An empirical survey of data augmentation for time series classification with neural networks

Iwana, Brian Kenji
Department of Advanced Information Technology, Kyushu University

Uchida, Seiichi
Department of Advanced Information Technology, Kyushu University

出版情報：PLOS ONE. 16 (7), pp.e0254841-, 2021-07-15. Public Library of Science (PLoS)
バージョン：

PLOS ONE

RESEARCH ARTICLE

An empirical survey of data augmentation for time series classification with neural networks

Brian Kenji Iwana*, Seiichi Uchida

Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan

* iwana@ait.kyushu-u.ac.jp

OPEN ACCESS

Citation: Iwana BK, Uchida S (2021) An empirical survey of data augmentation for time series classification with neural networks. PLoS ONE 16(7): e0254841. https://doi.org/10.1371/journal.pone.0254841

Editor: Friedhelm Schwenker, Ulm University, GERMANY

Received: February 7, 2021

Accepted: July 4, 2021

Published: July 15, 2021

Data Availability Statement: The data used for the experiments is publicly available and can be found at: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ The code is publicly available at: https://github.com/uchidalab/time_series_augmentation.

Funding: This work was supported by MEXT-Japan (Grant No. JP17H06100 and JP21K17808). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

In recent times, deep artificial neural networks have achieved many successes in pattern recognition. Part of this success can be attributed to the reliance on big data to increase generalization. However, in the field of time series recognition, many datasets are often very small. One method of addressing this problem is through the use of data augmentation. In this paper, we survey data augmentation techniques for time series and their application to time series classification with neural networks. We propose a taxonomy and outline the four families in time series data augmentation, including transformation-based methods, pattern mixing, generative models, and decomposition methods. Furthermore, we empirically evaluate 12 time series data augmentation methods on 128 time series classification datasets with six different types of neural networks. Through the results, we are able to analyze the characteristics, advantages and disadvantages, and recommendations of each data augmentation method. This survey aims to help in the selection of time series data augmentation for neural network applications.

1 Introduction

Time series classification attempts to categorize time series into distinct categories, and it is used for a wide range of applications. Some applications include the recognition of signals, biometrics, sequences, sound, trajectories, and more. The challenge of using time series is that time series are structural patterns that are dependent on element order. Note that *time* does not necessarily have to represent actual time and is just used to represent the sequence order.

Traditionally, time series classification was tackled using distance-based methods [1]. However, recently, artificial neural networks have had many successes in time series classification [2, 3]. For example, Recurrent Neural Networks (RNN) [4] have had many recent successes on time series in gait recognition [5, 6], biosignals [7, 8], and online handwriting [9, 10]. Also, recent work has shown that feedforward networks such as Multi-Layer Perceptrons (MLP) and temporal Convolutional Neural Networks (CNN) [11] can also achieve competitive and sometimes better results for time series recognition [3, 12, 13]. Part of the recent successes of neural networks is due to the recent availability of data [2]. Furthermore, it has been shown that increasing the amount of data can help with improving the generalization ability as well as the overall performance of the model [14, 15].

Competing interests: The authors have declared that no competing interests exist.

However, acquiring large amounts of data can be a problem for many time series recognition tasks. For example, the 2018 University of California Riverside (UCR) Time Series Archive [16] is one of the largest repositories of time series datasets, and out of 128 datasets, only 12 have more than a thousand training patterns. This demonstrates that there is a need for time series data.

One solution to acquiring more data is to generate synthetic patterns, i.e., *data augmentation*. Notably, data augmentation is a universal model-independent data side solution. Data augmentation attempts to increase the generalization ability of trained models by reducing overfitting and expanding the decision boundary of the model [17]. The need for generalization is especially important for real-world data and can help networks overcome small datasets [18] or datasets with imbalanced classes [19, 20].

For image recognition, data augmentation is already an established practice. Most of the original proposals of the state-of-the-art Convolutional Neural Network (CNN) [11] architectures used some form of data augmentation. For instance, AlexNet [21], one of the first deep CNNs that set a record benchmark on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset [22], used cropping, mirroring, and color augmentation. Other examples include the original proposal for the Visual Geometry Group (VGG) network [23] which used scaling and cropping, Residual Networks (ResNet) [24] which used scaling, cropping, and color augmentation, DenseNet [25] which used translation and mirroring, and Inception networks [26] which used cropping and mirroring.

While data augmentation is a common practice in image recognition with neural networks, it is not established as a standard procedure for time series recognition [27]. Similar to data augmentation for images, most data augmentation techniques for time series are based on random transformations of the training data. For example, adding random noise [28], slicing or cropping [29], scaling [30], random warping in the time dimension [28, 30], and frequency warping [31]. Examples of random transformation-based methods are shown in Fig 1. The figure shows an example pattern from the OliveOil dataset from the 2018 UCR Time Series Archive with eight random transformation-based data augmentation methods.

The problem with random transformation-based data augmentation is that there is a diverse amount of time series with each having different properties, and not every transformation is applicable to every dataset. For example, jittering (adding noise) assumes that it is normal for the time series patterns of the particular dataset to be noisy. While this might be true for sensor, audio, or Electroencephalogram (EEG) data, this is not necessarily true for time series based on object contours, such as the Adiac and Fish datasets from the 2018 UCR Time Series Archive. These datasets are pseudo-time series taken from the contours of the objects in images. Another example would be domain-specific transformations, such as frequency warping for audio.

An alternative to random transformations is to synthesize time series using information inherent to the data. Some examples of this are pattern mixing, generative models, and pattern decomposition methods. In pattern mixing, two or more existing time series are combined to produce new patterns. The idea is that mixing different existing patterns can create new samples with features from both patterns. Generative models take a less direct route and use the distributions of features in the datasets to generate new patterns. For example, many statistical models such as Gaussian trees [32] and handcrafted mathematical models [33] have been proposed. Another recent generative model for time series generation is the use of neural networks, such as Generative Adversarial Networks (GAN) [34]. Finally, the last family is decomposition methods. Decomposition methods extract features from the dataset, such as trend components [35] and independent components [36], and generate new patterns from those extracted features. The advantage of these families of methods is that they attempt to

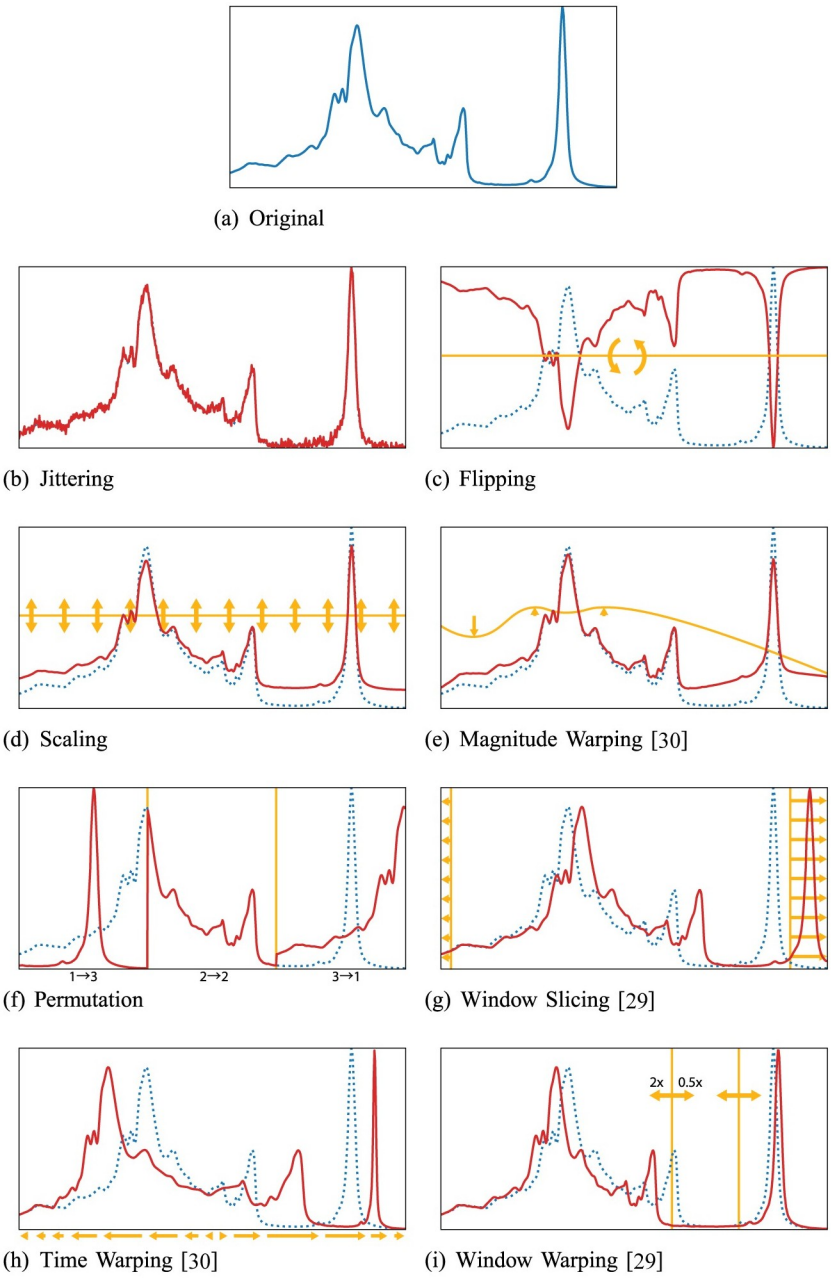

Fig 1. Examples of random transformation-based data augmentation on the OliveOil dataset. The dotted blue lines are the original patterns and the solid red lines are the generated patterns.

https://doi.org/10.1371/journal.pone.0254841.g001

preserve the distribution of time series in the dataset [37], whereas random transformations might unintentionally change the distribution.

A taxonomy of the time series data augmentation methods is shown in Fig 2. The taxonomy breaks down time series data augmentation methods into three primary hierarchical levels, family, domain, and method. The families of data augmentation methods include transformations, pattern mixing, generative models, and decompositions. The families are broken into their respective domains, or major subtypes.

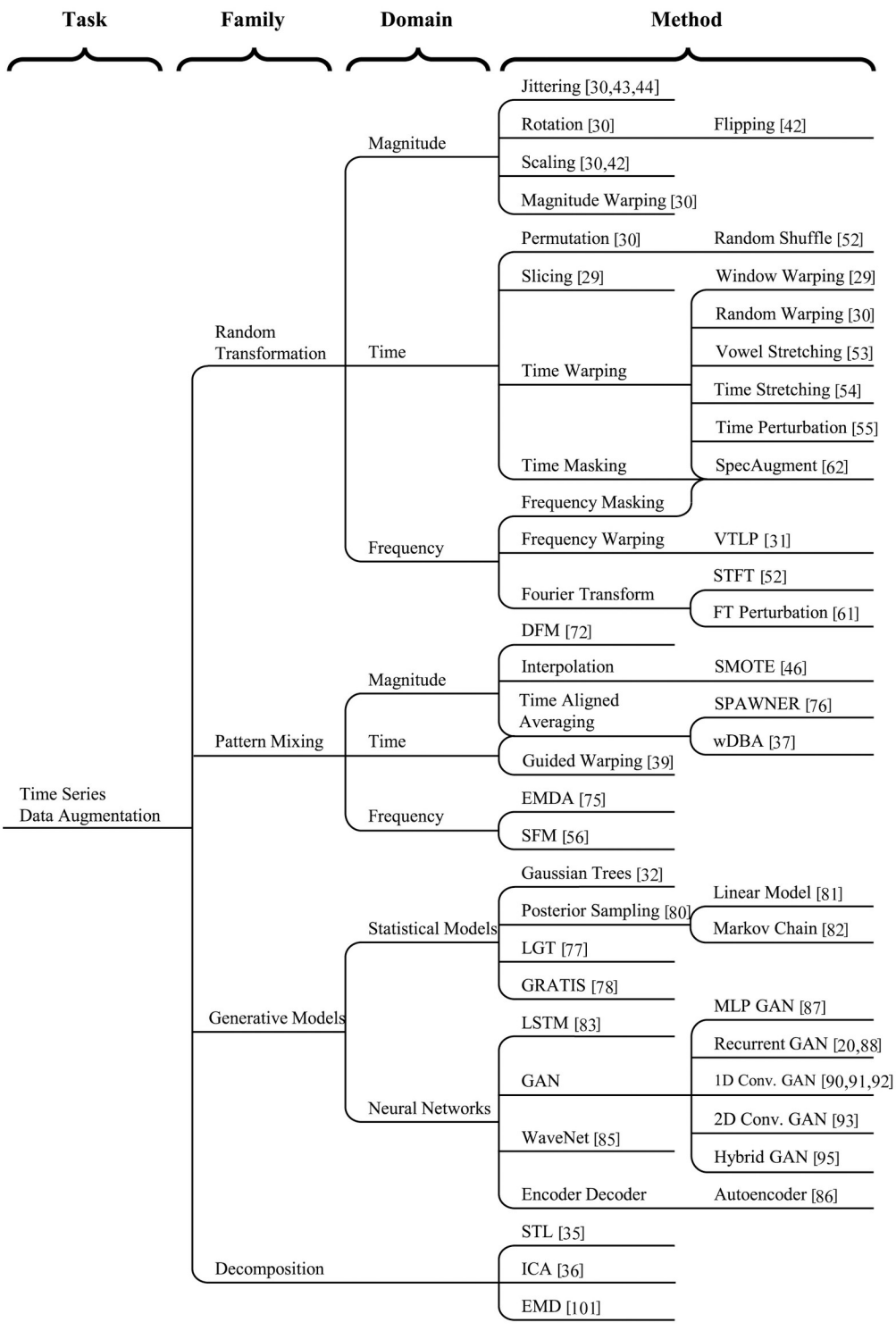

Fig 2. Taxonomy of time series data augmentation.

https://doi.org/10.1371/journal.pone.0254841.g002

The purpose of this paper is to gather and present many data augmentation techniques for time series. In addition, we aim to empirically evaluate some of the techniques using a wide variety of data types. Data augmentation can play an important part in the pattern recognition workflow. Therefore, this is important for progress in the field.

