Evaluating Preprocessing Strategies for Time Series Prediction Using Deep Learning Architectures

Sajitha Naduvil-Vadukootu
Georgia State University
Atlanta, Georgia

Rafal A. Angryk
Georgia State University
Atlanta, Georgia

Pete Riley
Predictive Science Inc.
San Diego, CA 92121

Abstract

We propose a novel approach to combine state-of-the-art time series data processing methods, such as symbolic aggregate approximation (SAX), with very recently developed deep neural network architectures, such as deep recurrent neural networks (DRNN), for time series data modeling and prediction. Time series data appear extensively in various scientific domains and industrial applications, yet the challenges in accurate modeling and prediction from such data remain open. Deep recurrent neural networks (DRNN) have been proposed as promising approaches to sequence prediction. We extend this research to the new challenge of the time series prediction space, building a system that effectively combines recurrent neural networks (RNN) with time series specific preprocessing techniques. Our experiments show comparisons of model performance with various data preprocessing techniques. We demonstrate that preprocessed inputs can steer us towards simpler (and therefore more computationally efficient) architectures of neural networks (when compared to original inputs).

1 Introduction

Time series data consist of real valued measurements of multiple parameters at equal intervals. Time series prediction tasks assume that future values in the series are a function of historical values of the same series. Electrocardiogram (ECG) analysis in medicine, stocks prediction in finance, energy usage prediction in power grids, weather prediction in meteorology and solar activity prediction in space weather are a few examples of important real-life applications of time series modeling and prediction.

Being naturally adept at recognizing shape-related or color-related patterns (e.g. patterns contained in images), humans can visually recognize some patterns that frequently occur in time series data with relative ease. Statistical and machine learning methods of modeling and predicting time series attempt to go one step further to uncover the nontrivial patterns. Inherent complexity of time series data stems from two aspects, namely, high dimensionality (with each time step being a dimension) and unique temporal properties (monotonic trends, seasonal variations or temporal autocorrelations). Deep recurrent neural networks (DRNN) have been proposed as promising approaches for sequence modeling and prediction approaches, but so far, extension to the time series domain has been sparse.

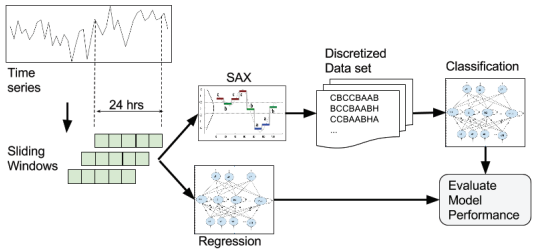

Figure 1: Overview of the prediction approach

In this work, we extend the sequence prediction approach to explore the suitability of deep learning algorithms to predicting time series data. Figure 1 shows an overview of the overall components and processes that constitute the system. We approach the time series prediction task via regression and classification. Regression is applied for predicting real values with input given as real values in a time series segment. While typical classification tasks aim to predict a label applied to a time series (e.g. predicting whether an Electroencephalogram indicates presence of stroke), we take a different approach. Input time series values are discretized using symbolic aggregate approximation (SAX), and the classification then predicts the next symbol in the sequence. Thus our classification approach is, in essence, a variation of sequence prediction. The predicted value indicates a range of possible values instead of a real value as given by regression approach. We report appropriate metrics for evaluating the efficiency of both approaches. The focus of this work is on effectiveness of preprocessing techniques. Therefore, the paper does not discuss novel deep neural network architectures or activation functions, nor does it claim superior scalability or accuracy compared to existing systems. These discussions are out of scope for this paper. Section 1 introduces the problem domain and gives an overview of time series modeling and prediction approaches. Section 2 gives related work in existing time series prediction and deep learning methods. Section 3 describes the proposed method in detail and experimental setup, and results are given in section 4. Section 5 contains conclusions and comments about

future work.

