypothesis sets and, by extension, the minimum attainable generalisation error. In multivariate time series analysis, the progression of time can often shift the target distribution. This shift increases the proportion of noise in historic data, making its inclusion in the learning process potentially detrimental. When data is scarce it is crucial to explore techniques capable of augmenting the sample. While numerous studies have demonstrated that augmentation can improve classification and regression accuracies, to date none have explored augmenting multivariate time series (i.e. time series with more than a single time-dependent variable) with exogenous variables despite the fact that the latter is commonly used by practitioners in regression analysis. Our research aims to fill this gap by identifying methods capable of successfully augmenting multivariate time series for regression analysis.

1.1. Sample sizes and model performance

Statistical learning and machine learning aim to identify reoccurring patterns in data by minimising a model's generalisation error. Because the joint probability distribution of $\mathbf{x}$, a feature vector, and y, a scalar target output, $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, is unknown, in practice, a direct minimisation of the generalisation error is impossible. Instead, in most statistical learning processes, an empirical error, and a regularisation term, such as the L2 norm, are minimised jointly.

Both bias plus variance decomposition and the Vapnik-Chervonenkis (VC) theory justify the above statistical learning process. The bias plus variance decomposition specifies a framework for better understanding and analysing the generalisation error. This framework exposes an almost unavoidable trade-off, termed the bias–variance trade-off, between a model's bias in parameter estimation and variance of parameter estimates. The bias–variance trade-off can be used to

☆ This work was funded by the Scholt Energy, 5555XA Valkenswaard, The Netherlands.
* Corresponding author.
E-mail address: s.demir@tue.nl (S. Demir).

https://doi.org/10.1016/j.apenergy.2021.117695
Received 19 April 2021; Received in revised form 16 August 2021; Accepted 22 August 2021
Available online 9 September 2021

show that the availability of training data determines the minimum attainable expected generalisation error. The VC theory expresses a generalised upper bound, termed the VC inequality, to the generalisation error. Similarly to the bias–variance trade-off, the VC inequality can be used to directly prove that a model's VC-dimension - i.e. a model's complexity captured by its number of effective parameters - is proportional to the number of training examples needed to attain a certain level of expected modelling performance - i.e. a certain generalisation bound [1]. Together the bias plus variance decomposition and VC inequality motivate studies which seek to reduce the expected generalisation error by expanding training set sizes.

Grounding theory in practice, note that studies, such as [2] for image classification, demonstrate that the relationship between model performance and data quantity is logarithmic. While, to the best of our knowledge, this relationship has yet to be extensively studied for regression models, back of the envelope calculations show that for every new training $(\mathbf{x}, y)$ example the expected generalisation error of, for instance, a linear regression modelling a linear target function decreases by $(eJ+e)/(N^2+N)$, where e is the best approximation error, J is the number of features, and N is the number of samples [1]. The above examples further underscore the importance of dataset sizes in minimising the expected generalisation loss, as well as the potential performance improvements that an expansion of the training set can yield.

1.2. Data augmentation

Linking modelling performance to data augmentation, in instances where it is impractical or even impossible to expand real training dataset sizes, due to costs, data scarcity, or time series structural breaks, it is worthwhile to consider whether augmented data could be used in place of additional real training data to boost modelling performance. While, to the best of our knowledge, a generalised proof establishing a direct relationship between the expected generalisation error and number of augmented data points has yet to be theoretically formulated, numerous studies, such as [3–10], empirically demonstrate that using augmented data can boost both classification and regression accuracies comparable to the addition of real data. Below, we introduce the augmentation methods explored in these studies.

Commencing with augmentation methods that have been utilised to boost classification accuracies, feature space augmentors, exploiting simple transformations such as symmetry, position, or style, have been observed to successfully generate data for both image and time series classification problems [3,4]. For instance, [4] observed a 5.1% accuracy increase in Parkinson's disease motor state classification, with a convolutional neural network, after augmenting wearable sensory data using random rotations. Despite this success, because of temporal relationships in time series, it must be stressed that applying feature space augmentations to time series can be risky. This is because, for augmentation methods to be useful they have to generate meaningful data originating from the underlying target distribution. Whilst we can observably conclude that images generated using feature space transformations continue to be meaningful, we cannot readily conclude this with time series.

Beyond feature space transformations, researchers have successfully applied augmentors utilising generative autoencoders to classification problems [5,6]. Autoencoders [11] are artificial neural networks that learn to encode, and subsequently decode model inputs, by performing unsupervised representation learning. Because encoding transforms input features from a feature space X to a latent space Z, autoencoders can generate data by applying perturbations in Z, or by performing Bayesian inference and sampling from a latent distribution. These methods, to the best of our knowledge, have never been used to augment multivariate time series; however, both have a potential theoretical advantage over feature space transformations that make them worth examining.

Generative adversarial networks (GANs) [12] have also been utilised to boost classification accuracies [7,13]. GANs, comprising of a generator G and a discriminator D, differ from generative autoencoders in their training approach. While the latter is trained by minimising the divergence between model inputs and outputs directly, GANs use D to indirectly train G. The process is comparable to a game between a forger and a detective, and allows G to learn to generate data from the distribution of the real data. Although, to the best of our knowledge GANs have never been used to augment multivariate time series, studies such as [7,8] demonstrate their ability to generate meaningful data and boost forecasting performances. Tran et al. [8] observed deep model classification accuracy improvements of 6%, across MNIST, CIFAR-10 and CIFAR-100 datasets, surpassing compound feature space transformation performances.

Finally describing augmentation methods for time series regression, to the best of our knowledge two studies, [9,10], have thus far attempted to boost regression accuracies using augmentation methods. In particular, [9] proposed generating univariate time series by replacing the error of deseasonalised and detrended time series with bootstrapped errors, while [10] proposed generating the same series by sampling Markov Chain Monte Carlo parameters and forecasting time series paths using those parameters. Across the M3-competition dataset, [9] demonstrated that the bootstrapping augmentation method, combined with bagging, significantly improves the prediction accuracies of exponential smoothing models. Across the same dataset, [10] observed a 5.7% reduction in symmetrical mean absolute percentage errors of long short-term memory recurrent neural networks. Whether these augmentation methods could be adapted to readily and successfully generate multivariate time series with exogenous inputs is uncertain.

Elaborating, hypothetically for example, to generate multivariate time series using [9], one would have to augment multiple univariate time series; replacing the errors of multiple deseasonalised and detrended time series with correlated bootstrapped errors. To achieve this without introducing an excessive amount of noise, cross correlations between the time series would probably have to be modelled. Moreover, the time series would have to be of the same length, with the same sampling periodicity. As the above, in and of itself, constitutes a new data augmentation method, as it significantly extends the use case presented in [9], we consider any attempts to implement it beyond the scope of our examination.

1.3. Electricity price forecasting

As a case study, we choose to forecast both Belgian and Dutch day-ahead electricity market (DAM) prices. We forecast DAM prices because in recent years regulatory changes and an increased emphasis on renewable energy production have increased the generalisation error of DAM forecasts [14]; establishing new price drivers, introducing structural breaks in the time series, and spurring the need for data augmentation.

Several papers, including [14–26], have studied DAM price forecasting. In [19,20], lagged DAM prices and exogenous variables such as load forecast were found to be important features for forecasting DAM prices. In [17], lagged DAM prices were used together with neighbouring market prices. Using Belgian and French DAM prices, the forecasting performance of neural networks (NNs) was improved. Similarly, data from multiple markets was stacked together to avoid overfitting in [22]. The forecasting performance of NNs was also improved.

In [15,18], the forecast accuracies of deep, ensemble, and statistical models were evaluated. Machine learning models were found to outperform statistical models in both studies. In [18], NNs were found to outperform long short-term memory networks (LSTM) and gated recurrent units (GRUs) in forecasting Belgian DAM prices. Other studies, for instance [14,21], have also compared the performances of deep and shallow machine learning models. In [14], deep models were found to outperform shallow models in forecasting Belgian DAM prices.

In [21], GRUs were found to outperform all other neural network structures such as NNs and LSTM and statistical methods such as seasonal autoregressive integrated moving average (SARIMA) and Markov models in forecasting Turkish DAM prices. Contrasting performances of artificial networks, such as NNs, GRUs and LSTM, across [18,21] can be explained by model complexities, data characteristics and data sizes.

In [24–26], hybrid models were shown to outperform individual benchmark models. In [24], the hybrid model was harmonised using a wavelet transformation, an autoregressive moving average model, a kernel extreme learning machine (KELM), and self-adaptive particle swarm optimisation (SAPSO). In [25], the hybrid model was constructed with a variational mode decomposition, SAPSO, SARIMA and deep belief networks. Model accuracies were assessed across three different DAMs in [25]. Exogenous variables, however, were not included in this study. In [26], the hybrid model was structured using a local forecasting paradigm, a general regression neural network, coordinate delay and an harmony search algorithm. Unlike in [25], exogenous variables were included in this study. Model accuracies, however, were assessed only on one DAM in [26]. In [23], applications of spike forecasting were further explored. Using the Borderline-SMOTE method to balance the number of samples in different target classes, [23] increased the number of spike samples in training data; yielding DAM forecasting accuracy improvements. For a more detailed review of the DAM forecasting literature/best practices we recommend interested readers read [15,16].

To the best of our knowledge, we are the first to explore using data augmentation with generative models as a method for boosting DAM forecasting accuracies. In our study, we include exogenous variables and neighbouring market prices, and evaluate our augmentation method on two different DAMs using deep, ensemble and statistical models.

1.4. Motivation and contributions

The literature review above exposes how researchers have thus far concentrated more on developing augmentation methods for classification than regression. Particularly, augmentation methods for multivariate time series with exogenous variables have to date not been researched. Motivated by a desire to fill this void, our paper concentrates on developing universally applicable augmentation methods for both univariate and multivariate time series regression analysis. Because of their prevailing use in classification studies as well as their ease of implementation, we evaluate three feature space augmentors: jittering, scaling, and magnitude-warping. Similarly, because of their significant achievements with classifiers, their capacity to model the underlying distribution of data, and their ability to generate data without any domain knowledge, we choose to develop and evaluate tailored model-based augmentors: autoencoders (AEs), variational autoencoders (VAEs), and Wasserstein GANs with a gradient penalty (WGAN-GP).

To fully explore the effectiveness of the above-mentioned augmentation methods, we examine their performance impact on autoregressive models with exogenous inputs (ARX) and two artificial neural networks (ANN). ARXs are chosen both to demonstrate the methods impacts on linear model performances, and because they have been shown to achieve reasonable DAM benchmark forecast accuracies. ANNs are chosen because they have been shown to outperform other state-of-the-art statistical and machine learning models in DAM forecasting [18].

Summarising the principle contributions of this paper, to the best of our knowledge, we are the first to: (1) explore the augmentation of multivariate time series with exogenous variables, (2) utilise feature space transformations, AEs, VAEs and WGAN-GPs for regression augmentation, and (3) apply augmentation methods to the forecasting of DAM prices. Outlining the structure of this paper, in Section 2 the DAM, as well as the theories and models underpinning our augmentation methods are described. In Section 3 our case study's methodology is outlined. In Section 4 the results are analysed. Finally, in Section 5 our study is concluded and avenues of potential future research are discussed.

2. Preliminaries

2.1. Day-ahead electricity markets

Energy producers and suppliers must together meet the energy needs of everyday consumers and business. In doing so, they must forecast hourly or quarter-hourly energy demand and plan a path for satisfying consumers' energy needs. They do this, while jointly trying to maximise profits and minimise price and volume risks. Numerous opportunities for trading electricity and meeting the demand for energy exist: from long-term futures with maturities ranging from weeks to years, to short-term contracts with maturities ranging from one hour to one day.

The day-ahead electricity market (DAM) offers 24 hourly energy contracts for next day delivery. Uniquely, DAM clearing prices (in €/MWh) are computed using a matching mechanism that aggregates all asks and bids submitted before noon. For the DAM's 24 hourly contracts, the matching engine sets prices where demand and supply intersect. Trade settlement occurs once prices are set, with hourly energy delivery commencing at 00:00 of the following day.

2.2. Autoregressive models with exogenous inputs

ARXs are linear time series models capable of capturing both autoregressive and exogenous relationships driving stochastic processes. Formally, a single output ARX(n, m) model is defined according to Eq. (1).

$$y_t = a_0 + \sum_{i=1}^{n} a_i y_{t-i} + \sum_{i=1}^{m} b_i u_i + \nu_t, \tag{1}$$

where y_{t-i} is the model output at time $t-i$, u_i is ith exogenous input, a_i and b_i are the ith autoregressive and exogenous parameters respectively, and ν_t is the residual noise. To train an ARX, the least absolute shrinkage and selection operator (LASSO) is employed; optimising $\mathbf{a}$ and $\mathbf{b}$ according to Eq. (2).

$$\arg\min_{\mathbf{a},\mathbf{b}} \mathcal{L}(\mathbf{a},\mathbf{b};\mathbf{y},\mathbf{u}) = \mathbb{E}_{y_t \sim p_{\mathbf{y}}(y_t)}\left[\left(y_t - (a_0 + \sum_{i=1}^{n} a_i y_{t-i} + \sum_{i=1}^{m} b_i u_i)\right)^2\right] + \tau\left(\|\mathbf{a}\|_1 + \|\mathbf{b}\|_1\right), \tag{2}$$

where τ is a regularisation parameter.

2.3. Artificial neural networks

ANNs are non-linear machine learning models consisting of input, output, and intermediate/hidden layers. Properly configured, ANNs can be used to theoretically approximate any continuous target function. Below, to introduce the fundamental building blocks of ANNs, two commonly used hidden layers, the fully connected layer (FCL) and the locally connected convolutional layer (CONV), are described.

ANN layers comprise of groups of interconnected nodes. Each node weights the outputs of previous layers according to $f(b + \mathbf{w}^\intercal \mathbf{z})$, where b is a bias term, $\mathbf{w}$ is a vector of trainable weights, $\mathbf{z}$ is a vector of previous layer outputs, and $f(.)$ is a differentiable activation function, such as a rectified linear unit (ReLU), capable of adding a non-linearity to a node's output. In an FCL, nodes in layer l are connected to all other nodes from layer $l-1$; computing the dot product of their outputs according to $f(b+\mathbf{w}^\intercal \mathbf{z}_{l-1})$. Each FCL yields $\mathrm{P}_{l-1} \times \mathrm{P}_l$ optimisable parameters, where P_l and P_{l-1} are the number of nodes in layers l and $l-1$ respectively.

In a CONV, multiple cascaded convolutions are applied across $\mathbf{z}_{l-1}$. Formally, kernels of size $(\mathrm{W} \times \mathrm{H})$, computing $f(b + \sum_k^{\mathrm{D}+k} \sum_j^{\mathrm{W}+j} \sum_i^{\mathrm{H}+i} w_{kji} z_{kji})$, where D is the depth of $\mathbf{z}_{l-1}$, are passed across $\mathbf{z}_{l-1}$. Unlike FCLs, CONVs implement parameter sharing, only optimising $\mathrm{C} \times \mathrm{W} \times \mathrm{H} \times \mathrm{D}$ parameters, where C is the number of CONV kernels.

2.4. Kullback–Leibler divergence

The Kullback–Leibler divergence (D_{KL}) is an asymmetric distance measure, measuring the divergence between two distributions. The forward D_{KL} is calculated as: $D_{KL}\left(p_X(\mathbf{x}) \parallel p_{\tilde{X}_\omega}(\tilde{\mathbf{x}})\right) = \sum_j p_X(x_j) \ln \frac{p_X(x_j)}{p_{\tilde{X}_\omega}(\tilde{x}_j)}$, where p_X and $p_{\tilde{X}_\omega}$ are real and generated data distributions respectively. The reverse D_{KL} is calculated as $D_{KL}\left(p_{\tilde{X}_\omega}(\tilde{\mathbf{x}}) \parallel p_X(\mathbf{x})\right)$. As we later demonstrate, the weights, ω, of autoencoders and GANs can be optimised by minimising the forward D_{KL} and reverse D_{KL} respectively. The impacts of minimising these metrics on data generation are highlighted below.

By reformulating the forward D_{KL} from Eq. (3), it can be shown that minimising the forward D_{KL} is equivalent to performing maximum likelihood estimation [27].

$$\begin{aligned}\arg\min_{\omega} \; D_{KL}(p_X(\mathbf{x}) \parallel p_{\tilde{X}_\omega}(\tilde{\mathbf{x}})) &= \arg\min_{\omega} \; \mathbb{E}_{\mathbf{x}\sim p_X}[-\log p_{\tilde{X}_\omega}(\tilde{\mathbf{x}}) + \log p_X(\mathbf{x})] \\ &= \arg\min_{\omega} \; \mathbb{E}_{\mathbf{x}\sim p_X}[-\log p_{\tilde{X}_\omega}(\tilde{\mathbf{x}})] - \mathcal{H}(p_X(\mathbf{x})) \\ &= \arg\max_{\omega} \; \mathbb{E}_{\mathbf{x}\sim p_X}[\log p_{\tilde{X}_\omega}(\tilde{\mathbf{x}})],\end{aligned} \tag{3}$$

where $\mathcal{H}(p_X(\mathbf{x}))$ is the entropy of real data, which is known and constant. The consequence of minimising the forward D_{KL} is a 'mean seeking' approximation, which spurs a model to cover the support of p_X, assigning a high probability mass where p_X is high, while centring $p_{\tilde{X}_\omega}$ around the mean of p_X.