There have been only a few works in the past that compile many data augmentation methods for time series data. In one example, Liu et al. [38] compare jittering, permutation, scaling, and time warping using a Fully Convolutional Network (FCN) and a ResNet. In another, Wen et al. [27] provide a high-level survey of time series augmentation methods. In addition to surveys, only a few works in time series recognition compare many data augmentation methods. These works are typically only performed on a limited amount of datasets [30] or using a limited amount of models [29, 39]. Thus, these works do not extensively evaluate the various data augmentation methods on many different datasets. In our work, we dive deeper into data augmentation methods for time series classification and evaluate more methods on a much larger amount of datasets with multiple neural network models. In addition, we provide analysis, observations, and recommendations from the results, which the previous time series data augmentation survey does not.

The contributions are as follows:

- We review time series data augmentation with a comprehensive taxonomy and categorization. In addition, we thoroughly outline and describe time series data augmentation methods.
- Using time series classification as a target, we perform a thorough comparative evaluation of time series data augmentation methods and demonstrate their effects on a variety of state-of-the-art neural network-based models. The data augmentation methods used for the evaluations include jittering, permutation, flipping, scaling, magnitude warping, time warping, slicing, window warping, SuboPtimAl Warped time series geNEratoR (SPAWNER) [40], weighted Dynamic Time Warping Barycentric Averaging (wDBA) [37], Random Guided Warping (RGW) [39], and Discriminative Guided Warping (DGW) [39]. Each of these methods is compared to no augmentation (or identity), using an MLP, a Long Short-Term Memory (LSTM) recurrent neural network, a Bidirectional Long Short-Term Memory (BLSTM) recurrent neural network, a VGG network, a ResNet, and a LSTM Fully Convolutional Network (LSTM-FCN). We test on all 128 datasets from the 2018 UCR Time Series Archive repository.
- We discuss the aspects of each time series data augmentation method including, the advantages and disadvantages, the characteristics, and suggestions for usage for different dataset types, dataset properties, and with different models.
- Thorough analysis is performed using visualizations of the data augmentation methods, correlation analysis between accuracy and dataset properties, and examination of the comparative evaluations.

The rest of this paper is structured as follows. Sections 2, 3, 4, and 5 detail transformation-based methods, pattern mixing, generative models, and decomposition methods, respectively. The evaluation, results, and discussion are described in Sections 6 and 7. Finally, the conclusion and future work are provided in Section 8.

2 Random transformation-based data augmentation

Many earlier time series data augmentation techniques are borrowed from image data augmentation, such as cropping, flipping, and noise addition. These augmentation methods rely

on random transformations of the training data. Namely, random transformation-based data augmentation generates pattern $\mathbf{x}'$ using some transformation function $g(\cdot)$, or:

$$\mathbf{x}' \leftarrow g(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a reference sequence $\mathbf{x} = x_1, \ldots, x_t, \ldots, x_T$ with T number of time steps from the training set. Each element x_t can be univariate or multivariate.

Transformations on time series can generally be divided into three domains, the magnitude domain, time domain, and frequency domain. Magnitude domain transformations transform the time series along the variate or value axes. Time domain transformations affect the time steps and frequency domain transformations warp the frequencies. There are also hybrid methods that use multiple domains. It should be noted that multiple transformation techniques can be used to augment the data set in serial [30] and in parallel [41, 42]. In the following subsections, we will detail each of these domains and the random transformation-based data augmentation methods associated with them.

2.1 Magnitude domain transformations

Magnitude domain transformation-based data augmentations are transformations that are performed on the values of the time series. The important characteristic of magnitude transformations is that only the values of each element are modified and the time steps are kept constant.

2.1.1 Jittering. One of the simplest, yet effective, transformation-based data augmentation methods is jittering, or the act of adding noise to time series. Jittering can be defined as:

$$\mathbf{x}' = x_1 + \epsilon_1, \ldots, x_t + \epsilon_t, \ldots, x_T + \epsilon_T, \tag{2}$$

where ϵ is typically Gaussian noise added to each time step t and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The standard deviation σ of the added noise is a hyperparameter that needs to be pre-determined. Adding noise to the inputs is a well-known method of increasing the generalization of neural networks [43, 44]. It is able to do this by effectively creating new patterns with the assumption that the unseen test patterns are only different from the training patterns by a factor of noise. In addition, jittering has been shown to help mitigate time series drift for various neural network models [28]. Time series drift when the data distribution changes due to the introduction of new data.

The use of jittering for time series has been most frequently used with sensor data. For example, Rashid and Louis [42] used a combination of jittering with other data augmentation techniques to improve the accuracy of LSTM for sensor data from construction equipment. Um et al. [30] also used jittering with ResNet for wearable sensor data for Parkinson's disease monitoring. However, the effects of jittering seem to be detrimental in the work by Um et al. Another example is Arslan et al. [45], who used a combination of Synthetic Minority Oversampling TEchnique (SMOTE) [46] and Gaussian noise for temperature, light, and air sensor data.

2.1.2 Rotation. Rotation is defined as:

$$\mathbf{x}' = Rx_1, \ldots, Rx_t, \ldots, Rx_T, \tag{3}$$

where R is an element-wise random rotation matrix for angle $\theta \sim \mathcal{N}(0, \sigma^2)$ for multivariate time series [30] and flipping for univariate time series [42]. While rotation data augmentation can create plausible patterns for image recognition, it might not be suitable for time series since rotating a time series can change the class associated with the original sample [47]. This is supported by rotation augmentation being seen to have either no effect or a detrimental effect on time series classification with neural networks [39, 42, 48]. Conversely, Um et al. [30]

found that rotation data augmentation did improve accuracy, especially when combined with other augmentation methods.

2.1.3 Scaling. Scaling changes the global magnitude, or intensity, of a time series by a random scalar value. With scaling parameter α, scaling is a multiplication of α to the entire time series, or:

$$\mathbf{x}' = \alpha x_1, \ldots, \alpha x_t, \ldots, \alpha x_T, \tag{4}$$

The scaling parameter α can be determined by a Gaussian distribution $\alpha \sim \mathcal{N}(1, \sigma^2)$ with σ as a hyperparameter [30], or it can be from a random value from a pre-defined set [42]. It should be noted that "scaling" in terms of time series is different than in the image domain. For time series, it refers to just increasing the magnitude of the elements and not enlarging the time series. Some examples of using scaling as data augmentation include classification of sensor data [30, 42]. Escano et al. [49] and Tran and Choi [50] used a combination of scaling with jittering and element-wise interpolation for Gait recognition.

2.1.4 Magnitude warping. Magnitude warping [30] is a time series specific data augmentation technique that warps a signal's magnitude by a smoothed curve. Namely, augmented time series $\mathbf{x}'$ is:

$$\mathbf{x}' = \alpha_1 x_1, \ldots, \alpha_t x_t, \ldots, \alpha_T x_T, \tag{5}$$

where $\alpha_1, \ldots, \alpha_t, \ldots, \alpha_T$ is a sequence created by interpolating a cubic spline $S(\mathbf{u})$ with knots $\mathbf{u} = u_1, \ldots, u_i, \ldots, u_I$. Each knot u_i is taken from a distribution $\mathcal{N}(1, \sigma^2)$ where the number of knots I and the standard deviation σ are hyperparameters. The idea behind magnitude warping is that small fluctuations in the data can be added by increasing or decreasing random regions in the time series. However, the downsides of magnitude warping for data augmentation is that it still assumes the random transformation is realistic and it depends on two pre-defined hyperparameters (the number of knots I and the standard deviation of the knot height σ) instead of one like many of the other transformation-based methods.

2.2 Time domain transformations

Time domain transformations are similar to magnitude domain transformations except that the transformation happens on the time axis. In other words, the elements of the time series are displaced to different time steps than the original sequence. The following methods are common examples of time domain transformations.

2.2.1 Slicing. Slicing is the time series data augmentation equivalent to cropping for image data augmentation. The general concept behind slicing is that the data is augmented by slicing time steps off the ends of the pattern, or:

$$\mathbf{x}' = x_\varphi, \ldots, x_t, \ldots, x_{W+\varphi} \tag{6}$$

where W is the size of a window and φ is a random integer such that $1 \leq \varphi \leq T - W$. Slicing in this way is also sometimes referred to as Window Slicing (WS) [29] due to the use of a window of size W.

2.2.2 Permutation. Permutation for data augmentation was proposed by Um et al. [30] as a method of rearranging segments of a time series in order to produce a new pattern. It should be noted that permutation does not preserve time dependencies. It can be performed in two ways, with equal sized segments and with variable sized segments [51]. Using permutation with equal sized segments splits the time series into N number of segments of length $\frac{T}{N}$ and permutes them. Using variable size segments uses segments of random sizes.

Random shuffling can be considered as a form of permutation which rearranges individual elements rather than segments. In one example of random shuffling, Eyobu and Han [52] incorporated a shuffling step into their data augmentation workflow of feature extraction, local averaging, shuffling, and local averaging again for sensor data classification with LSTMs.

2.2.3 Time warping. Time warping is the act of perturbing a pattern in the temporal dimension. This can be performed using a smooth warping path [30] or through a randomly located fixed window [29]. When using time warping with a smooth warping path, the augmented time series becomes:

$$\mathbf{x}' = x_{\tau(1)}, \ldots, x_{\tau(t)}, \ldots, x_{\tau(T)}, \tag{7}$$

where $\tau(\cdot)$ is a warping function that warps the time steps based on a smooth curve. The smooth curve is defined by a cubic spline $S(\mathbf{u})$ with knots $\mathbf{u} = u_1, \ldots, u_i, \ldots, u_I$. The height of the knots u_i taken from $u_i \sim \mathcal{N}(1, \sigma^2)$. In this way, the time steps of the series have a smooth transition between stretches and contractions.

Alternatively, a popular method of time warping called window warping was proposed by Le Guennec et al. [29]. Window warping takes a random window of the time series and stretches it by 2 or contracts it by $\frac{1}{2}$. While the multipliers are fixed to $\frac{1}{2}$ and 2, Le Guennec et al. note that they can be modified or optimized to other values.

Similarly, independently developed ideas are vowel stretching [53], dynamic time stretching [54], and time perturbation [55] for speech data augmentation. Vowel stretching targets vowels and extends them by interpolating frames. Time perturbation re-samples the input signal by a randomly selected factor.

2.3 Frequency domain transformations

Frequency domain transformations are transformations that are specific to periodic signals, such as acoustic data. The following are common methods of frequency domain transformations for data augmentation.

2.3.1 Frequency warping. In audio and speech recognition, frequency warping is a popular method of data augmentation [31, 56]. Vocal Tract Length Perturbation (VTLP) [31], for example, is an extension to Vocal Tract Length Normalization (VTLN) [57] that adds variability instead of removing it. In VTLP, frequency f is mapped to a new frequency f' using:

$$f' = \begin{cases} f\omega & f \leq F_{\text{hi}} \frac{\min(\omega,1)}{\omega}, \\ \frac{s}{2} - \frac{\frac{s}{2} - F_{\text{hi}} \min(\omega,1)}{\frac{s}{2} - F_{\text{hi}} \frac{\min(\omega,1)}{\omega}} & \text{otherwise}, \end{cases} \tag{8}$$

where ω is a random warp factor, s is the sampling frequency, and F_{hi} is a boundary frequency. The warping is applied directly to the Mel filter banks. VTLP is a popular data augmentation method and has been used for many audio applications, such as vocal tract shape conversion [58] and acoustic modeling [56, 59]. It has also been extended by using it in an end-to-end recognition framework [60].

2.3.2 Fourier transform-based methods. There have also been data augmentation methods that augment by manipulating the data under a Fourier transform. Gao et al. [61] proposed utilizing amplitude and phase perturbations in order to augment in the frequency domain. This is done by adding Gaussian noise to the amplitude and phase spectra found by a discrete Fourier transform. In another example, Eyobu and Han [52] use Short-Time Fourier transform (STFT) features as one of the features augmented using their method.

2.3.3 Spectrogram augmentation. Normally, frequency warping data augmentation is performed before conversion into a spectrogram. However, recently, a method called SpecAugment [62] was proposed that augments the spectrogram data itself. In order to augment the data, SpecAugment performs three key operations on the spectrogram: time warping, frequency masking, and time masking. SpecAugment's time warping works much like the window warping method, except with random duration. Frequency masking and time masking mask the spectrograms in their respective domains. In this way, SpecAugment is both a time domain and frequency domain augmentation method.

3 Pattern mixing

Pattern mixing combines one or more patterns to generate new ones. For random transformations, there is an assumption that the results of the transformations are typical of the dataset. However, not every transformation is applicable to every dataset. The benefit of pattern mixing is that it does not make this same assumption. Instead, pattern mixing assumes that similar patterns can be combined and have reasonable results.

3.1 Magnitude domain mixing

The most direct application of pattern mixing is to linearly combine the patterns at each time step. This is the idea behind magnitude domain mixing.

3.1.1 Averaging and interpolation. It is possible to create new patterns by simply averaging two patterns. In general, the reference patterns are selected randomly from the same class or selected using nearest neighbors. Interpolation extends averaging to more points between the two patterns instead of just the midpoint.