2 Background and Related Work

Moving average techniques (e.g. Auto Regressive Integrated Moving Average) and exponential smoothing techniques (e.g. Holt-Winters) (Asteriou and Hall 2011) are popular statistical methods applied to time series prediction domain. Supervised classification using machine learning algorithms such as support vector machines (SVM), decision trees (DT) and artificial neural networks (ANN) (Bishop 2006) aim to learn the parameters that define the best possible model by scanning the data repeatedly and adjusting parameter values with the goal of minimizing the error between the predicted outcome and the expected outcome. An extensive review by Schmidhuber et el. (Schmidhuber 2015) shows the expansion of ANN into deep learning aided by improvements in processing power using graphical processing units (GPU). In contrast to traditional artificial neural networks, deep neural networks typically apply multiple layers and sparse connections between layers, making the networks 'deep' and more complex. Each layer models an abstract and progressively lower level representation of the input, thus learning salient features in the input and recognizing patterns in them automatically. Deep neural networks have been found to be effective in tasks that are easy for humans, but incredibly difficult for machines, such as recognizing objects. Deep learning algorithms emulate human learning processes (e.g., vision for recognizing objects) in the structure of the network. Deep learning is said to be a good fit for problems where the information needed to complete the task is entirely contained in the input and the information to be learned is inherently structural. Deep recurrent neural networks (DRNN) are deep neural networks where feedback connections help the network learn from past input values, which makes them adept at learning sequences. Some of the variations of RNN include (1) Elman RNN (ERNN) (Elman 1990) which uses traditional artificial neurons, but adds weights in hidden layers connected to previous inputs, (2) Long short term memory (LSTM) (Greff et al. 2015) which uses a network that is built of units that consist of a memory cell and gates that control the memory cell's ability to accept input, provide output and remember values and (3) Gated recurrent unit (GRU) (Chung et al. 2014) which uses a simpler architecture based on gates but claims to match the performance of LSTM while being computationally effective. RNN are trained using algorithms based on gradient descent learning, such as: Stochastic gradient descent (SGD) (Bottou 2010), Adadelta (Schaul, Zhang, and LeCun 2013), Adagrad (Duchi, Hazan, and Singer 2011), Adam (Kingma and Ba 2014) and Rmsprop (Tieleman and Hinton). Various configurations of RNN networks have been effectively used in a variety of application areas, including: speech recognition (Graves, Mohamed, and Hinton 2013), handwriting analysis (Graves 2013), natural language modeling (Graves and Schmidhuber 2005) (Mesnil et al. 2013) and audio analysis and classification (Chung et al. 2014) (Eck and Schmidhuber 2002) (Lyu, Wu, and Zhu 2015). Efforts have been made to extend the success of RNN with sequences into problems involving modeling and predicting time series (Lee et al. 2009) related to traffic flow prediction (Lv et al. 2015) and energy prediction (Busseti, Osband, and Wong 2012). In addition to RNN, deep belief networks (DBN) (Kuremoto et al. 2014) and deep convolutional networks (Zheng et al. 2014) have also been explored. Piece-wise aggregate approximation (PAA) (Faloutsos, Ranganathan, and Manolopoulos 1994) and symbolic aggregate approximation (SAX) (Lin et al. 2007) are techniques that compresses the time series while preserving enough information for further searching, mining or learning tasks. PAA splits the time series into segments of approximately equal size and averages the values in a segment. SAX maps the approximated values to symbols using a predefined alphabet by splitting the time series value domain into equal probability regions. In addition to dimensionality reduction, applying PAA and SAX allows application of natural language and sequence prediction related deep learning classifier models and techniques to time series prediction problem.

3 Methodology

Our approach to evaluating the effectiveness of combining preprocessing strategies with deep learning methods is summarized in Figure 2. The various components of the system are: (1) preprocessing pipeline, (2) hyper parameter optimization, (3) training, (4) validation and (5) evaluation. The following subsections describe these components and their working in detail.

