Improving the Forecasting and Classification of Extreme Events in Imbalanced Time Series Through Block Resampling in the Joint Predictor-Forecast Space

XIAOQIAN CHEN, LALIT GUPTA, AND SPYROS TRAGOUDAS
School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University, Carbondale, IL 62901, USA
Corresponding author: Lalit Gupta (lgupta@siu.edu)

ABSTRACT A novel resampling strategy is introduced to improve the forecasting and classification accuracies of events in imbalanced time series (ITS) containing a mix of low probability extreme observations and high probability normal observations. The lag-based strategy mitigates the imbalance problem by modelling an ITS as a composition of normal and extreme observations, combining the input predictor variables and the associated forecast output into moving blocks, categorizing the blocks as extreme event (EE) or normal event (NE) blocks, and selectively resampling the blocks. Combining the predictor variables and the associated forecast enables resampling of the input and output simultaneously in the joint predictor-forecast (PF)-space. Imbalance is decreased by oversampling the minority EE blocks and undersampling the majority NE blocks. The EE blocks are oversampled using a modification of block bootstrapping and a modification of the synthetic minority oversampling technique. The Box-Cox transform is employed to decrease the pattern complexity caused by the mixing of disparate extreme and normal observations in the ITS. Convolution neural networks and long-short term memory deep neural networks (DNNs) are selected for forecast modelling and tested on a set of simulated and real sub-basin outflow ITS. The root mean square errors, forecast plots, and classification accuracies show that the hybrid forecasting and classification DNN models trained on the block-balanced training sets extracted from the Box-Cox transformed ITS dramatically outperform the corresponding baseline models which are trained directly with the ITS.

INDEX TERMS Extreme event forecasting and classification, imbalanced time series, block bootstrapping, SMOTE, deep neural networks, LSTM, CNN, box-cox transform.

I. INTRODUCTION

Extreme events such as floods, droughts, hurricanes, tsunamis, forest fires, heat waves, power outages, and stock market crisis can have disastrous impacts on human lives, environment, and infrastructure [1], [2], [3], [4], [5], [6], [7], [8], [9]. In general, the accurate forecasting of extreme events is a complex problem that is challenging researchers from a broad range of engineering and related disciplines for improved forecasting models to help decrease the devastations attributed to extreme events. The specific goal of this study is to design deep neural network (DNN) models to accurately forecast events in imbalanced time series (ITS) containing a mix of low probability extreme observations and high probability normal observations. Although the focus is on developing forecasting models, the hybrid models that are introduced output both the forecast value and the category of the forecast to obtain a better understanding of the performance of the models. The classification accuracy of extreme events can serve as a useful measure for comparing the performances of forecasting models with respect to the number of extreme events that are detected or missed. Furthermore, an extreme event can be classified correctly even if the forecasted value of the event is not very accurate.
This knowledge can contribute to improving real-time early warning systems for extreme events. Although the primary focus on DNNs, the overall forecasting strategy that is developed is application-independent and it is not tied to any specific type of supervised forecasting model. Therefore, the strategy can be adapted to improve the performance of various models that are used to forecast extreme events in a broad range of disciplines such as engineering, climatology, social sciences, ecology, hydrology, and earth sciences.

Classical forecasting models such as exponential [10], [11] and ARIMA [12], [13], are quite effective in predicting observations in normal time series but tend to perform poorly on time series with extreme events due to violations of the model assumptions. Through the inclusion of additional exogeneous variables, multi-input models such ARIMAX show improvements in forecasting extreme events when compared with ARIMA [14]. Non-linear DNN models such as recurrent long short-term memory (LSTMs) also perform effectively in normal time series forecasting problems including those that are not handled well by the classical models. However, the prediction accuracy of data-driven DNN predictors is also poor on time series with extreme events primarily due to the lack of representative training data. The main goal of this study is to improve the prediction accuracy of DNN forecasting models for time series containing a mix of normal and extreme events because they have the potential for improvement through innovative augmented training. Furthermore, there is a dearth of studies on the development of DNNs for such problems. The forecasting problem, therefore, is formulated as a supervised learning problem. The DNNs selected are the LSTM networks and convolution neural networks (CNNs) which have shown to be very successful in numerous machine learning applications. LSTMs are, in general, a clear choice for time series processing due to their recurrent structure [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]. Although CNNs were originally designed to classify images [25], they have been shown to be quite successful for time series classification [26], [27], [28], [29]. CNNs appear to be a questionable choice for forecasting time series because of their feedforward structure, however, they have also been shown to be quite successful in predicting future observations [20], [23], [24], [30]. Most of the DNN studies have focused on normal time series and a notable few have concentrated on extreme event prediction [31], [32], [33], [34]. A method that combines an auto-encoder for feature extraction with an LSTM for special event forecasting is reported in [31]. The study in [32] incorporates extreme value loss derived from extreme value theory with an adaptive memory network module to memorize extreme events for time series prediction with extreme events. A strategy using a densely connected multiscale deep CNN neural network with relative entropy loss function for calibrating rescaled output data is proposed for the prediction of extreme events and anomalous features from data [33]. The study in [34] develops a labelling scheme to train a CNN for the model-free prediction of the occurrence of extreme events both in time and in space. Clearly, the methods are quite different in their respective developments and have their own merits. In this study we introduce a novel block resampling and balancing training strategy that modifies block bootstrap resampling which is widely used in normal time series analysis and modifies the synthetic minority oversampling technique which is used extensively in balancing features in classification problems. The strategy enables the numerous supervised models designed primarily to accurately forecast normal time series to also forecast extreme events accurately without sacrificing the accuracies of normal events.

A time series with extreme events is composed of a majority of normal observations and a minority of extreme observations. Due to this imbalance, DNNs trained to predict both normal and extreme events tend to overfit and underfit with respect to the normal and extreme events, respectively. Both affect the prediction performance negatively but the underfitting problem is more serious because it results in excessively poor prediction of the more important extreme observations. The obvious approach to improve prediction performance is to increase and decrease the numbers of extreme and normal observations, respectively, through resampling. Although observation resampling is a straightforward process, inserting the oversampled extreme observations and removing the undersampled normal observations while maintaining the temporal relationships of the observations across the time series, is a non-trivial problem with no simple solution. Furthermore, resampling is applied only to the training time series and not to the test time series. Consequently, the training and test time series will have different temporal properties due to the insertions and deletions. An additional problem that affects the performance adversely is attributed to the mixing of normal and extreme events which increases the complexity of the patterns in the time series. Consequently, the prediction difficulty increases because the DNNs are faced with the additional challenge of learning these complex patterns which include transitions that occur from normal to extreme events, and vice versa.

A. AIMS AND APPROACH

This study aims specifically at enabling the numerous DNN models designed to accurately forecast events in normal time series to also accurately forecast both normal and extreme events in imbalanced time series by decreasing the negative effects of the underfitting, overfitting, and pattern complexity problems identified above. Instead of attempting to generate a balanced training time series by resampling observations, we introduce novel block resampling strategies to decrease the imbalance problem. Specifically, we introduce (a) moving blocks which combine the lag predictor variables and the forecast into blocks, (b) the categorizing of the blocks as extreme event (EE) or normal event (NE) blocks based on the forecast within each block, (c) the categorizing of predictor variables within each block as a predictor vector for an extreme or normal event, (d) a combined predictor-forecast (PF)-space in which the blocks can be resampled such that
the predictor vectors and the associated forecast values are resampled simultaneously, (e) a block bootstrapping (BB) oversampling strategy to increase the number of minority EE blocks through block duplication, (f) a modification of the synthetic minority oversampling technique (SMOTE) used in imbalanced classification problems, to increase the number of minority EE blocks through block synthesis, (g) the generation of balanced training sets for DNN training through the modified BB and modified SMOTE algorithms, (h) the use of the Box-Cox transform to decrease the pattern complexity in the imbalanced time series, and (i) objective analyses of the root-mean-square errors (RMSEs) and classification accuracies (CAs) of extreme and normal events separately and across the combined events to have a better understanding of the performance of the forecasting models with respect to each type of event. Furthermore, we introduce a modification of SMOTE to generate pseudo time series consisting of synthetic blocks as an alternative to standard block bootstrapping which generates pseudo-time series through block duplication. The RMSEs and CAs of the DNNs trained on block-balanced training sets are compared against their respective baseline forecasting models and against each other using simulated and real imbalanced time series. The baseline DNNs are trained directly with the unbalanced set of blocks extracted from the original imbalanced time series. The performances are also compared subjectively by examining plots of the predicted events against the actual events.

Combining the predictor vectors and forecast into blocks and assigning categories to the blocks is the crux of the block balancing strategy because each block carries the relevant information for predicting future events. That is, the EE blocks contain the information for predicting extreme events and the NE blocks contain information for predicting normal events. Furthermore, the predictor vectors and associated forecasts are resampled simultaneously through block resampling. Balancing blocks is equivalent to balancing the number of predictor vectors for the extreme and normal events. To the best of our knowledge, the strategy outlined to improve the forecasting of imbalanced time series through block resampling and balancing in conjunction with the Box-Cox transform is quite unique with no overlap with any reported method on imbalanced time series forecasting.

B. ORGANIZATION OF THE PAPER

This paper is quite long with many sections. A preview of the rest of the paper is described in this section in order to show the flow of the subsequent sections leading to the block diagram which summarizes the entire sequence of steps to train, test, and evaluate the performances of DNNs trained with the block balanced training sets.

Section II: describes the representations of normal time series and imbalanced time series which are used throughout the study.

Section III: describes the lag-based time series forecasting problem with respect to the time series representations. A description of hybrid forecasting-classification models for imbalanced time series which output both the forecast value and the category of the forecast is also contained in this section.

Section IV: discusses important issues related to training DNNs for time series forecasting.

Section V: presents a brief review of block bootstrapping which is widely used in normal time series and SMOTE which is used extensively in imbalanced classification because the two block resampling and balancing strategies developed in this study are based on modifications of block bootstrapping and SMOTE.

Section VI: presents the formal definitions of the joint predictor-forecast blocks which are fundamental to the development of the two block resampling and balancing strategies. The blocks are categorized as extreme event and normal event blocks and it is emphasized that balancing involves balancing these blocks and not ITS observations.

Section VII: describes the modifications applied to standard block bootstrapping and SMOTE for balancing joint predictor-forecast blocks.

Section VIII: describes different ways in which the blocks can be balanced through resampling in the PF-space.

Section IX: reviews the Box-Cox transform which is used in this study to reduce the discrepancy between the normal events and extreme events in imbalanced time series to improve performance and also discusses time series normalization which is used to facilitate training.

Section X: describes the performance measures which include the separate computations of the RMSEs and classification accuracies for the entire time series, extreme events, and the normal events. This section also describes the method for visually comparing forecast estimates.

Section XI: presents a block diagram which summarizes the entire sequence of steps to train, test, and evaluate the performances of DNNs trained with the block balanced training sets.

Section XII: describes the simulated and two real imbalanced time series used to train and test the DNN models.

Section XIII: describes the architectures and hyperparameters of the CNNs and LSTMs used in the experiments.