One famous interpolation method is called SMOTE [46]. SMOTE was designed to combat data with imbalanced classes by interpolating patterns from under-represented classes. In SMOTE, a random sample $\mathbf{x}$ is selected from the under-represented class and another random sample $\mathbf{x}_{\text{NN}}$ is selected from the reference sample's k-nearest neighbors. Next, the difference between the two samples is multiplied by a random value λ in a range of $\{0, 1\}$. The result is a pattern between the two original patterns, or:

$$\mathbf{x}' = \mathbf{x} + \lambda|\mathbf{x} - \mathbf{x}_{\text{NN}}|. \tag{9}$$

SMOTE has been shown to perform well in many time series applications, such as sensor data [45], gene sequences [63], high-dimensional data [19]. There have also been a number of improvements on SMOTE, such as Safe-Level-SMOTE [64], SMOTE based on the furthest neighbor (SMOTEFUNA) [65], Cost Minimization Oriented SMOTE (CMO-SMOTE) [66], Density-Based SMOTE (DBSMOTE) [67], etc. SMOTE has also been used in a feature space modeled by an Echo state network (ESN) [68].

In addition to SMOTE, there have been other proposals of interpolation for data augmentation. For example, Sawicki and Zielinski [69] used interpolation in combination with LSTMs on sensor data. In addition, an interpolation method similar to SMOTE was extended by DeVries and Taylor [70] to extrapolation by allowing λ in Eq (9) to be $\{0, \infty\}$.

3.1.2 Deviation from the mean. Yeomans et al. [71] proposed a method of time series data augmentation using the deviation from the mean (DFM). To simulate new time series, they use the following process. First, the signals are smoothed with a Savitzky-Golay filter [72] and an offset is used to ensure that all values are greater than 0. Next, the bounding curves of the smoothed signals for each class are calculated. A mean curve is then calculated using the bounding curves and the DFM for each pattern is determined based on the difference between the pattern and the mean curve of its class. Random segments of DFMs from multiple patterns

are then combined to create a surrogate DFM curve. Finally, the surrogate DFM is multiplied with the class mean curve to create new simulated patterns.

3.2 Time domain mixing

Guided warping [39] combines time series by time warping a reference pattern by a teacher pattern using Dynamic Time Warping (DTW) [73]. DTW is a method of measuring the distance between time series using elastic element matching found by dynamic programming. Guided warping uses the dynamic alignment function of DTW to warp the elements of a reference pattern to the elements of a teacher pattern. In this way, the reference pattern is set to the time steps of the teacher pattern. This is different from averaging in that the mixing happens only in the time domain. There are two variants, Random Guided Warping (RGW) which uses a random intra-class teacher and Discriminative Guided Warping (RGW) which uses a directed discriminative teacher [39].

3.3 Frequency domain mixing

Pattern mixing can also be performed in the frequency domain. Takahashi et al. [74] proposed a method called Equalized Mixture Data Augmentation (EMDA), which mixes two sounds of the same class with randomly selected timings. In addition to mixing sounds, EMDA perturbs the sound by boosting or attenuating particular frequency bands. In another example, Stochastic Feature Mapping (SFM) [56] converts one speaker's speech data to another speaker by mapping features using an acoustic model. Due to the nature of using the frequency domain, data augmentations in this area are generally used for sound recognition, e.g., EMDA has been used for acoustic event detection [74] and animal audio classification [75] and SFM has been used for speech [56].

3.4 Mixing in multiple domains

There are also methods that use pattern mixing across multiple domains. For instance, the following methods mix the patterns in multiple domains.

3.4.1 Suboptimal element alignment averaging. SuboPtimAl Warped time series geNEratoR (SPAWNER) [40] was introduced by Kamycki et al. as a method of generating patterns through a novel method called suboptimal time warping. Suboptimal time warping uses the warping ability of DTW but adding an additional constraint that forces the warping path through a random point. By using the suboptimal time warping, SPAWNER is able to create an almost unlimited number of new time series by averaging aligned patterns.

3.4.2 Barycentric averaging. DTW Barycentric Averaging (DBA) [76] is a method of averaging multiple discrete time series by finding the center of time aligned elements. It does this through an iterative process of adding sample patterns that are time aligned by DTW to a cumulative centroid pattern. The advantage of using DBA over linear averaging is that the underlying pattern is preserved, whereas linear averaging might smooth features (for example, linear averaging time series that are just offset in time would lose distinct features).

For data augmentation, Forestier et al. [37] proposed a weighted version of DBA (wDBA). They propose three weighting schemes, Average All (AA), Average Selected (AS), and Average Selected with Distance (ASD). AA weights all of the time series in a class input into wDBA by a flat Dirichlet distribution. AS selects a reference time series and weights two of the five nearest neighbors by a large constant amount and all the rest by a small constant amount. ASD is similar to AS except that it weights based on the distance to the reference. Fawaz et al. [47] also used wDBA AS with a ResNet for time series classification.

4 Generative models

Instead of using random transformations or mixing patterns, it is possible to sample time series from feature distributions using generative models. We classify generative models into two categories, statistical models and neural network-based models.

4.1 Statistical generative models

There are a wide variety of statistical, mathematical, or stochastical models used for time series generation and augmentation. Typically, these augmentation methods build a statistical model of the data and are often used in forecasting. For example, the Local and Global Trend (LGT) [77] is a time series forecasting model that uses nonlinear global trends and reduced local linear trends to model the data. LGT-based data augmentation has been shown to improve forecasting results with LSTMs [77]. In Cao et al. [32], a mixture of Gaussian trees was used to oversample imbalanced classes for time series classification. GeneRAting TIme Series (GRATIS) [78] was recently introduced, and it uses mixture autoregressive (MAR) models in order to simulate time series. GRATIS can be used to generate non-Gaussian and nonlinear time series by modeling with MAR and adjusting the parameters. There are also works that use the posterior sampling technique proposed by Tanner and Wong [79], such as a dynamic linear model with a hyperparameter approximated from marginal probability functions [80] and posterior sampling a Markov chain Monte Carlo [81] model.

4.2 Neural network-based generative models

Generative models based on neural networks have become popular in recent times. However, while many generative networks have been proposed, not every model was used for data augmentation. In this section, we will discuss the networks that have specifically been used for data augmentation.

The most basic application of neural networks for time series generation is direct sequence-to-sequence networks such as LSTMs and temporal CNNs. This technique is especially useful for natural language processing (NLP) tasks. For example, Hou et al. [82] tackle language understanding and demonstrate the effectiveness of an LSTM-based input-feeding neural machine translation (NMT) model with attention in generating sentences for data augmentation. Longpre et al. [83] use a back-translation data augmentation strategy with sequence-to-sequence networks for the question-answer task. Another example is the use of WaveNet [84], a speech generation network that uses dilated causal convolutions, which has been used for data augmentation [85].

4.2.1 Encoder-decoder networks. Encoder-decoder networks take a high-dimensional or structural input, encode it into a latent space lower-dimensional vector, and then decode it back to a high-dimensional or structural output. Data augmentation methods generate new patterns by decoding vectors sampled from the latent space. In one example, an LSTM-based autoencoder (LSTM-AE) [86] was used to generate data for a classification LSTM on skeleton-based human action recognition. However, the results were mixed when comparing the results of data augmentation using the LSTM-AE, rotation, scaling, and no augmentation. On one dataset, the data augmentation from LSTM-AE had the best results, but on the two others, no augmentation was better. In another example, DeVries and Taylor [70] combined samples generated from an LSTM-based variational autoencoders (VAE) and combined it with interpolation and extrapolation to augment time series.

4.2.2 Generative adversarial networks. GANs [34] are a class of generative networks that use adversarial training to jointly optimize two neural networks, a generator and a discriminator. Similar to encoder-decoder networks, in order to generate samples, the GAN is trained

using the training dataset and then z-vector is sampled and used with the generator to create new time series. There have been numerous time series GANs proposed. However, most target generation only and not data augmentation. Due to this, we will only focus on the works that are specifically used for data augmentation.

The underlying networks of GANs for time series can be roughly separated into four architectures, GANs based on fully-connected networks or MLPs, recurrent GANs that use RNNs, GANs with temporal CNNs or 1D CNNs, and GANs that generate spectrum based images with 2D CNNs. Lou et al. [87] is an example of a GAN built on a fully-connected network. They combine an autoencoder network with a Wasserstein GAN (WGAN) in order to augment time series regression data.

Some examples of recurrent GANs include [88] and [20]. In the former, Harada et al. [88] use a straightforward implementation of a deep LSTM-based GAN for the data augmentation of ECG and EEG signals. They found an improvement in accuracy when compared to noise addition, interpolation, and generation with a Hidden Markov Model (HMM). Similar results were found in [20] for recognizing network traffic and [89] for synthetic and medical time series.

Temporal Convolutional GANs (T-CGAN) [90] are GANs that use 1D convolutional layers for time series generation. Electronic Health Records GAN (ehrGAN) [91] is another 1D convolutional GAN that differs from a T-CGAN in that it incorporates an encoder with variational contrastive divergence in the generator. Chen et al. [92] proposed EmotionalGAN which is also based on 1D CNNs for classifying emotions from long ECG patterns. They found significant improvements when augmenting data for Random Forests and Support Vector Machines (SVM). However, Hatamian et al. [93] had mixed results when comparing a 1D convolutional GAN to a Gaussian Mixture Model (GMM) on ECG data. There are also spectrogram-based augmentation methods there use 2D convolutional layers in their GAN, such as WaveGAN [94].

Finally, there are hybrid GAN models, such as the BLSTM-CNN GAN proposed by Zhu et al. [95]. In Zhu et al., they found that their hybrid BLSTM-CNN performed better than other LSTM-based GANs.

Conditional GANs (cGAN) [96] have also been used for data augmentation. cGANs are GANs that are provided a condition, or parameter, to the generator and discriminator in order to control the generated patterns. Nikolaidis et al. [97] showed modest improvements on ECG data using a recurrent cGAN with an MLP, random forest, k-nearest neighbor, and SVM. Similar results were found in [98]. However, it is not clear if the use of cGANs is better for data augmentation because Sheng et al. [99] compared a traditional GAN and a cGAN on speech recognition and found that the traditional GAN performed better on average. Wang et al. [100] also proposed a selective WGAN (sWGAN) and selective VAE (sVAE) that showed better performance than a standard conditional WGAN (cWGAN).

5 Time series decomposition

Decomposition methods generally decompose time series signals by extracting features or underlying patterns. These features can either be used independently, recombined, or perturbed for generating new data for augmentation. Empirical Mode Decomposition (EMD) [101] is a method of decomposing nonlinear and non-stationary signals. EMD has shown to improve classification by using it as a decomposition method for data augmentation of noisy automobile sensor data in a CNN-LSTM [102]. Another example of a decomposition method used for data augmentation was proposed in [36], where the use of Independent Component Analysis (ICA) [103] was combined with a Dynamical-Functional Artificial Neural Network

(D-FANN) for filling gaps in time series. This work assumes that the observed signals are generated from independent sources and estimates the mixture using ICA. Using the transformed space from ICA, D-FANN is used for each component and then transformed back into the signal. Using this technique, Eltoft was able to increase the performance of an MLP.

There have also been methods that used Seasonal and Trend decomposition using Loess (STL) [104]. STL is traditionally used to decompose signals into seasonal, trend, and remainder components. Bergmeir et al. [35] exploited STL by decomposing the signal into these components and bootstrapping the remainder using a moving block bootstrap. They then assembled a new signal using the bootstrapped remainder. Robust Time series Anomaly Detection (RobustTAD) [61] also used a version of STL, namely RobustSTL [105]. In RobustTAD, the signals are decomposed and augmented using a variety of time and frequency domain transformations in order to augment data for anomaly detection.

6 Comparative evaluations

In this section, we describe the comparative evaluations performed using the data augmentation methods. The purpose of the evaluations is to empirically compare data augmentation methods on a variety of neural network models and time series datasets.

6.1 Datasets

We used all of the datasets in the 2018 UCR Time Series Archive [16]. Information about each of the datasets is outlined in S1 Table. In total, 128 datasets are used, including 9 device, 6 ECG, 2 electrooculography (EOG), 2 electrical penetration graph (EPG), 3 hemodynamics (Hemo), 1 High-Resolution Melting (HRM), 32 object contours from images, 17 motion, 1 power, 30 sensor, 8 simulated, 8 spectro, 4 spectrum, 2 traffic, and 3 trajectory time series. The datasets have fixed training sets and test sets.

The time series are rescaled so that the smallest value in the training set is -1 and the largest value in the training set is 1. In addition, datasets that vary in length are zero-padded for batch training and missing values are replaced with zeroes.

6.2 Evaluated network models

In order to evaluate the data augmentation methods, we used a variety of neural network models. This includes a 1D VGG, 1D ResNet, MLP, LSTM, BLSTM, and LSTM-FCN, as shown in Fig 3. These networks were chosen due to being state-of-the-art in time series recognition as well as representing a wide range of models with different attributes.