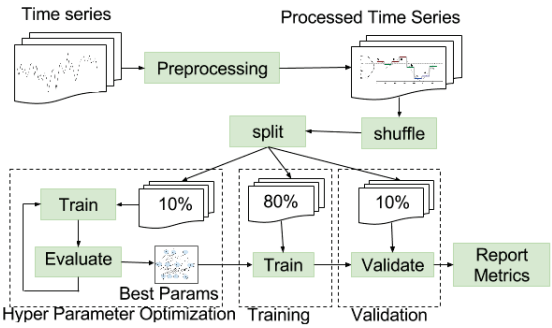

Figure 2: System Architecture

Preprocessing Pipeline

The preprocessing pipeline combines a series of steps with the goal of compressing the input time series and transforming it into a representation suitable for applying deep learning models. Figure 3 shows the stages in the pipeline. The specific configuration of this pipeline, including the processing steps and their order is determined through exploratory data analysis (EDA). EDA is geared towards understanding the data exploratory visualizations (histograms, box plots, scatter plots etc.) and correlation analysis which bring out the relationships and other patterns. The cleaning step in the pipeline can include (1) detecting N/A values (2) detecting gaps and (3) outlier analysis. Affected instances may either be removed, replaced with interpolated values or preserved depending on application needs. A sliding window approach

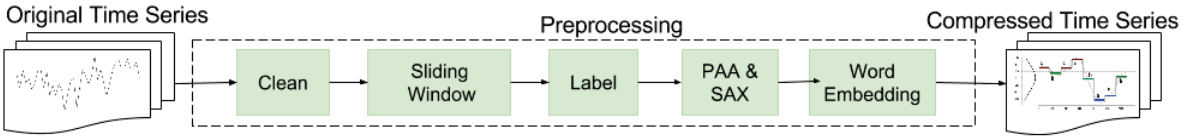

Figure 3: Various Stages of a Preprocessing Pipeline

is used to transform a continuous time series into discrete sequences. The window length, corresponding to the length of the resulting sequences, and the step size can be adjusted. For example, a sequence 1-2-3-4-5-6 with sliding window size of 3 and step size of 1 outputs a set of sequences {1-2-3, 2-3-4, 3-4-5, 4-5-6}. Labeling the sequences with meta-data encodes additional, potentially relevant, information about the input data. The meta-data labels are not fed into the artificial neural network model as an input feature, but are used to stratify or balance the data set for training and testing purposes. PAA and SAX further compresses and encodes the time series as a sequence of symbols. If the original time series is of length w and the PAA is applied to it with a number of segments n, the time series is compressed by a factor of $\frac{n}{w}$. If the number of segments is set too high, PAA will not result in good compression; at the same time, if the number of segments is set too low, relevant information may be lost. Each SAX symbol (the set of symbols is called the SAX alphabet) represents a slice of values in the domain. Too large an alphabet will result in effective compression, while too small an alphabet will lose information. Word embedding (Mikolov et al. 2013) is a technique used in natural language modeling. In our approach, rather than learning the embedding via unsupervised training, we use preset random embedding to serve the purpose of translating a sequence of SAX symbols into a unique real-valued matrix. The length of the embedding vector should be kept minimal to avoid complexity, but long enough to ensure uniqueness in mapping.

Hyper Parameter Optimization

Hyper parameters in a neural network model are parameters that define the model and do not change during the learning process (e.g., number of hidden layers and neurons, learning rate, learning algorithm used etc.). Hyper parameter optimization (Bengio 2012) includes a search through the hyper parameter space, i.e., training the model and evaluating predictions for each possible combination of hyper parameter values with the goal of finding an optimal set of values. Table 1 lists the hyper parameters and value set used in experimentation in our work. We partition the data set into three parts: 10% for optimization, 80% for training and 10% for validation using stratified shuffle split. A manual grid search through the parameter space is used for finding optimal models, which are then used for training and comparative evaluation.