In contrast, minimising the reverse D_{KL} encourages 'mode seeking' approximations of p_X by assigning a high probability mass to the mode of the real distribution [27]. Eq. (4) formally expresses this.

$$\arg\min_{\omega} \; D_{KL}(p_{\tilde{X}_\omega}(\tilde{\mathbf{x}}) \parallel p_X(\mathbf{x})) = \arg\min_{\omega} \; \mathbb{E}_{\tilde{\mathbf{x}}\sim p_{\tilde{X}_\omega}}[-\log p_X(\mathbf{x})] - \mathcal{H}(p_{\tilde{X}_\omega}(\tilde{\mathbf{x}})), \tag{4}$$

where $\mathcal{H}(p_{\tilde{X}_\omega}(\tilde{\mathbf{x}}))$ is the entropy of generated data. Eq. (4) highlights that when minimising the reverse D_{KL}, a generative model learns to assign low probabilities to $p_{\tilde{X}_\omega}$ where p_X is low.

2.5. Autoencoders

Autoencoders are lossy networks that learn to encode and subsequently decode model inputs. Generally, autoencoders are implemented in a $J/K/J$ or bottleneck architecture, where J is the dimension of input and output layers, K is the dimension of hidden layers/latent space, and $J > K$. Under such a structure the encoder extracts the most salient features from the input vector, $\mathbf{x}$; transforming inputs from the feature space $X \in \mathbb{R}^J$ to an unstructured and generally more compact latent space $Z \in \mathbb{R}^K$. To find the optimal encoder and decoder parameters, ϕ and θ respectively, autoencoders are trained by minimising a reconstruction loss according to Eq. (5) [28].

$$\hat{\theta}, \hat{\phi} = \arg\min_{\theta,\phi} \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})} \ell(\mathbf{x}, D_\theta(E_\phi(\mathbf{x}))), \tag{5}$$

where $E_\phi(.)$, $D_\theta(.)$, and $\ell(\mathbf{x}, .)$ are encoding, decoding, and distance functions respectively, and $\mathbf{x}$ is a vector sampled from the training set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. As [29] argues and as we further explain in Section 3.5.2, it is the dimensionality reduction, which occurs during encoding, that makes latent space transformations more likely to generate data from the underlying distribution than feature space transformations.

2.6. Variational autoencoders

VAEs [30] are a special class of autoencoder, modelling observed and latent variable probability distributions while learning structured latent space representations $\mathbf{z}$ of real observations $\mathbf{x}$. To learn these representations, VAEs use variational inference to approximate the true posterior, $p(\mathbf{z}|\mathbf{x})$, with a variational posterior $q_\lambda(\mathbf{z}|\mathbf{x})$, where λ is a collection of variational parameters. From a neural net perspective, VAEs perform variational expectation maximisation to optimise the parameters ϕ of an inference network $q_\phi(\mathbf{z}|\mathbf{x})$ that outputs λ. VAE encoders are called inference networks because they parametrise the inference of a true posterior. VAE decoders, performing the parametrised probabilistic decoding $p_\theta(\mathbf{x}|\mathbf{z})$, are called generation networks.

Deriving the VAE objective function [30], we note that directly approximating the true distribution of $\mathbf{x}$ as: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) dz = \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) dz$ is intractable because it requires evaluating all configurations of $\mathbf{z}$. However, by reformulating the above using Bayes' rule, a tractable objective function representing the lower bound of $\mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})} \log p_\theta(\mathbf{x})$ can be derived. Specifically, to optimise the parameters of the inference and generation networks, ϕ and θ respectively, the value function $\mathcal{V}$ is maximised according to Eq. (6).

$$\arg\max_{\phi,\theta} \mathcal{V}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})]. \tag{6}$$

Since $\mathcal{V}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}\left[\log p_\theta(\mathbf{x}) - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})]\right] \leq \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\log p_\theta(\mathbf{x})]$ [31], maximising $\mathcal{V}(\theta, \phi; \mathbf{x})$ is equivalent to minimising the forward $D_{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})]$. This is because, as the forward D_{KL} approaches 0, $\mathcal{V}$ approaches $\mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})} \log p_\theta(\mathbf{x})$. Linking $\mathcal{V}$ to autoencoders, by taking the negative of $\mathcal{V}$ we can identify similarities and differences between VAE and autoencoder objective functions. Formally, the negative of $\mathcal{V}$ is equal to Eq. (7).

$$\arg\min_{\theta,\phi} \mathcal{L}(\theta, \phi; \mathbf{x}) = -\left[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})]\right]. \tag{7}$$

In Eq. (7), the first term on the right measures the expected negative log-likelihood of $\mathbf{x}$. This term is equivalent to an autoencoder's reconstruction loss. The second term measures the information loss from variational approximation: it is a regulariser term controlling the structuredness of $\mathbf{z}$ and distinguishing VAEs from autoencoders.

2.7. Wasserstein generative adversarial networks

GANs are adversarial networks that learn to generate samples from the underlying probability distribution of data without explicitly modelling said distribution. They do this by pitting two neural networks G_ϕ, a generator, and D_θ, a discriminator, against each other in a minmax game. To train D_θ, [32] proposed maximising the negative cross-entropy of a binary classifier, outputting variable y, and separating real ($\mathbf{x} \sim p_X(\mathbf{x}|y = 1)$) and generated ($\tilde{\mathbf{x}} \sim p_{G_\phi}(\tilde{\mathbf{x}}|y = 0)$) data. Formally, θ and ϕ are optimised according to Eqs. (8) and (9).

$$\arg\max_{\theta} \mathcal{V}_D(\phi, \theta; \mathbf{x}, \mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[\log D_\theta(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log(1 - D_\theta(G_\phi(\mathbf{z})))], \tag{8}$$

$$\arg\max_{\phi} \mathcal{V}_G(\phi, \theta; \mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log D_\theta(G_\phi(\mathbf{z}))]. \tag{9}$$

Equating GANs to VAEs, by treating the binary targets y of D_θ as observed variables, and the inputs $\acute{X} = \{\mathbf{x}, \tilde{\mathbf{x}}\}$ of D_θ as latent variables, [31] established a formal connection between a GAN's objective functions and variational expectation maximisation. Similarly to a VAE's inference network, [31] highlighted that G_ϕ can be thought of as performing posterior inference; with the variational posterior $p_\phi(\acute{X}|y)$ approximating the posterior $q^r(\acute{X}|y) \propto q_{\theta_0}(1 - y|\acute{X})\left[\mathbb{E}_{y\sim p_Y(y)} p_{\phi_0}(\acute{X}|y)\right]$, where θ_0 and ϕ_0 are D and G weights from the previous training iteration respectively, and, $q_{\theta_0}(y|\acute{X})$ is equal to $D_{\theta_0}(\acute{X})$. Eq. (10) emphasises this connection.

$$\begin{aligned}\nabla_\phi \mathbb{E}_{\acute{X}\sim p_\phi(\acute{X}|y)p(y)} \log q_{\theta_0}(1 - y|\acute{X}) = \nabla_\phi JSD\left(p_{G_\phi}(\tilde{\mathbf{x}}|y = 0) \| p_X(\mathbf{x}|y = 1)\right) \\ - \nabla_\phi \mathbb{E}_{y\sim p(y)}\left[D_{KL} p_\phi(\acute{X}|y) \| q^r(\acute{X}|y)\right].\end{aligned} \tag{10}$$

In Eq. (10), the JSD term, whose impact becomes negligible once the reverse D_{KL} is sufficiently minimised, is upper bounded by the reverse D_{KL} term [31]. The optimisation of G_ϕ according to Eq. (9) is thus equivalent to the minimisation of the reverse D_{KL}.

There are two critical drawbacks with optimising ϕ according to Eq. (9): unstable gradient updates and mode collapse [33]. For a fixed G_ϕ, as D_θ improves and approaches optimality ($\theta \rightarrow \theta^*$) the gradient norm of the objective function - $\|\nabla_\phi \log D_\theta\big(G_\phi(\mathbf{z})\big)\|$ - rapidly explodes. An instability of gradient updates spurs network saturation/instability and increases the complexity of hyperparameter tuning. Further, as minimising the reverse D_{KL} encourages 'mode seeking' approximations of p_X, optimising ϕ according to Eq. (9) can provoke mode collapse, which occurs when G_ϕ, with varying input vectors, begins generating data centred at a single mode of a complex multimodel dataset.

Wasserstein GANs (WGANs) [34] are a variant of traditional GANs that offer an alternative training approach to [32] centred around the optimisation of a more stable objective function: the Wasserstein distance. The Wasserstein distance, a symmetric measure of distribution similarity, is mathematically defined in its primal form as the infimum (greatest lower bound) energy cost of transforming $p_{\tilde{X}}$ into p_X. Unlike the JSD and reverse D_{KL}, the Wasserstein distance is continuous everywhere and yields smooth and meaningful gradients even when the manifolds of p_X and $p_{\tilde{X}}$ have disjoint supports. Moreover, it limits mode collapse, by dissuading 'mode seeking' approximations of p_X. Because computing the infimum is intractable, in practice a dual form of the Wasserstein distance, expressed in Eq. (11), is computed.

$$W(p_X, p_{\tilde{X}}) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[f(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}\sim p_{\tilde{X}}(\tilde{\mathbf{x}})}[f(\tilde{\mathbf{x}})], \tag{11}$$

where sup is the supremum (least upper bound), $f : X \rightarrow \mathbb{R}$ is a 1-Lipschitz function with $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$. Arjovsky et al. [34] derived the above objective function using the Kantorovich–Rubinstein duality; turning a minimisation problem with an infimum into a maximisation problem with a supremum. Formalising the WGAN training procedure, using f in place of D, ϕ and θ are optimised according to Eq. (12).

$$\arg\min_{\phi}\max_{\theta} W(\phi, \theta; \mathbf{x}, \mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[f_\theta(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_Z(\mathbf{z})}[f_\theta(G_\phi(\mathbf{z}))]. \tag{12}$$

Note, f_θ must satisfy the Lipschitz constraint, because only Lipschitz continuous functions produce feasible/optimal solutions for both the dual/primal forms of the Wasserstein loss respectively. To satisfy the Lipschitz constraint, [34] proposed using gradient clipping when optimising θ. Alternatively, a gradient penalty, underlined in Eq. (13), may be added to the objective function [35].

$$\begin{aligned}\arg\min_{\theta} W(\theta; \mathbf{x}, \tilde{\mathbf{x}}) = {} & \mathbb{E}_{\tilde{\mathbf{x}}\sim p_{\tilde{X}}(\tilde{\mathbf{x}})}[f_\theta(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}[(f_\theta(\mathbf{x}))] \\ & + \underbrace{\lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{X}}(\hat{\mathbf{x}})}[(\|\nabla_{\hat{\mathbf{x}}} f_\theta(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{gradient penalty}},\end{aligned} \tag{13}$$

where λ is the gradient penalty coefficient and $\hat{\mathbf{x}}$ is an interpolated randomly sampled vector: $\alpha\tilde{\mathbf{x}} + (1-\alpha)\mathbf{x}$ with $0 \leq \alpha \leq 1$. The gradient penalty ensures that the gradient norm of the objective function does not exceed 1, without requiring extensive hyperparameter tuning.

3. Methods

3.1. Data

As a case study, to evaluate the robustness of our augmentation methods, we forecast Belgian and Dutch DAM prices. To forecast these, we use lagged DAM prices [14,18], day-ahead grid load forecasts [19], day-ahead available generation forecasts [18] and meteorological features, namely actual and day-ahead forecasts of temperature, wind speed, solar irradiance, and precipitation. From [36,37] we gather our dataset, using data from 01/01/2016 to 31/12/2016 for model training and data augmentation (training), data from 01/01/2017 to 31/12/2017 for hyperparameter tuning (validation), and data from 01/01/2018 to 31/12/2018 for augmentation evaluation (test). DAM prices for each split are shown in Fig. 1. An example of weekly DAM prices is also given in Fig. 2. Summary statistics of DAM prices are presented in Table 1. Training, validation, and test means (**Mean**) and standard deviations (**SD**) are presented for every hour (**h**) denominated DAM contract and the average (**avg**) of contracts.

Analysing Table 1, we observe that the dataset of DAM prices displays varying statistical characteristics across countries (Belgium/The Netherlands), contracts (hour), and time (training/validation/test). On average, the Mean and SD of DAM prices are higher in Belgium than The Netherlands. Similarly, the Mean and SD of DAM prices are frequently higher for peak hours (between 08h–20h) than for off-peak hours. Finally, the Mean and SD of DAM prices are observed to generally increase across the training, validation, and test sets, i.e. across time.

While the above are general observations with exceptions – the Belgian 14h contract for instance displays a decreasing Mean and SD across time – overall we strongly believe that modelling such statistically diverse data will enable us to thoroughly evaluate the robustness of our augmentation methods.

3.2. Data processing

To facilitate data augmentation using generator networks, we use Min-Max normalisation, which binds data in the range $[0, 1]$. Such data can be readily generated by applying a sigmoid activation function across final layer network outputs as in [38].

3.3. Feature selection

Many researchers have modelled the DAM as a multiple output regression problem. We choose to model the DAM as a single output regression problem, forecasting the prices of every DAM hourly contract independently. This allows us to: (1) tailor our modelling approach to the varying statistical characteristics of every contract, (2) capture any price drivers which may be unique to a specific contract, and (3) avoid using too many features in the forecasting of any single contract. Feature selection is used to shrink an initial feature pool of 2160 features to a reduced feature space of at most 35 features. Our initial feature pool, for any DAM contract, consists of the features mentioned in Section 3.1 spanning a seven-day-lagged period. Specifically, the total of 2160 features comprises of 1440[1] meteorological features, 384[2] generation and load features, and 336[3] price features. For example, the Dutch feature pool contains seven-day-lagged 00h–23h: Dutch prices, Belgian prices, Dutch generation and load forecasts, and Dutch meteorological features. Note that we use Belgian and Dutch prices together for both the Dutch and Belgian feature pools to increase the explanatory variability coming from neighbouring countries.

To understand why we constrain our modelling input size to at most 35 features refer back to our discussion of the generalisation bound in Section 1.1. The VC-bound[4] for a regression can be expressed according to Eq. (14) [39]:

$$\mathbb{P}\left(E_{\text{out}}(h) \leq \frac{E_{\text{in}}(h)}{\left(1 - c\sqrt{\gamma}\right)_+}\right) = 1 - \delta, \tag{14}$$

[1] 360 temperature features, 360 wind speed features, 360 solar irradiance features and 360 precipitation features. Each of the above consists of 7 * 24 (seven-day) lagged actuals + 7 * 24 (seven-day) lagged forecasts + 24 next day forecasts.

[2] 192 generation features and 192 load features; each consisting of 7 * 24 (seven-day) lagged forecasts + 24 next day forecasts.

[3] 168 Belgian and 168 Dutch DAM prices; each consisting of 7 * 24 (seven-day) lagged actuals.

[4] VC-theory assumes that sample points are independently and identically distributed (i.i.d.). While DAM prices are not uniformly i.i.d., in determining the maximum number of features we make this assumption to obtain reasonable bounds.

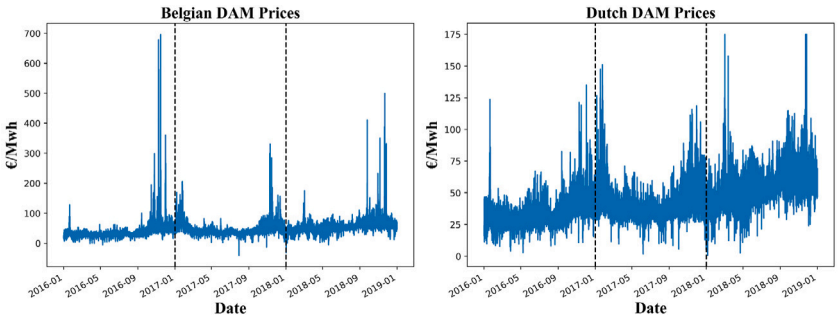


Fig. 1. Historic DAM prices, covering the training, validation and test datasets. The dashed black lines represent the train/validation/test splits.

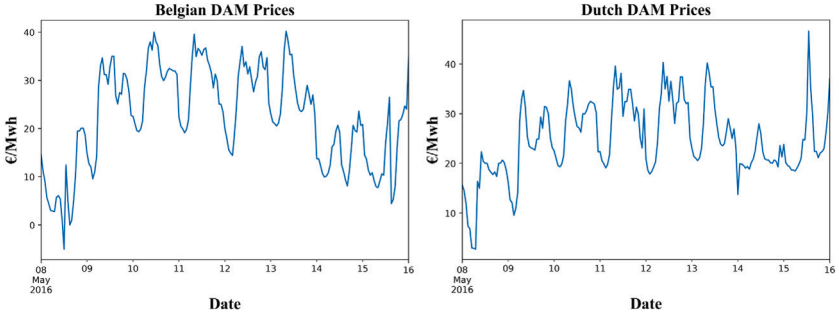


Fig. 2. Single week of DAM prices from 08/05/2016 to 16/05/2016.