Section XIV: describes the series of experiments conducted to evaluate and compare the performances of the DNN forecasting models using the real and simulated time series. The results are also presented and discussed in this section.

Section XV: summarizes the contributions of this study in improving the performance of DNN models for extreme event forecasting.

Furthermore, in order to facilitate the flow from section to section, each section begins with a brief introduction and concludes with comments leading to the next section.

II. IMBALANCED TIME SERIES REPRESENTATION

In the study of time series (TS), it is important to distinguish between TS that contain extreme observations and those that do not. In order to do so, each observation will be referred to as event and each event will be assigned to one of two
categories: normal ($n$) or extreme ($e$). An extreme event can be a single observation. A TS composed solely of normal events will be referred to as a normal time series (NTS) and a TS that contains a majority of normal and a minority of extreme events will be referred to as an imbalanced time series (ITS). Extreme amplitude events in ITS can be characterized by large increases (upward-extremes), large decreases (downward-extremes), or a combination of large increases and decreases (upward & downward-extremes) in the event amplitudes. Given a univariate ITS $Z_k = (z_1, z_2, \ldots, z_K)$ consisting of a sequence of $K$ events at regular intervals of time, an event $z_i$ in an upward-extreme ITS is assigned to a category according to:

$$z_i \in \begin{cases} n, & \text{if } z_i < \tau_u \\ e, & \text{if } z_i \geq \tau_u \end{cases}$$

where $\tau_u$ is the up-extreme amplitude threshold and $e$ is used as the category assignment symbol. Similarly, events in the downward-extreme and upward & downward-extreme ITS are assigned categories according to:

$$z_i \in \begin{cases} n, & \text{if } z_i > \tau_d \\ e, & \text{if } z_i \leq \tau_d \end{cases}$$

and

$$z_i \in \begin{cases} n, & \text{if } \tau_1 \leq z_i \leq \tau_2 \\ e, & \text{otherwise} \end{cases}$$

respectively, where $\tau_d$ is the down-extreme threshold, and $\tau_1$ and $\tau_2$ are the lower and upper thresholds for segmenting the upward & down-extreme ITS.

Similar thresholding approaches can be developed to categorize extreme events with respect to event frequencies $P(z_i)$ and event durations $D(z_i)$. In general, thresholds can be selected using expert knowledge or by using suitable statistical measures such as percentiles. The most important characteristics of the types of ITS considered in this study is that the probability of an extreme event is much smaller than the probability of a normal event, that is, $P(e) \ll P(n)$. As a result, the number, $K_e$, of minority extreme events is much smaller than the number, $K_n$, of majority normal events. The event-balance ratio $\rho$, defined as $(K_e/K_n)$, provides a measure of the balance between extreme and normal events in an ITS. Ideally, the ratio should be one or close to one. Although a very small $\rho$ indicates a low proportion of extreme events, effective forecasting models can still be designed if $K_e$ is sufficiently large enough to train and test the model. In such cases, a simple undersampling of the majority events can be employed to balance the two types of events. The real challenge, which is the primary focus of this study, is to design forecasting models not only when the event-balance ratio is small but also when $K_e$ is too small to train and test the models effectively.

### III. LAG-BASED ITS FORECASTING AND HYBRID MODELS

Given an ITS $Z_k = (z_1, z_2, \ldots, z_K)$, the goal of lag-based forecasting is to determine a forecasting function $f(\cdot)$ which estimates future events of the time series from lag predictors consisting of the present event and a set of recent events from prior time steps ordered in time. The lag predictors can be solely normal, solely extreme, or a mix of normal and extreme events. If $h$ is the forecast horizon and $\delta$ is the number of lag predictors, the forecast at time $(T+h)$ which could be either a normal or an extreme event, can be expressed as

$$\tilde{z}_{(T+h)} = f(z_T, z_{(T-1)}, \ldots, z_{(T-\delta+1)})$$

where the accent $\sim$ is used to denote the estimate. The forecast is the real valued function of a $\delta$-dimensional vector, that is, $f(\cdot): R^\delta \rightarrow R$. Without loss of generality, it will be assumed that $h = 1$.

### A. HYBRID FORECASTING AND CLASSIFICATION MODELS

As stated in the introduction, the approach used in this study is to frame the forecasting problem as a supervised learning problem. Although supervised learning is applicable to both classification and prediction problems, they are treated separately because they have different goals. For classification problems, the output is a class label (category) and for prediction problems such as forecasting, the output is a real value. That is, classification models are not designed to determine real valued outputs and prediction models are not designed to output class labels. In the previous section, we defined an ITS as a time series consisting of two types of events: normal and extreme. This categorization presents us an opportunity to design models which not only predict the output forecast value but also assign the forecast to a class. For example, the categorical assignment for the forecast in the upward-extreme case is as follows:

$$\tilde{z}_{(T+h)} \in \begin{cases} n, & \text{if } \tilde{z}_{(T+h)} < \tau_u \\ e, & \text{if } \tilde{z}_{(T+h)} \geq \tau_u \end{cases}$$

We refer to models which output the forecast value and the category associated with the forecast as “hybrid forecasting-classification (HFC)” models. The classification accuracies of each type of event can be estimated by comparing the categories of $\tilde{z}_{(T+h)}$ and $z_{(T+h)}$. Along with the RMSEs, classification accuracies provide useful measures for gaining insights into the performance of a forecasting model and for comparing the performances of various forecasting models.

### IV. CNN AND LSTM DNNs

The primary reason for selecting CNNs and LSTMs in this study is they are data-driven and are, therefore, not constrained by model assumptions. Furthermore, unlike classical models, these DNNs are highly scalable. CNNs and LSTMs are well-studied DNNs which are described in detail in recent textbooks (e.g., [35], [36], and [37]) and reviewed exhaustively in recent in papers (e.g., [38], [39], and [40]). Therefore, instead of repeating these details, we focus on issues related to the design of CNNs and LSTMs for time series forecasting. Recurrent LSTMs are an obvious choice because they are specifically designed to process sequential data and, therefore, can exploit the temporal context in
time series to improve prediction accuracy. Although CNNs have been shown to yield outstanding performance in numerous classification and regression problems, their feedforward structure makes them questionable for processing temporal data. Interestingly, several notable studies have shown that CNNs can also be designed to predict future values effectively in sequential observations [20], [23], [24], [30].

A. LAG-BASED TRAINING OF DNNs FOR TIME SERIES FORECASTING

In the design of DNN predictors for time series, the series is split into a training set consisting of the first α% of the events in the TS and evaluated on the remaining (100 − α)% of the events which constitute the test set. Therefore, the training and test sets are also time series. The usual approach for training both CNN and LSTM DNN models for lag-based time series forecasting is to train the model with the training TS in its natural order. That is, the DNN is trained sequentially with the predictor vector \([z_{(T−δ+1)}, z_{(T−δ+2)}, \ldots, z_T]\) to forecast the next event \(z_{(T+1)}\) in the TS. The predictor vector, therefore, serves as contextual information for the forecast. At any given time step, the δ predictors and the target forecast form an input-output training pair. In terms of implementation, the input vectors are stored in an ordered matrix \(X\) such that the first row \(X(1,\cdot)\) contains the first predictor vector, the second row contains the next predictor vector \(X(2,\cdot)\), and so forth. The target forecast is stored in an ordered column vector \(Y\) in which the element \(Y(j)\) is the associated forecast for the corresponding predictor vector in row \(X(j,\cdot)\) of the input predictor matrix. The ordered training set can be represented compactly as \(\{X(j,\cdot), Y(j)\}_{j=1}^{K}\) pairs, where \(K\), in this case, is the number of training pairs. We refer to this training method in which the input-output pairs are extracted in an ordered fashion from the training TS as “natural-order training.”

The order of the \(\{X(j,\cdot), Y(j)\}_{j=1}^{K}\) pairs in the training set is not important for CNNs due to the non-recurrent structure. Therefore, the performance should not be affected if the pairs are randomly shuffled which is an attractive feature that is exploited for the block balancing approach developed in this study. Contrastingly, for LSTMs, the natural order of the training pairs has to be maintained to exploit the temporal context for accurate forecasting. Therefore, shuffling the training pairs can have a negative impact on the performance of LSTMs. Training DNNs with randomly ordered training pairs will be referred to as “unordered-training.” The two block balancing methods introduced in this study generate balanced but randomly ordered blocks in the training set which could be used to form balanced pseudo time series for LSTM training through block concatenation. However, as emphasized in the Introduction, there is no simple solution for balancing an ITS by inserting and deleting resampled events without destroying the fundamental temporal relationships between the events across the entire training ITS. Answering the question as to where and how to insert and/or delete events does not help because any form of insertion and/or deletion will affect the temporal properties of the time series. Moreover, given that resampling is applied only to the training set, the training and test sets will have different temporal properties. This particular problem is mitigated to some extent by resampling blocks in which the temporal patterns of the events within the blocks are maintained.

B. COMPARING DNNs

As noted earlier, the goal is to demonstrate that the performances of baseline CNNs and LSTMs which are trained with the ordered training set extracted directly from the training ITS can be improved by training them with the modified training set generated by the two block balancing methods introduced in this study. An additional goal is to compare the performances of the two DNN forecasting models. Given that the operations in CNNs and LSTMs are quite different, it is extremely difficult to compare the performances of the two DNNs in a fair manner. Our approach to compare the performances is to start with somewhat meaningless “matching architectures” to give some sense of equivalence and then adjust the parameters of both DNNs to yield similar baseline performances. The matching architectures equate the number of CNN convolution layers and LSTM hidden layers, equate the number of filters in each CNN layer to the hidden units in each LSTM layer, and use identical fully connected networks at the output. Furthermore, the two DNN predictors will be trained on the same training sets and also trained identically with respect to initialization, normalization, order of presentation, number of epochs, dropout probabilities, and optimization algorithm. The architectures and hyperparameters of the LSTMs and CNNs used in this study are described in Section XIII.

V. BLOCK BOOTSTRAPPING AND SMOTE

At this point, we re-emphasize that our strategy to mitigate the imbalance problem is block-based and not observation based. The next section presents our formal definitions of normal and extreme event blocks and Section VII describes how block bootstrapping and SMOTE are modified to balance the two types of blocks. This section presents a brief review of block bootstrapping and SMOTE to facilitate the understanding of the modifications.