Training, Validation and Evaluation Metric

The best performing model obtained from the hyper parameter optimization is trained using the 80% partition. After training, the 10% validation partition is used for reporting evaluation metric. Root mean squared error (RMSE) is used for evaluating regression models. For classification, we report multi-class extensions of accuracy and F1 score as metrics. Accuracy is indicated as the ratio of correct predictions to total number of predictions. F1 Score is a harmonic mean of two metrics, namely precision and recall. Precision focuses on the impact of negative prediction rate, i.e., classifying a product as defective when it is actually defective. Recall on positive prediction rate, i.e., classifying a product as not-defective when it is actually not defective. F1 score is better suited in cases where the data set contains unbalanced classes.

Hyper Parameter	Options
Cell Type	ERNN,LSTM,GRU
Learning Algorithm	SGD,AdaDelta, Adam,RmsProps
Hidden Layers	1,3
Size of Hidden Layers	50,100,200, 250,500,1000
Learning Rate	0.005,0.05,0.5

Table 1: Hyper Parameters

4 Experiments

Experiments are conducted on five time series data sets sampled from the interplanetary magnetic field (IMF) and solar wind data set provided by the solar physics data facility (SPDF) obtained from NASA's COHO Web (King and Papitashvili 2005). Deep neural network models are implemented in theano (Bastien et al. 2012) (Bergstra et al. 2010). Experiments are conducted on a 2.8 GHz Intel Core i7 machine with 16GB memory and 1 GPU, Nvidia GeForce GT 750M.

Application and Data Set Details

Interplanetary magnetic field (IMF) and solar wind parameters in the space between the Sun and Earth are measured by various satellites near earth's surface and consolidated by the Solar Physics Data Facility (SPDF). Fluctuations and extreme values in interplanetary magnetic field are often indicators of increased geomagnetic activity and possibly coronal mass ejection events (CME). Interplanetary coronal mass ejection (ICME) events interfere with the Earth's magnetic field and can cause electric and communication network outages. Predicting IMF values may help us understand and predict ICME events, which in turn, help avoid damage to power grids and ensure that spacecrafts and air-

planes operate safely. IMF is reported in radial-tangential-normal (RTN) coordinate system, where $\hat{R}$ points radially away from the sun, $\hat{T}$ points in the direction of planet's motion and $\hat{N}$ completes the system. In this paper, we attempt to predict next value of the N-component of IMF (also called B_N) from 24 hours of historical B_N data. The data set contains 16 years (1999-2015) of hourly averaged values, containing 149,016 time steps. Although there are 8 features present in the data set that may aid in prediction, for the scope of this paper, we only use B_N.

Preprocessing

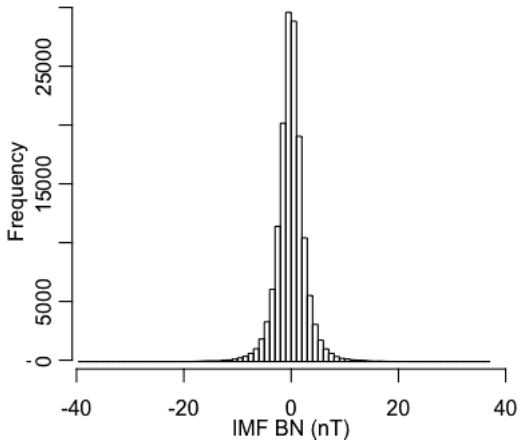

Figure 4: IMF B_N Distribution

B_N follows a distribution shown in Figure 4. Table 2 shows the preprocessing steps applied and resulting cardinality and distribution changes in the data set. The data set uses a filler value of 999.9 where measurements are not available. This value will be filtered out in the cleaning step.

Step	Data Size	Remarks
Original Series	149016	Before Preprocessing
Cleaning	148977	0.026% of original size
Sliding Window	148863	Sequences of length 25
Labeling	23412	(ICME) 15.71%
	125451	(Non-ICME) 84.27%
SAX	17634	A 11.84%
	17650	B 11.85%
	19470	C 13.07%
	18314	D 12.30%
	18669	E 12.54%
	18261	F 12.26%
	19333	G 12.98%
	19532	H 13.12%
WordEmbedding	148863	No Change

Table 2: Preprocesing Results

For IMF B_N prediction, each time step is labeled with a flag to indicate presence of interplanetary coronal mass ejection (ICME) events as given in Dr Ian Richardsons catalog (Cane and Richardson 2003). Since the IMF BN data set is hourly averaged and we may be risking loss of salient information, we do not apply PAA in our pipeline. SAX alphabet size was set to 8 to limit the hyper parameter search due to practical considerations. Table 3 shows the range of IMF B_N values for each alphabet. Embedding converts the sequence into a real valued (in the range [0,1]) matrix of shape 24 x 10.