Table 1
Summary statistics of Belgian and Dutch DAM prices (in €/MWh).

h	Belgian DAM prices						Dutch DAM prices					
	Training		Validation		Test		Training		Validation		Test	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
00	30.74	11.49	38.78	12.94	50.34	15.74	26.40	6.37	33.98	6.37	47.07	11.27
01	27.15	9.83	35.45	11.38	44.61	14.15	24.67	5.52	32.30	5.90	43.23	9.99
02	25.61	9.12	32.81	10.58	42.26	13.86	23.87	5.44	30.65	5.38	41.28	9.39
03	23.78	8.94	30.75	9.88	39.72	13.30	22.99	5.65	29.61	5.61	39.56	9.55
04	23.31	8.92	30.15	9.64	39.57	13.30	22.88	5.95	29.07	5.63	39.60	9.08
05	25.39	9.87	32.32	10.82	41.79	13.53	23.87	5.91	30.22	5.71	40.98	9.22
06	31.61	14.03	39.02	14.85	49.64	16.68	28.08	6.78	34.81	7.12	48.03	11.22
07	39.71	22.84	48.05	24.42	58.71	23.47	34.20	9.89	40.92	10.67	55.80	14.45
08	42.59	24.73	51.78	27.04	62.56	27.59	37.02	11.72	44.31	12.44	59.29	14.73
09	43.48	23.51	52.58	24.66	64.59	25.85	38.34	12.13	46.08	14.21	62.33	15.32
10	43.05	23.71	51.58	23.70	62.45	23.44	37.67	11.95	45.23	14.38	59.99	15.05
11	42.04	21.56	50.79	22.43	61.11	23.13	37.28	11.77	43.92	13.67	57.31	13.90
12	40.10	20.23	47.75	21.26	58.49	21.12	35.32	10.80	41.24	12.32	54.49	13.13
13	38.26	19.79	46.11	21.13	55.87	20.88	34.47	11.23	40.39	12.58	52.72	13.15
14	36.72	22.58	43.98	21.25	53.38	19.98	32.94	11.02	39.06	12.50	51.09	13.28
15	35.29	18.46	43.30	21.66	52.14	19.68	32.65	11.37	38.77	12.84	50.44	13.40
16	35.95	19.20	44.44	22.20	53.31	19.59	34.03	13.06	40.83	14.99	52.44	13.60
17	40.72	22.80	50.59	27.35	60.19	25.78	38.66	17.15	47.48	20.10	59.78	17.75
18	51.59	64.19	57.61	36.54	68.64	43.05	39.76	13.68	48.41	19.21	62.63	17.83
19	47.89	31.08	56.00	24.36	68.79	28.97	39.02	10.37	46.66	12.98	63.40	17.93
20	42.92	20.07	51.05	17.91	64.79	26.86	36.05	8.19	43.56	9.05	59.95	14.71
21	37.85	13.54	46.25	13.69	58.41	17.18	32.86	6.65	40.50	6.07	55.39	12.39
22	37.75	12.25	46.27	13.40	59.33	17.56	31.71	6.11	39.32	5.57	54.05	11.42
23	35.18	11.82	42.48	13.33	55.73	17.28	29.10	5.92	35.95	5.34	49.83	9.96
Avg	36.78	23.69	44.59	21.62	55.27	23.54	32.29	11.35	39.31	12.76	52.53	15.18

where $\gamma = a/N\,[VC(H) + VC(H)\ln(bN/VC(H)) - \ln(\delta/4)]$, a, b, c are constants, N is the sample size, E_{out} and E_{in} are the generalisation and empirical errors, $1-\delta$ is the probability that the bound holds, and $VC(H)$ is the VC-dimension of the family of regression models $\mathcal{F} = \{f(\mathbf{x}, h)\} : \mathbb{R}^J \rightarrow \mathbb{R}$ indexed by $h \in H$. The VC-bound exposes how changes in $VC(H)$ and N impact the expected generalisation error. Because the generalisation bound follows the same monotonicity as the VC-bound, we can use Eq. (14) to determine the N needed to train a family of models $\mathcal{F}$, or the $\mathcal{F}$ that a constrained N can reasonably train. Numerous studies, such as [1], indicate that by applying a rule of 10, mathematically expressed as $10 \times VC(H) \leq N$, a reasonable generalisation error - i.e. a level where the VC-bound is meaningful with a probability close to 1 - can be attained. The rule of 10 can either be applied to determine an appropriate N for a given $VC(H)$ or the reverse. Given our N of 365, the rule of 10 suggests that an $\mathcal{F}$ with a $VC(H) \leq 36$ is best suited to our case study.

Linking the $VC(H)$ to our final modelling feature size, it can be shown that the $VC(H)$ of a J-dimensional linear classifier and regression is equal to $J+1$ — i.e. its degrees of freedom. The $VC(H)$ of a real-valued function $\mathcal{F} = \{f(\mathbf{x}, h)\}$ is equal to the $VC(H)$ of its

respective binary classifier $\mathbb{1}(f(\mathbf{x},h)-[f(\mathbf{x},h)]>0)$ [40]. Because the expressiveness and by extension $VC^{J}_{NN}(H)$ of J-dimensional real-valued neural networks is at least as large as the $VC^{J}_{LR}(H)$ of a J-dimensional linear regression we can determine a limit for the maximum number of feature inputs for our case study. Formally, $\mathcal{F}^{J}_{LR} \subseteq \mathcal{F}^{J}_{NN}$ implies that $\left[VC^{J}_{LR}(H)=J+1\right] \leq VC^{J}_{NN}(H)$, and therefore $J \leq VC^{J}_{LR}(H) \leq VC^{J}_{NN}(H)$. From the above we discern that strictly no more than 35 feature inputs, J, should be selected by our feature selection method.

Similarly to [41], we use random forests (RF) to identify and select features with the greatest explanatory power. RF, an ensemble machine learning model, fits numerous decision trees to random samples of a training dataset. Because each tree is trained to recursively split data by maximising an information gain, estimates of a feature's importance can be computed. This property, as well as RF's computational speed, and robustness to outliers and noise [42], make RF an effective feature selection method.

The RF feature selection method is trained on the training set and tuned by minimising the R^2 across the validation set. We set a feature importance cut-off that prevents the RF method from selecting more than 35 features. Fig. 3 displays the total number of features selected for every Belgian and Dutch DAM contract. One-day-lagged and seven-day-lagged dependent variables are frequently selected, capturing the well-known seasonalities in DAM prices. For example, the following 17 features are selected for the Dutch 14h contract: BE^{11}_{-7}, BE^{14}_{-7}, NL^{14}_{-7}, generation_fc$^{14}_{-7}$, BE^{15}_{-7}, NL^{15}_{-7}, NL^{16}_{-7}, NL^{17}_{-2}, NL^{14}_{-1}, NL^{16}_{-1}, BE^{17}_{-1}, NL^{17}_{-1}, NL^{18}_{-1}, load_fc$^{14}_{-1}$, and wind_fc$^{14}_{-1}$. The NL^{14}_{-7} and generation_fc$^{14}_{-7}$ represent the seven-day-lagged Dutch 14h DAM price and seven-day-lagged Dutch 14h generation forecast respectively. While both the one-day-lagged and seven-day-lagged dependent variables are selected for the Dutch 14h contract, because we do not constrain our RF feature selection method, no lagged dependent variables are selected in 6 out of 48 cases: the Belgian: {00h, 01h, 14h}, and the Dutch: {00h, 10h, 13h}.

3.4. Forecasting models

In order to establish reasonable benchmark forecast accuracies we model DAM prices using ARXs. Further, following [18], we model prices using a two-layer neural network (**2NN**) consisting of two intermediate FCLs. Finally, to evaluate the performance impacts of augmentation on convolutional neural networks, we model DAM prices using a joint three-layer network (**2CNN_NN**) consisting of two intermediate CONVs, two optional max-pooling or average-pooling layers, and a single intermediate FCL. Layers are stacked to identify non-linear relationships, and prevent an explosion in the number of network parameters. Early stoppage, L2 regularisation, dropout, ReLU activation functions, Adam, learning rate scheduling, and learning rate decay are used to combat node saturation, speed-up model training, and reduce the likelihood of overfitting. Implementation details can be found in Appendix. To identify the hyperparameters of the above evaluation models, Bayesian optimisation, which often outperforms random grid search [43], is employed. A sample of selected hyperparameters is displayed in Tables A.1 and A.2.

3.5. Data augmentation

When augmenting multivariate time series, both X, real explanatory variables, and Y, real response variables, can be simultaneously used to generate $\tilde{X}$, augmented explanatory variables, and $\tilde{Y}$, augmented response variables. X and Y represent measurements indexed through time, forming a set of time series $\{\mathbf{t}\in T \mid \mathbf{t}=\mathbf{x}\cup y,\ \forall \mathbf{x}\in X,\ y\in Y\}$. An ability to generate a set of augmented time series $\{\tilde{\mathbf{t}}\in \tilde{T} \mid \tilde{\mathbf{t}}=\tilde{\mathbf{x}}\cup\tilde{y},\ \forall\tilde{\mathbf{x}}\in\tilde{X},\ \tilde{y}\in\tilde{Y}\}$ simplifies augmentation by alleviating the need for any pseudo-labelling of $\tilde{X}$. Examples of generated time series are presented in Fig. 4. In Sections 3.5.1–3.5.4, we describe how, given a set of normalised training inputs T, normalised augmented outputs $\tilde{T}$ can be generated using a multitude of augmentation methods.

3.5.1. Feature space augmentation

From [4] we choose to evaluate jittering, scaling and magnitude-warping. Briefly describing how these methods generate data, jittering adds varying amounts of noise ($\epsilon \sim N(0,\sigma^2 I)$) to $\mathbf{t}$. Scaling generates data by multiplying $\mathbf{t}$ by a random scalar ($s \sim N(1,\sigma^2)$). Magnitude-warping, similarly to scaling, multiplies $\mathbf{t}$ by a smoothly-varying random curve with k knots and a standard deviation σ. To select the optimal hyperparameters σ and k, Bayesian optimisation is employed. Table A.3 presents examples of selected hyperparameters.

3.5.2. Autoencoder augmentation

We utilise AEs, introduced in Section 2.5, to augment multivariate time series. Similarly to [38], as a distance function, $l(\mathbf{x},.)$, to measure the reconstruction loss in Eq. (5) we choose the binary cross-entropy (BCE). We found that using the BCE, an asymmetric loss function, outperformed using a symmetric loss, such as the mean squared error.[5] The BCE a special instance of the cross-entropy $\mathcal{H}(p_X(\mathbf{x}),p_{\tilde{X}}(\tilde{\mathbf{x}})) = -\sum_j p_X(x_j)\log p_{\tilde{X}}(\tilde{x}_j)$, is calculated according to Eq. (15).

$$\begin{aligned}\mathrm{BCE}(\mathbf{x},D_\theta(E_\phi(\mathbf{x}))) = -\sum_{j=1}^{J}\Big[&p_X(x_j)\log p_{\tilde{X}}(D_\theta(E_\phi(\mathbf{x}))_j)\\ &+\big(1-p_X(x_j)\big)\log\big(1-p_{\tilde{X}}(D_\theta(E_\phi(\mathbf{x}))_j)\big)\Big],\end{aligned} \tag{15}$$

where J is the dimensionality of X. Optimising the BCE pointwise is equivalent to optimising the forward D_{KL}: $D_{KL}(p_X(\mathbf{x}) \parallel p_{\tilde{X}}(\tilde{\mathbf{x}})) = \mathcal{H}(p_X(\mathbf{x}),p_{\tilde{X}}(\tilde{\mathbf{x}})) - \mathcal{H}(p_X(\mathbf{x}))$, because $\mathcal{H}(p_X(\mathbf{x}))$, the entropy of p_X, is known and constant. For our AEs' training, we use Leaky ReLU with a negative slope ζ to combat neuron saturation, early stoppage and L2 regularisation with a regularisation parameter τ to limit model overfitting, and Adam to speed-up convergence. The search spaces of ζ and τ for Bayesian optimisation can be found in Appendix.

AEs, once trained, can generate time series, $\tilde{\mathbf{t}}$, by applying perturbations to encoder outputs in the latent space Z according to $\tilde{\mathbf{t}} = D_{\tilde{\theta}}(f(E_{\tilde{\phi}}(\mathbf{t})))$, where $f: Z \to \mathbb{R}^J$ is a perturbation function. This is advantages because encoding model inputs from the feature space T to Z increases the relative volume occupied by the real distribution p_T [29]. To allow tailored data generation, we identify optimal encoding and decoding functions for every DAM hourly contract using Bayesian optimisation. Table A.4 presents examples of encoding and decoding functions.

As shown in Fig. 5, encoder outputs are scaled by a scaling matrix $S \sim N(1, diag(\boldsymbol{\upsilon}^\intercal))$, where $\boldsymbol{\upsilon}^\intercal$ is equal to the column-wise standard deviation (SD) of encoder outputs, $[\sigma_1,\dots\sigma_K]$, multiplied by γ. Formally, we apply a scaling latent space perturbations according to: $f(\mathbf{z}) = S\mathbf{z}$. Bayesian optimisation is used to identify an optimal γ. See Appendix for the search space of γ.

[5] To demonstrate why an asymmetric loss function, specifically the BCE, is better suited to modelling p_X than a symmetric loss function such as the mean squared error (MSE), below we calculate the forecasting losses ($\mathcal{L}$) and gradient magnitudes ($|\nabla|$) for two model predictions. Firstly, we consider the $\mathcal{L}$ and $|\nabla|$ when a model underpredicts a scaled target value (0.1) in the left tail of p_X by 0.075. Using the MSE, we record $\mathcal{L}^{MSE}_{0.025} = (0.025-0.1)^2 = 0.0056$ and $|\nabla|^{MSE}_{0.025} = |2\tilde{x}_j - 2x_j| = 0.15$, while with the BCE we get: $\mathcal{L}^{BCE}_{0.025} = -0.1\log(0.025) - 0.9\log(0.975) = 0.392$ and $|\nabla|^{BCE}_{0.025} = |-x_j/\tilde{x}_j + (1-x_j)/(1-\tilde{x}_j)| = 3.077$. Secondly, we consider the $\mathcal{L}$ and $|\nabla|$ when the model overpredicts the same target by 0.075. The MSE again records a loss of 0.0056 and gradient magnitude of 0.15, while the BCE yields: 0.3474 and 0.519. It is clear that the MSE treats both underprediction and overprediction of target values in the left tail of p_X identically ($\{\mathcal{L}^{MSE}_{0.025},|\nabla|^{MSE}_{0.025}\} = \{\mathcal{L}^{MSE}_{0.175},|\nabla|^{MSE}_{0.175}\}$), while the BCE places an emphasis on correcting underprediction ($\{\mathcal{L}^{BCE}_{0.025},|\nabla|^{BCE}_{0.025}\} > \{\mathcal{L}^{BCE}_{0.175},|\nabla|^{BCE}_{0.175}\}$). Notice also that BCE $|\nabla|$ dominate those of the MSE. Such asymmetric weight updating is desirable when modelling a non-uniform probability distribution as it promotes mass covering, while strictly preventing outlier generation. The results are: fewer generated data points near 0 and 1, and more generated data points near the central tendency.

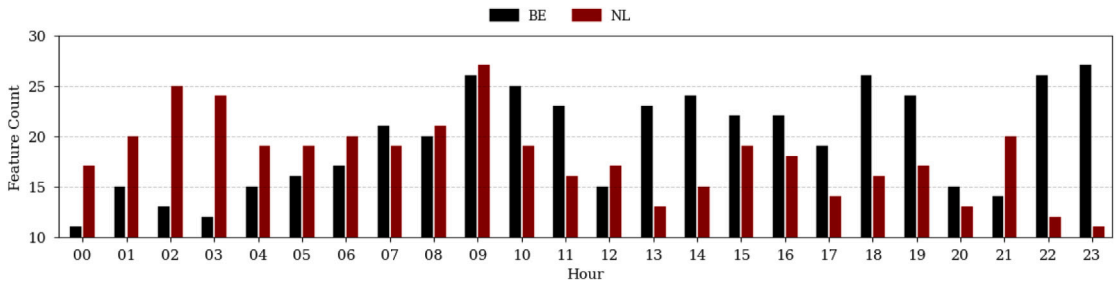


Fig. 3. Visualisation presenting the number of selected features for every Belgian (BE) and Dutch (NL) DAM hour denominated contract. On average, roughly 20 features are selected for Belgian contracts and 18 for Dutch contracts.