A. BLOCK BOOTSTRAPPING FOR NTS

Bootstrapping is a resampling technique that is used to estimate population statistics from a given sample of size \(K\) by generating a sampling distribution by repeatedly drawing \(K\) random samples from the given sample, with replacement [41]. Bootstrapping is not suitable for time series because the correlations between the observations cannot be retained in the bootstrapped sample. That is, it is not possible to construct a pseudo-time series which has the same order-dependent statistics as the original time series from the bootstrapped sample. In the analysis of NTS, block bootstrapping (BB), a variant of the basic bootstrapping, is used to generate multiple pseudo time series from a given time series of length \(K\) by partitioning the time series into blocks, sampling the
blocks with replacement, and joining the blocks together to form pseudo time series of length \( K \) in which the correlations of the observations within the blocks are maintained [42], [43], [44], [45], [46], [47], [48], [49]. Several variants of BB which include simple BB (SBB), moving BB (MBB), circular BB (CBB), and stationary BB (STBB), have been developed. They differ primarily in the way the blocks are defined. The time series is partitioned into non-overlapping blocks in SBB [43], overlapping blocks in MBB [44], and variable length blocks in STBB [47]. CBB is a simple modification of MBB aimed at overcoming the edge effects by assuming the time series is circular [46]. For each case, a pseudo time series with the same length as the original time series is obtained by concatenating the blocks in the bootstrap sample in the order they were drawn. The selection of the optimal block length with respect to the strength of the correlation of the time series as measured by the correlogram is described in [50]. The pseudo time series generated through any form of bootstrapping suffers from between-block discontinuous at the points where the blocks are joined together. Also, important to note is that bootstrapping involves duplication of samples or blocks, therefore, no new information is added through bootstrapping.

B. SMOTE FOR IMBALANCED CLASSIFICATION

SMOTE is a well-known algorithm that is applied extensively to augment the features vectors of the minority class in imbalanced classification problems [51]. Given the training set of the minority class, the basic form of SMOTE synthesizes a minority feature vector by randomly selecting a minority feature vector from the training set (can also be looped through all minority feature vectors), randomly selecting one of its \( k \) nearest neighbors, and synthesizing a feature vector by randomly selecting a point between the two feature vectors in the feature space. The steps are repeated to generate as many synthetic feature vectors as needed for balancing. Several variants of the basic SMOTE, such as Borderline SMOTE [52], ADASYN [53], Safe-Level Smote [54], LN-SMOTE [55] and MWMOTE [56] have been developed to address some of the shortcomings of the basic algorithm described in [51].

C. SMOTE FEATURE SPACE AND BLOCK-VECTOR SPACE

In classification problems, a \( \delta \)-dimensional feature vector is regarded as a point in a fixed \( \delta \)-dimensional feature space. The ensemble of feature vectors forms a cluster in the feature space. SMOTE, by definition, synthesizes points in the feature space through interpolation between neighboring points in the cluster. Except for STBB which has blocks of varying lengths, the fixed-length blocks of BB can also be regarded as vectors and thus points in a fixed dimensional vector space. An ensemble of block vectors also forms clusters in the block-vector space. BB, therefore, corresponds to drawing randomly selected points from the cluster with replacement. A similar vector space will be defined for resampling the fixed-length predictor-forecast blocks defined in the next section.

VI. JOINT PREDICTOR-FORECAST BLOCKS AND JOINT PREDICTOR-FORECAST SPACE

The strategy to decrease the imbalance problem is based on defining minority and majority blocks in ITS and using BB or SMOTE to oversample the minority blocks and, if needed, undersample the majority blocks. The first step, therefore, is to define blocks in ITS. The blocks are defined in a unified manner so that they can be used to form balanced training sets for both CNN and LSTM training. Given that the DNN forecast models are lag-based predictors, the predictor vector is the input, and the associated forecast is the output. Taking this underlying input-output property into consideration, the criteria for grouping of events into blocks are (a) a block, in isolation criteria, should contain the input predictor vector required to produce the target forecast and (b) the block, in isolation, should also contain the output forecast. We, therefore, combine the input predictor vector and the output forecast into a block. We refer to blocks which combine the \( \delta \)-dimensional predictor vector and associated forecast value as joint “predictor-forecast” (PF) blocks. The \((\delta + 1)\)-dimensional space formed by combining predictor vectors and associated forecasts is referred to as the “joint predictor-forecast (PF)-space.

Combining the input predictor vector and output forecast into a block is worthy of elaboration. In imbalanced classification problems, the goal is to balance the output labels by balancing the associated input feature vectors. Bootstrapping or SMOTE is applied to the minority-class feature vectors as a means for generating duplicates or synthetic feature vectors, respectively. The class of the duplicates or synthetic feature vectors is the same as the class of the minority feature vectors set, therefore, there is no need to synthesize the output label. An additional point that is obvious but is relevant to note for this study is that the input and output are always held separately and not combined into a single vector. As noted in Section IV, in lag-based forecasting of NTS, the model is trained to predict the output forecast value \( z_{(T+1)} \) given the input \( \{ z_{(T-\delta+1)}, z_{(T-\delta+2)}, \ldots z_T \} \). In this case too, the input \( X(j,:) \) and the associated output \( Y(j) \) are held separately in the training set \( \{X(j,:), Y(j)\}\}_{j=1}^K \). Duplicates and synthetic input predictor vectors can be obtained by applying bootstrapping or SMOTE blindly to the input vectors in prediction problems. This, however, is quite meaningless because the output value associated with each synthetic input would not be known. Synthetic values of the output can also be obtained through bootstrapping or SMOTE, but this too would not help because the assignment of a synthetic output to a synthetic input is unknown. Therefore, our approach of combining the predictor vector and the forecast in the same block enables bootstrapping or SMOTE to yield input predictor vectors and associated outputs simultaneously. The inputs and outputs are extracted from the duplicated or synthetic blocks to form the separated input-output training pairs after balancing. Essentially, block balancing is aimed at balancing the two types of predictor vectors.
ITS Z

\[ B_2^{[e]} = \{n_2, n_3, e_4\}; \ B_3^{[e]} = \{n_3, e_4, e_5\}; \ B_4^{[e]} = \{e_4, e_5, e_6\}; \ B_7^{[e]} = \{n_7, n_8, e_9\}; \ B_9^{[e]} = \{e_9, n_{10}, e_{11}\}; \]
\[ B_{10}^{[e]} = \{n_{10}, e_{11}, e_{12}\}. \]

**FIGURE 1.** Generation of \((\delta + 1)\)EE blocks.

### A. GENERATION OF PF BLOCKS

The PF blocks are generated by overlapping moving blocks of \((\delta + 1)\) consecutive events. The stride used is equal to unity, therefore, the overlap between successive blocks is \(\delta\) events. The joint-training matrix formed by the PF blocks is represented by the \([K \times (\delta + 1)]\) matrix \(\{XY(j,:)\}_{j=1}^{K}\) where, \(XY(j,:)\) denotes the concatenation of \(X(j,:)\) and \(Y(j)\) into a block. Hereafter, the term “joint-training matrix” is reserved for the matrix that combines the predictors and forecast and the term “training set” is used for the predictor-forecast separated representation.

In the following description of the PF blocks which constitute the rows of \(\{XY(j,:)\}_{j=1}^{K}\), it is assumed that the extreme events in the ITS have already been identified by using an appropriately selected threshold as described in Section II. The training ITS is represented by \(Z\), the length of \(Z\) by \(K_Z\), the numbers of extreme and normal events by \(N_e\) and \(N_n\), respectively, and the block length by \(L\). Furthermore, it is assumed that \(K_Z\) is divisible by \(L\) so that the number of blocks \(b = (K_Z/L)\) is an integer. The block length \(L\) is fixed to \((\delta + 1)\). In accordance with the criteria for grouping events into blocks, the first \(\delta\) events constitute the predictor vector, and the last event is the target forecast. The ITS is partitioned into a set of \((K_Z - L) + 1\) overlapping PF blocks of length \(L\) such that the \(jth\) block in the matrix is given by

\[ XY(j,:) = \{e_i\}_{i=1}^{jL} \cup \{e_{j+1}\}_{j=1}^{(j+1)\delta - 1}, j = 1, 2, \ldots, (K_Z - L) + 1. \]  

(6)

The blocks may or may not contain extreme events. Without losing generality, it will be assumed that the extreme events are upward-extremes. Any block in the set is defined as an extreme event block, represented by \((\delta + 1)\)EE, if the last event, that is, the \((\delta + 1)st\) event is an extreme event. The remaining blocks are referred to as normal event blocks represented by \((\delta + 1)NE\). The number of extreme events in a \((\delta + 1)EE\) block ranges from 1 to \((\delta + 1)\). The \((\delta + 1)EE\) block terminating at \(q\) is given by

\[ X_eY_e(q - L + 1, :) = \{e_i\}_{i=q}^{q+L-1}; \ e_q \in e. \]  

(7)

The \((\delta + 1)NE\) blocks can also contain extreme events with the number ranging from 0 to \(\delta\). Similarly, the \((\delta + 1)NE\) block terminating at \(r\) is given by

\[ X_eY_e(r - L + 1, :) = \{e_i\}_{i=r}^{r+L-1}; \ e_r \in n. \]  

(8)

Figure 1 is a simple illustration of how \((\delta + 1)EE\) blocks are generated assuming \(\delta = 2\), that is, \(L = 3\). The boxed row is the ITS \(Z\) of length \(K_Z = 13\), and the normal and extreme events occurring at times \(i\) and \(j\) are denoted by \(n_i\) and \(e_j\), respectively. In order to simplify the notations, the blocks \(X_eY_e(a, :)\) and \(X_nY_n(b, :)\) are denoted by \(B_{0}^{[e]}(a)\) and \(B_{0}^{[n]}(n)\), respectively. The pool of \((\delta + 1)EE\) blocks \(B_{1}^{[e]}, B_{2}^{[e]}, B_{3}^{[e]}, B_{4}^{[e]}, B_{7}^{[e]}, B_{9}^{[e]}, B_{10}^{[e]}\), generated from this ITS are listed below the ITS.

In Figure 1, observe that the events near the start and end of the ITS occur in fewer blocks because they are not included as frequently as those further away from the ends. This problem can be remedied simply by extending the ITS by \((L - 1)\) events in a circular fashion. The resulting ITS of length \((K_Z + (\delta + 1))\) can then be partitioned into \(K_Z\) overlapping blocks of length \(L\). Hereafter, it will be assumed that the ITS has been extended circularly. As a result of circularity, the sum of \((\delta + 1)EE\) and \((\delta + 1)NE\) blocks is equal to the number of extreme events \(N_e\) and the number of normal events \(N_n\), respectively. Therefore, the total number \((N_e + N_n)\) of blocks is equal to \(K_Z\). With the circularity assumption, the \((\delta + 1)NE\) blocks in Figure 1 are given by:

\[ B_{1}^{[n]} = \{n_1, n_2, n_3\}; \ B_{5}^{[n]} = \{e_5, e_6, n_7\}; \]
\[ B_{8}^{[n]} = \{e_6, n_7, n_8\}; \ B_{8}^{[n]} = \{n_8, e_9, n_{10}\}; \]
\[ B_{11}^{[n]} = \{e_{11}, e_{12}, n_{13}\}; \ B_{12}^{[n]} = \{n_{12}, n_{13}, n_1\}; \]
\[ B_{13}^{[n]} = \{n_{13}, n_1, n_2\}. \]

The \((\delta + 1)EE\) and \((\delta + 1)NE\) blocks are combined in their natural order to form the natural-ordered joint-training matrix \(\{XY(j,:)\}_{j=1}^{K_Z}\). In order to construct a block-balanced joint-training matrix, BB or SMOTE is applied to the \((\delta + 1)EE\) blocks extracted from \(\{XY(j,:)\}_{j=1}^{K_Z}\).