Each evaluation used hyperparameters taken from state-of-the-art time series classification models found in their respective literature. In addition, each evaluation was trained with their recommended optimizer, learning rate, and batch size. The only exception is that they all were trained for the same 10,000 iterations, had a learning rate reduction of 0.1 on plateaus of 500 iterations, and that the training dataset was augmented with four times the number of patterns of the original dataset. The number of iterations was kept constant with no early stopping across all of the datasets in order to give each model the same opportunity for training so that intra-model comparison between data augmentation methods is more consistent. A total of 10,000 iterations was chosen because the training loss for all of the models on each of the datasets tended to converge before then. Also, since the number of iterations is kept constant, the control experiment with no augmentation is the same as repeating identical patterns to match the augmented experiments. In addition, it should be noted that the hyperparameters of the models have little tuning due to the inter-model accuracy being not as important as the intra-model accuracy for the purposes of data augmentation comparison. However, the

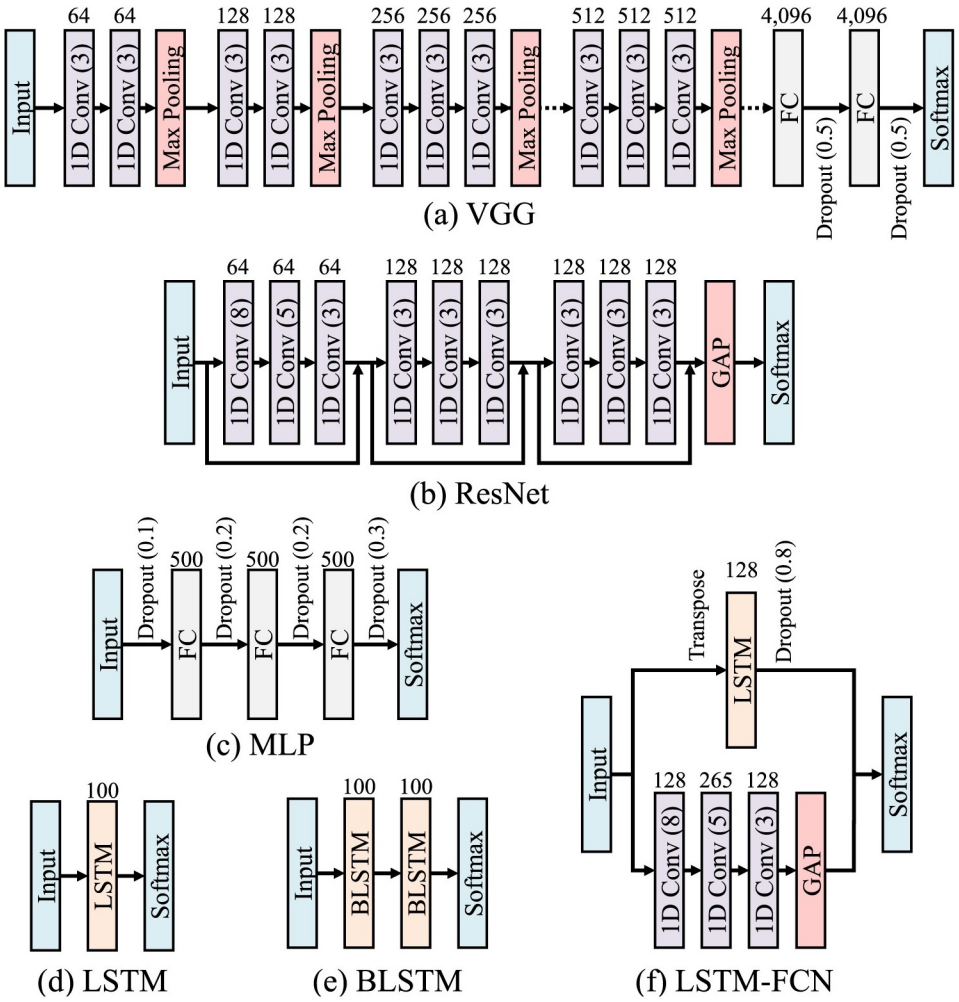

Fig 3. The architectures of the neural networks used to evaluate the data augmentation methods.

https://doi.org/10.1371/journal.pone.0254841.g003

hyperparameters used in the experiments were chosen based on a good faith attempt to find the best hyperparameters found in literature. The details of the hyperparameters and the training scheme are described as follows.

6.2.1 1D Visual Geometry Group (VGG) network. Temporal CNNs are CNNs [11] that use 1D convolutions instead of 2D convolutions that are traditionally used for image recognition. Thus, we adapted a VGG [23] for time series through the use of 1D convolutions. We use the hyperparameters of the original VGG as shown in Fig 3(a). However, because the datasets in the 2018 UCR Time Series Archive come in a wide variety of sequence sizes, the standard VGG is not appropriate due to the possibility of excessive pooling. Thus, we use a modified version that contains different numbers of convolutional and max pooling blocks depending on the input length [39]. The number of blocks B used is:

$$B = \text{round}(\log_2(T)) - 3, \tag{10}$$

where T is the number of time steps. This keeps the output of the final max pooling to be between 5 and 12 time steps [13]. Similar to the original VGG, the first two blocks of convolutional layers have two consecutive convolutional layers of 64 and 128 filters, respectively. Every

block thereafter contains three consecutive convolutional layers. 256 filters are used for the third block and every subsequent block contains 512 filters. Each block is followed by max pooling with filter size 2 at stride 2. The convolutional layers all use ReLU activation functions, filter length of 3 at stride 1, and initialized using the uniform variance proposed by He et al. [106]. Furthermore, after the final max pooling layer, there are two fully-connected hidden layers with 4,096 nodes each, ReLU, and dropout with a probability of 0.5. For training, the VGG evaluation was trained using batch size 256 with Stochastic Gradient Descent (SGD) with a learning rate of 0.01, weight decay of 5×10^{-4}, and momentum of 0.9.

6.2.2 1D Residual Network (ResNet). A ResNet is a deep CNN that uses residual connections between blocks [24]. The hyperparameters for the 1D ResNet used in the evaluation were taken from Fawaz et al. [47], who saw improvements using data augmentation for time series. As shown in Fig 3(b), this version deviates from the original proposal of ResNet [24] by containing only three residual blocks with varying filter lengths and no max pooling. The first block uses three layers of 64 1D convolutions, the second block has three layers of 128 1D convolutions, and the third block has three layers of 128 convolutions. Each residual block contains three convolutional layers with filter size 8, 5, and 3. Each convolution is followed by batch normalization [107] and ReLU activation functions. In addition, the residual connection connects the input of each residual block to the input of the next block using an addition operation. The last two layers of the ResNet include a Global Average Pooling (GAP) layer and an output layer with softmax. All of the network parameters for the ResNet were initialized using Glorot's Uniform initialization [108]. Finally, the network was trained using Adam optimizer [109] with the initial learning rate set to 0.001 and the exponential decay rates of the first and second momentum estimates set to 0.9 and 0.999 respectively, as per [47].

6.2.3 Multi Layer Perceptron (MLP). For this evaluation, we used a fully-connected MLP network. While not traditionally used for time series, MLPs have shown [3] to be as effective as time series specific models. This network was selected as an evaluated model to represent a neural network that does not take structural relationships into consideration. In order to use an MLP for time series, the time series is flattened so that each time step is one element in the input vector. The version of MLP that was used was the MLP proposed by Wang et al. [3] and is shown in Fig 3(c). The network is constructed of three hidden layers with 500 nodes each, rectified linear unit (ReLU) activations, and an output layer with softmax. Dropout is used between each layer with a rate of 0.1 after the input layer, 0.2 between the hidden layers, and 0.3 before the output layer. As suggested by Wang et al., the MLP is optimized using Adadelta [110] with a learning rate of 0.1, $\rho = 0.95$, and $\epsilon = 10^{-8}$. All datasets and data augmentation methods were trained with a batch size of 256.

6.2.4 Long Short-Term Memory (LSTM). RNN-based models use memory states with recurrent connections to address the difficulties with time series recognition. Thus, using RNN-based models is useful to assess the effects of the data augmentation techniques on time distortion invariant models. LSTMs [111] are one of the most commonly used RNNs. The hyperparameters selected for the experiments were determined by the LSTM hyperparameter survey in [112]. Namely, as shown in Fig 3(d), the LSTM has one LSTM layer with 100 units. Both models were trained using Nesterov Momentum Adam (Nadam) [113] optimizers with an initial learning rate of 0.001 and batch size of 32.

6.2.5 Bidirectional Long Short-Term Memory (BLSTM). BLSTMs [114] are a bidirectional variant of LSTMs that use a forward and backward recurrent connection. The idea of using both a standard LSTM and a bidirectional one is to observe the differences that having forward and backward recurrent connections has on different data augmentation methods. The hyperparameters and training of the BLSTM are identical to the LSTM, except that the

BLSTM had two layers. In addition, the BLSTM uses concatenation as the output merging method.

6.2.6 Long Short-Term Memory Fully Convolutional Network (LSTM-FCN). The final model for the data augmentation evaluations is using the hybrid of LSTM and CNN referred to as an LSTM-FCN [115]. An LSTM-FCN is a two-stream network that combines a fully convolutional stream and a recurrent stream. The LSTM-FCN and hyperparameters are shown in Fig 3(f). The convolutional stream has three 1D convolutional layers, each with batch normalization and ReLU. The convolutional filters are initialized using He's Uniform. For the recurrent stream, there is one LSTM layer with 128 units and an aggressive dropout rate of 0.8. It should be noted that the input of the LSTM layer is transposed so that the LSTM receives a T dimensional vector that is one time step long. While this is a non-standard use of LSTM, it has shown to be more effective than the standard use [116]. The output of the LSTM is concatenated with the output of GAP from the convolutional stream. For training, as suggested, Adam optimizer is used with batches of 128 and an initial learning rate of 0.001.

6.3 Evaluated data augmentation methods

The following time series data augmentation methods are evaluated. These data augmentation methods were selected as the methods that fell under two criteria. First, the evaluated data augmentation methods are general methods that can be used with any time series. For example, we do not use any frequency domain methods due to them not being applicable to non-periodic time series. Second, we did not select data augmentation methods that required external training, such as the generative models.

- **None**: "None" refers to no augmentation, which is the control experiment against which the data augmentation methods can be compared.
- **Jittering**: For Jittering, Gaussian noise with a mean $\mu = 0$ and standard deviation $\sigma = 0.03$ is added to the time series, as suggested by Um et al. [30]
- **Rotation**: Because the 2018 UCR Time Series Archive contains univariate time series, we use flipping as the Rotation data augmentation method. To do so, 50% of the patterns of the training set are inverted at random for each data augmentation set [42].
- **Scaling**: Scaling multiplies training set time series with random scalars from a Gaussian distribution with $\mu = 1$ and $\sigma = 0.2$ [30]. In this way, the time series are scaled by a single multiplier for all time steps.
- **Magnitude Warping**: For the magnitude warping evaluation, we use the augmentation method proposed by [30]. In this method, the magnitudes of the time series are multiplied by a warping amount determined by a cubic spline line with four knots at random locations and magnitudes. The knots have peaks or valleys with $\mu = 1$ and $\sigma = 0.2$.
- **Permutation**: For this implementation, we use permutation with two to five equal sized segments [30].
- **Slicing**: Slicing, specifically window slicing [29], crops the time series to 90% of the original length. The starting point of the window slice is chosen at random. Also, in order to be directly compared to the other data augmentation methods, we interpolate the time series back to the original length.
- **Time Warping**: The warping path is defined by a smooth cubic spline-based curve with four knots. The knots have random magnitudes with $\mu = 1$ and a $\sigma = 0.2$ [30].

- **Window Warping**: As outlined in [29], our Window Warping implementation selects a random window, that is 10% of the original time series length and warps the time dimension by 0.5 times or 2 times.
- **SuboPtimAl Warped time series geNEratoR (SPAWNER)**: SPAWNER [40] is a pattern mixing data augmentation method that "suboptimally" averages two intra-class randomly selected patterns. We use the standard symmetric slope constraint for DTW with a warping path boundary constraint of 10% of the time series length. In addition, SPAWNER adds noise with $\mu = 0$ and $\sigma = 0.5$ in order to further transform the data.
- **Weighted DTW Barycentric Averaging (wDBA)**: wDBA exploits traditional DBA for the use of data augmentation by weighting patterns in the average. We use the Average Selected with Distance (ASD) version due to it showing the best results [37]. A symmetric slope constraint is used as DTW's slope constraint and 10% of the length is used as the warping window.
- **Random Guided Warping (RGW)**: RGW selects random intra-class patterns and warps the elements of one pattern to the time steps of the other. As in [39], we use the RGW-D version which uses standard DTW with a symmetric slope constraint and a warping path boundary constraint of 10% of the time series length.
- **Discriminative Guided Warping (DGW)**: DGW is similar to RGW, except it selects the most discriminative pattern in a batch as the teacher [39]. For this implementation, DGW-sD is used. DGW-sD extends the standard DGW-D but uses shapeDTW instead of the typical DTW. This is used for the evaluation as the results were the most competitive variation of guided warping [39].

7 Results and discussion

The average results of each data augmentation method and each model on the 128 datasets in the 2018 UCR Time Series Archive are shown in Tables 1–3. In addition to the average results for each data augmentation method and model combination, the tables include a paired t-test which compares the differences between no augmentation and each augmentation method. Specifically, the t-value is shown along with indicators for low two-tailed p-values.

The tables show that the different data augmentation methods have dramatic differences in results depending on the neural network architecture. Overall, there were mixed results. Some data augmentation methods improved the accuracy and some methods were detrimental. Slicing, Window Warping, and DGW tended to have the most positive effects for each model while Rotation, Permutation, and Time Warping had significantly degraded accuracies. Furthermore, every data augmentation tended to improve the VGG model the most with the largest gain from using Slicing with VGG. It is also notable that the data augmentation with MLP and LSTM-FCN was mostly detrimental, sometimes significantly.