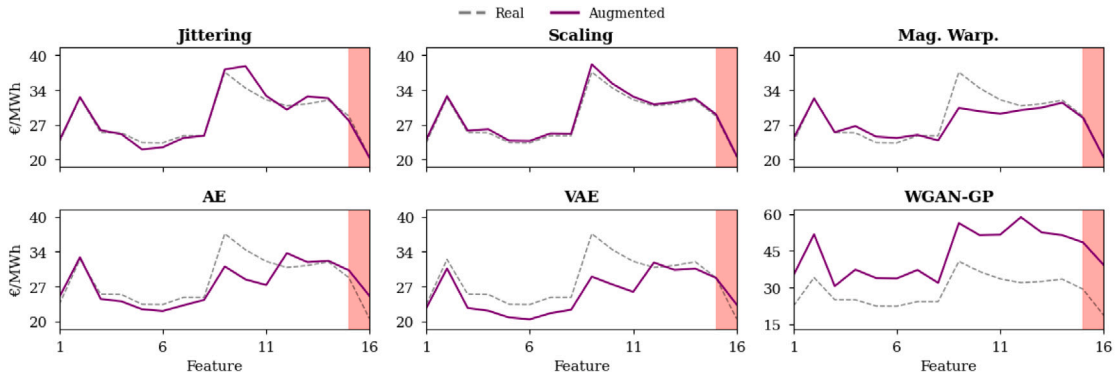


Fig. 4. Generated time series examples for the Belgian 01h contract. Note that $\mathbf{x}$ consists of lagged prices (in €/MWh) for this contract. The training (Real) time series is used as an input in the generation of the augmented series. The violet band displays how we slice $\tilde{\mathbf{t}}$ to give $\tilde{\mathbf{x}}$, in the feature range $[1, 15]$, and $\tilde{y}$, in the feature range $(15, 16]$.

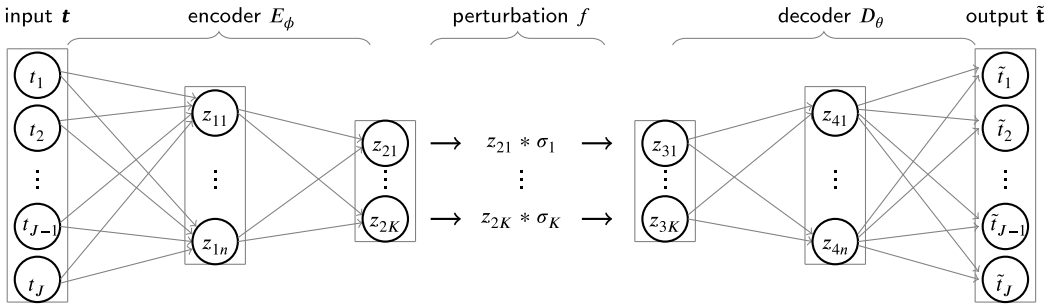


Fig. 5. AE network architecture showing the input–output relationship for multivariate time series generation. Encoder and decoder are drawn with one hidden layer for simplicity. Each, however, usually consists of more hidden layers.

3.5.3. *Variational autoencoder augmentation*

Introducing our VAE architectures, all inference networks use a single latent variable layer with a normal prior ($p(\mathbf{z}) \sim N(0, I)$) and variational parameters λ: μ and σ^2. These parameters are approximated as: $N(\mathbf{z}|\mu_\phi(\mathbf{t}), \sigma^2_\phi(\mathbf{t}))$, where $\mu_\phi(\mathbf{t})$ and $\sigma^2_\phi(\mathbf{t})$ are parametrised latent variational estimates of λ. To train our VAEs, we use a variant of Eq. (7). The weight of the forward D_{KL} regulariser in Eq. (7) is adjusted by a Bayesian optimised parameter δ [44]. Additionally, an L2 penalty, weighted by a Bayesian optimised parameter τ, is added. For a minibatch of m time series ($\{\mathbf{t}^i\}_{i=1}^m \sim p_T$), we train our VAE according to Eq. (16).

$$\arg\min_{\theta,\phi} \mathcal{L}(\theta, \phi; \mathbf{t}) = \frac{1}{m}\sum_{i=1}^{m}\left(-\left(\sum_{j=1}^{J}\Omega_{i,j}\right) + \delta\left(-\frac{1}{2}\sum_{k=1}^{K}Y_{i,k}\right)\right) + \tau(\|\theta\|_2^2 + \|\phi\|_2^2), \tag{16}$$

where $\Omega_{i,j} = p_T(t^i_j)\log p_\theta(t^i_j|\mathbf{z}^i) + \big(1 - p_T(t^i_j)\big)\log\big(1 - p_\theta(t^i_j|\mathbf{z}^i)\big)$, $Y_{i,k} = 1 + \log\sigma_\phi(\mathbf{t}^i)^2_k - \mu_\phi(\mathbf{t}^i)^2_k - \sigma_\phi(\mathbf{t}^i)^2_k$, and J/K are the dimensionalities of feature/latent variable outputs respectively [30]. In Eq. (16), the first term on the right is equivalent to the BCE, while the second term is a closed-form expression of the forward D_{KL}. We use Bayesian optimisation to identify optimal inference and generation networks for each DAM contract. Table A.4 presents examples of optimal inference and generation networks. Similarly to AEs, we apply Leaky ReLU with a negative slope ζ, early stoppage, and Adam. The search spaces of δ, τ and ζ for Bayesian optimisation can be found in Appendix. Once trained, by minimising the above VAE loss, the prior ($p(z) \sim N(0, I)$) is sampled and passed through the generation network to yield VAE augmented data as shown in Fig. 6.

3.5.4. *Wasserstein generative adversarial network augmentation*

Summarising three generators that motivated our WGAN-GP generation method, [46] proposed generating outputs, $\tilde{\mathbf{x}}_2$, by performing $G_\phi(\mathbf{x}_1) \rightarrow \tilde{\mathbf{x}}_2$: inputs, $\mathbf{x}_1$, are passed into G_ϕ and dropout is used to provide noise to G_ϕ. Similarly, [47] proposed generating outputs by performing $\mathcal{T}_\phi(\mathbf{x}_1) \rightarrow \tilde{\mathbf{x}}_2$ using a transformation network comparable to an autoencoder, $\mathcal{T}_\phi$, in place of G_ϕ. Finally, [48] proposed generating outputs by performing $G_\phi(\mathbf{z}, \sigma) \rightarrow \tilde{\mathbf{x}}$. Noise, $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I)$, is added in every layer of G_ϕ to improve energy-based GAN training stability.

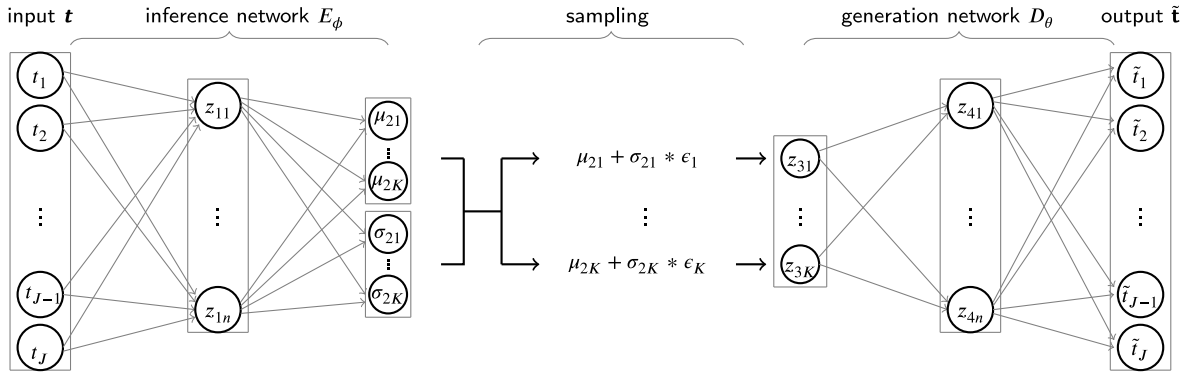


Fig. 6. VAE network architecture showing the input–output relationship for multivariate time series generation. Inference and generation networks are drawn with one hidden layer for simplicity. Each, however, usually consists of more hidden layers. Note that a vector $\boldsymbol{\epsilon} \sim N(0, I)$ yields $\{\epsilon_1, \ldots, \epsilon_K\}$.

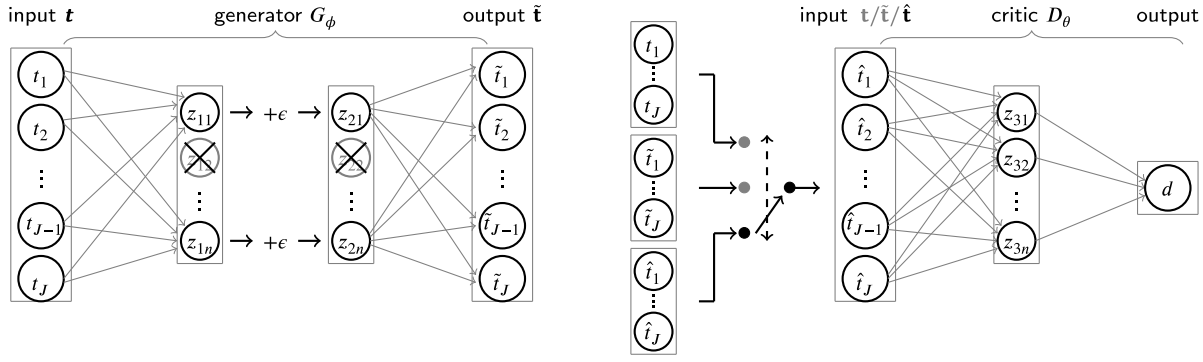


Fig. 7. WGAN-GP network architecture showing the input–output relationship separately for a generator and a critic. Crosses in generator nodes i.e. z_{1n_1-1} represent our use of dropout. Meanwhile $\hat{\mathbf{t}} = \alpha\mathbf{t} + (1-\alpha)\tilde{\mathbf{t}}$, where $\alpha \sim U[0,1]$. The generator and the critic are drawn with one hidden layer for simplicity. Each, however, usually comprises of more than one hidden layer. To obtain the WGAN-GP loss, all inputs, namely t, $\tilde{t}$, and $\hat{t}$, must individually be passed through the critic.

Algorithm 1 WGAN-GP for multivariate time series augmentation. The need to control both training duration and identify early instances of mode collapse is weighed against the need to avoid impeding model convergence. Following [35], default variables $\lambda = 10$ and $n_{critic} = 5$ are used.

Require: learning rates α_g and α_d, gradient penalty coefficient λ, regularisation parameter τ, standard deviations σ, minimum spread coefficient ψ, batch size m, early stopping parameters k, number of training iterations K, the number of critic updates per generator update n_{critic}

Require: initial discriminator parameters θ, initial generator parameters ϕ

for $k = 1, \ldots, K$ **do**
  for $n = 1, \ldots, n_{critic}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample training time series $\mathbf{t} \sim P_T$, and a random number $\alpha \sim U[0,1]$
      $\tilde{\mathbf{t}} \leftarrow G_\phi(\mathbf{t}, \sigma_g)$
      $\hat{\mathbf{t}} \leftarrow \alpha\mathbf{t} + (1-\alpha)\tilde{\mathbf{t}}$
      $\mathcal{L}^{(i)} \leftarrow D_\theta(\tilde{\mathbf{t}}, \sigma_d) - D_\theta(\mathbf{t}, \sigma_d) + \lambda(\| \nabla_{\hat{\mathbf{t}}} D_\theta(\hat{\mathbf{t}}, \sigma_d) \|_2 - 1)^2$
    end for
    $\theta \leftarrow \text{RMSprop}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^m \mathcal{L}^{(i)},\ \theta,\ \alpha_d\right)$
  end for
  Sample a batch from the training time series $\{\mathbf{t}^{(i)}\}_{i=1}^m \sim P_T$
  $\phi \leftarrow \text{RMSprop}\left(\nabla_\phi \frac{1}{m} \sum_{i=1}^m \left[- D_\theta(G_\phi(\mathbf{t}^i, \sigma_g), \sigma_d)\right] + \tau \parallel \phi \parallel_2^2,\ \phi,\ \alpha_g\right)$
  if $k > k_{check}$ **then**
    $\tilde{T} \leftarrow G_\phi(T, \sigma)$
    if $\sum_{j=1}^J \text{SD}(\tilde{T}) < \psi \sum_{j=1}^J \text{SD}(T)$ or $-W_D(\theta; T, \tilde{T})$ has not set new low for k_{break} **then**
      early stopping **break**
    end if
  end if
end for

*Noise, $\epsilon \sim N(0, \sigma^2 I)$, is added element-wise to intermediate G layer outputs and, similarly to [45], optionally to D.

Using elements of [46–48], we propose performing $G_\phi(\mathbf{t}, \sigma) \rightarrow \tilde{\mathbf{t}}$ to augment time series. Our G_ϕ accepts time series $\mathbf{t}$ as inputs and adds noise, $\epsilon \sim N(0, \sigma^2 I)$, to intermediate layer outputs. Modifying Eqs. (12) and (13) to reflect our generation method, we train our WGAN-GP according to Eqs. (17) and (18).

$$\arg\min_\theta W_D(\theta; \mathbf{t}, \tilde{\mathbf{t}}) = \mathbb{E}_{\tilde{\mathbf{t}} \sim p_{\tilde{T}}(\tilde{\mathbf{t}})}[D_\theta(\tilde{\mathbf{t}})] - \mathbb{E}_{\mathbf{t} \sim p_T(\mathbf{t})}[(D_\theta(\mathbf{t}))] + \lambda \mathbb{E}_{\hat{\mathbf{t}} \sim p_{\hat{T}}(\hat{\mathbf{t}})}[(\|\nabla_{\hat{\mathbf{t}}} D_\theta(\hat{\mathbf{t}})\|_2 - 1)^2], \tag{17}$$

$$\arg\min_\phi W_G(\phi; \mathbf{t}) = -\mathbb{E}_{\mathbf{t} \sim p_T(\mathbf{t})}[D_\theta(G_\phi(\mathbf{t}, \sigma))] + \tau\|\phi\|_2^2, \tag{18}$$

where D_θ, referred to as the critic in WGANs, approximates the 1-Lipschitz function $f : T \rightarrow \mathbb{R}$. The L2 penalty, $\tau\|\phi\|_2^2$, is added to the generator loss to further improve training stability. During a non-exhaustive empirical evaluation we found that L2 regularisation facilitated WGAN-GP training and increased the diversity of generated data (See [45]).

Algorithm 1 presents our proposed method for training WGAN-GP time series augmentors. The hyperparameters referenced in Algorithm 1 and further described in Appendix are selected using Bayesian optimisation for every DAM hourly contract. Similarly to [35], the negative critic loss, $-W_D(\theta; \mathbf{t}, \tilde{\mathbf{t}})$, is used in our convergence criterion. Once trained, G_ϕ is employed to generate multivariate time series. Example G_ϕ and D_θ network architectures are presented in Fig. 7 and Table A.4.

Across a randomly selected sample of DAM contracts, we empirically compared the training stability of WGAN-GPs with the training stability of vanilla GANs, GANs with soft real and fake labels, WGANs, and context encoders, which generate regions of a target and use a compound loss function comprising of a reconstruction loss plus an adversarial loss. Overall, WGAN-GPs were found to be the most stable and easy to train. They suffered significantly fewer instances of mode collapse and were less sensitive to small hyperparameter changes.

4. Results and discussion

4.1. Augmented data exploration

To highlight the capabilities and limitations of the augmentation methods, Section 4.1.1 compares the internal linear structure of real training and generated data by visualising their principal component projections. Section 4.1.2 analyses the distributions of both real training and generated data by visualising the statistical differences between them. Note that no link between the distributions of generated data and augmentation performance should be assumed.

4.1.1. Principal components

Principal component analysis [49] orthogonally transforms a set of variables onto a linearly uncorrelated orthogonal basis set of principal components. Variable variance is maximised in the first principal components, enabling the informative visualisation of multivariate datasets, and the linear correlations and temporal relationships between time series features. To generate Fig. 8 we performed singular value decomposition, and projected both our real and generated data onto the first two principal components, PC1 and PC2, of the training series.

Analysing Fig. 8, positive $\%\Delta\text{SD}_{PC}$s as well as stochastic and relatively small shifts in both $\overline{\text{PC1}}$ and $\overline{\text{PC2}}$ highlight how feature space transformations increase the principal component variability of T without critically distorting the temporal relationships in T. Feature space augmentors effectively generate $\tilde{T}$ by adding noise, $E \sim N\left(0, diag(\sigma)I\right)$, to T. E adds variability to the projection of T; $T^{\intercal}[\mathbf{v}_1, \mathbf{v}_2] + E^{\intercal}[\mathbf{v}_1, \mathbf{v}_2]$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the principal and second principal eigenvectors of T. Since E is zero-centred and tuned to avoid distorting the temporal relationships in T, a majority of generated $\tilde{\mathbf{t}}$s generally remain in the manifold of T. Out of the three feature space augmentors examined, jittering is observed to generate the most varied $\tilde{\mathbf{t}}$s, with larger $\%\Delta\text{SD}_{PC}$s.