Summarizing this section, PF-blocks are introduced as the first key step in the effort to decrease the imbalance problem in ITS. The overlapping blocks are generated easily using the moving block approach. The manner in which the blocks are combined and assigned categories is the crux of the block balancing strategy because it enables simultaneous resampling of the predictor vectors and associated forecast quite elegantly in a single step. The PF blocks offer a high degree of flexibility because both block bootstrapping and SMOTE can be applied, with the appropriate modifications, to generate block-balanced joint-training matrices. The predictor vectors and forecast are separated easily after resampling for training set generation. The next section describes how block bootstrapping and SMOTE are modified to oversample the \((\delta + 1)EE\) blocks.
VII. BLOCK BOOTSTRAPPING AND SMOTE IN THE PF-SPACE

We modify block bootstrapping and SMOTE so that they are applicable to balancing the \((\delta + 1)\)EE and \((\delta + 1)\)NE blocks through resampling in the \((\delta + 1)\)-dimensional PF-space.

A. \((\delta + 1)\) BLOCK BOOTSTRAPPING

Unlike BB for NTS analysis which is aimed at generating multiple pseudo time series of the same length as the original by bootstrapping all blocks, our goal is to generate block-balanced trainings set by selectively oversampling only the minority \((\delta + 1)\)EE blocks. Multiple block-balanced training sets can be generated by repeating the process. The categorization of blocks and selectively bootstrapping of the \((\delta + 1)\)EE blocks are the key modifications applied to regular BB. Because bootstrapping is conducted in the \((\delta + 1)\)-dimensional PF-space, we refer to this procedure as \((\delta + 1)\)BB. The next step is to combine the original set of undersampled \((\delta + 1)\)NE blocks, set of original \((\delta + 1)\)EE blocks, and the resampled \((\delta + 1)\)EE blocks into the balanced joint-training matrix.

B. \((\delta + 1)\) SMOTE

As noted earlier, bootstrapping is basically a duplicating operation which does not add new information into the training set. An alternate approach for generating a set of balanced blocks is to apply SMOTE to synthesize the required number of \((\delta + 1)\)EE blocks from the original set of \((\delta + 1)\)EE blocks. It was explained in Section VI that SMOTE cannot be applied directly to forecasting problems because the input variables and output values are separated. However, SMOTE can be applied to the \((\delta + 1)\)EE blocks because the input predictor vectors and the associated output forecast values are combined in the blocks. The space in which SMOTE is applied is the \((\delta + 1)\)-dimensional PF-space, therefore, this modification is referred to as \((\delta + 1)\)SMOTE. The synthetic blocks are combined with the original set of extreme event blocks and the undersampled normal event blocks to construct the balanced joint-training matrix. Note that any version of SMOTE can be used to synthesize the \((\delta + 1)\)EE blocks.

Figure 2 illustrates the application of the \((\delta + 1)\)SMOTE algorithm to generate synthetic blocks for training forecasting models given the ITS segment:

\[
\{.,.,e_4,e_5,e_6,n_7,n_8,e_9,n_{10},e_{11},e_{12},...,\}. \tag{9}
\]

A lag of \(\delta = 2\) is assumed to yield the original joint training matrix consisting of input-output pairs composed of 2 input lag predictor variables and the associated output. The figure only shows the original partial joint-training matrix consisting of the four \((\delta + 1)\)EE blocks that are involved in block synthesis. In this example, it is assumed that SMOTE is applied to generate 4 synthetic blocks to double the number of \((\delta + 1)\)EE blocks. The synthesized blocks are represented by \(\{a_1,a_2,a_3\},\{b_1,b_2,b_3\},\{c_1,c_2,c_3\},\) and \(\{d_1,d_2,d_3\}\). The figure also shows the SMOTE augmented joint-training matrix and training set consisting only of extreme events.

VIII. PF BLOCK BALANCING

This section presents a formal description of how \((\delta + 1)\)BB and the \((\delta + 1)\)SMOTE algorithm are applied to produce block-balanced block joint-training matrices. Balance is improved by resampling the two types of PF blocks. Equating the number of blocks by undersampling the majority \((\delta + 1)\)NE blocks is ruled out because that number is assumed to be too small to train and test forecast models effectively. If the number of majority blocks is not excessively large, the minority blocks can be oversampled to equalize the numbers of both types of blocks. A combination of oversampling and undersampling can be used if the number of majority events is excessively large. Given that the number of extreme events vary across both types of blocks, it is likely that the numbers of extreme and normal events will be different upon block balancing. However, this is not a problem because the goal
is to balance the number of extreme event and normal event predictor vectors which is similar to balancing the number of feature vectors and not the contents of the feature vectors in binary imbalanced classification problems.

A. \((\delta + 1)\)BB BALANCED JOINT-TRAINING MATRICES

If the goal is to oversample the minority blocks to equal the number of majority blocks, the block-balanced joint-training matrix consisting of an equal number of \((\delta + 1)\)EE and \((\delta + 1)\)NE blocks is obtained by drawing a bootstrap sample of \((N_n - N_e)\) blocks from the minority \((\delta + 1)\)EE block pool. Using the compact notations for blocks introduced in Section VI, the joint-training matrix of size \(2N_n\) blocks can be written as

\[ \begin{bmatrix} B_{i}^{[e,n]} \end{bmatrix}_{i=1}^{2N_n} = \begin{bmatrix} B_{i}^{[e]} \end{bmatrix}_{i=1}^{N_e} \cup \begin{bmatrix} B_{i}^{[e]} \end{bmatrix}_{i=1}^{(N_n-N_e)} \cup \begin{bmatrix} B_{i}^{[n]} \end{bmatrix}_{i=1}^{N_n}, \]

where \(B_{i}^{[e]}\) is the \(i\)th bootstrapped \((\delta + 1)\)EE block and the symbol \(\cup\) denotes the combining operation. In this case, each pool will have \(N_n\) blocks and the combined joint-training matrix will contain almost twice the original number of blocks given that \(N_e\) is quite small.

In order to avoid overfitting, the size of the balanced training set can be made smaller by oversampling \((\delta + 1)\)EE blocks and undersampling \((\delta + 1)\)NE blocks. The joint-training matrix size, for example, can be adjusted to be equal to the number of original blocks \(K_Z\) by adding a bootstrap sample of size \((N_n - N_e)/2\) of \((\delta + 1)\)EE blocks and undersampling the \((\delta + 1)\)NE blocks to \(K_Z/2\). The balanced block-set for this case can be written as

\[ \begin{bmatrix} B_{i}^{[e,n]} \end{bmatrix}_{i=1}^{K_Z} = \begin{bmatrix} B_{i}^{[e]} \end{bmatrix}_{i=1}^{N_e} \cup \begin{bmatrix} B_{i}^{[e]} \end{bmatrix}_{i=1}^{(N_n-N_e)/2} \cup \begin{bmatrix} B_{i}^{[n]} \end{bmatrix}_{i=1}^{K_Z/2}. \]

A similar approach can be used to create a balanced joint-training matrix of any desired size.

B. \((\delta + 1)\)SMOTE BALANCED JOINT-TRAINING MATRICES

The procedure for balancing using \((\delta + 1)\)SMOTE algorithm is the same as that used for balancing the \((\delta + 1)\)BB set in the sense that same number of additional \((\delta + 1)\)EE blocks are needed for balancing. The main difference is that additional \((\delta + 1)\)EE blocks needed for balancing and adjusting the joint-training matrix size are synthesized. If undersampling is employed, the blocks are removed randomly from the original pool of \((\delta + 1)\)NE blocks.

In general, the block-balanced joint-training matrix can be represented by \(\{\tilde{X}(\tilde{j},:)\}_{\tilde{j}=1}^{\tilde{K}}\), where \(\tilde{X}(\tilde{j},:)\) can be a BB or SMOTE oversampled \((\delta + 1)\)EE block, original \((\delta + 1)\)EE block, or an original \((\delta + 1)\)NE block. The number of blocks \(\tilde{K}\) is equal to \(K_Z\) if original size of the training set is maintained. The block-balanced training set is \(\{\tilde{X}(\tilde{j},:)\}_{\tilde{j}=1}^{\tilde{K}}\) obtained by separating the predictor vector and the target forecast in \(\{\tilde{X}(\tilde{j},:)\}_{\tilde{j}=1}^{\tilde{K}}\). Unlike the natural-ordered joint-training matrix \(\{XY(j,:):j=1,...,K\}\), the ordering in \(\{\tilde{X}(\tilde{j},:)\}_{\tilde{j}=1}^{\tilde{K}}\) depends on where the oversampled blocks are inserted and which undersampled blocks are deleted. That is, the balanced joint-training matrix and, hence, the balanced training set is unordered.

C. BLOCK-BALANCED PSEUDO-TS

The block bootstrapping approach to generate pseudo-time NTS can be used to generate an approximately balanced pseudo-TS having the same length \(K_Z\) as the original training ITS by concatenating a bootstrap sample of size \(b = (K_Z/(\delta + 1))\) blocks (rows), in the order they were drawn, from \(\{\tilde{X}(\tilde{j},:)\}_{\tilde{j}=1}^{\tilde{K}}\). The resulting pseudo-ITS can contain duplicates of \((\delta + 1)\)EE blocks if \((\delta + 1)\)BB is used and will contain synthetic \((\delta + 1)\)EE blocks if \((\delta + 1)\)SMOTE is used for oversampling. Multiple balanced pseudo-ITS can be generated by repeating the operation which will change the number of block duplications or the synthetic blocks as well as the order of blocks for concatenation. Although no attempt is made to generate balanced pseudo-ITS for training LSTM models directly, this method can be exploited to produce synthetic pseudo normal time series as described next.

D. \((\Lambda)\)SMOTE FOR NORMAL TIME SERIES

The procedure to generate balanced pseudo-ITS can be modified to generate pseudo-NTSs by concatenating synthetic blocks for normal time series analysis. If the block size selected for analyses is \(\Lambda\), a total of \(b = (K_Z/\Lambda)\) blocks are synthesized using SMOTE and concatenated. Multiple realizations of pseudo-NTSs are obtained by repeating the procedure. We refer to this method as \((\Lambda)\)SMOTE. Since SMOTE is a form of interpolation between closely spaced points in feature space, the properties of the synthesized blocks in the pseudo NTS should not differ by much from the properties of the blocks of the original NTS. This SMOTE-based procedure for synthesizing pseudo-NTS is a notable contribution in general time series analysis because it offers an alternative to standard block bootstrapping which generates pseudo-NTS through block duplication.