B_N Value Range	Symbol
-50.7 to -2.6	A
-2.6 to -1.4	B
-1.4 to -0.6	C
-0.6 to 0	D
0 to 0.6	E
0.6 to 1.3	F
1.3 to 2.5	G
2.5 to 37.0	H

Table 3: Applying SAX breakpoints to IMF B_N

Optimization, Training and Validation

We created five approximately equally sized data sets by random sampling without replacement from the COHOWeb IMF data set, imposing stratification and balancing constraints as described in Table 4. Creating these data sets allowed us to study how the classification results change when class distribution in the input (or 'richness' of data set) is controlled. Where balancing was required, larger classes were randomly under-sampled. For each data set, the optimization, training and validation data sets are created using stratified shuffle split with 10%-80%-10% splits.

Model Performance

We experiment with RNN consisting of 1 or 3 layers and basic (ERNN), LSTM or GRU cells. A layer 1 model is a shallow model, but is implemented using deep learning framework for effective comparison. A linear regression layer is added on top for regression experiments and a softmax layer for classification experiments.

For regression model, the input is a univariate time series of length 24 (corresponding to 24 hours) and output is a real number corresponding to the next hour into the future. The input is cleaned and standardized to a [0,1] range, but not preprocessed otherwise. The prediction results are given in Table 5. Compared to the baseline of repeating the last time step value in the series, which yields an RMSE 2.26 in IMF data set, out-of-the-box statistical methods do not perform well. Machine learning methods do better, with SVM and a multi-layer RNN model yielding results better than baseline.

For classification model, using data sets II to V, the input is extensively preprocessed. Input is time series segments, transformed into real valued matrices by the preprocessing pipeline. The model predicts 8 class labels, corresponding to SAX symbols. Data set I is fed into the classifier without preprocessing, with 24 real valued inputs.

	Description	Size	Class Distribution(%)
I	No Constraints	49,000	877 possible distinct values(-50.7 to 37.0), shown in Figure 4.
II	Balanced on ICME labels	46,824	50% ICME,50% Non-ICME
III	Stratified on ICME labels	49,621	15.72% ICME,84.72% Non-ICME
IV	Balanced on SAX labels	48,000	12.5% A, 12.5% B, 12.5% C, 12.5% D, 12.5% E, 12.5% F, 12.5% G, 12.5% H
V	Stratified on SAX labels	49,619	8.06% A, 10.29% B, 14.07% C, 16.92% D, 17.54% E, 15.04% F, 9.42% G, 8.66% H

Table 4: Description of data sets

Classification results are shown in Table 6. The model description short hand given as "n-layer-m-hidden-celltype" reads - an RNN consisting of n number of layers, with m number of celltype (Basic (ERNN), LSTM or GRU) neurons in each hidden layer. For classification, the first base case is accuracy of random choice for 8 classes, i.e. 12.5%. The second base case is predictions that repeat the last value in the series, i.e. 33% accuracy in the IMF data set. Deep learning models consistently out perform out-of-the box machine learning methods over all preprocessed datasets such as SVM and tree classifiers and are better than random chance prediction and repeating last time step values, yielding up to 44% accuracy. From the results, not preprocessing the input for classification models results in extremely poor performance. This may be because there are 877 classes to be predicted instead of 8. The best accuracy and F1 score obtained with our most complex classification model (3 layers, 250 hidden neurons) for data set I is an order less than the accuracy obtained using a much simpler models (e.g. 1 layer, 100 hidden neurons) for preprocessed data sets II,III,IV and V. Results show that effective preprocessing can simplify the task and thus lead to simpler models which can be equally effective.