Turning to model-based augmentors, AEs, VAEs, and WGAN-GPs generate $\tilde{\mathbf{t}}$s more tightly clustered around their central measures of tendency and lower principal component variability. More tightly clustered $\tilde{\mathbf{t}}$s result because of early stoppage and how model-based augmentors optimise their respective objective functions. While empirically evaluating various model-based augmentors, we observed that AEs and VAEs initially learn to generate data that varies across PC1, before proceeding to learn to generate data that varies across other principal components. Early stoppage ceases model training before AEs and VAEs learn to generate data with greater overall principal component variability than T, resulting in less varied but more meaningful $\tilde{\mathbf{t}}$s. Similarly, early stoppage ceases WGAN-GP training before G begins overfitting the training data. Overall, Fig. 8 demonstrates that AEs, VAEs, and WGAN-GP can generate $\tilde{\mathbf{t}}$s on the manifold of T. While $\Delta\overline{\text{PC}}$s are non-zero, their magnitudes suggest that model-based augmentors do not generate $\tilde{\mathbf{t}}$s with critically distorted temporal relationships.

4.1.2. Generated distributions

Distribution differences between T and $\tilde{T}$ for each feature j, namely T_j and $\tilde{T}_j$, are visualised in Fig. 9 to demonstrate our augmentors' ability to successfully approximate the moments of the real distribution, and to emphasise their capacity to approximate both the lagged dependent and the lagged independent variables in T.

Examining the generated distributions of feature space augmentors, because jittering, scaling and magnitude-warping do not model the underlying distribution of T, perturbations with a potential to significantly alter T_js are generally avoided. Near zero $\overline{\text{Means}}$ and positive $\overline{\text{SDs}}$ in Fig. 9 highlight how feature space transformations impact the moments of the real distribution. It can be shown that the addition of $\epsilon \sim N(0, \sigma)$ to t_j increases the variance and decreases the skewness of feature j. Formally, $\mathbb{E}[t_j] = \mathbb{E}[t_j] + \mathbb{E}[\epsilon]$, $\mathbb{E}[t_j^2] - \mathbb{E}[t_j]^2 \leq \mathbb{E}[(t_j+\epsilon)^2] - \mathbb{E}[t_j]^2$, and $\left|\mathbb{E}[(\frac{t_j-\bar{t}_j}{s_j})^3]\right| \geq \left|\mathbb{E}[(\frac{t_j+\epsilon-\bar{t}_j}{\tilde{s}_j})^3]\right|$, where $\bar{t}_j$ is the mean of T_j, and s_j and $\tilde{s}_j$ are the standard deviations of T_j and $\tilde{T}_j$ respectively. Similarly, it can be shown that scaling t_j by $s \sim N(1, \sigma)$ increases the feature's variance and skewness. Formally, $\mathbb{E}[t_j] = \mathbb{E}[st_j]$, $\mathbb{E}[t_j^2] - \mathbb{E}[t_j]^2 \leq \mathbb{E}[st_j^2] - \mathbb{E}[t_j]^2$, and $\left|\mathbb{E}[(\frac{t_j-\bar{t}_j}{s_j})^3]\right| \leq \left|\mathbb{E}[(\frac{st_j-\bar{t}_j}{\tilde{s}_j})^3]\right|$.

Analysing the generated distributions of autoencoders, in Fig. 9 AE and VAE are observed to successfully approximate the means of the real distribution. A majority of $\tilde{T}_j$s have a negative $\%\Delta$SD and $\%\Delta$Skewness. Explaining these results, Sections 2.6 and 3.5.2 underlined that both AEs and VAEs perform parameter optimisation by minimising the forward D_{KL}. As is explained in Section 2.4, minimising the forward D_{KL} spurs an autoencoder to perform 'mean seeking' approximations, centring $p_{\tilde{T}}$ around the mean of p_T. Because mean centring is prioritised, it generally occurs before early stoppage, and before the second and third standardised moments of p_T are fully approximated. Finally, examining the generated distributions of WGAN-GP, WGAN-GP generates $\tilde{T}_j$s with the highest $|\overline{\text{Mean}}|$ and $|\overline{\text{SD}}|$, and both positive and negative $\%\Delta$ SDs. Minimising the Wasserstein loss encourages GANs to approximate the second standardised moments of the real distribution without inducing 'mean seeking' approximations of p_T.

4.2. Evaluation setup

We aim to determine whether data augmentation can significantly boost the prediction accuracies of multivariate time series regression models. To achieve this aim, across the test set we conduct an empirical and statistical analysis of regression models trained with (Aug: $[T, \tilde{T}]$, size $= 2N_{train}$) and without (Bench: $[T]$, size $= N_{train}$) augmentation. The specifics of our empirical and statistical evaluation procedures are detailed below.

Describing our empirical evaluation procedure, to assess the overall impacts of data augmentation, in Section 4.3 we compute and evaluate the mean absolute errors (MAE), the root mean squared errors (RMSE), and the symmetric mean absolute percentage errors (sMAPE) of regression forecasts with and without augmentation. Additionally, similarly to [3], we compute Win / Tie / Loss scores to measure an augmentor's efficiency. Instances where benchmark RMSEs decrease/increase across the test set are classified as Wins/Losses. Instances where an augmentor fails to improve upon benchmark RMSEs across the validation set are classified as Ties. Beyond an overall evaluation, in Section 4.3.1 we analyse cumulative absolute error differences between Bench and Aug model forecasts to determine whether augmentation performance remains consistent after model refitting, and to evaluate how augmentation impacts forecast accuracies across the test set. To evaluate whether augmentation performance is impacted by model complexity, we examine the relationship between an augmentor's performance and forecasting model complexity in Section 4.3.2. To determine whether augmentation performance is invariant to changes in the target distribution, in Section 4.3.3 we evaluate the relationship between an augmentor's performance and the target's variance. Finally, to investigate how augmentation improves Bench model accuracies, in Section 4.3.4 we analyse forecast accuracies of targets inside and outside the interquartile range.

Describing our statistical evaluation procedure, to determine the statistical significance of any accuracy improvements, we perform one-sided Wilcoxon signed-rank tests [50]. The Wilcoxon test is a non-parametric test used to compare distributions of paired samples. We use this test to compare Bench and Aug RMSEs: $\{\text{RMSE}_i^{bench}\}_{i=1}^n$ and $\{\text{RMSE}_i^{aug}\}_{i=1}^n$ respectively. The test assumes symmetry between positive and negative sample differences: $d_i = \text{RMSE}_i^{aug} - \text{RMSE}_i^{bench}$, $\forall\, i \in \{1, \ldots, n\}$. To compute the Wilcoxon statistic, the rank sums of positive and negative differences are calculated, and their minimum is taken. Note, following the recommendations of [51], we use the Pratt ranking method to obtain conservative estimates of the Wilcoxon statistic. We perform a one-sided Wilcoxon test with the null hypothesis H_0: $\text{Median}(\{d_i\}_{i=1}^n) \geq 0$ and alternative hypothesis H_1: $\text{Median}(\{d_i\}_{i=1}^n) < 0$.

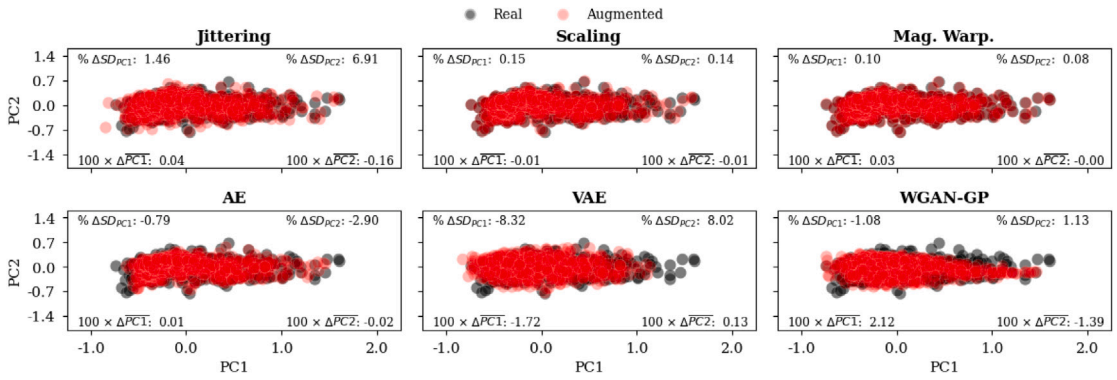


Fig. 8. Projections of the real and generated Dutch 20h time series. Scaled nominal differences between mean generated and mean real principal component projections ($100\times\Delta\overline{PC}$); and percentage differences between generated and real principal component standard deviations ($\%\Delta\text{SD}_{PC}$) are displayed. A positive $\%\Delta\text{SD}_{PC}$ records an increase in PC variability..

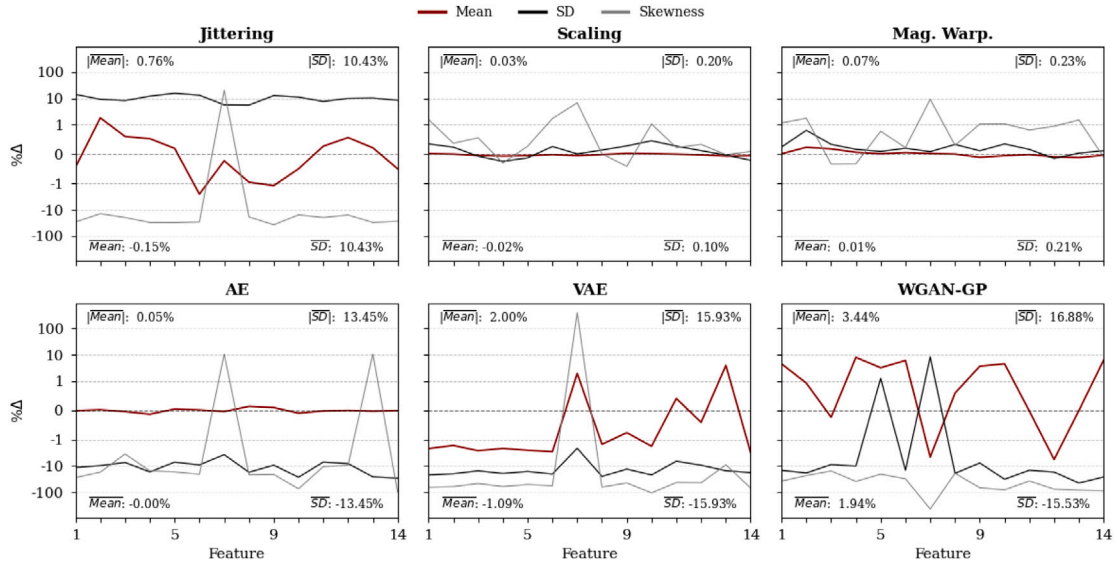


Fig. 9. Statistical differences between real training and generated Dutch 20h time series. Exogenous variables, namely lagged temperatures and wind forecasts, are stored as features $\{7, 13\}$. For feature j, $\%\Delta\text{Mean}_j$ is calculated according to: $100\times[(\text{Mean}(\tilde{T}_j) - \text{Mean}(T_j)) \div \text{Mean}(T_j)]$. The $\%\Delta$ SD (standard deviation) and $\%\Delta$ Skewness are calculated similarly. Nominal and absolute averages of the above mentioned metrics – e.g. the $\overline{\text{Mean}}$ and $|\overline{\text{Mean}}|$ - are presented.

When the Wilcoxon statistic translates to a p-value less than 0.05, we reject H_0 and accept H_1. A p-value less than 0.05 indicates a statistically significant performance improvement at a 5% level.

In addition to the aforementioned analyses, in Section 4.3.5 we assess the prediction accuracies of models trained using a combination of $\tilde{T}$s, generated using multiple model-based augmentors. For example, performances of forecasting models trained using both AE $\tilde{T}$ and VAE $\tilde{T}$ ($[T, \tilde{T}_{AE}, \tilde{T}_{VAE}]$, size $= 3N_{train}$) are evaluated. Motivated by [9], we also evaluate the performances of ensemble augmentation methods, which average AE, VAE, and WGAN-GP forecasts. Both simple averaging (AVG) and weighted averaging (Weighted) ensemble methods are assessed. The AVG method assigns the same weights to augmentors' forecasts, while the Weighted method assigns weights based on augmentors' previous day's absolute forecast errors. To calculate an AVG forecast for day d we perform: $\hat{y}_d^{\text{AVG}} = 1/3(\hat{y}_d^{\text{AE}} + \hat{y}_d^{\text{VAE}} + \hat{y}_d^{\text{WGAN-GP}})$. A Weighted forecast for day d is calculated according to Eq. (19).

$$\hat{y}_d^{\text{Weighted}} = \frac{1}{\varpi}\left(\frac{\hat{y}_d^{\text{AE}}}{|y_{d-1} - \hat{y}_{d-1}^{\text{AE}}|} + \frac{\hat{y}_d^{\text{VAE}}}{|y_{d-1} - \hat{y}_{d-1}^{\text{VAE}}|} + \frac{\hat{y}_d^{\text{WGAN-GP}}}{|y_{d-1} - \hat{y}_{d-1}^{\text{WGAN-GP}}|}\right), \tag{19}$$

where $\varpi = 1/|y_{d-1} - \hat{y}_{d-1}^{\text{AE}}| + 1/|y_{d-1} - \hat{y}_{d-1}^{\text{VAE}}| + 1/|y_{d-1} - \hat{y}_{d-1}^{\text{WGAN-GP}}|$. Under the Weighted method, a greater weight is assigned the forecasts of historically successful augmentors.

4.3. Numerical results and discussion

To highlight overall performance changes, summary augmentation results are presented in Table 2. Mean forecast error percentage changes, $\%\Delta\mathbf{E}$, for each country and evaluation model pair, are calculated according to: $100\times[(\text{E}_{aug} - \text{E}_{bench}) \div \text{E}_{bench}]$, where E_{aug} and E_{bench} are the augmentation and ***Bench.*** mean forecast errors respectively, and E is either the MAE, RMSE, or sMAPE. To calculate **Summary** forecast error percentage changes, $\%\Delta$**E**s are averaged across the six evaluation cases. Bold p-values, from Wilcoxon tests, indicate statistically significant performance improvements at a 5% level. Grey highlights show the best results for each row.

The results indicate that not all augmentation methods can significantly boost the regression accuracies of multivariate time series models. While AEs, VAEs, and WGAN-GPs, on average, reduce benchmark summary MAEs, RMSEs, and sMAPEs by more than 2%, yielding p-values $<$ 0.05, jittering and magnitude-warping fail to significantly improve summary benchmark MAEs and sMAPEs, yielding p-values $\geq$ 0.05. Beyond the summary results, AEs, VAEs, and WGAN-GPs are observed to significantly boost benchmark performances in 6/6, 6/6 and 4/6 evaluation cases respectively. Moreover, they are observed to improve 58.33% to 87.4% of benchmark RMSEs; yielding $\%\Delta$ RMSEs between −4.36% and −0.64%. Meanwhile, jittering, scaling and magnitude-warping are observed to yield p-values $<$ 0.05 in

Table 2
Augmentation results displaying average percentage changes in benchmark (***Bench.***) mean forecast errors: specifically MAEs, RMSEs, and sMAPEs.