IX. ITS TRANSFORMATIONS AND NORMALIZATION

One of the problems that makes accurate forecasting of ITS difficult is that the forecasting model must essentially learn patterns of two types of highly differing events as well as the large transitions between the two event types. In general, time series with simpler patterns simplify the prediction problem and lead to more accurate forecasts. Therefore, the accuracy can be expected to improve by decreasing the disparities between the normal and extreme events. For ITS characterized by extreme amplitudes, mathematical transforms can be used to reduce the gap between normal and extreme events in order to improve the consistency across the entire ITS. The Box-Cox transform [57] is often used to convert skewed data into data that has approximately Gaussian distribution. For time series, the transform is used to stabilize the variance across the time series. When applied to ITS, the transform tends to map a narrow range of low values in the input into a wider range of output values. It also maps a wide range of high values into a narrower range of output values. Consequently, the transitions between the normal and extreme events are decreased making the transformed ITS more homogeneous across its entire length. The Box-Cox
inverse transform is given by

\[ y_i = \begin{cases} 
\log(z_i), & \text{if } \lambda = 0 \\
(z_i^\lambda - 1)/\lambda, & \text{otherwise},
\end{cases} \]

(12)

where \( z_i \) and \( y_i \) are the corresponding events in the original ITS \( Z \) and the Box-Cox transformed ITS \( Y \), respectively. The inverse transform is given by

\[ z_i = \begin{cases} 
\exp(y_i), & \text{if } \lambda = 0 \\
\exp((\lambda y_i + 1)^{1/\lambda}), & \text{otherwise}.
\end{cases} \]

(13)

The optimal \( \lambda \) for a given time series is determined through maximum likelihood estimation. The above set of equations are applicable only to positive \( z_i \). Another set of equations involving two parameters are included in [57]. Alternatively, the Yeo-Johnson transform can be used for time series containing both positive and negative events [58]. Figure 3 illustrates the application of the Box-Cox transform on a simulated ITS. It is clear that the transformed ITS is more homogeneous or “simpler” than the original ITS with respect to the amplitudes. The benefits of using the Box-Cox transform can be determined by comparing the forecasts conducted on the transformed ITS against the forecasts conducted on the original ITS.

A. NORMALIZATION

Various normalization technique such as min-max, z-score, vector, and robust scaling are applied to the time series to facilitate training of deep learning models. In this study, we employ min-max normalization to the Box-Cox transformed time series. In general, min-max normalization is not effective for ITS because the normalization is biased by the large maximum extreme value. As a result, the low-amplitude normal events are suppressed when normalized. Figure 4(a) shows direct min-max normalization applied to the ITS shown in Figure 3(a). Min-max normalization after Box-Cox transforming the ITS is shown in Figure 4(b). The suppression of the normal events is clear in Figure 4(a), whereas the normalization in Figure 4(b) is more balanced.

X. PERFORMANCE MEASURES

A set of objective and visual measures are used to evaluate the performance of each forecasting model and to compare the performances across the models. The objective measures include the root-mean-square-error (RMSE) and classification accuracy (CA).

A. ROOT MEAN SQUARE ERRORS

We define RMSEs separately for extreme and normal events and also include the usual RMSE computed across the combined events. Splitting the RMSEs in this manner provides a better understanding of the performance of the model with respect to each type of event. The RMSEs for the extreme, normal, and combined events are defined as follows:

\[ \text{RMSE}_e = \left[ \frac{1}{N_e} \sum_{j=1}^{N_e} (\hat{z}_j^e - z_j^e)^2 \right]^{1/2}, \]

(14)

\[ \text{RMSE}_n = \left[ \frac{1}{N_n} \sum_{j=1}^{N_n} (\hat{z}_j^n - z_j^n)^2 \right]^{1/2}, \]

(15)

\[ \text{RMSE}_{n,e} = \left[ \frac{1}{(N_e + N_n)} \left( \sum_{j=1}^{N_e} (\hat{z}_j^e - z_j^e)^2 + \sum_{j=1}^{N_n} (\hat{z}_j^n - z_j^n)^2 \right) \right]^{1/2}, \]

(16)

where, \( z_j^e \) and \( z_j^n \) are the actual and predicted events, respectively. In the above equations, \( N_e \) and \( N_n \) are used to represent the number of extreme and normal events in the test set.

B. CLASSIFICATION ACCURACIES

Although CA is not generally used to measure the performance of a predictor, we include CAs in this study because it provides additional performance information not reflected in RMSEs. As in the RMSEs, we define CAs for extreme, normal, and combined events in the following manner:

\[ CA_e = \left( \frac{\hat{N}_e}{N_e} \right) \times 100, \]

(17)
LSTM models during testing are identical. The CNN and LSTM models are tested on the moving testing. The CNN and LSTM models are tested on the moving

where, $\tilde{N}_e$ and $\tilde{N}_n$ are the number of extreme and normal events that are classified correctly. Extreme events are correctly classified if the predicted output is greater or equal to the threshold $\tau_u$ that was set to assign the events to their respective categories. That is, an extreme event in the test set is classified correctly if $\tilde{z}_j^{[e]} \geq \tau_u$. Similarly, a normal event is classified correctly if $\tilde{z}_j^{[n]} < \tau_u$. $CA_e$ and $CA_n$ can be interpreted as the sensitivity and specificity of the forecast models, respectively. $(N_e - \tilde{N}_e)$ is a useful comparison measure because it indicates the number of extreme events that are missed by the models.

C. VISUAL ANALYSES

The performance of a forecasting model is also evaluated through visual analysis of the plots of the predicted events against the actual events in the test set. In this case too, the analysis can be split into the visual comparisons of extreme event segments, normal event segments, and across the entire ITS.

XI. BLOCK DIAGRAM OF THE BLOCK-BALANCING DNN MODELS FOR ITS FORECASTING

Figure 5 is an illustration of the complete block-balancing strategy developed to forecast events in ITS. The top part shows the training phase while the testing and evaluation phases are shown at the bottom. The entire ITS, represented by $Z$, is split into the training and test sets consisting of the first naturally ordered $\alpha\%$ of the events and the remaining naturally ordered $(100 - \alpha)\%$ events, respectively. The training set is represented by $Z_t$ and the test set by $Z_{(100-\alpha)}$. The block diagram also illustrates the following important difference in the training and testing phases: block balancing through resampling is performed only during training and not during testing. The CNN and LSTM models are tested on the moving lag-predictor blocks, of size $\delta$, extracted in their natural order from the test ITS. Importantly, the inputs to the CNN and LSTM models during testing are identical.

XII. TIME SERIES DATA

The performances of the LSTM and CNN forecasting models are evaluated using a simulated NTS, a simulated ITS and two real ITS. These time series are described next.

A. NTS AND ITS SIMULATION

The advantage of training and testing forecast models on simulated ITS is that the performance trends can be judged with respect the expected trends. If the simulated ITS has the general characteristics of real ITS, a poor match between the experimental and expected trends can serve as warning that the model is flawed. For example, if no improvement is seen in the prediction and classification accuracies of a model with respect to the baseline model, it would be a clear indication that model would perform poorly if deployed on real ITS. Contrarily, a good match is encouraging because it is an initial validation of the model and there is hope that the model will also perform as expected on real ITS. Furthermore, “for what if analyses”, the characteristics of the ITS can be varied by adjusting the simulation parameters which include the length, polarities of the extreme events, the imbalance ratio, amplitude differences between normal and extreme events, durations of extreme events, and intervals between extreme events.

1) NORMAL TIME SERIES SIMULATION

Figure 6(a) shows the normal 20,000-event simulated TS that is used in the experiments. This NTS was simulated by an AR($\delta = 3$) model of the form:

$$z_t = a_0 + a_1z_{(t-1)} + a_2z_{(t-2)} + a_3z_{(t-3)} + \epsilon_t,$$

$$t = 4, 5, \ldots, 20000,$$

(20)

with $a_0 = 1, a_1 = 0.2, a_3 = 0.1,$ and $\epsilon_t = N(0, 1)$, where $N(\mu, \sigma^2)$ is a Gaussian random variable with mean $\mu$ and variance $\sigma^2$.

2) ITS SIMULATION

Figure 6(b) which is the same as Figure 3(a), shows the simulated ITS used in the evaluations. Narrow chunks of extreme events containing varying amplitudes were added.
to the NTS in Figure 6(a) to obtain the simulated ITS containing upward extreme events. The shapes of the chunks were roughly triangular and the amplitudes within the chunks exceeded the maximum amplitude of $z_t$. The chunks were placed in segments in which the amplitudes of the normal TS were approximately rising. The durations and the gaps between the chunks varied across the time series. A total of 1,000 extreme events were added to the 20,000 NTS events for the event balance ratio $\rho$ to be 0.05.

### B. REAL OUTFLOW ITS

1) REAL HOURLY OUTFLOW ITS

The real ITS selected for performance evaluations consisted of the hourly outflow rate of Sub-basin 5 within the Tuolumne River Basin located in Sonoma and Mendocino Counties in California. The hourly data, consisting of 185,544 observations, were collected over the 1980-2001 time period across the Tuolumne Basin from the gridded meteorological dataset [59]. We selected this particular outflow time series, shown in Figure 7, for performance evaluations because it contained segments that have the typical characteristics of extreme events. The large infrequent increases (upward-extremes) in the outflow over random and brief time periods are clearly observable in the figure.

2) REAL DAILY OUTFLOW ITS

The real daily outflow ITS shown in Figure 8 was obtained by averaging across 24 hours, the original hourly outflow time series displayed in Figure 7. The reason for doing so was to evaluate the performance of the forecast models on...
ITS which contain spike-like extreme events which are visible in the figure. In general, spike-like extreme events present a challenge for any resampling technique.

XIII. DNN ARCHITECTURES AND HYPERPARAMETERS
This section describes the architectures and hyperparameters that need to be defined to implement the various DNN models.

A. SELECTION OF LAG
Given that the DNNs forecast models are lag-based, the lag is an important parameter that has to be set in the models. The lag \( \delta \) has to be selected to form the \((\delta + 1)\)EE and \((\delta + 1)\)NE blocks and to define the input dimension of the DNNs. The lag selected for the simulated ITS was the same as that used to simulate the NTS \((\delta = 3)\). For the real flow data, the significant lag determined from the partial autocorrelation function (PACF) [60] of the training ITS was initially selected. The final lag that was selected was determined by experimenting with lags around the initial lag and picking the one that yielded the best performance.

B. DNN MODEL PARAMETERS
Baseline LSTM
Number of hidden layers = 4
Number of units in each layer = 16, 32, 64, 16
Numbers of neurons in dense layer = 48, 24, 12, 1
Output activation function = Linear
Training options: initialization = uniform random, optimizer = Adam, learning rate = 0.001, number of epochs = 30, drop out probabilities = 0.15

Baseline CNN
Number of convolution layers = 4
Filter dimensions in convolution layer 1 = \((1 \times \delta)\); Number of filters = 16
Filter dimensions in convolution layer 2 = \((3 \times 1)\); Number of filters = 32
Filter dimensions in convolution layer 3 = \((3 \times 1)\); Number of filters = 64
Filter dimensions in convolution layer 4 = \((3 \times 1)\); Number of filters = 16
Pooling layer dimension = \((2 \times 1)\); stride = 2

Numbers of neurons in the fully connected layer = 48, 24, 12, 1
Output activation function = Linear
Training options: initialization = uniform random, optimizer = Adam, learning rate = 0.001, number of epochs = 30, drop out probabilities = 0.15

XIV. EXPERIMENTS AND RESULTS
Several experiments were designed to evaluate the performances of the forecasting models on the simulated NTS, simulated ITS and the two real ITS described in Section XII. The first set of experiments were aimed at demonstrating that CNNs and LSTMs can be designed to forecast events in NTS quite accurately but perform poorly in forecasting extreme events in ITS formed by adding infrequent extreme events to the NTS. The remaining set of experiments were aimed at improving the forecasting of events in the simulated and real ITS. For each type of ITS, the following CNN and LSTM models were developed:
(a) The model that yielded the best results trained on the normalized ITS. The best result was determined experimentally. The architecture and hyperparameters of this model defined the baseline model.
(b) The baseline model trained on the balanced normalized training set generated by \((\delta + 1)\)BB.
(c) The baseline model trained on the balanced normalized training set generated by \((\delta + 1)\)SMOTE.
(d) The baseline model trained on the balanced Box-Cox normalized training set generated by \((\delta + 1)\)BB.
(e) The baseline model trained on the balanced Box-Cox normalized training set generated by \((\delta + 1)\)SMOTE.