7.1 Differences between augmentation methods

Some of the accuracy differences can be explained by examining the effects that the data augmentation has on the data. Fig 4 shows a comparison of the data augmentation methods using Principal Component Analysis (PCA). In the figure, the training set of a sample dataset, GunPoint from the 2018 UCR Time Series Archive, is visualized using PCA. The solid points are the original training set points and the hollow points are the generated samples. The two colors represent the different classes. By comparing the generated patterns when projected into the

Table 1. Comparative results for magnitude domain transformation-based data augmentation methods.

Model	None	Jittering		Rotation		Scaling		Magnitude Warping	
	Ave. (%)	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*
MLP	70.23±21.91	70.52±21.67	1.55	69.13±21.70	-1.95*	70.24±21.97	0.05	69.43±22.50	-2.91***
VGG	73.02±23.05	74.21±22.57	0.91	71.87±22.14	-0.84	73.69±23.86	0.53	74.42±22.63	1.45
ResNet	81.39±17.19	80.87±18.08	-0.87	78.01±19.93	-.31***	81.92±16.81	1.05	81.33±17.15	-0.08
LSTM	53.46±28.10	54.72±27.10	1.09	49.64±26.93	-2.95***	53.69±26.94	0.21	54.35±27.29	0.93
BLSTM	62.51±24.74	60.23±24.87	-2.01**	57.76±24.76	-3.51***	62.13±25.89	-0.37	62.00±26.51	-0.44
LSTM-FCN	81.54±17.53	79.18±19.87	-4.89***	78.87±19.37	-3.66***	80.73±18.37	-1.74*	79.46±19.68	-3.15***

* $p < 0.1$,
** $p < 0.05$,
*** $p < 0.01$

https://doi.org/10.1371/journal.pone.0254841.t001

first two principal axes, we can infer some differences between the generated patterns of the data augmentation methods.

For example, wDBA and Jittering created time series that were very similar to the original time series in the GunPoint dataset. This is unsurprising since the former weights similar patterns more when mixing and the latter only adds noise. The change in accuracy reflected this with Tables 1 and 3 demonstrating that wDBA and Jittering only had minor effects on the

Table 2. Comparative results for time domain transformation-based data augmentation methods.

Model	Permutation		Slicing		Time Warping		Window Warping	
	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*
MLP	68.17±22.17	-3.89***	70.00±22.51	-0.49	67.20±22.31	-5.64***	69.73±22.50	-0.87
VGG	73.75±22.00	0.81	76.33±24.47	2.68**	75.50±20.82	1.86*	75.88±21.63	2.57**
ResNet	80.67±18.31	-1.16	81.51±17.64	0.24	79.62±19.20	-2.25**	82.32±16.93	1.99**
LSTM	49.65±26.90	-3.56***	52.44±27.81	-0.96	51.09±27.14	-2.28**	54.41±28.23	1.73*
BLSTM	60.89±24.73	-1.38	64.60±24.01	1.93*	61.42±25.22	-0.97	62.63±24.40	0.10
LSTM-FCN	81.25±17.22	-0.74	80.30±19.27	-1.67*	78.82±18.69	-4.66***	79.73±19.35	-1.94*

* $p < 0.1$,
** $p < 0.05$,
*** $p < 0.01$

https://doi.org/10.1371/journal.pone.0254841.t002

Table 3. Comparative results for pattern mixing-based data augmentation methods.

Model	SPAWNER		wDBA		RGW		DGW	
	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*	Ave. (%)	*t*
MLP	69.15±22.07	-2.84***	69.49±21.68	-2.15**	70.01±22.09	-0.57	69.60±22.70	-1.19
VGG	75.77±21.18	2.28**	73.49±22.33	0.36	74.51±22.18	1.54	75.78±22.20	2.25**
ResNet	80.47±17.02	-1.25	81.46±18.59	0.12	81.25±18.10	-0.22	81.99±17.38	1.29
LSTM	54.97±28.33	1.73*	52.21±27.29	-1.19	54.04±28.17	0.51	54.48±28.52	1.06
BLSTM	60.53±25.66	-1.50	61.41±24.81	-1.12	62.41±26.09	-0.10	64.82±24.34	2.21**
LSTM-FCN	79.06±18.57	-4.75***	79.90±18.93	-3.70***	78.55±20.16	-3.09***	80.25±19.94	-1.49

* $p < 0.1$,
** $p < 0.05$,
*** $p < 0.01$

https://doi.org/10.1371/journal.pone.0254841.t003

accuracy (with exception to LSTM-FCN which had significant losses in accuracy for all data augmentation methods). Furthermore, the data augmentation methods which transform the patterns so much that the classes are overlapped in Fig 4, such as Time Warping, Permutation, and Rotation, reflected significant losses in accuracy. Conversely, as expected, the data augmentation methods which push the boundaries of the classes in the PCA space, tended to perform better.

7.2 Relationship between dataset properties and accuracy

In order to understand the differences between the data augmentation methods, we analyze the relationships between the methods and different properties, or characteristics, of the datasets. To do this, we find the correlation between the dataset properties and the change in accuracy ΔAcc from the un-augmented model to the augmented models. The following dataset properties are used:

- **Training set size**: The number of training patterns in each dataset.
- **Patterns per class**: The average number of training patterns in each class.
- **Time series length**: The maximum number of time steps of the patterns in the training set for each dataset.
- **Dataset variance**: The average variance of the elements of the time series across each dataset.
- **Intra-class variance**: The average variance within each class.
- **Class Imbalance**: The difference between the size of the classes in the training set.

Each dataset property and the expectation of each correlation is detailed below. These dataset properties are selected because they provide fundamental differences between time series data.

The results of the correlation analysis are shown in Fig 5. The first row, Fig 5(a), displays the change in accuracy ΔAcc for each model and data augmentation method. The subsequent rows are the Spearman's Rank Correlation Coefficients between ΔAcc and each property. Spearman's Rank Correlation Coefficient is used because it is robust to outliers and can be used with skewed variable distributions [117]. As many of the dataset properties is skewed, e.g. most of the datasets have small training sets and most of the datasets have balanced class membership. However, for reference, Pearson's Product Moment Correlation Coefficient was also used and can be found in S1 Fig.

7.2.1 Training set size. The correlation between the change in accuracy ΔAcc and the number of patterns in the training set is shown in Fig 5(b). We expect the correlation to be negative because larger datasets are known to already generalize well [14, 15]. In other words, data augmentation should have a larger effect on smaller datasets, so we would expect a larger increase in accuracy due to data augmentation as the dataset size decreases.

For most model and data augmentation pairs, our expectation is confirmed. ResNet and LSTM-FCN especially have strong negative correlations for most of the methods. But, even MLP, VGG, and BLSTM had slightly negative correlations between ΔAcc and training set size. This implies that for those methods, our expectation is confirmed and the accuracy is inversely related to the number of patterns in the training set. However, for some methods, such as Scaling with MLP and LSTM, there is a positive correlation and for many others, there is almost no correlation. Thus, for many models, such as MLP, LSTM, and BLSTM, training set size is not a strong indicator for improvement in accuracy.

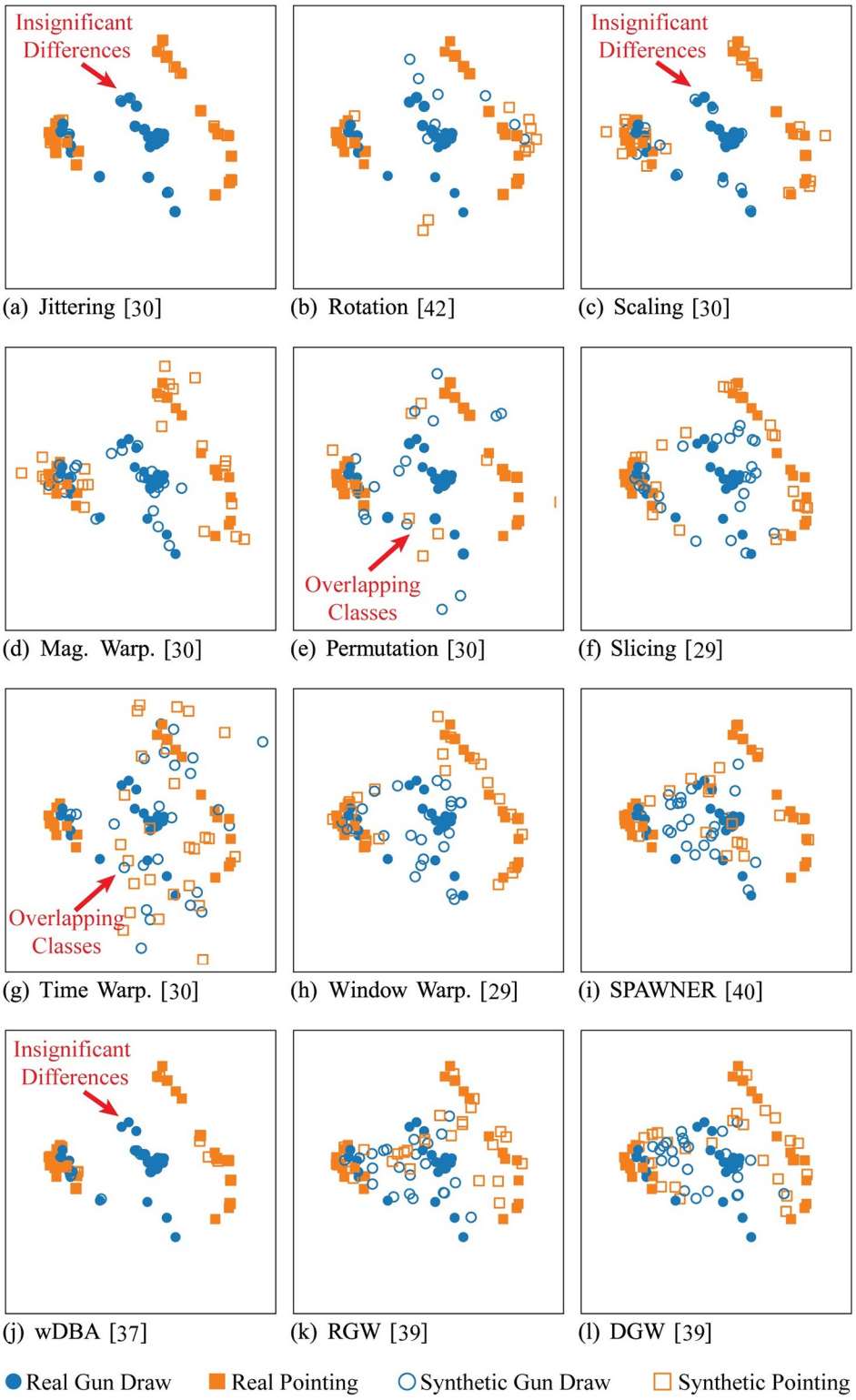

Fig 4. Visualization of the GunPoint dataset using PCA and the compared augmentation methods. The solid shapes are the original time series, the hollow shapes are the generated time series, and the colors indicate different classes.

https://doi.org/10.1371/journal.pone.0254841.g004

7.2.2 Patterns per class. The correlation of the change in accuracy to the average number of patterns per class is similar to training set size. This property is shown because it is possible for some datasets to have a large number of classes, thus the total training set size is not accurately portrayed. Like the total training set size, it would be expected that there is a negative correlation between ΔAcc and patterns per class. However, the results do not differ significantly from the correlation to training set size. The correlations between ΔAcc and patterns per class for MLP, VGG, ResNet, and LSTM were very similar to the correlation between ΔAcc and training set size. Furthermore, in general, the correlations are weaker than the total training set size. Thus, consideration based on the other properties or should be used to select the data augmentation method.

The only major difference from Training Size and Patterns Per Class is LSTM-FCN. LSTM-FCN had unexpected positive correlations with Jittering, Time Warping, SPAWNER, and wDBA. This indicates that LSTM-FCN might not react as positively to data augmentation when the number of patterns per class is smaller.

7.2.3 Time series length. This trial examines the relationship between time series length and change in accuracy. The expectation here would be that the correlation between the two is zero because naively, there should be no strong relationship. However, from the results in Fig 5(d), it can be observed that there are strong correlations. For example, for Slicing, there are positive correlations for the models with CNN components: VGG, ResNet, and LSTM-FCN. This means that as the time series grows larger, the gain in accuracy goes up. One explanation for this might be due to longer time series containing less information at the endpoints, which Slicing crops. Conversely, for the RNN-based networks, LSTM and BLSTM, the correlations were generally negative and LSTM-FCN had a strong negative correlation for Jittering, SPAWNER, and wDBA.

7.2.4 Dataset variance. Next, the correlation between ΔAcc and the dataset variance $\bar{\sigma}^2_{\text{DS}}$ is found for each data augmentation method and network model combination. The dataset variance is the average element-wise variance across the entire training dataset, or:

$$\bar{\sigma}^2_{\text{DS}} = \frac{1}{T}\sum_{t=1}^{T}\sigma_t^2, \tag{11}$$

where T is the number of time steps and σ_t^2 is the variance at time step t across the whole dataset. The correlation between ΔAcc and $\bar{\sigma}^2_{\text{DS}}$ explains the relationship between the change and accuracy and the differences between the patterns in the dataset. A positive correlation is expected because datasets with small variations have similar patterns, even between classes, and would be disturbed from the transformations of the data augmentation methods.

From Fig 5(e), we find that the expected positive correlations were observed for all of the models and most of the data augmentation, especially the CNN-based models. The only outlier is LSTM which had negative correlations for magnitude domain augmentation methods.