			Jittering	Scaling	Mag. Warp.	AE	VAE	WGAN-GP	*Bench.*
Belgian DAM prices	2NN	%Δ **MAE**	−0.36	−0.94	−2.23	−3.26	−4.07	−3.96	*9.26*
		%Δ **RMSE**	−0.98	−1.15	−2.14	−3.18	−3.50	−4.36	*16.47*
		%Δ **sMAPE**	−0.24	−1.03	−2.28	−3.49	−4.52	−4.04	*17.27*
		Win / Tie / Loss	6/13/5	4/16/4	12/7/5	15/4/5	21/1/2	15/1/8	–
		p-value	0.38	0.42	**0.03**	**0.01**	**0.00**	0.12	–
	2CNN_NN	%Δ **MAE**	0.39	0.58	1.90	−3.82	−5.79	−3.03	*9.30*
		%Δ **RMSE**	0.25	0.57	1.74	−2.72	−2.92	−3.54	*16.38*
		%Δ **sMAPE**	0.76	0.60	1.99	−3.27	−5.07	−2.41	*17.09*
		Win / Tie / Loss	7/12/5	4/16/4	5/10/9	16/1/7	17/0/7	16/0/8	–
		p-value	0.33	0.55	0.93	**0.01**	**0.01**	**0.03**	–
	ARX	%Δ **MAE**	0.57	−0.70	0.50	−1.96	−2.80	−6.03	*10.02*
		%Δ **RMSE**	0.22	−0.54	0.25	−1.58	−2.37	−4.23	*18.25*
		%Δ **sMAPE**	0.57	−0.55	0.43	−1.73	−2.77	−5.82	*18.25*
		Win / Tie / Loss	6/7/11	14/5/5	11/0/13	17/1/6	17/1/6	15/2/7	–
		p-value	0.84	**0.01**	0.66	**0.00**	**0.00**	0.06	–
Dutch DAM Prices	*2NN*	%Δ **MAE**	0.43	0.52	0.19	−0.76	−0.94	−1.33	*6.69*
		%Δ **RMSE**	0.28	0.15	−0.05	−0.64	−0.94	−1.13	*9.53*
		%Δ **sMAPE**	0.40	0.56	0.30	−0.72	−0.78	−1.15	*12.96*
		Win / Tie / Loss	6/13/5	7/10/7	6/11/7	16/2/6	14/1/9	16/0/8	–
		p-value	0.38	0.65	0.55	**0.04**	**0.02**	**0.05**	–
	2CNN_NN	%Δ **MAE**	−0.89	−0.02	0.45	−2.27	−0.60	−1.77	*6.90*
		%Δ **RMSE**	−0.55	0.08	0.42	−1.95	−0.79	−2.32	*9.73*
		%Δ **sMAPE**	−0.77	0.25	0.76	−2.35	−0.94	−1.85	*13.36*
		Win / Tie / Loss	9/15/0	4/16/4	5/13/6	17/4/3	16/2/6	19/1/4	–
		p-value	**0.00**	0.53	0.70	**0.00**	**0.03**	**0.00**	–
	ARX	%Δ **MAE**	0.05	−1.07	−0.67	−1.31	−2.18	−1.71	*6.92*
		%Δ **RMSE**	−0.08	−1.14	−0.94	−1.00	−2.52	−2.47	*9.97*
		%Δ **sMAPE**	0.22	−1.00	−0.47	−1.20	−1.98	−1.72	*13.37*
		Win / Tie / Loss	9/13/2	13/8/3	13/5/6	16/2/6	17/0/7	19/0/5	–
		p-value	**0.03**	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	–
Summary		%Δ **MAE**	0.03	−0.27	0.02	−2.23	−2.73	−2.97	*8.18*
		%Δ **RMSE**	−0.14	−0.34	−0.12	−1.85	−2.17	−3.01	*13.39*
		%Δ **sMAPE**	0.16	−0.19	0.12	−2.13	−2.67	−2.83	*15.39*
		Win / Tie / Loss	43/73/28	46/71/27	52/46/46	97/14/33	102/5/37	100/4/40	–
		p-value	0.08	**0.01**	0.27	**0.00**	**0.00**	**0.00**	–

2/6, 2/6, and 2/6 evaluation cases, and improve 16.67% to 58.33% of benchmark RMSEs; producing %Δ RMSEs between −2.14% and 1.74%. Overall, out of the augmentors evaluated, VAEs and WGAN-GPs achieve the greatest forecast error improvements. VAEs produce the highest number of Wins, while WGAN-GPs produce the greatest average %Δ MAE, %Δ RMSE, and %Δ sMAPE improvements. To facilitate further analyses, Belgian, and Dutch ARX, 2NN, and 2CNN_NN augmentation performances are analysed separately below.

To explain the relative underperformance of jittering, scaling, and magnitude-warping, we reiterate that the success of feature space augmentation is highly data-dependent. Broadly, feature space augmentation either increase forecast accuracies by spurring the identification of long-term trends or decrease forecast accuracies by adding too much noise and scrambling any potential trends in time series. Analysing average ARX %Δ RMSE improvements, scaling (−0.84%) is found to perform better than magnitude-warping (−0.35%) and jittering (0.07%). We postulate that scaling, in contrast to jittering, successfully improves linear model performances, because it better regulates the amount of noise added to input time series by transforming both trends and residual errors.

In the case of deep models, it must also be noted that feature space augmentations can increase overfitting by not adding enough noise; extending training times without facilitating the identification of non-linear long-term trends. Analysing average ANN %Δ RMSE improvements, jittering (−0.25%) is found to perform better than magnitude-warping (−0.09%) and scaling (−0.01%). We postulate that, with ANNs, scaling, in contrast to jittering, generally fails to generate sufficiently distinct time series, i.e. add enough noise, to meaningfully facilitate the identification of non-linear long-term trends.

Beyond feature space augmentation, in Table 2 we find overwhelming evidence that model-based augmentors consistently generate meaningful multivariate time series capable of significantly boosting the regression accuracies of both ARXs and ANNs. All model-based augmentors decrease Belgian, and Dutch ARX, 2NN, and 2CNN_NN MAEs, RMSEs, as well as sMAPEs. Moreover, they yield more than 14 (58.33%) Wins across all evaluation cases. Analysing performance differences between Belgian and Dutch evaluation cases, AEs, VAEs, and WGAN-GPs are observed to reduce Belgian forecasting errors more than Dutch errors. Varying feature counts, used in the forecasting of Belgian and Dutch DAM prices, may explain these differences. Because Dutch prices are, on average, forecasted using fewer features, it is reasonable to postulate that less complex ANNs are used to forecast Dutch prices. Elaborating, per Section 1.1, decreased model complexity reduces the need for large training sets and, by extension, the expected forecast error reduction attainable from data augmentation. The impacts of model complexity on augmentation performances are analysed further in Section 4.3.2.

Finally, analysing model-based augmentation performances across ARXs and ANNs, on average, model-based augmentors are observed to reduce ARX MAEs, RMSEs, and sMAPEs by 2.67%, 2.36%, and 2.54%, and ANN MAEs, RMSEs, and sMAPEs by 2.63%, 2.33%, and 2.55% respectively. While similar ARX and ANN improvements are surprising, note that they mostly result from WGAN-GP's Belgian ARX performance. On average, AEs and VAEs achieve higher forecast error reductions with ANNs than ARXs.

4.3.1. Cumulative absolute error differences

To adjust for potential shifts in the target distribution, model refitting is implemented. Describing the refitting process, after forecasting 183 consecutive targets from the test set, historic test set data is

combined with the training and validation data to form a combined training set. Using this set, model-based augmentors are refitted, and a further 183 $\tilde{\imath}$s are generated. After generating additional time series, the evaluation models are refitted as well. In order to analyse the evolution of forecast accuracies and to evaluate the impacts of model refitting, Fig. 10 visualises the evolution of cumulative absolute error differences (CAED) between benchmark and augmentation model forecasts.

Analysing the evolution $\overline{\text{CAED}}$s, Fig. 10 further exposes the performance gulf between feature space and model-based augmentors. While AE, VAE, and WGAN-GP yield $\overline{\text{CAED}}_{183}$s of −27.87, −33.19, and −30.35 respectively, jittering, scaling, and magnitude-warping only yield $\overline{\text{CAED}}_{183}$s of −1.99, −3.76, and −5.52. Moreover, while all model-based augmentors yield a negative $\overline{\text{CAED}}_{365}$, only scaling, buoyed by ARX improvements, results in a negative $\overline{\text{CAED}}_{365}$. As the updating of model weights, on average, negatively impacts ANN scaling $\overline{\text{CAED}}$s, as well as both ARX and ANN jittering and magnitude-warping, we postulate that feature space augmentors must, in general, add a sub-optimal quantity of noise when generating additional $\tilde{\imath}$s.

Further analysing model-based $\overline{\text{CAED}}$s, in Fig. 10 AE, VAE, and WGAN-GP are observed to gradually decrease $\overline{\text{CAED}}$s across the test set. Up until the 183rd evaluation day, VAE outperforms AE and WGAN-GP. After refitting, however, WGAN-GP commences outperforming both AE and VAE augmentors, resulting in the lowest $\overline{\text{CAED}}_{365}$. Note that as a result of higher average target prices, the decrease in $\overline{\text{CAED}}$s marginally intensifies after, roughly, the 225th evaluation day.

4.3.2. The impact of model complexity

In Section 1.1 we introduced the bias–variance trade-off, and formalised how an expansion of the sample size decreases a linear regression's expected generalisation error. In support of the bias–variance trade-off, it can be shown that, provided overfitting is avoided, the magnitude of expected generalisation error improvements increases as model complexity increases. Below we set out to explore whether the above relationship holds for our augmentation methods. For this, ANN model complexity and augmentation performance (%ΔRMSE) are visualised in Fig. 11. Because computing an ANN model's true complexity is impractical, we estimate model complexities using the number of forecast model parameters per input feature (Params/Features). Augmentors which display an average drop in RMSE for both a majority of complex and simple evaluation models, regardless of model architecture, are considered to be successful augmentors. Moreover, augmentors which additionally display a greater average decrease in RMSE with complex models are considered comparable to a real expansion of sample sizes. For the purposes of our analysis, complex evaluation models are defined to be models with a Params/Features ratio greater than or equal to our case study's median ANN Params/Features ratio of 503.

Analysing Fig. 11, scaling and magnitude-warping are observed to perform better with simple models than with complex models; yielding Win/Loss ratios of 12/6 and 16/12, compared to 7/13 and 12/15. Scaling increases both complex 2NN and 2CNN_NN benchmark RMSEs, while magnitude-warping is observed to increase both simple and complex 2CNN_NN RMSEs. Consequently, based on the above results, we consider scaling and magnitude-warping to be unsuccessful augmentors. Evaluating jittering results, jittering improves a majority of both simple and complex benchmark models; yielding Win/Loss ratios of 14/11, and 14/4 respectively. Jittering reduces complex model errors more than simple model errors; on average yielding higher %Δ RMSE reductions. As jittering, however, fails to improve simple 2NN RMSEs, we do not considered to be comparable to the real expansion of sample sizes.

Similarly to jittering, all model-based augmentors reduce a majority of both simple and complex model RMSEs. Specifically, AE, VAE and WGAN-GP yield Win/Loss ratios of 30/13, 33/16 and 34/15 across simple models, and ratios of 34/8, 35/8 and 32/13 across complex models. Unlike jittering, however, model-based augmentors on average improve all simple and complex model 2NN and 2CNN_NN benchmark RMSEs. Moreover all model-based augmentors, apart from the WGAN-GP 2NN, reduce complex model RMSEs more than simple model RMSEs. From the evidence displayed in Fig. 11, model-based augmentation is considered comparable in impact to a real expansion of sample sizes.

4.3.3. The impact of a standard deviation change

In Section 4.1.2 the differences between generated and real time series were analysed. We highlighted that feature space augmentors increase the standard deviations of time series inputs, while model-based augmentors generally decrease their standard deviations. To implicitly evaluate whether the standard deviation of generated time series affects augmentation performance, Fig. 12 visualises the impacts of target standard deviation changes on ARX and ANN %Δ RMSEs. We tranche standard deviation changes into two groups: SD^- ($\%\Delta\text{SD}(Y) < 0$), and SD^+ ($\%\Delta\text{SD}(Y) > 0$), before analysing augmentation performances separately across SD^- and SD^+.

In line with expectations, in Fig. 12 feature space augmentors are observed to improve benchmark regression accuracies more often when $\%\Delta$ SD(Y) increases. In total, feature space augmentors yield a Win / Loss ratio of 110/76 across SD^+ compared to a Win / Loss ratio of 31/25 across SD^-. Jittering, scaling, and magnitude-warping respectively yield Win / Loss ratios of 32/22, 38/20 and 40/34 across SD^+ and Win / Loss ratios of 11/6, 8/7 and 12/12 across SD^-. Despite varying total counts across the two SD regions, a majority of feature space augmentors are observed to individually narrowly improve benchmark model performances across both SD^+ and SD^-.

AE, VAE and WGAN-GP overwhelmingly improve a majority of benchmark model performances across both SD^+, with Win / Loss ratios of 76/25, 81/29 and 80/30, and SD^-, with Win / Loss ratios of 21/8, 21/8 and 20/10 respectively. Unintuitively, given that model-based augmentors generally generate $p_{\tilde{T}}$s with a lower standard deviation, model-based augmentors are observed to improve benchmark regression accuracies marginally more often when $\%\Delta$ SD(Y) increases. Given these results, we find no evidence of an innate relationship between the standard deviations of $p_{\tilde{T}}$s and augmentation performances under varying $\%\Delta$ SD(Y) shifts.

4.3.4. Augmentation performance across the target distribution

Below absolute forecast error changes, between the benchmark and augmentation models, are analysed for targets: below the lower quartile (lower tail), $y < Q1$; inside the interquartile range (IQR), $Q1 \leq y \leq Q3$; and above the upper quartile (upper tail), $Q3 < y$. For each target region, Fig. 13 visualises the mean Win over Loss (%MWOL) as well as the mean absolute error difference (MAED) to highlight how often and by how much, on average, augmentation improves benchmark forecast errors respectively. An augmentor is considered to be robust if on average it improves a majority of errors, yielding a MAED < 0 and a %MWOL > 50%, for all target regions. Augmentors which only reduce forecast errors in the IQR, or IQR and tail are considered comparable to outlier removal techniques or regularisation. Augmentors which robustly reduce lower tail, IQR, and upper tail forecast errors are considered comparable to the introduction of additional real training data. In order to avoid skewing our examination, the impact of refitting, discussed in Section 4.3.1, is filtered out. Only forecasts error changes up to the 183rd day are considered.

Analysing Fig. 13, on average, feature space augmentors are observed to reduce lower tail forecast errors more than IQR errors, and IQR forecasts errors more than upper tail errors. While jittering and magnitude-warping both reduce lower tail forecast errors, only scaling reduces IQR and upper tail forecast errors. To understand the potential reasons for this behaviour, note that the probability of feature space transformations generating $\tilde{t}$ in p_T decreases as the input series t moves further from the mode of p_T. Because the distribution of DAM prices is positively skewed with a long right tail, the probability of feature space augmentors generating meaningful time series in the lower tail or IQR

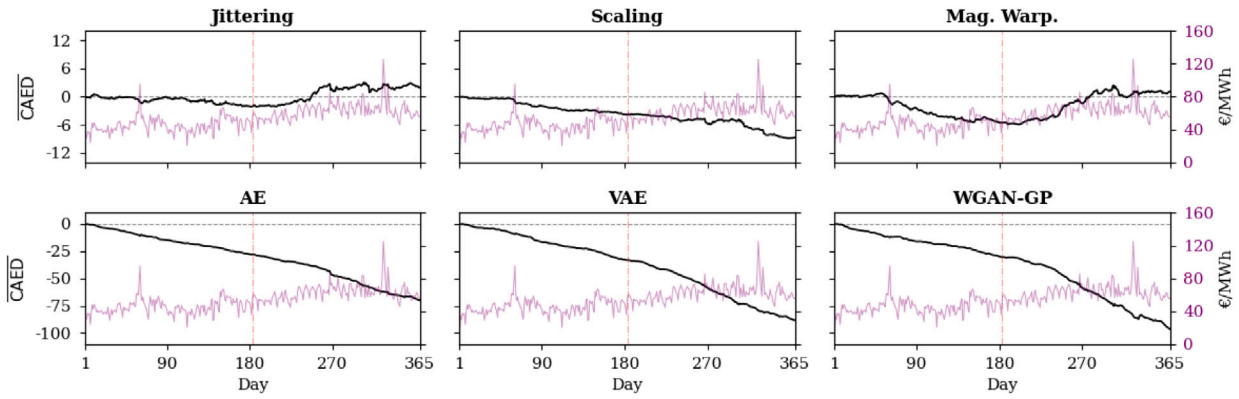


Fig. 10. Visualising the daily $\overline{\text{CAED}}$ evolution across both Belgian and Dutch DAM contracts. The left axes in black track the $\overline{\text{CAED}}$ between augmented and benchmark models. For a single contract, the $\text{CAED}_D = \sum_{d=1}^{D} |e_d^{aug}| - |e_d^{bench}|$, where $|e_d|$ is the absolute forecast error for day d. The $\overline{\text{CAED}}_D$ is calculated by averaging all CAED_Ds across both Belgian and Dutch DAM contracts. The right axes in purple track average DAM prices, averaged across both Belgian and Dutch DAM contracts. Model refitting is highlighted with a dashed red vertical line.

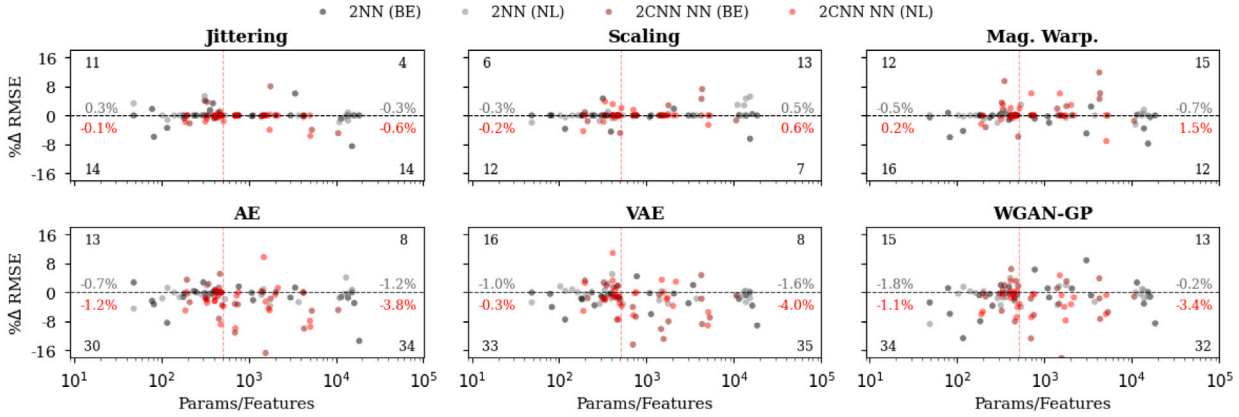


Fig. 11. Visualising how model complexity (Params/Features) impacts ANN augmentation performances (%Δ RMSE) across Belgian (BE) and Dutch (NL) evaluation cases. The red dashed vertical line separates simple models (Params/Features < 503) from complex models (Params/Features ≥ 503). Loss and Win counts are presented in black at the top and bottom of the graphs respectively for both simple and complex models. Additionally, average 2NN %Δ RMSE and average 2CNN NN %Δ RMSE are presented in grey and red respectively for both simple and complex models.