That is, 6 CNN and 6 LSTM models were designed for each ITS. The goal of these experiments was to systematically demonstrate that the forecasting and classification accuracies of extreme events can be improved significantly by training the baseline models with the training sets generated by the two block-balancing methods from the non-transformed ITS and the Box-Cox transformed ITS. The threshold to categorize events as extreme or normal in ITS was the 95th percentile for the event balance ratio \(\rho\) to be 0.05. For each experiment involving ITS forecasting, the mean RMSEs and mean CAs of the extreme events, normal events, and combined events are presented compactly in tables in this section and as bar graphs in the Appendix. The mean RMSEs and mean CAs were computed across 50 test repetitions which involved initializing the DNN models with different sets of random weights. The bar graphs include error bars which represent +/-1 standard error of the mean. Plots of the original test ITS and the mean forecast are also included in the Appendix for visual evaluations. In all cases, the training set consisted of the first 70% events and the test set consisted of the remaining 30% of events. In order to maintain fairness across the ITS related experiments, the training sets for the CNN and LSTM models were extracted from the same joint-training matrix. In every experiment, the input during testing consisted of \(\delta\)-dimensional moving predictor vectors presented in the order they appeared in the test ITS.
A. SET A EXPERIMENTS - SIMULATED TIME SERIES

1) EXPERIMENT A1
Aim A1: Determine the performance of the DNN forecasting models on normal time series directly.

This set of experiments involved forecasting the simulated NTS shown in Figure 6(a). Natural-order training (see Section IV) was used to train the DNNs. No block resampling was employed during training. The models were tested on the moving predictor blocks in the order they appeared in the test NTS. The RMSEs are shown in Figure 9 and the superimposed plots of the original test NTS and the mean forecast are shown in Figures 10. The results show that the CNN and LSTM models perform equally well in forecasting NTS. Classification accuracies were not computed for these experiments because the time series contained only normal events.

Conclusion A1: The LSTM and CNN models are quite effective in forecasting NTS. Although CNNs are not designed for processing sequential data, the performance is comparable to that of LSTMs which are designed specifically for sequential data processing.

2) EXPERIMENT A2
Aim A2: Determine the performance of the DNN forecasting models on normal time series using block resampling.

The goal of this set of experiments is to have a better understanding how changes in the natural order of the blocks in the training set through resampling affects the performance of the models on NTS. Unlike Experiment A1 which used natural-order training, the DNNs were trained using unordered training (see Section IV). The ordered predictor blocks extracted from the training NTS in Experiment 1 were shuffled randomly to form the training set. In this particular case, a repetition involved a different shuffle. The results, shown in Figures 9 and 10, indicate that the CNN model is not affected by changing the natural order of the blocks. However, the performance of the LSTM model was negatively impacted by the shuffling. Furthermore, the wider error bar indicates relatively higher variability in the LSTM forecast.

Conclusion A2: It can be expected that, in general, the improvement in performance of LSTM models may not be as high as CNN models when block resampling is employed.

3) EXPERIMENT A3
Aim A3: Determine the performance of the baseline DNN models on simulated ITS.

This set of experiments are similar to A1 except that the time series is the simulated ITS shown in Figure 6(b). Other than normalization, no other operation was performed on the training ITS. That is, DNNs were trained directly using natural-order training. No block resampling was used during training. The results are summarized in Table 1 and the plots are displayed in Figures 11-13 in the Appendix. The best result in each column of the table is highlighted in bold font. The results for the extreme events, normal events, and combined events in the figures are denoted by E, N, and C on the horizontal axis, respectively. Clearly, the inclusion of extreme events adversely affects the RMSEs. The CAs of the extreme events are quite poor. The plots also confirm the poor performance in forecasting both normal and extreme events. The LSTM models perform better than the CNN models for the normal segments of the ITS. These results serve as the baselines for comparisons with the models that use block resampling.

Conclusion A3: This experiment confirms that the baseline CNN and LSTM models trained directly with ITS perform poorly on both normal and extreme events.

4) EXPERIMENT A4
Aim 4: Determine the performance of the DNN models trained on balanced training sets generated by (δ + 1)BB.

In this set of experiments involving the forecasting of the simulated ITS shown in Figure 6(b), the DNN models were trained with the balanced training sets generated by (δ + 1)BB. The results are summarized in Table 1 and the plots are shown in Figures 11-13 in the Appendix. The improvements over the respective baseline models are clearly evident. In an overall sense, the RMSEs, CAs, and the plots show that the CNN models perform better than the LSTM models in forecasting the extreme events while the forecasting of normal events is comparable. Observe that the numbers of extreme events detected correctly by the CNN model is higher than the numbers of the LSTM models. This observation supports the relatively better performance of the CNN.

Conclusion A4: The performance of the baseline DNN models can be improved by training them with the balanced training sets by generated by (δ + 1)BB.

5) EXPERIMENT A5
Aim 5: Determine the performance of the DNN models trained on balanced training sets generated by (δ + 1)SMOTE.

This set of experiments are similar to those in Experiment A4 except that the balanced training sets were generated by (δ + 1)SMOTE. The results are summarized in Table 1 and the plots are shown in Figures 11-13. The improvements over the baseline and (δ + 1)BB trained models are evident. In general, the CNN models outperform the LSTM models in forecasting and classifying the extreme events.
Conclusion A5: The performance of the baseline CNN and LSTM models can be improved by training them with the balanced training sets generated by \((\delta + 1)\text{SMOTE}\), respectively. The models using \((\delta + 1)\text{SMOTE}\) balancing outperform those using \((\delta + 1)\text{BB}\) on both extreme and normal events. Furthermore, the CNN models using \((\delta + 1)\text{SMOTE}\) balancing perform better than the LSTM models using \((\delta + 1)\text{SMOTE}\) balancing on extreme events. These conclusions drawn from forecasting simulated ITS are an initial validation of the performance improvements that can be expected by applying the block balancing approaches developed in this study.

6) EXPERIMENTS A6-A8

Aim 6-8: Determine the performance of the DNN models on Box-Cox transformed simulated ITS.

Experiments A3-A5 are repeated on the Box-Cox transformed simulated ITS in Figure 3(b). The results of the experiments, labelled A6, A7, and A8, are summarized in Table 1 and the plots are shown in Figures 14-16 in the Appendix. When compared with non-transformed results, it is observed that the RMSEs are substantially smaller for both extreme and normal events and the CAs are substantially higher for the extreme events. Moreover, the plots show a much better match between the original and the mean forecasted ITS. For the extreme events, the overall trends across the RMSEs, CAs, and plots are the same as observed for the non-transformed ITS.

Conclusion A6-A8: The application of the Box-Cox transform has a dramatic effect in improving the performance of the block-balanced DNN models over their respective baseline models. The best average performance is obtained by the CNN models trained on the \((\delta + 1)\text{SMOTE}\) block-balanced training sets generated from the Box-Cox transformed ITS.

B. SET B EXPERIMENTS - REAL HOURLY OUTFLOW ITS

Experiments A3-A8 were repeated on the real outflow ITS shown in Figure 7. The lag used for this ITS was 72. The results of the Set B experiments, labelled B3-B8, are summarized in Table 2 and the plots are displayed in Figures 17-22 in the Appendix. The overall performance trends for the real hourly ITS (Table 2) and the simulated ITS (Table 1) are similar for the extreme events.
TABLE 1. Performance of DNN forecasting models on the simulated ITS.

| Transformation | Predictor | Experiment | RMSE | (CA)% |
|----------------|-----------|------------|------|-------|
|                |           |            | RMSE | RMSE | C_A  | C_A,σ |
| Non CNN A3     | 43.36     | 27.97      | 20.79| 147/300=49.00| 5679/5679=99.68| 97.15 |
| A4             | 20.79     | 23.36      | 17.24| 228/300=76.00| 5670/5679=99.53| 98.35 |
| A5             | 14.65     | 21.97      | 13.54| 239/300=86.33| 5683/5679=99.75| 99.08 |
| LSTM A3        | 44.65     | 24.54      | 19.55| 666/300=22.00| 5697/5679=100.00| 96.1 |
| A4             | 21.97     | 20.79      | 17.36| 187/300=62.33| 5695/5679=99.96| 98.08 |
| A5             | 16.38     | 19.94      | 15.0 | 203/300=67.67| 5697/5679=100.00| 98.38 |
| BoxCox CNN A6  | 53.61     | 2.29       | 3.04 | 196/300=65.33| 5688/5679=99.84| 98.12 |
| A7             | 6.34      | 2.93       | 0.97 | 296/300=98.67| 5684/5679=99.77| 99.72 |
| A8             | 1.61      | 0.31       | 0.37 | 300/300=100.00| 5697/5679=100.00| 100.00|
| LSTM A6        | 34.41     | 1.35       | 3.86 | 178/300=59.33| 5691/5679=99.89| 97.87 |
| A7             | 8.89      | 0.30       | 0.73 | 263/300=87.67| 5699/5679=99.88| 99.38 |
| A8             | 5.06      | 0.36       | 0.61 | 294/300=98.00| 5697/5679=100.00| 99.9 |

TABLE 2. Performance of DNN forecasting models on the real hourly ITS.

| Transformation | Predictor | Experiment | RMSE | (CA)% |
|----------------|-----------|------------|------|-------|
|                |           |            | RMSE | RMSE | C_A  | C_A,σ |
|                |           |            | RMSE | RMSE | C_A  | C_A,σ |
| Non CNN B3     | 150/300   | 910        | 3771 | 123/34=36.76| 3653/3657=99.88| 98.23 |
| B4             | 105/348   | 693        | 2897 | 679/34=92.51| 3636/3657=99.86| 97.72 |
| B5             | 138/355   | 299        | 2745 | 719/34=97.96| 3652/3657=99.92| 98.88 |
| LSTM B4        | 214/613   | 957        | 5185 | 109/34=14.85| 3639/3657=99.87| 98.19 |
| B5             | 118/529   | 115        | 347  | 697/34=94.96| 3652/3657=99.99| 98.89 |
| BoxCox CNN B6  | 140/185   | 25         | 192  | 134/34=18.26| 3635/3657=99.99| 98.38 |
| B7             | 38/24     | 30         | 110  | 270/34=80.10| 3630/3657=98.84| 98.81 |
| LSTM B5        | 26/48     | 45         | 97   | 724/34=98.64| 3651/3657=99.96| 99.93 |
| B7             | 36/09     | 67         | 304  | 99/34=13.49| 3657/3657=100.00| 98.29 |
| B8             | 43/02     | 97         | 181  | 706/34=96.19| 3631/3657=99.87| 99.8 |
| A8             | 30/09     | 75         | 134  | 723/34=98.50| 3654/3657=99.97| 99.94 |

Conclusions Set B: The performance trends of the CNN and LSTM models on the hourly flow time series are similar to the trends observed for the synthetic ITS. The results in Table 2 and the plots in Figure 22 confirm that the CNN models trained on the (δ+1) SMOTE block-balanced training sets generated from the Box-Cox transformed ITS yield the best overall performance.