7.2.5 Intra-class variance. Next, we use the intra-class variance, or the variance within classes, as a property of the dataset. The intra-class variance $\bar{\sigma}^2_{IC}$ is the average element-wise variance, or:

$$\bar{\sigma}^2_{IC} = \frac{1}{NC}\sum_{c=1}^{C}\frac{N_c}{T}\sum_{t=1}^{T}\sigma_t^2, \tag{12}$$

where C is the number of classes, N is the total number of training patterns, N_c is the number of patterns in class c, T is the number time steps, and σ_i^2 is the variance of the patterns in class c

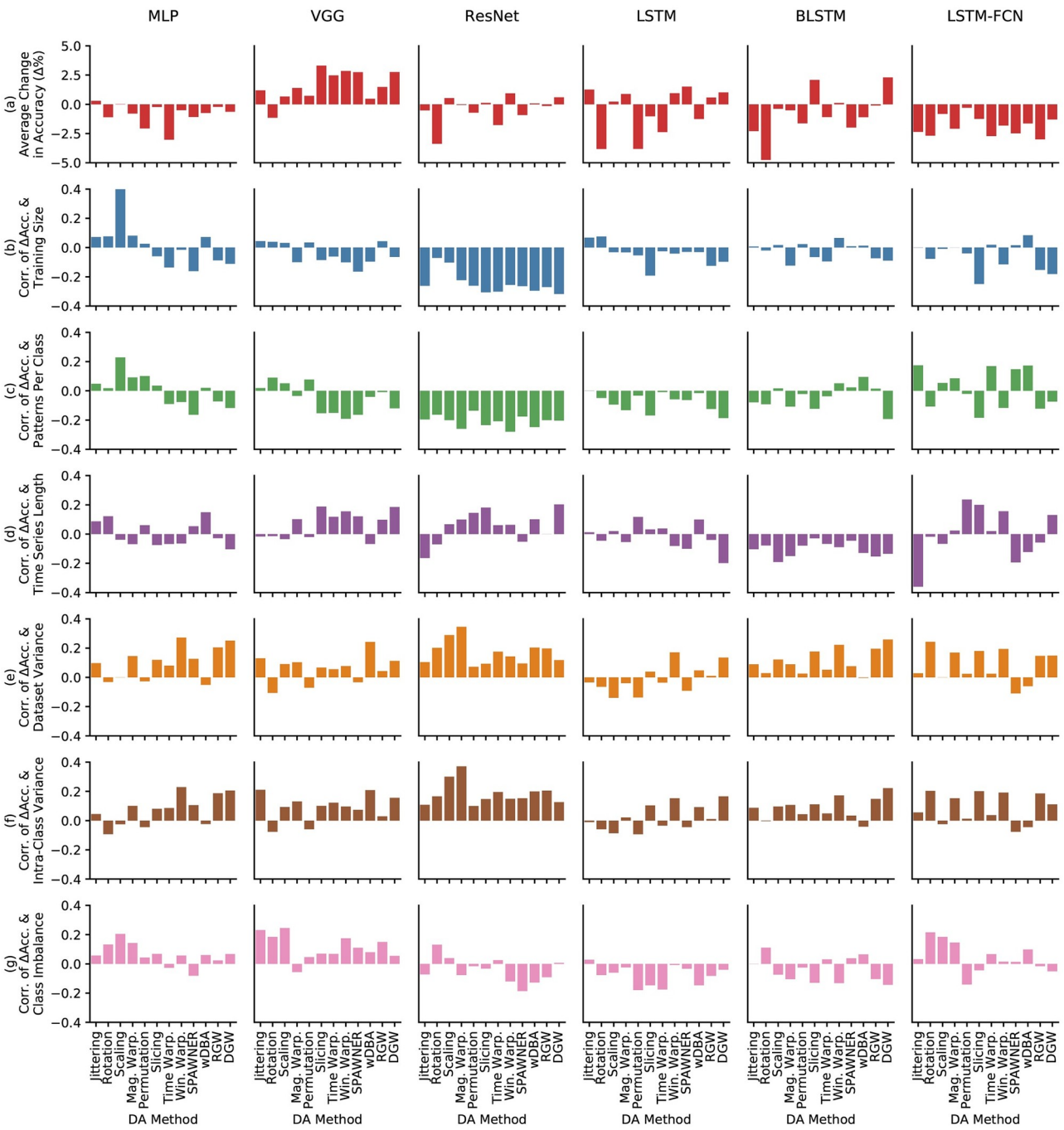

Fig 5. Spearman's Rank Correlation Coefficients between change in accuracy (Δ Acc) and various dataset characteristics. The top row in red is the change in accuracy and the subsequent rows are the correlations.

https://doi.org/10.1371/journal.pone.0254841.g005

at time step *i*. Similar to the dataset variance, the expected correlation between $\bar{\sigma}^2_{IC}$ and ΔAcc is positive.

The correlations are shown in Fig 5(f) and it shows that the correlations between intra-class variance and change in accuracy act similar to the overall dataset variance. ResNet especially has a strong correlation between intra-class variation and change in accuracy.

7.2.6 Class imbalance. Finally, we consider the class imbalance as a property. Class imbalance is how imbalanced the size of the classes in the training set is. Augmentation methods such as SMOTE [46] attempt to fix class imbalance by generating extra patterns in the minority classes. To measure class imbalance, Imbalance-Degree (ID) proposed by Ortigosa Hernandez [118] is used. ID is a robust class imbalance measure that is designed for multi-class problems, much like the datasets used in the experiments. ID is calculated

$$ID = \frac{d(\boldsymbol{\zeta}, \boldsymbol{\beta})}{d(\boldsymbol{\iota}, \boldsymbol{\beta})} + (C_m - 1), \tag{13}$$

where $d(\cdot, \boldsymbol{\beta})$ is a statistical distance between distributions ζ or $\boldsymbol{\iota}$ and $\boldsymbol{\beta}$. $\zeta = [\zeta_1, \ldots, \zeta_c, \ldots, \zeta_C]$ denotes a vector of the true distributions ζ_c of the classes c in the dataset where:

$$\zeta_c = \frac{N_c}{N}, \tag{14}$$

and N_c is the number of patterns in each class c and N is the total number of training patterns. Vector $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_c, \ldots, \beta_C]$ is the distribution of a balanced dataset with each $\beta_c = \frac{1}{C}$. C_m is the number of minority classes. A minority class is a class that has with $\zeta_c < \frac{1}{C}$. Finally, $\boldsymbol{\iota} = [\iota_1, \ldots, \iota_c, \ldots, \iota_C]$ is a vector representing the class distributions that is has the maximal distance from $\boldsymbol{\beta}$ with the same C_m, or a vector of C_m number of $\iota_c = 0$, $C - C_m - 1$ number of $\iota_c = \frac{1}{C}$, and one $\iota_c = 1 - \frac{C - C_m - 1}{C}$. A balanced dataset would have an $ID = 0$ and unbalanced datasets have larger ID scores. In the experiment, the Hellinger distance is used as the distance measure $d(\cdot, \boldsymbol{\beta})$ due to it having the highest Pearson correlation coefficient between imbalance and neural network performance among the distance functions tested in Ortigosa-Hernandez et al. [118]

Unlike SMOTE, the data augmentation methods in the experiments sample the classes at the same distribution as what already exists in the training dataset. While small classes can benefit from more training samples, the large classes also increase in size. Despite this, there are correlations that can be found from the results.

In Fig 5(g), there are positive correlations for Jittering, Rotation, Scaling, and Magnitude Warping for MLP and LSTM-FCN, and Jittering, Rotation, and Scaling for VGG. This is interesting to note because these methods are magnitude domain augmentations. In other words, this shows that as class imbalance rises, the ΔAcc tends to rise for magnitude domain data augmentation methods for MLP, VGG, and LSTM-FCN. Conversely, the time domain data augmentations tend have negative or very small correlations for the same models.

7.3 Computation time

The theoretical computational complexity of many of the methods are similar; the simple transformations are $O(T)$ and the complex transformations and pattern mixing methods are $O(T^2)$, where T is the number of time steps. However, as shown in Table 4, the observed execution time varies a lot between the methods. To compare the execution time, all of the datasets in the 2018 UCR Time Series Archive were augmented once and timed using a computer with a 2.60 GHz Intel Xeon CPU using the Python implementation at https://github.com/uchidalab/time_series_augmentation.

The average observed execution times for the transformation-based methods are negligible. They only took a fraction of a second on average to double the dataset size. On the other hand, the pattern mixing methods are much slower with SPAWNER and RGW taking about a minute on average to execute and wDBA and DGW taking 2,300 and 4,290 seconds, respectively. The reason for the slow run times is primarily due to the speed of DTW, which each of the

Table 4. Algorithm comparison with average augmentation time per dataset and tunable parameters.

DA Method	Average Time (s)	Tunable Parameters
Jittering	<0.01	σ
Rotation	<0.01	σ
Scaling	<0.01	σ
Magnitude Warping	0.11	σ, # of knots
Permutation	0.01	window size, # of permutations
Slicing	0.02	window size
Time Warping	0.12	σ, # of knots
Window Warping	0.04	window size, warping amount
SPAWNER	66.7	σ, DTW constraints
wDBA	2,300	weighting, DTW constraints
RGW	70.5	DTW constraints
DGW	6,380	batch size, DTW constraints

https://doi.org/10.1371/journal.pone.0254841.t004

methods relies on for element alignment. For most datasets, this is not a problem, but ones with long time series, such as HandOutlines with 2,709 time steps, can take extraordinary amounts of time compared to the transformation-based methods. Accordingly, pattern mixing methods which do not use DTW, such as interpolation, might not face the same issue. DGW and wDBA take extra long because for each generated time series, multiple DTW calculations must be performed. In addition, the DGW implementation used in the experiment uses shapeDTW which takes significantly longer to execute than standard DTW.

7.4 Number of tunable parameters

The number of tunable parameters is another aspect of the data augmentation methods that one needs to consider. The hyperparameter, design choice, and variation of the method need to be selected and the effectiveness of the augmentation method can depend on the parameters. Thus, methods with many parameters may need many adjustments and evaluations to be effectively used. A list of the parameters that need to be defined manually is displayed in Table 4. In general, while the random transformations have fewer parameters, they are more dependent on the hyperparameters due to the random element. While more complex, the pattern mixing methods on the other hand rely on the patterns in the dataset for randomness, thus, they have fewer choices to tune.

7.5 Recommendations on data augmentation usage

Each data augmentation method has different effects depending on the model and dataset, as shown in Tables 1–3 and Fig 5. Thus, it is an arduous task to determine which data augmentation method to use in which situation. In this section, we will attempt to provide recommendations to solve this problem.

7.5.1 Dataset type and data augmentation method. The datasets can be broken into categories, such as ECG, sensor, etc. To categorize the datasets, we use the labels provided by [16]. In Table 5, we show the methods with the top 5 highest average rank for each category and model combination. In the table, execution time is the deciding factor for instances of ties. This table can be used as a general guide in combination with Fig 5.

7.5.2 Magnitude domain transformations. Jittering, Magnitude Warping, and Scaling had similar results. They tend to act similarly since they are similar transformations in that they only differ in how many directions magnitude gets scaled. For example, in general, these magnitude domain transformations work well with VGG and with LSTM.

Table 5. Data augmentation recommendations for data type and model type.

Data Type	MLP	VGG	ResNet	LSTM	BLSTM	LSTM-FCN
Device	Sl, Ro, Sc, J, N	Sl, M, W, T, RW	N, W, T, P, SP	RW, J, W, Sc, T	J, T, RW, DW, wD	T, P, M, N, RW
ECG	M, Ro, Sc, J, N, SP	Sc, Ro, M, W, RW	J, Sc, N, P, M	Sc, J, Ro, Sl, W	Sl, Sc, N, Ro, J	Sc, M, P, N, wD
EOG	W, wD, RW, J, DW	SP, W, P, T, wD	DW, P, RW, T, SP	Sc, Sl, W, J, P	J, SP, N, T, M	DW, Sc, W, Ro, Sl
EPG	N, Sc, J, Sl, M	J, SP, Sc, T, DW	N, Sc, J, Sl, P	Sc, J, Sl, W, P	N, Sc, J, Sl, P	N, Sc, J, Sl, W
Hemo	RW, Ro, M, N, P	T, P, DW, Sl, W	Sl, Ro, SP, wD, N	T, Sl, P, DW, N	Sl, DW, P, SP, Ro	Sl, N, P, DW, T
HRM	Sl, DW, N, P, RW	T, Sl, Ro, W, P	Sl, P, W, RW, SP	DW, J, SP, Ro, Sl	Sl, RW, P, SP, Ro	N, P, W, RW, wD
Image	Sc, J, Ro, RW, N	W, DW, RW, Sl, T	W, N, Sc, M, wD	RW, M, SP, W, J	DW, Sc, M, T, W	N, W, P, Sl, DW
Motion	W, DW, Sl, RW, Sc	DW, W, RW, Sl, T	Sc, wD, Sl, W, Ro	W, N, M, RW, J	RW, Sl, SP, W, RW	Sc, W, N, DW, M
Power	N, Sc, Ro, wD, J	N, Sc, Ro, J, M	RW, Ro, SP, wD, W	N, Ro, RW, M, SP	Ro, DW, N, J, RW	Sc, J, Ro, wD, N
Simulated	DW, SP, Sc, RW, J	W, DW, Sc, Sl, N	Sc, DW, W, N, wD	W, DW, Sl, Sc, J	RW, DW, W, M, wD	W, DW, N, Ro, RW
Spectro	Sl, W, DW, RW, T	Sl, W, DW, wD, J	W, DW, RW, SP, Sl	DW, SP, M, T, J	M, T, DW, Sc, SP	W, RW, Sl, Sc, N
Spectrum	J, Sc, RW, N, W	Sl, N, DW, Ro, P	W, DW, Sl, SP, M	Sc, SP, T, RW, W	W, T, J, SP, N	P, SP, Sc, RW, N
Traffic	Sl, Sc, RW, DW, wD	Sc, J, Ro, T, wD	P, wD, J, Ro, N	W, T, Sc, T, RW	W, wD, Sl, DW, N	N, W, J, SP, wD
Trajectory	J, Sc, M, DW, SP	RW, N, DW, Sc, W	Sc, M, wD, W, RW	M, Sc, J, Sl, DW	DW, W, J, Ro, Sc	P, RW, Sc, N, DW
Overall	Sc, J, RW, W, Sl	W, DW, Sl, RW, T	W, Sc, DW, N, wD	W, RW, J, Sc, M	DW, Sl, RW, W, M	N, W, DW, P, Sc

The suggested augmentation method has the top five highest average rank within each data type. Ties between methods are broken based on execution time.
N: None, J: Jittering, Sc: Scaling, Ro: Rotation, M: Magnitude Warping, P: Permutation, T: Time Warping, Sl: Slicing, W: Window Warping, SP: SPAWNER, wD: wDBA, RW: RGW, DW: DGW

https://doi.org/10.1371/journal.pone.0254841.t005

As described previously, Rotation (flipping in this implementation), seems to be not suitable as a general time series data augmentation method. The overall accuracy for Rotation, in Table 1, shows a decrease in accuracy for all models. Furthermore, in Table 5, Rotation only had the highest average rank for one combination, power data with BLSTM. This was the second worst showing next to Permutation in Table 5. The poor performance of Rotation is intuitive, however. The 2018 UCR Time Series Archive contains many shape-based datasets and datasets where flipping is inappropriate.