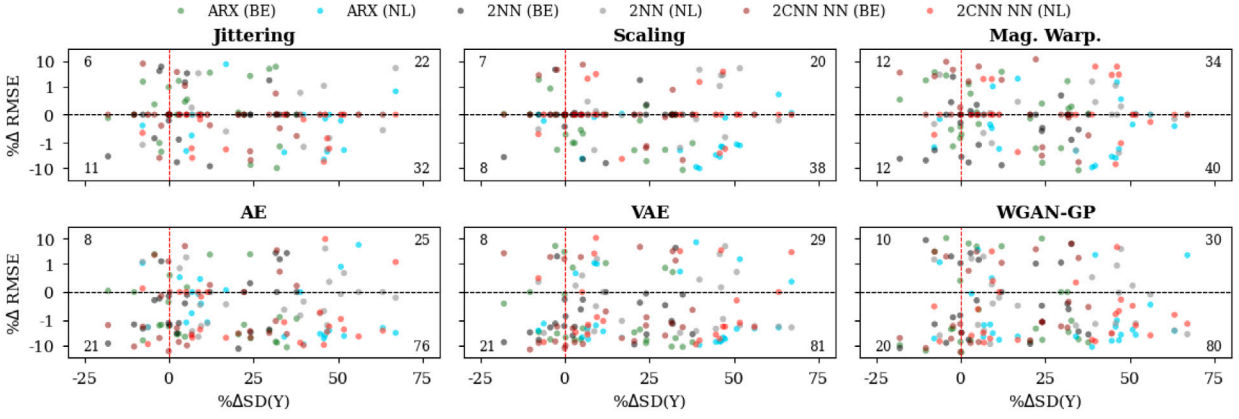


Fig. 12. Visualising how target standard deviation shifts (%Δ SD(Y)) impact augmentation performances (%Δ RMSE) of Belgian (BE) and Dutch (NL) ARXs and ANNs. %ΔSD(Y) is calculated as: $100 \times [(\text{SD}(Y_{test}) - \text{SD}(Y_{train})) \div \text{SD}(Y_{train})]$. The dashed red vertical line at %Δ SD(Y) = 0 divides each graph into two regions: a region of decreasing target variance SD^- on the left side and a region of increasing target variance SD^+ on the right side. Win and Loss counts are presented at the bottom and top of each graph respectively for both regions.

of p_T is, generally, higher than the probability of generating meaningful time series in the upper tail. This underlines potential reasons for varying feature space augmentation performances across the lower tail, IQR, and upper tail.

Similarly to feature space augmentors, AE improves lower tail forecast errors more often than IQR errors, and IQR errors more often than upper tail errors. Unlike feature space augmentors, however, AE achieves robust forecast error reductions across every target region. This is significant because it distinguishes AE performances from

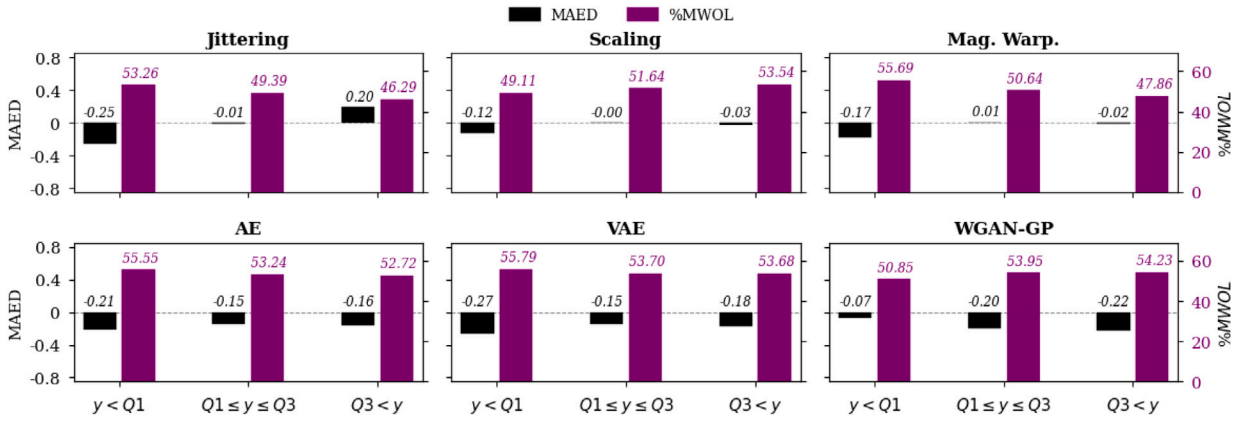


Fig. 13. Breakdowns of forecasts error changes across all evaluated Belgian and Dutch DAM contracts. The MAED is calculated as: $\frac{1}{\eta}\sum\frac{\sum|e_{aug}|-|e_{bench}|}{n}$, where $|e_{aug}|$ and $|e_{bench}|$ are the augmented and benchmark absolute errors respectively, n is the number of forecast errors for a specific DAM contract, and η is the total number of evaluated DAM contracts where augmentation improves benchmark validation RMSEs ($\eta \leq 144$). The %MWOL is calculated as: $\frac{100}{\eta}\sum\frac{n^{Win}}{n^{Win}+n^{Loss}}$, where n^{Win} is the total number of absolute forecast errors that improve ($|e_{aug}| < |e_{bench}|$) and n^{Loss} is the total number absolute forecast errors that worsen.

jittering, scaling, and magnitude-warping, underscoring how/why latent space augmentors generally outperform feature space augmentors. Explaining AE's outperformance, we reiterate that applying transformations in the latent space better ensures that generated time series originate from p_T. Encoding increases the relative volume occupied by T, making transformations in the latent space more likely to generate meaningful $\tilde{t}$s across p_T than feature space transformations.

Improving upon the performance of AE, in Fig. 13 VAE is observed to yield greater %MWOLs and MAED reductions than AE across every target region. Because of the positive skewness of DAM prices and VAE's optimisation of the forward D_{KL}, VAE continues to improve lower tail benchmark performances more than upper tail performances similarly to AE. We postulate that VAE improves upon the overall performance of AE by embracing the modelling and sampling of latent space distributions. Modelling the latent space distribution ensures that less residual noise is added when subsequently generating any time series, resulting in more meaningful $\tilde{t}$s across p_T.

Similarly to AE and VAE, WGAN-GP achieves robust error reductions across every target region. WGAN-GP, however, reduces IQR forecast errors more than AE and VAE. To understand a potential reason why, recall from Section 2.4 that optimising the reverse D_{KL} encourages 'mode seeking' approximations of p_X. While WGAN-GPs optimise a more stable variant of the reverse D_{KL}, WGAN-GPs continue to spur mode/median seeking approximations of p_T more than AEs or VAEs. We postulate that the above underscores why WGAN-GP yields the highest IQR %MWOLs and MAED reductions.

4.3.5. Combined and ensemble augmentation methods

Having identified multiple performance differences between model-based augmentors, below we assess the prediction accuracies of models trained using a combination of $\tilde{T}$s. Combined augmentors such as AE + WGAN-GP, which uses both AE and WGAN-GP $\tilde{T}$s to train evaluation models, are evaluated. Additionally, ensemble augmentors, which average the forecasts from AEs, VAEs, and WGAN-GPs using either simple averaging (AVG) or weighted averaging (Weighted) methods, are evaluated. Results of both combined and ensemble augmentation methods are presented in Table 3. Further, a breakdown of forecast error changes is visualised in Fig. 14.

Analysing the summary results in Table 3, we observe that combined augmentors, utilising two varying sets of $\tilde{T}$s, on average outperform their individual augmentation counterparts. VAE + WGAN-GP, for example, yields higher summary MAE and sMAPE reductions than either VAE or WGAN-GP augmentors. As a general rule, we observe that the greater the sum of individual forecast error changes the greater the combined augmentation forecast error change. VAE + WGAN-GP yields a higher summary sMAPE reduction than AE + WGAN-GP, while AE + WGAN-GP yields a higher sMAPE reduction than AE + VAE. Given that AEs, VAEs, and WGAN-GPs change benchmark summary sMAPEs by −2.13%, −2.67%, and −2.83%, the above ordering is correctly predicted by the sum of individual augmentor sMAPEs; $|-2.67-2.83| > |-2.13-2.83| > |-2.13-2.67|$. Explaining the results, we postulate that combined augmentors, such as VAE + WGAN-GP, potentially outperform individual augmentors because of a logarithmic relationship between model performance and augmented data quantity. Note, however, that the failure AE + VAE + WGAN-GP to, on average, reduce forecast errors more than VAE + WGAN-GP suggests that an upper bound to this relationship must exist.

Examining the performances of ensemble augmentors, in all 6 evaluation cases AVG and Weighted augmentors produce statistically significant RMSE reductions. Evaluated using the number of Wins, AVG and Weighted outperform individual and combined augmentors across all 6 evaluation cases. Additionally, they generally yield higher forecast error improvements than combined augmentors. Most notably, Weighted yields greater summary %ΔMAEs, %ΔRMSEs, and %ΔsMAPEs than any other augmentor.

Analysing breakdowns of forecast error changes, in Fig. 14 all combined augmentors are observed to robustly improve a majority of lower tail, IQR, and upper tail benchmark forecast errors. In comparison to individual augmentors, combined augmentors generally yield marginally smaller lower tail forecast error reductions. However, they produce greater upper tail forecast error improvements. Similarly to the results in Table 3, the performances of combined augmentors are impacted by the performances of their individual component $\tilde{T}$s. Elaborating with an example, we find that combined augmentors utilising WGAN-GP $\tilde{T}$s improve upper tail MAEDs and %MWOLs. Linking these results to the performances of individual augmentors, recall that in Fig. 13 we observed that WGAN-GP yields greater upper tail forecast error improvements than either AE or VAE.

Evaluating ensemble augmentation performances, both AVG and Weighted are observed to also robustly improve a majority of benchmark errors across all three target regions. Compared to individual and combined augmentors, ensemble augmentors yield noticeably higher %MWOLs. While ensemble augmentors generally fail to replicate the IQR and upper tail MAED boosts of combined augmentors, they nevertheless succeed in yielding comparable MAED boosts to AEs, VAEs, and WGAN-GPs. Overall, we conclude that ensemble augmentors offer a less risky path to attaining forecast error reductions than either individual or combined augmentors.

Table 3
Combined and ensemble augmentation results.

			AE + WGAN-GP	AE + VAE	VAE + WGAN-GP	AE + VAE + WGAN-GP	AVG	Weighted
Belgian DAM prices	2NN	%Δ MAE	−3.54	−3.93	−3.95	−3.37	−3.66	−4.14
		%Δ RMSE	−3.51	−3.62	−3.72	−3.16	−3.43	−3.66
		%Δ sMAPE	−3.56	−3.95	−4.38	−3.86	−3.84	−4.28
		Win / Tie / Loss	12/1/11	16/1/7	15/1/8	12/1/11	21/1/2	20/1/3
		p-value	0.40	**0.02**	0.08	0.33	**0.00**	**0.00**
	2CNN_NN	%Δ MAE	−5.56	−5.89	−5.46	−5.36	−4.87	−5.33
		%Δ RMSE	−4.14	−3.72	−3.49	−3.49	−3.32	−3.45
		%Δ sMAPE	−4.21	−4.68	−4.19	−3.62	−4.34	−4.78
		Win / Tie / Loss	17/0/7	16/0/8	14/0/10	17/0/7	18/0/6	18/0/6
		p-value	**0.01**	**0.02**	**0.03**	**0.03**	**0.00**	**0.00**
	ARX	%Δ MAE	−3.85	−2.39	−4.63	−4.11	−4.12	−5.46
		%Δ RMSE	−2.17	−1.73	−2.99	−2.30	−3.04	−3.73
		%Δ sMAPE	−3.71	−2.17	−4.69	−4.04	−4.03	−5.34
		Win / Tie / Loss	18/0/6	18/1/5	17/0/7	19/0/5	20/0/4	20/0/4
		p-value	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Dutch DAM Prices	2NN	%Δ MAE	−1.19	−1.25	−1.24	−1.47	−1.54	−1.79
		%Δ RMSE	−0.98	−0.91	−0.99	−1.22	−1.35	−1.48
		%Δ sMAPE	−1.36	−1.38	−1.46	−1.62	−1.39	−1.58
		Win / Tie / Loss	17/0/7	16/0/8	14/0/10	17/0/7	21/0/3	20/0/4
		p-value	**0.02**	**0.03**	0.06	**0.01**	**0.00**	**0.00**
	2CNN_NN	%Δ MAE	−2.30	−1.73	−2.46	−2.93	−2.46	−2.76
		%Δ RMSE	−2.67	−1.68	−2.59	−3.08	−2.28	−2.48
		%Δ sMAPE	−2.30	−1.88	−2.38	−2.90	−2.49	−2.75
		Win / Tie / Loss	16/1/7	15/1/8	17/0/7	16/0/8	21/0/3	22/0/2
		p-value	**0.00**	**0.02**	**0.00**	**0.00**	**0.00**	**0.00**
	ARX	%Δ MAE	−2.64	−2.20	−2.93	−2.76	−2.65	−3.10
		%Δ RMSE	−2.67	−2.10	−3.51	−2.86	−2.82	−3.18
		%Δ sMAPE	−2.56	−2.05	−2.83	−2.70	−2.49	−2.87
		Win / Tie / Loss	20/0/4	17/0/7	22/0/2	19/0/5	24/0/0	24/0/0
		p-value	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
Summary		%Δ MAE	−3.18	−2.90	−3.44	−3.33	−3.22	−3.76
		%Δ RMSE	−2.69	−2.29	−2.88	−2.68	−2.71	−3.00
		%Δ sMAPE	−2.95	−2.69	−3.32	−3.12	−3.10	−3.60
		Win / Tie / Loss	100/2/42	98/3/43	99/1/44	100/1/43	125/1/18	124/1/19
		p-value	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

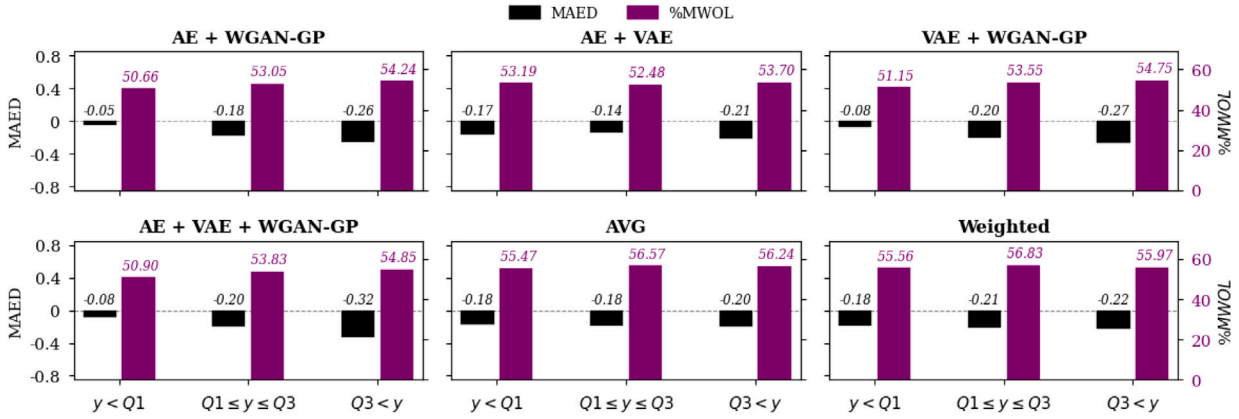


Fig. 14. Breakdowns of forecasts error changes across all evaluated Belgian and Dutch DAM contracts. MAEDs and %MWOLs are calculated per formulas specified in Fig. 13.

5. Conclusion

Our study demonstrates that multivariate time series augmentation methods can significantly boost the regression accuracies of both autoregressive models with exogenous inputs and artificial neural networks. While jittering, scaling, and magnitude-warping generally struggle to improve a majority of forecast errors, AEs, VAEs, and WGAN-GPs are found to significantly reduce a majority of forecast errors; on average reducing benchmark MAEs by 2.23%, 2.73% and 2.97% respectively. Taking every result into consideration, VAEs and WGAN-GPs are found to be our best and most stable individual multivariate time series augmentors, decreasing 70.83% and 69.44% of benchmark errors.

Beyond individual augmentors, we find evidence that utilising combinations of generated time series from multiple model-based augmentors can reduce forecast errors further. VAE+WGAN-GP for instance on average reduces benchmark MAEs by 3.44%: 26.01% more than VAE and 15.82% than WGAN-GP. Similarly, we find evidence that ensemble augmentors, which average the forecasts from AEs, VAEs, and WGAN-GPs, reduce the inherent risks in multivariate time series augmentation. AVG for instance improves 86.81% of benchmark errors: 22.56% more than VAE and 25.01% more than WGAN-GP.