C. SET C EXPERIMENTS - REAL DAILY OUTFLOW ITS

Experiments B3-B8 were repeated on the real daily outflow ITS shown in Figure 8. The lag used for this ITS was 3. The results are shown in Table 3 and Figures 23-28 in the Appendix. The overall performance trends are similar to those observed for the simulated and hourly ITS. In this case, the application of the Box-Cox transform has a large impact in decreasing the RMSEs and increasing the CAs. It is somewhat difficult to judge whether the average performance of the (δ + 1)SMOTE trained CNN models is better than that of the LSTM models in Figure 28. An overshoot and an undershoot is observed for the extreme events in the CNN and LSTM plots, respectively. However, the classification results in Table 3 show that the number of extreme events detected by the CNN models is higher than the number detected by the LSTM models. That is, the LSTM models miss more extreme events than the CNN models. The average forecast of the CNN models in Figure 28(e) can be judged to be good given the difficulty in forecasting spike-like extreme events.

Conclusions Set C: The overall trends are similar to those observed for the simulated and hourly ITS.

D. RESAMPLING AND BALANCING APPLIED TO SVM

Although the primary focus of this study is to improve the forecasting accuracies of DNNs for extreme events, the training sets generated by (δ + 1)BB and (δ + 1)SMOTE can also be used to improve extreme event forecasting by other supervised forecasting models. In order to demonstrate this fact, SVM regression models were implemented to forecast the three ITS and the results are presented in Table 4. The results in Table 4 include the baseline results obtained by training the models directly with the ITS and the results obtained by training the models with the non-transformed and Box-Cox transformed training sets generated by (δ + 1)BB and (δ + 1)SMOTE. The SVM performance trends are consistent with those obtained with the DNN models which confirms that the block balanced training sets generated by (δ + 1)BB and (δ + 1)SMOTE can not only improve the performance of DNN models for extreme event forecasting but can also improve the performance of other supervised models designed primarily for normal time series forecasting. When compared with the DNN results in Tables 1-3, it is clear that the DNN models outperform the SVM models.

For comparison purposes, balanced pseudo training time series were also generated from the three ITS using an observation resampling method described in [61]. This particular method was selected because it is closely related to our SMOTE balancing strategy in the sense that it involves the application of SMOTE for imbalanced time series forecasting in conjunction with standard regression algorithms including SVMs. The method oversamples observations in bins of consecutive observations exceeding a relevance
threshold, uses a 2-step process which first applies SMOTE to synthesize cases giving preference to nearby observations and then applies interpolation to synthesize the forecast value, undersamples cases in low relevance bins, and uses the F1 score to evaluate overall performance. Other than focusing on the general goal of improving prediction accuracy through SMOTE resampling, there is little commonality with our block-balancing approach. This method, referred to as SM_TPhi, was implemented using code extracted from: https://github.com/nunompmoniz/TSResampStrat_JDSA2017, to generate observation balanced training time series from the three ITS used in the experiments. The baseline SVM forecasting models for each ITS in Table 4 were trained with the respective SM_TPhi generated training sets and tested on exactly the same test ITS used to evaluate the performances of the \((\delta + 1)\text{BB}\) and \((\delta + 1)\text{SMOTE}\) SVM models in Table 4. The results, included in Table 4, show that SM_TPhi offers improvements over the baseline models, however, the improvements are not as dramatic as those obtained by \((\delta + 1)\text{BB}\) and \((\delta + 1)\text{SMOTE}\). The reasons for the relatively poor performance are due to the problems that were identified with attempts to balance observations in time series in the Introduction and in Section IV.

E. SUMMARY OF RESULTS

The experimental results confirm that natural-order trained DNNs can forecast events in NTS accurately but fail to accurately forecast extreme events in ITS. The training sets generated by the two block resampling and balancing strategies enabled the DNNs to forecast both the extreme and normal events in ITS accurately. The DNNs trained using the two block-balancing methods on the Box-Cox transformed ITS outperformed the baseline models quite dramatically for forecasting both extreme and normal events. In general, the best results were obtained by the DNN models trained on the \((\delta + 1)\text{SMOTE}\) block-balanced training sets generated from the Box-Cox transformed ITS. Furthermore, the CNN models performed better than the LSTM models. For the extreme events in the simulated ITS, the average RMSE of the baseline CNN models dropped from 43.36 to 1.61 when the CNN models were trained with the Box-Cox and \((\delta + 1)\text{SMOTE}\) block-balanced combination.
This reduction amounts to a \((43.36/1.61) = 26.93\)-fold relative improvement in the RMSE. Correspondingly, the relative improvements for the extreme events in the hourly and daily ITS were \((150430/2648) = 56.80\)-fold and \((94863/2159) = 43.94\)-fold, respectively. The improvement in classifying extreme events was also significant. The corresponding relative improvements in the classification accuracies of the extreme events were \((100/49) = 2.04\)-fold, \((98.64/16.76) = 5.88\)-fold, and \((96.77/19.36) = 4.98\)-fold. These high relative improvements confirm that block-balanced training in conjunction with the Box-Cox transform can improve the forecasting and classification accuracies of extreme events quite substantially. The RMSEs of the normal events also decreased substantially. Furthermore, the improvements in forecasting extreme and normal events were attained without sacrificing the classification accuracies of the normal events.

The dramatic improvements in performance with respect to the baseline models using PF-block balancing can be attributed to reducing the underfitting of extreme events and overfitting of normal events encountered in training models with ITS as well as the reduction of the discrepancies between the extreme and normal events through the Box-Cox transform. The temporal patterns of the events are maintained within the blocks and resampling generates blocks containing both the predictor vectors and associated forecast. Balancing PF-blocks through resampling is quite similar to resampling feature vectors in classification problems which has proven to be highly successful in improving imbalanced classification. Furthermore, the two PF-block resampling methods are based on modifications of well-established methods of block-bootstrapping and SMOTE. Based on the discussion in Section IV, it was expected that LSTM models would not benefit as much as the CNN models due to the unordered training sets generated by block balancing. The improved performance of the CNN models can be attributed to the fact that the filters in the convolution layers are capable of extracting complex features by coupling the within-block temporal information across the predictor variables in the first convolution layer and cross-coupling the successive coupled information in the subsequent convolution layers. Moreover, on average, the CNN models trained five times faster than the LSTM models.

Finally, it is important to note that the two block-balancing methods combined with the Box-Cox transform are quite general in the sense that they are independent of any particular ITS forecasting application and are not tied to any specific supervised forecasting model. Consequently, the performances of a wide range of supervised forecasting models can be improved by training the models with the block-balanced training sets generated from the Box-Cox transformed ITS.

**XV. CONCLUSION**

The goal of this study was to overcome the imbalance curse that plagues the performance of forecast models on time series containing a mix of a low probability extreme-observations and high probability normal-observations. Instead of attempting to balance observations directly, a novel block balancing strategy was introduced to decrease the imbalance problem. At the core of this strategy was the manner in which the blocks were defined. The overlapping moving blocks combined the input predictor vectors and the associated output forecast values and each block was assigned to one of two categories: extreme or normal. By doing so, the predictor vectors for the extreme and normal events were separated and the input predictor vector and the associated output forecast value in a block could be resampled simultaneously in the PF-space. After block balancing through resampling, the training set was formed by separating the predictor vectors and the associated forecasts. The Box-Cox transform was used as an additional means to improve forecasting accuracy by decreasing the complexity of the ITS.

A detailed set of experiments was developed to evaluate the performance of the HFC DNNs trained on \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) block-balanced training sets generated from non-transformed and Box-Cox transformed ITS. The results on simulated and real ITS showed that block-balancing combined with the Box-Cox transform dramatically outperformed the baseline DNNs models on forecasting both extreme and normal events. The improvements in the classification of extreme events were also dramatic. In general, the best performances were obtained by the CNN models trained on the \((\delta + 1)SMOTE\) block-balanced training sets generated from the Box-Cox transformed ITS. The balanced training sets developed for DNN training can be used in conjunction with other supervised regression algorithms. Furthermore, the proposed modification of SMOTE to generate multiple pseudo-time series with synthetic blocks as an alternative to block duplication through standard block bootstrapping is a noteworthy contribution for normal time series analysis. Future efforts will focus on extending the block balancing strategies to accurately forecast and classify events in multivariate imbalanced time series.

In summary, the accurate forecasting of extreme events in ITS is a challenging problem. Models designed to accurately forecast normal events tend to forecast extreme events poorly. We have clearly demonstrated that the performance of baseline DNN forecasting models which perform quite well on normal events but perform poorly on extreme events can be improved dramatically by incorporating the generalized training strategy which combines block resampling and balancing with the Box-Cox transform. Although DNNs are a good choice because of their data-driven property and scalability, the generalized training strategy can be used in conjunction with other supervised forecasting models. Therefore, and most importantly, a wide range of forecasting models developed in various fields of engineering and related disciplines can be improved to decrease the disastrous impacts of infrequent dangerous extreme events through the incorporation of the generalized training strategy developed in this study.
APPENDIX

FIGURE 11. RMSEs of the baseline DNN and the DNN models on simulated ITS using $(\delta + 1)BB$ and $(\delta + 1)SMOTE$ balancing.

FIGURE 12. CAs of the baseline DNN and the DNN models on simulated ITS using $(\delta + 1)BB$ and $(\delta + 1)SMOTE$ balancing.
FIGURE 13. Original simulated test ITS (blue) and forecasted ITS (red).
FIGURE 14. RMSEs of the baseline, \((i + 1)BB\) and \((i + 1)SMOTE\) balanced DNN models for the Box-Cox simulated ITS.

FIGURE 15. CAs of the baseline, \((i + 1)BB\) and \((i + 1)SMOTE\) balanced DNN models for the Box-Cox simulated ITS.
FIGURE 16. Original Box-Cox transformed simulated test ITS (blue) and forecasted ITS (red).
FIGURE 17. RMSEs of the baseline DNN and the DNN models on real hourly ITS using \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balancing.

FIGURE 18. CAs of the baseline DNN and the DNN models on real hourly ITS using \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balancing.
FIGURE 19. Original real hourly test ITS (blue) and forecasted ITS (red).
FIGURE 20. RMSEs of the baseline DNN, the \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balanced DNN models for the Box-Cox transformed real hourly ITS.