7.5.3 Time domain transformations. Similar to Rotation, Permutation had severely detrimental effects on accuracy. However, this is somewhat expected because Permutation breaks the time dependency of the time series. The only time that it would intuitively make sense for Permutation to be used would be for periodic time series or very sparse time series.

As a general purpose data augmentation method for time series, the other time domain transformations far outperform Permutation. Specifically, Slicing and Window Warping performed well on most datasets and models. In particular, the CNN-based models, VGG and ResNet, were significantly improved by Slicing and Window Warping (Table 2). Considering the positive effect they have as data augmentation methods and the very fast computation time (Table 4), it seems like they should be a first choice when selecting time series data augmentation.

However, Time Warping showed a poor performance. This is likely due to over transforming the time series causing significant noise, as illustrated in Fig 4(g). The parameters used for this implementation of Time Warping were the parameters suggested by Um et al. [30]. This reinforces the flaw with random transformation-based data augmentation methods in which the parameters must be carefully selected. The difficulty is that the user has to balance transforming the patterns enough so that the generalization is increased, but not so much that the classes are confused.

7.5.4 Pattern matching methods. As mentioned previously, the largest downside to the pattern matching methods is the slow computation time. While it is possible to achieve better

generalization and results using these methods, one has to consider the extra time it would take to generate the data. However, this issue is only with longer patterns and the execution time is also negligible for short time series. Furthermore, from Fig 5(d), the correlation between change in accuracy and time series length is negative for these methods. Thus, using these pattern mixing methods would be recommended for shorter time series.

However, for most datasets, wDBA had disappointing results. The primary reason for this is due to wDBA not creating diverse enough results. In the evaluation, we used the ASD because Forestier et al. [37] found it the most competitive weighting method. The ASD weights the nearest neighbors of the reference pattern more than farther away patterns. Due to this, as shown in Fig 4(j), the new patterns generated from wDBA were not significantly different from the existing patterns in the dataset. Although, it should be noted that this could be an issue with the ASD weighting scheme.

As for SPAWNER, RGW, and DGW, there were mixed results. In some models and some datasets, they performed better than other augmentation methods and on others, they performed worse. Notably, DGW had the most performance increase compared to no augmentation out of all of the pattern mixing methods and the highest average rank on BLSTM compared to all augmentation methods (Table 5). As mentioned before, the downside of DGW is that it is also the slowest algorithm of all the data augmentation methods used for the comparative evaluations. For datasets with many long time series, such as HandOutlines, StarLightCurves, and NonInvasiveFetalECGThorax1/2, this could mean many orders of magnitude longer than the simple transformations.

8 Conclusion

In this paper, we performed a comprehensive survey of data augmentation methods for time series. The survey categorizes and outlines the various time series data augmentation methods. We include transformation-based methods across the related time series domains and time series pattern mixing, generative models, and decomposition methods. Furthermore, a taxonomy of time series data augmentation methods was proposed.

In addition, an empirical comparative evaluation of 12 time series data augmentation methods on six neural network models and 128 discrete finite time series datasets was performed. Namely, we use all of the datasets in the 2018 UCR Time Series Archive [16] to evaluate jittering, rotation, scaling, magnitude warping, permutation, slicing, time warping, window warping, SPAWNER, wDBA, RGW, and DGW. The training datasets of each dataset are augmented by four times the size and are trained and tested using an MLP, VGG, ResNet, LSTM, BLSTM, and LSTM-FCN. Through the empirical evaluation, we are able to compare the data augmentations and analyze the findings.

By using all 128 datasets of the 2018 UCR Time Series Archive, we are able to test the data augmentation methods on a wide variety of time series and make recommendations on data augmentation usage. As a general, easy-to-use, and effective method, the Window Warping algorithm that was proposed by Le Guennec et al. [29] is the most recommended algorithm. Window warping had the highest average rank across all of the datasets for VGG, ResNet, and LSTM and consistently performed well with the other datasets as well. In addition, slicing, i.e., window slicing proposed by the same authors, also showed to be very effective in most cases. While there were algorithms that periodically performed better than window warping and slicing, the drawbacks of speed or sensitivity to parameter tuning hindered the other methods. Alternatively, we found that the time domain pattern mixing method, DGW, also performed well in most cases. However, it is only recommended on time series that are shorter in length due to the high computational requirement.

The results also revealed some key aspects of data augmentation with time series based neural networks. For example, LSTM-FCN does not respond well to data augmentation. Part of this could be because LSTM-FCN uses dropout with a very high rate or it could be that the design of the architecture is just not suitable for data augmentation. MLP, to a lesser extent, also did not respond well. Conversely, the accuracy of VGG was often improved with most of the data augmentation methods. The other models, ResNet, LSTM, and BLSTM had mixed results, and careful augmentation method selection is required.

We also analyzed different aspects of time series datasets and the effects data augmentation had on datasets with them. First, the correlations between properties of time series datasets and the change in accuracy from augmentation were found. The findings showed that there generally was a negative correlation between change in accuracy and training set size (and number of patterns per class) and a positive correlation for dataset variance and intra-class variance. We found that time series length generally had a negative correlation for RNN-based models, but positive correlations for CNN-based models. There was a strong positive correlation between change in accuracy and class imbalance for MLP, VGG, and LSTM-FCN for the magnitude domain data augmentation methods. Second, using ranking, we found the top augmentation methods for each model and dataset type combination.

This survey serves as a guide for researchers and developers in selecting the appropriate data augmentation method for their applications. Using this, it is possible to refine the selection of data augmentation based on dataset type, property, and model. An easy to use implementation of the algorithms evaluated in this survey are provided at https://github.com/uchidalab/time_series_augmentation.

Since data augmentation for time series is not established as much as data augmentation for images, there is a lot of room for time series data augmentation to grow. For example, like most works, this survey only uses a single data augmentation method for each model. It is possible that multiple data augmentation methods can synergize well and be used serially. With exception to Um et al. [30], this idea is not used. There are also many other advanced data augmentation methods used in the image domain [17] that are not used by time series, such as style transfer, meta-learning, and filters. There is a lot of potential for new time series data augmentation research.

Supporting information

S1 Table. Details of the 2018 UCR Time Series Archive datasets.
(PDF)

S1 Fig. Pearson's Product Moment Correlation Coefficients between change in accuracy (Δ Acc) and various dataset characteristics. The top row in red is the change in accuracy and the subsequent rows are the correlations.
(PDF)

Author Contributions

Conceptualization: Brian Kenji Iwana, Seiichi Uchida.

Data curation: Brian Kenji Iwana.

Formal analysis: Brian Kenji Iwana.

Funding acquisition: Seiichi Uchida.

Investigation: Brian Kenji Iwana.

Methodology: Brian Kenji Iwana.

Project administration: Brian Kenji Iwana, Seiichi Uchida.

Resources: Brian Kenji Iwana, Seiichi Uchida.

Software: Brian Kenji Iwana.

Supervision: Seiichi Uchida.

Validation: Brian Kenji Iwana.

Visualization: Brian Kenji Iwana.

Writing – original draft: Brian Kenji Iwana.

Writing – review & editing: Brian Kenji Iwana, Seiichi Uchida.

References

1. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E. Querying and mining of time series data. Proc Very Larg Data Base Endow. 2008; 1(2):1542–1552.
2. Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015; 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003 PMID: 25462637
3. Wang Z, Yan W, Oates T. Time series classification from scratch with deep neural networks: A strong baseline. In: IJCNN; 2017.
4. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nat. 1986; 323(6088):533–536. https://doi.org/10.1038/323533a0
5. Chen Q, Liang B, Wang J. A comparative study of LSTM and phased LSTM for gait prediction. Int J Artificial Intelli & App. 2019; 10(4):57–66. https://doi.org/10.5121/ijaia.2019.10405
6. Kluwak K, Nizynski T. Gait classification using LSTM networks for tagging system. In: IEEE SoSE; 2020.
7. Xu G, Xing G, Jiang J, Jiang J, Ke Y. Arrhythmia detection using gated recurrent unit network with ECG signals. J Medical Imag and Health Inform. 2020; 10(3):750–757. https://doi.org/10.1166/jmihi.2020.2928
8. Kim BH, Pyun JY. ECG identification for personal authentication using LSTM-based deep recurrent neural networks. Sensors. 2020; 20(11):3069. https://doi.org/10.3390/s20113069 PMID: 32485827
9. Sun L, Su T, Liu C, Wang R. Deep LSTM networks for online Chinese handwriting Recognition. In: ICFHR; 2016.
10. Carbune V, Gonnet P, Deselaers T, Rowley HA, Daryin A, Calvo M, et al. Fast multi-language LSTM-based online handwriting recognition. Int J Doc Analy and Recogn. 2020; 23(2):89–102. https://doi.org/10.1007/s10032-020-00350-4
11. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998; 86(11):2278–2324. https://doi.org/10.1109/5.726791
12. Bai S, Kolter JZ, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:180301271. 2018;.
13. Iwana BK, Uchida S. Time series classification using local distance-based features in multi-modal fusion networks. Pattern Recogn. 2020; 97:107024. https://doi.org/10.1016/j.patcog.2019.107024
14. Banko M, Brill E. Scaling to very very large corpora for natural language disambiguation. In: AMACL; 2001.
15. Torralba A, Fergus R, Freeman WT. 80 Million tiny images: A large data set for nonparametric object and scene recognition. IEEE Trans Pattern Anal and Mach Intell. 2008; 30(11):1958–1970. https://doi.org/10.1109/TPAMI.2008.128 PMID: 18787244
16. Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, et al. The UCR time series classification archive; 2018.
17. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data. 2019; 6(1). https://doi.org/10.1186/s40537-019-0197-0
18. Olson M, Wyner A, Berk R. Modern neural networks generalize on small data sets. In: NeurIPS; 2018. p. 3619–3628.

19. Blagus R, Lusa L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics. 2013; 14 (1). https://doi.org/10.1186/1471-2105-14-106 PMID: 23522326

20. Hasibi R, Shokri M, Dehghan M. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. arXiv preprint arXiv:190100204. 2019;.

21. Krizhevsky A. Learning multiple layers of features from tiny images. University of Tront Thesis. 2009;.

22. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis. 2015; 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y

23. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014;.

24. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: IEEE CVPR; 2016.

25. Huang G, Liu Z, Maaten LVD, Weinberger KQ. Densely connected convolutional networks. In: IEEE CVPR; 2017.

26. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: IEEE CVPR; 2015.

27. Wen Q, Sun L, Song X, Gao J, Wang X, Xu H. Time series data augmentation for deep learning: A survey. arXiv preprint arXiv:200212478. 2020;.

28. Fields T, Hsieh G, Chenou J. Mitigating drift in time series data with noise augmentation. In: ICCSCI; 2019.

29. Le Guennec A, Malinowski S, Tavenard R. Data augmentation for time series classification using convolutional neural networks. In: IWAATD; 2016.

30. Um TT, Pfister FMJ, 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: ACM ICMI; 2017. p. 216–220.

31. Jaitly N, Hinton GE. Vocal tract length perturbation (VTLP) improves speech recognition. In: ICML WDLASLP; 2013.

32. Cao H, Tan VYF, Pang JZF. A parsimonious mixture of Gaussian trees model for oversampling in imbalanced and multimodal time-series classification. IEEE Trans Neural Networks and Learning Sys. 2014; 25(12):2226–2239. https://doi.org/10.1109/TNNLS.2014.2308321 PMID: 25420245

33. Wendling F, Bellanger JJ, Bartolomei F, Chauvel P. Relevance of nonlinear lumped-parameter models in the analysis of depth-EEG epileptic signals. Bio Cybernetics. 2000; 83(4):367–378. https://doi.org/10.1007/s004220000160 PMID: 11039701

34. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: NeurIPS; 2014. p. 2672–2680.

35. Bergmeir C, Hyndman RJ, Benítez JM. Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. Int J Forecasting. 2016; 32(2):303–312. https://doi.org/10.1016/j.ijforecast.2015.07.002

36. Eltoft T. Data augmentation using a combination of independent component analysis and non-linear time-series prediction. In: IJCNN; 2002. p. 448–453.