We hope that our work spurs others to research data augmentation methods for multivariate time series regression. We recommend that researchers looking to extend our work consider evaluating variants of our best augmentors, such as adversarial VAEs and Info-GANs, or, given

the successes of combined and ensemble augmentors, explore more sophisticated bootstrapping algorithms and novel averaging methods for ensemble augmentation. Our empirical evaluation of context encoders leads us to believe that GANs could also be used for scenario generation. A myriad of GANs exist, including TransGANs, that could be integrated into our adversarial augmentation algorithm to generate correlated time series. We encourage researchers to consider using context encoders, context-conditional GANs, or conditional WGAN-GPs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Scholt Energy Control for funding and enabling us to conduct this study. Additionally, we thank Prof. Tom Kelsey and Walter van Alst for their invaluable insights and guidance.

All authors approved the version of the manuscript to be published.

Acronyms

AE Autoencoder

ANN Artificial Neural Network

ARX Autoregressive Model with Exogenous Inputs

AVG Averaging Ensemble Augmentation Method

BCE Binary Cross-Entropy Error

BE Belgian

CAED Cumulative Absolute Error Difference

CONV Locally Connected Convolutional Layer

DAM Day-Ahead Electricity Market

FCL Fully Connected Layer

GANs Generative Adversarial Networks

IQR Interquartile Range

LASSO Least Absolute Shrinkage and Selection Operator

MAE Mean Absolute Error

MAED Mean Absolute Error Difference

MSE Mean Squared Error

NL Dutch

PC Principal Component

ReLU Rectified Linear Unit

RF Random Forest

RMSE Root Mean Squared Error

SD Standard Deviation

sMAPE Symmetric Mean Absolute Percentage Error

VAE Variational Autoencoder

VC Vapnik–Chervonenkis

Weighted Weighted Averaging Ensemble Augmentation Method

WGANs Wasserstein GANs

WGAN-GP Wasserstein Generative Adversarial Network with a Gradient Penalty

%MWOL Mean Win over Loss

2NN Two-layer Neural network consisting of two intermediate FCLs

2CNN_NN Three-layer network consisting of two intermediate CONVs, two optional max-pooling or average-pooling layers, and a single intermediate FCL

Appendix. Model architectures and hyperparameters

Bayesian optimisation is used to identify optimal architectures and hyperparameters of every model in our study. Models are trained on the training set and evaluated on the validation set. The hyperparameters are selected based on the best validation score, i.e. the lowest root mean squared error. Note that the augmented data is not used for optimisation. Samples of selected hyperparameters are presented in Tables A.1–A.4. Tables A.1 and A.2 present hyperparameters for sample ARX and ANN evaluation models, introduced in Section 3.4. For ARX, bayesian optimisation is used to select the L1 regularisation parameter: $\tau \sim U\left[0, 1\right]$, the maximum number of training iterations: $\Xi \sim U\left[800, 825, 850, \dots, 1600\right]$, and the optimisation tolerance: $\log(\mathrm{T}) \sim U\left[\log(10^{-5}), \log(10^{-3})\right]$. For ANN, bayesian optimisation is used to identify the ANN hidden layers, the dropout: $\iota \sim U\left[0, 0.25, 0.5\right]$, and the learning rate: $\log(\alpha) \sim U\left[\log(10^{-4}), \log(2 \times 10^{-2})\right]$. Note, in Table A.2 CONVs are represented using tuples specifying kernel height and width, and the number of filter outputs. FCLs are represented using integers specifying neuron count. An average pooling layer, using a one dimensional kernel of size 2 with a stride of 2, is represented using the acronym AvgPool. Explaining the above notation with an example: the NL 06h 2CNN_NN consists of two CONVs, with one dimensional kernels and channel outputs of size 4 and 4, and 8 and 32 respectively, connected to a single FCL with 64 neurons. This network is specified as [(1, 4, 8), (1, 4, 32), 64] in Table A.2. Early stoppage and learning rate scheduling and decay are employed while training ANNs. ANN training continues until either an early stoppage criterion, requiring a minimum training loss improvement of 1e-6 over 22 epochs, is satisfied, or the maximum number of training epochs, 200, is reached. The learning rate is reduced by a factor of 10 after approximately every 50 epochs. Additionally, it is reduced by a factor of 4/3 after 7 epochs of the training loss plateauing.

Table A.3 shows a sample of selected hyperparameters for feature space augmentors, introduced and described in Section 3.5.1. Bayesian optimisation is employed to select $\log(\sigma) \sim U\left[\log(9 \times 10^{-3}), \log(10^{-1})\right]$ and $k \in 4, \dots, 8$.

Finally, Table A.4 presents a sample of optimal architectures for our model-based augmentation methods: AE, VAE, and WGAN-GP, described in Sections 3.5.2–3.5.4 respectively. To combat neuron saturation, leaky ReLU activations, with a negative slope ζ, are applied to intermediate layer outputs of both the generator and critic. For AEs, Bayesian optimisation is used to select: $\zeta \sim U[0,\ 2 \times 10^{-1}]$; $\log(\gamma) \sim U[\log(10^{-4}), \log(10^{-2})]$; and $\log(\tau) \sim U[\log(10^{-5}), \log(5{\times}10^{-3})]$. For VAEs, it is used to select: $\zeta \sim U[0, 2{\times}10^{-1}]$; $\log(\delta) \sim U[\log(10^{-4}), \log(7.5{\times}10^{-3})]$; and $\log(\tau) \sim U[\log(10^{-5}), \log(5 \times 10^{-3})]$. For WGAN-GPs, it is used to select: $\log \alpha_g \sim U[\log(10^{-5}), \log(2 \times 10^{-2})]$; $\log(\alpha_d) \sim U[\log(10^{-5}), \log(2 \times 10^{-2})]$; $\log(\sigma_g) \sim U[\log(10^{-4}), \log(10^{-1})]$; $\sigma_d \sim U[0, 10^{-2}]$; $\zeta \sim U[0, 2 \times 10^{-1}]$, and $\log(\tau) \sim U[\log(10^{-5}), \log(10^{-1})]$. Hyperparameters impacting WGAN-GP training duration are set by trial and error to: $m = 32$; $K = 3000$; $k_{check} = 360$; $k_{end} = 720$; $k_{break} = 600$; and $\psi = 2.5 \times 10^{-1}$.

Table A.1
Optimised ARX model hyperparameters for the Belgian 15h and 17h, and Dutch 06h and 22h.

	ARX (BE 15h)	ARX (BE 17h)	ARX (NL 06h)	ARX (NL 22h)
Regularisation parameter	1.78×10^{-3}	1.70×10^{-3}	1.29×10^{-4}	2.11×10^{-4}
Maximum iterations	1450	1175	925	1425
Tolerance	2.40×10^{-4}	6.42×10^{-4}	1.49×10^{-5}	3.66×10^{-4}

Table A.2
ANN hyperparameters for the Belgian 15h and 17h, and Dutch 06h and 22h.

	2NN (BE 17h)	2CNN_NN (BE 15h)	2NN (NL 22h)	2CNN_NN (NL 06h)
Hidden layers	[500, 500]	[(1, 5, 48), AvgPool, (1, 3, 32), 12]	[97, 43]	[(1, 4, 8), (1, 4, 32), 64]
Dropout	0.25	0.00	0.00	0.00
Learning rate	5.62×10^{-3}	9.07×10^{-4}	1.36×10^{-3}	1.27×10^{-4}

Table A.3
Optimised feature space augmentation hyperparameters for the Belgian 13h and Dutch 17h.

	Jittering	Scaling	Mag. Warp.
2NN (BE 13h)	$\sigma = 1.78 \times 10^{-2}$	$\sigma = 4.52 \times 10^{-2}$	$k = 4$, $\sigma = 3.20 \times 10^{-2}$
2CNN_NN (NL 17h)	$\sigma = 1.01 \times 10^{-2}$	$\sigma = 1.22 \times 10^{-2}$	$k = 6$, $\sigma = 1.51 \times 10^{-2}$
ARX (BE 13h)	$\sigma = 3.67 \times 10^{-2}$	$\sigma = 9.87 \times 10^{-2}$	$k = 7$, $\sigma = 8.98 \times 10^{-2}$

Table A.4
Optimised model-based augmentation network architectures for the Belgian 13h and Dutch 17h and 22h.

	AE		VAE		WGAN-GP	
	Encoding	Decoding	Inference	Generation	Generator	Critic
2NN (BE 13h)	[25, 13, 4]	[9, 17, 24]	[17, 12, 2]	[7, 12, 24]	[27, 24]	[32, 21]
2CNN_NN (NL 17h)	[9, 3]	[9, 17, 15]	[31, 3]	[7, 12, 15]	[52, 31, 15]	[31, 1]
ARX (NL 22h)	[17, 5]	[7, 12, 13]	[31, 3]	[17, 13]	[27, 13]	[27, 13, 1]

References

[1] Abu-Mostafa YS, Magdon-Ismail M, Lin H-T. Learning from data. vol. 4, AMLBook New York, NY, USA; 2012.

[2] Sun C, Shrivastava A, Singh S, Gupta A. Revisiting unreasonable effectiveness of data in deep learning era. In: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, p. 843–52. http://dx.doi.org/10.1109/ICCV.2017.97.

[3] Guennec AL, Malinowski S, Tavenard R. Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD workshop on advanced analytics and learning on temporal data. Riva Del Garda, Italy; 2016.

[4] Um TT, Pfister FM, Pichler D, Endo S, Lang M, Hirche S, et al. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In: Proceedings of the 19th ACM international conference on multimodal interaction. ICMI 2017, New York, NY, USA: ACM; 2017, p. 216–20. http://dx.doi.org/10.1145/3136755.3136817.

[5] DeVries T, Taylor GW. Dataset augmentation in feature space. 2017, pre-print arXiv:1702.05538.

[6] Jorge J, Vieco J, Paredes R, Sánchez J-A, Benedí J-M. Empirical evaluation of variational autoencoders for data augmentation. In: VISIGRAPP. 2018.

[7] Luo Y, Lu B. Eeg data augmentation for emotion recognition using a conditional wasserstein gan. In: 2018 40th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018, p. 2535–8. http://dx.doi.org/10.1109/EMBC.2018.8512865.

[8] Tran T, Pham T, Carneiro G, Palmer L, Reid I. A bayesian data augmentation approach for learning deep models. In: Advances in neural information processing systems. 2017, p. 2797–806.

[9] Bergmeir C, Hyndman R, Benítez J. Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. Int J Forecast 2016;32:303–12. http://dx.doi.org/10.1016/j.ijforecast.2015.07.002.

[10] Smyl S, Kuber K. Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In: 36th International Symposium on Forecasting. 2016.

[11] Hinton GE, Zemel RS. Autoencoders, minimum description length and Helmholtz free energy. In: Cowan JD, Tesauro G, Alspector J, editors. Advances in neural information processing systems. vol. 6, Morgan-Kaufmann; 1994, p. 3–10.

[12] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in neural information processing systems. vol. 27, Curran Associates, Inc.; 2014, p. 2672–80.

[13] Frid-Adar M, Diamant I, Klang E, Amitai M, Goldberger J, Greenspan H. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. Neurocomputing 2018;321:321–31. http://dx.doi.org/10.1016/j.neucom.2018.09.013.

[14] Demir S, Mincev K, Kok K, Paterakis NG. Introducing technical indicators to electricity price forecasting: A feature engineering study for linear, ensemble, and deep machine learning models. Appl Sci 2019;10(1). http://dx.doi.org/10.3390/app10010255.

[15] Lago J, Marcjasz G, De Schutter B, Weron R. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. Appl Energy 2021;293:116983. http://dx.doi.org/10.1016/j.apenergy.2021.116983.

[16] Weron R. Electricity price forecasting: A review of the state-of-the-art with a look into the future. Int J Forecast 2014;30(4):1030–81. http://dx.doi.org/10.1016/j.ijforecast.2014.08.008.

[17] Lago J, Ridder FD, Vrancx P, Schutter BD. Forecasting day-ahead electricity prices in europe: The importance of considering market integration. Appl Energy 2018;211:890–903. http://dx.doi.org/10.1016/j.apenergy.2017.11.098.

[18] Lago J, Ridder FD, Schutter BD. Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. Appl Energy 2018;221:386–405. http://dx.doi.org/10.1016/j.apenergy.2018.02.069.

[19] Uniejewski B, Nowotarski J, Weron R. Automated variable selection and shrinkage for day-ahead electricity price forecasting. Energies 2016;9(8):621. http://dx.doi.org/10.3390/en9080621.

[20] Uniejewski B, Weron R. Efficient forecasting of electricity spot prices with expert and LASSO models. Energies 2018;11:2039. http://dx.doi.org/10.3390/en11082039.

[21] Ugurlu U, Oksuz I, Tas O. Electricity price forecasting using recurrent neural networks. Energies 2018;11(5). http://dx.doi.org/10.3390/en11051255.

[22] Gunduz S, Ugurlu U, Oksuz I. Transfer learning for electricity price forecasting. 2020, pre-print arXiv:2007.03762.

[23] Shi W, Wang Y, Chen Y, Ma J. An effective two-stage electricity price forecasting scheme. Electr Power Syst Res 2021;199:107416. http://dx.doi.org/10.1016/j.epsr.2021.107416.

[24] Yang Z, Ce L, Lian L. Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods. Appl Energy 2017;190:291–305. http://dx.doi.org/10.1016/j.apenergy.2016.12.130.

[25] Zhang J, Tan Z, Wei Y. An adaptive hybrid model for short term electricity price forecasting. Appl Energy 2020;258:114087. http://dx.doi.org/10.1016/j.apenergy.2019.114087.

[26] Elattar EE, Elsayed SK, Farrag TA. Hybrid local general regression neural network and harmony search algorithm for electricity price forecasting. IEEE Access 2021;9:2044–54. http://dx.doi.org/10.1109/ACCESS.2020.3048519.

[27] Goodfellow I, Bengio Y, Courville A. Deep learning. The MIT Press; 2016.

[28] Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning. ICML '08, New York, NY, USA: ACM; 2008, p. 1096–103. http://dx.doi.org/10.1145/1390156.1390294.

[29] Bengio Y, Mesnil G, Dauphin Y, Rifai S. Better mixing via deep representations. In: International conference on machine learning. 2013, p. 552–60.
[30] Kingma DP, Welling M. Auto-encoding variational Bayes. 2013, pre-print arXiv:1312.6114.
[31] Hu Z, Yang Z, Salakhutdinov R, Xing EP. On unifying deep generative models. 2017, pre-print arXiv:1706.00550.
[32] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Proceedings of the 27th International conference on neural information processing systems. NIPS'14, vol. 2, Cambridge, MA, USA: MIT Press; 2014, p. 2672–80.
[33] Arjovsky M, Bottou L. Towards principled methods for training generative adversarial networks. 2017, pre-print arXiv:1701.04862.
[34] Arjovsky M, Chintala S, Bottou L. Wasserstein GAN. 2017, pre-print arXiv:1701.07875.
[35] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of wasserstein gans. In: Advances in neural information processing systems. 2017, p. 5767–77.
[36] ENTSO-e. 2020, available online https://www.entsoe.eu/data/transparency-platform/ [accessed on 16 January 2020].
[37] Scholt energy control, weather data. 2020, available online https://www.scholt.com [accessed on 05 January 2020].
[38] Creswell A, Arulkumaran K, Bharath AA. On denoising autoencoders trained to minimise binary cross-entropy. 2017, pre-print arXiv:1708.08487.
[39] Cherkassky VS, Mulier F. Learning from data: concepts, theory, and methods. first ed.. USA: John Wiley & Sons, Inc.; 1998.
[40] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. Springer series in statistics, New York, NY, USA: Springer New York Inc.; 2001.
[41] Ludwig N, Feuerriegel S, Neumann D. Putting big data analytics to work: Feature selection for forecasting electricity prices using the LASSO and random forests. J Decis Sys 2015;24(1):19–36. http://dx.doi.org/10.1080/12460125.2015.994290.
[42] Breiman L. Random forests. Mach Learn 2001;45(1):5–32. http://dx.doi.org/10.1023/A:1010933404324.
[43] Thornton C, Hutter F, Hoos HH, Leyton-Brown K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2013, p. 847–55.
[44] Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, et al. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In: ICLR. 2017.
[45] Petzka H, Fischer A, Lukovnicov D. On the regularization of wasserstein GANs. 2017, pre-print arXiv:1709.08894.
[46] Isola P, Zhu J-Y, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 1125–34.
[47] Wang C, Xu C, Wang C, Tao D. Perceptual adversarial networks for image-to-image transformation. IEEE Trans Image Process 2018;27(8):4066–79. http://dx.doi.org/10.1109/TIP.2018.2836316.
[48] Zhao JJ, Mathieu M, LeCun Y. Energy-based generative adversarial network. 2016, pre-print arXiv:1609.03126.
[49] Jolliffe I. Principal component analysis. Springer; 2011.
[50] Wilcoxon F. Individual comparisons by ranking methods. Biom Bull 1945;1(6):80–3.
[51] Mann HB, Whitney DR. On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 1947;50–60.