FIGURE 21. CAs of the baseline DNN, the \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balanced DNN models for the Box-Cox transformed real hourly ITS.
FIGURE 22. Original Box-Cox transformed real hourly test ITS (blue) and forecasted ITS (red).
FIGURE 23. RMSEs of the baseline DNN and the DNN models on real daily ITS using $(\delta + 1)BB$ and $(\delta + 1)SMOTE$ balancing.

FIGURE 24. CAs of the baseline DNN and the DNN models on real daily ITS using $(\delta + 1)BB$ and $(\delta + 1)SMOTE$ balancing.
FIGURE 25. Original real daily test ITS (blue) and forecasted ITS (red).
**FIGURE 26.** RMSEs of the baseline DNN, the \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balanced DNN models for the Box-Cox transformed real daily ITS.

**FIGURE 27.** CAs of the baseline DNN, the \((\delta + 1)BB\) and \((\delta + 1)SMOTE\) balanced DNN models for the Box-Cox transformed real daily ITS.
FIGURE 28. Original Box-Cox transformed real daily test ITS (blue) and forecasted ITS (red).
REFERENCES

[1] S. Albeverio, V. Jentsch, and H. Kantz, “Extreme Events in Nature and Society.” Berlin, Germany: Springer, 2006.

[2] C. C. Ummenhofer and G. A. Meehl, “Extreme weather and climate events with ecological relevance: A review,” Phil. Trans. Roy. Soc. B, vol. 372, no. 1723, 2017. Art. no. 20160135.

[3] L. E. McPhillips, H. Chang, M. V. Chester, Y. Depietri, E. Friedman, N. B. Grinn, J. S. Kominoski, T. McPherson, P. Méndez-Lázaro, E. J. Rossi, and J. S. Shiva, “Defining extreme events: A cross-disciplinary review,” Earth’s Future, vol. 6, no. 3, pp. 441–455, Mar. 2018.

[4] F. M. Ren, B. Trewin, M. Brunet, P. Dushmanta, A. Walter, O. Baddour, and M. Korber, “A research progress review on regional extreme events,” Adv. Climate Change Res., vol. 9, no. 3, pp. 161–169, Sep. 2018.

[5] C. Raymond, R. M. Horton, J. Zscheischler, O. Martinis, A. AghaKouchak, J. Balch, S. G. Bowen, S. J. Carnargo, J. Hess, K. Kornhuber, M. Oppenheimer, A. C. Ruane, T. Wahl, and K. White, “Understanding and managing connected extreme events,” Nature Climate Change, vol. 10, no. 7, pp. 611–621, Jul. 2020.

[6] L. H. Brooks, W.-R. Poganyetz, and S. Vögele, “Extreme events defined—A conceptual discussion applying a complex systems approach,” Futures, vol. 115, Jan. 2020. Art. no. 102940.

[7] J. E. Walsh, T. J. Ballinger, E. S. Euskirchen, E. Hanna, J. Märd, J. E. Overland, H. Tangen, and T. Vihma, “Extreme weather and climate events in northern areas: A review,” Earth-Sci. Rev., vol. 209, Oct. 2020, Art. no. 103324.

[8] W. A. Robinson, “Climate change and extreme weather: A review focusing on the continental United States,” J. Air Waste Manag. Assoc., vol. 71, no. 10, pp. 1186–1209, Oct. 2021.

[9] K. L. Ebi, J. Vanos, J. W. Baldwin, J. E. Bell, D. M. Hondula, N. A. Errett, K. Hayes, C. E. Reid, S. Saha, J. Spector, and P. Berry, “Extreme weather and climate change: Population health and health system implications,” Annu. Rev. Public Health, vol. 42, pp. 293, Apr. 2021.

[10] G. E. Box, M. G. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control. Hoboken, NJ, USA: Wiley, 2015.

[11] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting. Cham, Switzerland: Springer, 2002.

[12] E. S. Gardner, H. Tangen, and T. Vihma, “Exponential smoothing: The state of the art—Part II,” Int. J. Forecasting, vol. 22, no. 4, pp. 637–666, Oct. 2006.

[13] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, Forecasting With Exponential Smoothing: The State Space Approach. Berlin, Germany: Springer, 2008.

[14] G. P. D. Søhlin, “Analysis of the effect of fuel price policy on Jakarta inflation by using multi-input intervention model,” AIP Conf. Proc., vol. 2014, no. 1, 2018. Art. no. 020125.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[16] A. Sagheer and M. Kott, “Time series forecasting of petroleum production using deep LSTM recurrent networks,” Neurocomputing, vol. 323, pp. 203–213, Jan. 2019.

[17] A. Khodabakhsh, I. Ari, M. Bakır, and S. M. Alagoz, “Forecasting multivariate time-series data using lstm and mini-batches,” in Proc. 7th Int. Conf. Contemp. Issues Data Sci. Cham, Switzerland: Springer, 2019, pp. 121–129.

[18] Z. Karevan and J. A. K. Suykens, “Transductive LSTM for time-series event forecasting with neural networks,” in Proc. 52th Int. Conf. Contemp. Issues Data Sci. Cham, Switzerland: Springer, 2021.

[19] Z. Kar and J. A. K. Suykens, “Transductive LSTM for time-series prediction: An application to weather forecasting,” Neuro Inf. Process. Lett., vol. 125, pp. 1–9, May 2020.

[20] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutta, “AI in healthcare: Time-series forecasting using statistical, neural, and ensemble architectures,” Frontiers Big Data, vol. 3, p. 4, Mar. 2020.

[21] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, “Financial time series forecasting with deep learning: A comprehensive literature review: 2005–2019,” Appl. Soft Comput., vol. 90, May 2020, Art. no. 106181.

[22] B. Lindemann, T. Müller, H. Vietz, N. Jazdzi, and M. Weyrich, “A survey on long short-term memory networks for time series prediction,” Proc. CBRP, vol. 99, pp. 650–655, Jan. 2021.

[23] R. Casado-Vara, A. M. D. Rey, D. Pérez-Palau, L. de-la-Fuente-Vallen, and J. M. Corcho, “Web traffic time series forecasting using LSTM neural networks with distributed asynchronous training,” Mathematics, vol. 9, no. 4, pp. 421, Feb. 2021.

[24] P. Lara-Benitez, M. Carranza-García, and J. C. Riquelme, “An experimental review on deep learning architectures for time series forecasting,” Int. J. Neural Syst., vol. 31, no. 3, Mar. 2021, Art. no. 2130001.
[52] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-smote: A new over-
sampling method in imbalanced data sets learning,” in Proc. Int. Conf.
Intell. Comput. Berlin, Germany: Springer, 2005, pp. 878–887.
[53] H. He, Y. Bai, E. A. Garcia, and S. Li, “ADASYN: Adaptive syn-
thetic sampling approach for imbalanced learning,” in Proc. IEEE Int.
Joint Conf. Neural Netw., IEEE World Congr. Comput. Intell., Jun. 2008,
pp. 1322–1328.
[54] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, “Safe-level-
SMOTE: Safe-level-synthetic minority over-sampling technique for han-
dling the class imbalanced problem,” in Proc. Pacific–Asia Conf. Knowl.
Discovery Data Mining. Berlin, Germany: Springer, 2009, pp. 475–482.
[55] T. Maciejewski and J. Stefanowski, “Local neighbourhood extension of
SMOTE for mining imbalanced data,” in Proc. IEEE Symp. Comput. Intell.
Data Mining (CIDM), Apr. 2011, pp. 104–111.
[56] S. Barua, M. Islam, X. Yao, and K. Murase, “MWMOTE–majority
weighted minority oversampling technique for imbalanced data set learn-
ing,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 2, pp. 405–425, Feb. 2012.
[57] G. E. Box and D. R. Cox, “An analysis of transformations,” J. Roy. Stat.
Soc. B. Methodol., vol. 26, no. 2, pp. 211–243, 1964.
[58] I.-K. Yeo and R. A. Johnson, “A new family of power transformations to
improve normality or symmetry,” Biometrika, vol. 87, no. 4, pp. 954–959,
Dec. 2000.
[59] B. Livneh, T. J. Bohn, D. W. Pierce, F. Munoz-Arriola, B. Nijssen, R. Vose,
D. R. Cayan, and L. Brekke, “A spatially comprehensive, hydrometeoro-
logical data set for Mexico, the U.S., and southern Canada 1950–2013,”
Sci. Data, vol. 2, no. 1, Aug. 2015, Art. no. 150042.
[60] P. J. Brockwell and R. A. Davis, “ARMA models,” in Introduction to Time
Series and Forecasting. New York, NY, USA: Springer, 2016, pp. 73–96.
[61] N. Moniz, P. Branco, and L. Torgo, “Resampling strategies for imbalanced
time series,” in Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA),
Oct. 2016, pp. 282–291.

**XIAOQIAN CHEN** received the B.S. degree in electronic & communication engineering and the M.S. degree in electronic & information engineering from the Civil Aviation University of China, in 2014 and 2018, respectively. She is currently pursuing the Ph.D. degree with the School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University, Carbondale, IL, USA. Her research interests include developing generalized machine learning models for diverse applications related to imbalanced time series forecasting and classification.

**SPYROS TRAGOUDAS** received the Diploma degree in computer engineering from the University of Patras, Greece, in 1986, and the M.S. and Ph.D. degrees in computer science from The University of Texas at Dallas, in 1988 and 1991, respectively.

He is currently a Professor and the Director of the School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University Carbondale (SIUC), and the Director of the Consortium for Embedded Systems, SIUC. He has held prior appointments with the Faculty Member of the Electrical and Computer Engineering Department, The University of Arizona, and the Faculty Member of the Computer Science Department, SIUC. His current research interests include the areas of VLSI design/test automation, embedded systems, and deep learning. He has published over 300 papers in peer-reviewed journals and conference proceedings, including five best paper awards. His research has been funded by federal agencies and industry. He has served and currently serving on the editorial board of several journals and the technical program committees of many conferences. He has supervised numerous Ph.D. dissertations and M.S. theses.

**LALIT GUPTA** received the B.E. degree (Hons.) in electrical engineering from the Birla Institute of Technology and Science, Pilani, India, the M.S. degree in digital systems from Brunel University, Middlesex, U.K., and the Ph.D. degree in electrical engineering from Southern Methodist University, Dallas, TX, USA. He is currently a Professor Emeritus with the School of Electrical, Computer, and Biomedical Engineering (ECBE), Southern Illinois University, Carbondale (SIUC), IL, USA. He was a Professor with the ECBE School, SIUC, until May 2020, where he also worked as the Director of the Biomedical Engineering Program. His research interests include developing machine learning models for diverse applications ranging from automatic target recognition to neural time series classification. He has received funding from the Army Research Office, Sea-gate Technology, and Cleveland Medical Devices, Neuronetrix, and Think-A-Move to conduct research on NIH funded projects. He was also a Co-PI on a 1.5 million grant from NSF aimed at increasing retention and graduate rates in the College of Engineering, SIUC. He has served as an Associate Editor for the Pattern Recognition journal and has numerous publications in the areas of multisensor information-fusion strategies, neural networks, neural time series analysis and classification, and human–machine interfacing.