37. Forestier G, Petitjean F, Dau HA, Webb GI, Keogh E. Generating synthetic time series to augment sparse datasets. In: IEEE ICDM; 2017.

38. Liu B, Zhang Z, Cui R. Efficient Time Series Augmentation Methods. In: CISP-BMEI; 2020.

39. Iwana BK, Uchida S. Time series data augmentation for neural networks by time warping with a discriminative teacher. In: ICPR; 2021.

40. Kamycki K, Kapuscinski T, Oszust M. Data augmentation with suboptimal warping for time-series classification. Sensors. 2019; 20(1):98. https://doi.org/10.3390/s20010098 PMID: 31877970

41. Huang CL. Exploring effective data augmentation with TDNN-LSTM neural network embedding for speaker recognition. In: IEEE ASRUW; 2019. p. 291–295.

42. Rashid KM, Louis J. Time-warping: A time series data augmentation of IMU data for construction equipment activity identification. In: ISARC; 2019.

43. Bishop CM. Training with noise is equivalent to Tikhonov regularization. Neural Computation. 1995; 7 (1):108–116. https://doi.org/10.1162/neco.1995.7.1.108

44. An G. The effects of adding noise during backpropagation Training on a Generalization Performance. Neural Computation. 1996; 8(3):643–674. https://doi.org/10.1162/neco.1996.8.3.643

45. Arslan M, Guzel M, Demirci M, Ozdemir S. SMOTE and Gaussian noise based sensor data augmentation. In: ICCSE; 2019.

46. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. J Art Intelli Research. 2002; 16:321–357. https://doi.org/10.1613/jair.953

47. Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller PA. Data augmentation using synthetic data for time series classification with deep residual networks. In: IWAATD; 2018.

48. Ohashi H, Ahmed S, Akiyama T, Sato T, Nguyen P, Nakamura K, et al. Augmenting wearable sensor data with physical constraint for DNN-based human-action recognition. In: ICML Workshops; 2017.

49. Delgado-Escano R, Castro FM, Cozar JR, Marin-Jimenez MJ, Guil N. An end-to-end multi-task and fusion CNN for inertial-based gait recognition. IEEE Access. 2019; 7:1897–1908. https://doi.org/10.1109/ACCESS.2018.2886899

50. Tran L, Choi D. Data augmentation for inertial sensor-based gait deep neural network. IEEE Access. 2020; 8:12364–12378. https://doi.org/10.1109/ACCESS.2020.2966142

51. Pan Q, Li X, Fang L. Data augmentation for deep learning-based ECG analysis. In: Feature Eng. and Comput. Intell. in ECG Monitor. Springer; 2020. p. 91–111.

52. Eyobu OS, Han D. Feature representation and data augmentation for human activity classification based on wearable IMU sensor data using a deep LSTM neural network. Sensors. 2018; 18(9):2892. https://doi.org/10.3390/s18092892

53. Nagano T, Fukuda T, Suzuki M, Kurata G. Data augmentation based on vowel stretch for improving children's speech recognition. In: IEEE ASRUW; 2019. p. 502–508.

54. Nguyen TS, Stuker S, Niehues J, Waibel A. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. In: IEEE ICASSP; 2020.

55. Vachhani B, Bhat C, Kopparapu SK. Data augmentation using healthy speech for dysarthric speech recognition. In: Interspeech; 2018.

56. Cui X, Goel V, Kingsbury B. Data augmentation for deep neural network acoustic modeling. In: IEEE ICASSP; 2014.

57. Lee L, Rose R. A frequency warping approach to speaker normalization. IEEE Trans Speech and Audio Process. 1998; 6(1):49–60. https://doi.org/10.1109/89.650310

58. Adachi S, Takemoto H, Kitamura T, Mokhtari P, Honda K. Vocal tract length perturbation and its application to male-female vocal tract shape conversion. J Acousitical Soc of America. 2007; 121(6):3874–3885. https://doi.org/10.1121/1.2730743 PMID: 17552734

59. Ko T, Peddinti V, Povey D, Khudanpur S. Audio augmentation for speech recognition. In: Interspeech; 2015.

60. Kim C, Shin M, Garg A, Gowda D. Improved vocal tract length perturbation for a state-of-the-art end-to-end speech recognition system. In: Interspeech; 2019.

61. Gao J, Song X, Wen Q, Wang P, Sun L, Xu H. RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks. arXiv preprint arXiv:200209545. 2020;.

62. Park DS, Chan W, Zhang Y, Chiu CC, Zoph B, Cubuk ED, et al. SpecAugment: A simple data augmentation method for automatic speech recognition. In: Interspeech; 2019.

63. Khadijah, Endah SN, Kusumaningrum R, Rismiyati. The study of synthetic minority over-sampling technique (SMOTE) and weighted extreme learning machine for handling imbalance problem on multiclass microarray classification. In: ICICoS; 2018.

64. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C. Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Adv. in Knowl. Disc. and Data Mining. Springer; 2009. p. 475–482.

65. Tarawneh AS, Hassanat ABA, Almohammadi K, Chetverikov D, Bellinger C. SMOTEFUNA: Synthetic minority over-sampling technique based on furthest neighbour algorithm. IEEE Access. 2020; 8:59069–59082. https://doi.org/10.1109/ACCESS.2020.2983003

66. Zhou C, Liu B, Wang S. CMO-SMOTE: Misclassification cost minimization oriented synthetic minority oversampling technique for imbalanced learning. In: IHMSC; 2016.

67. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C. DBSMOTE: Density-based synthetic minority over-sampling technique. Applied Intell. 2011; 36(3):664–684. https://doi.org/10.1007/s10489-011-0287-y

68. Gong Z, Chen H. Model-based oversampling for imbalanced sequence classification. In: ACM ICIKM; 2016.

69. Sawicki A, Zieliński SK. Augmentation of segmented motion capture data for improving generalization of deep neural networks. In: CISIM. Springer; 2020. p. 278–290.

70. DeVries T, Taylor GW. Dataset augmentation in feature space. arXiv preprint arXiv:170205538. 2017;.

71. Yeomans J, Thwaites S, Robertson WSP, Booth D, Ng B, Thewlis D. Simulating time-series data for improved deep neural network performance. IEEE Access. 2019; 7:131248–131255. https://doi.org/10.1109/ACCESS.2019.2940701

72. Savitzky A, Golay MJE. Smoothing and differentiation of data by simplified least squares procedures. Analytical Chemistry. 1964; 36(8):1627–1639. https://doi.org/10.1021/ac60214a047

73. Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoustics, Speech, and Sig Process. 1978; 26(1):43–49. https://doi.org/10.1109/TASSP.1978.1163055

74. Takahashi N, Gygli M, Pfister B, Gool LV. Deep convolutional neural networks and data augmentation for acoustic event recognition. In: Interspeech; 2016.

75. Nanni L, Maguolo G, Paci M. Data augmentation approaches for improving animal audio classification. Ecological Informatics. 2020; 57:101084. https://doi.org/10.1016/j.ecoinf.2020.101084

76. Petitjean F, Ketterlin A, Gançarski P. A global averaging method for dynamic time warping, with applications to clustering. Pattern Recogn. 2011; 44(3):678–693. https://doi.org/10.1016/j.patcog.2010.09.013

77. Smyl S, Kuber K. Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In: ISF; 2016.

78. Kang Y, Hyndman RJ, Li F. GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. Stat Anal and Data Mining: The ASA Data Sci J. 2020;. https://doi.org/10.1002/sam.11461

79. Tanner MA, Wong WH. The calculation of posterior distributions by data augmentation. J American Stat Assoc. 1987; 82(398):528–540. https://doi.org/10.2307/2289463

80. Fr uhwirth-Schnatter S. Data augmentation and dynamic linear models. J Time Series Anal. 1994; 15 (2):183–202. https://doi.org/10.1111/j.1467-9892.1994.tb00184.x

81. Meng XL. Seeking efficient data augmentation schemes via conditional and marginal augmentation. Biometrika. 1999; 86(2):301–320. https://doi.org/10.1093/biomet/86.2.301

82. Hou Y, Liu Y, Che W, Liu T. Sequence-to-sequence data augmentation for dialogue language understanding. arXiv preprint arXiv:180701554. 2018;.

83. Longpre S, Lu Y, Tu Z, DuBois C. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In: WMRQA. ACL; 2019.

84. Oord Avd, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, et al. Wavenet: A generative model for raw audio. arXiv preprint arXiv:160903499. 2016;.

85. Wang J, Kim S, Lee Y. Speech augmentation using Wavenet in speech recognition. In: IEEE ICASSP; 2019.

86. Tu J, Liu H, Meng F, Liu M, Ding R. Spatial-temporal data augmentation based on LSTM autoencoder network for skeleton-based human action recognition. In: IEEE ICIP; 2018.

87. Lou H, Qi Z, Li J. One-dimensional data augmentation using a Wasserstein generative adversarial network with supervised signal. In: CCDC; 2018.

88. Haradal S, Hayashi H, Uchida S. Biosignal data augmentation based on generative adversarial networks. In: EMBC; 2018.

89. Esteban C, Hyland SL, Rätsch G. Real-valued (medical) time series generation with recurrent conditional GANs. arXiv preprint arXiv:170602633. 2017;.

90. Ramponi G, Protopapas P, Brambilla M, Janssen R. T-CGAN: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. arXiv preprint arXiv:181108295. 2018;.

91. Che Z, Cheng Y, Zhai S, Sun Z, Liu Y. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In: IEEE ICDM; 2017.

92. Chen G, Zhu Y, Hong Z, Yang Z. EmotionalGAN: Generating ECG to enhance emotion state classification. In: AICS; 2019.

93. Hatamian FN, Ravikumar N, Vesal S, Kemeth FP, Struck M, Maier A. The effect of data augmentation on classification of atrial fibrillation in short single-lead ECG signals using deep neural networks. In: IEEE ICASSP; 2020.

94. Madhu A, Kumaraswamy S. Data augmentation using generative adversarial network for environmental sound classification. In: ESPC; 2019.

95. Zhu F, Ye F, Fu Y, Liu Q, Shen B. Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. Scientific Reports. 2019; 9(1). https://doi.org/10.1038/s41598-019-42516-z PMID: 31043666

96. Mirza M, Osindero S. Conditional generative adversarial nets. arXiv preprint arXiv:14111784. 2014;.

97. Nikolaidis K, Kristiansen S, Goebel V, Plagemann T, Liestøl K, Kankanhalli M. Augmenting physiological time series data: A case study for sleep apnea detection. In: ECML/PKDD; 2019.

98. Harada S, Hayashi H, Uchida S. Biosignal generation and latent variable analysis with recurrent generative adversarial networks. IEEE Access. 2019; 7:144292–144302. https://doi.org/10.1109/ACCESS.2019.2934928

99. Sheng P, Yang Z, Qian Y. GANs for children: A generative data augmentation strategy for children speech recognition. In: IEEE ASRUW; 2019. p. 129–135.

100. Wang F, hua Zhong S, Peng J, Jiang J, Liu Y. Data augmentation for EEG-based emotion recognition with deep convolutional neural networks. In: ICMM; 2018. p. 82–93.

101. Huang NE, Shen Z, Long SR, Wu MC, Shih HH, Zheng Q, et al. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proc Royal Society of London Series A: Math, Physical and Eng Sci. 1998; 454(1971):903–995. https://doi.org/10.1098/rspa.1998.0193

102. Nam GH, Bu SJ, Park NM, Seo JY, Jo HC, Jeong WT. Data augmentation using empirical mode decomposition on neural networks to classify impact noise in vehicle. In: IEEE ICASSP; 2020.

103. Comon P. Independent component analysis, A new concept? Sig Process. 1994; 36(3):287–314. https://doi.org/10.1016/0165-1684(94)90029-9

104. Cleveland RB, Cleveland WS, McRae JE, Terpenning I. STL: A seasonal-trend decomposition. J Official Stat. 1990; 6(1):3–73.

105. Wen Q, Gao J, Song X, Sun L, Xu H, Zhu S. RobustSTL: A robust seasonal-trend decomposition algorithm for long time series. AAAI Conf Artificial Intell. 2019;33:5409–5416.

106. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: IEEE ICCV; 2015.

107. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML; 2015. p. 448–456.

108. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: AISTATS; 2010. p. 249–256.

109. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.

110. Zeiler MD. Adadelta: An adaptive learning rate method. arXiv preprint arXiv:12125701. 2012;.

111. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation. 1997; 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 PMID: 9377276

112. Reimers N, Gurevych I. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. arXiv preprint arXiv:170706799. 2017;.

113. Dozat T. Incorporating Nesterov momentum into Adam. In: ICLR Workshops; 2016.

114. Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE Trans Sig Process. 1997; 45(11):2673–2681. https://doi.org/10.1109/78.650093

115. Karim F, Majumdar S, Darabi H, Chen S. LSTM fully convolutional networks for time series classification. IEEE Access. 2018; 6:1662–1669. https://doi.org/10.1109/ACCESS.2017.2779939

116. Karim F, Majumdar S, Darabi H. Insights into LSTM fully convolutional networks for time series classification. IEEE Access. 2019; 7:67718–67725. https://doi.org/10.1109/ACCESS.2019.2916828

117. Mukaka Mavuto M. A guide to appropriate use of correlation coefficient in medical research Malawi Med. J. 2012; 24(3):69–71.

118. Ortigosa-Hernandez J, Inza I, Lozano JA. Measuring the class-imbalance extent of multi-class problems. Pattern Recogn. Letters. 2017; 98:32–38. https://doi.org/10.1016/j.patrec.2017.08.002