Model	RMSE
Simple Exponential Smoothing	16.87
ARIMA(model=1,1,1)	16.07
1-layer-45-hidden-LSTM	0.04
RBF SVM	**0.03**
3-layer-30-hidden-GRU	**0.03**

Table 5: Regression model performance

	Model	Accuracy	F1 Score
I	Linear SVM	0.32	0.32
	ExtraTreesClassifier	0.31	0.30
	3-layer-500-hidden-Basic	0.011	0.12
II	Linear SVM	0.35	0.35
	ExtraTreesClassifier	0.35	0.35
	1-layer-100-hidden-Basic	**0.44**	**0.44**
III	Linear SVM	0.32	0.32
	ExtraTreesClassifier	0.23	0.23
	1-layer-500-hidden-Basic	0.38	0.36
IV	LinearSVM	0.32	0.32
	ExtraTreesClassifier	0.22	0.22
	1-layer-100-hidden-Basic	0.39	0.38
V	LinearSVM	0.33	0.33
	ExtraTreesClassifier	0.24	0.24
	1-layer-100-hidden-Basic	0.39	0.38

Table 6: Classification model performance

5 Conclusions and Future Work

The paper describes a system that facilitates exploration of suitability of deep learning methods in time series prediction, and particularly in predicting B_N time series. We explore two approaches, namely regression and classification, comparing the effect of preprocessing the data set for classification. A preprocessing pipeline is described that compresses the time series input data without losing information required to train a deep neural network on classification task. We demonstrate through experiments that classification can be a viable method for prediction of time series when combined with preprocessing techniques. Compared to models required for un-preprocessed inputs, the model for preprocessed inputs is simpler in terms of layers and hidden neurons needed to achieve effective classification.

In future research, we intend to extend this framework to predict more than one value into the future. We envision that it would also be beneficial to extend the input layer to handle multiple input features, i.e., multivariate time series. We would also explore the suitability of other existing deep learning models, such as encoders, in the context of time series prediction. To improve efficiency of the framework, we intend to use alternate methods such as automated grid searches and random search for hyper parameter optimization.

6 Acknowledgments

We acknowledge use of NASA/GSFC's Space Physics Data Facility's OMNIWeb (or CDAWeb) service, and OMNI data. This project has been supported in part by funding from CISE, MPS and GEO Directorates under NSF award #1443061, and by funding from the LWS Program, under NASA award #NNX15AF39G.

References

Asteriou, D., and Hall, S. G. 2011. Arima models and the box-jenkins methodology. *Applied Econometrics* 2(2):265–

286.

Bastien, F.; Lamblin, P.; Pascanu, R.; Bergstra, J.; Goodfellow, I.; Bergeron, A.; Bouchard, N.; Warde-Farley, D.; and Bengio, Y. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bengio, Y. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*. Springer. 437–478.

Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; and Bengio, Y. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, 1–7.

Bishop, C. M. 2006. Pattern recognition. *Machine Learning* 128.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Busseti, E.; Osband, I.; and Wong, S. 2012. Deep learning for time series modeling. Technical report, Technical report, Stanford University.

Cane, H., and Richardson, I. 2003. Interplanetary coronal mass ejections in the near-earth solar wind during 1996–2002. *Journal of Geophysical Research: Space Physics* 108(A4).

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Eck, D., and Schmidhuber, J. 2002. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, 747–756. IEEE.

Elman, J. L. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Faloutsos, C.; Ranganathan, M.; and Manolopoulos, Y. 1994. *Fast subsequence matching in time-series databases*, volume 23. ACM.

Graves, A., and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, 2047–2052. IEEE.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. IEEE.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; and Schmidhuber, J. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

King, J., and Papitashvili, N. 2005. Solar wind spatial scales in and comparisons of hourly wind and ace plasma and magnetic field data. *Journal of Geophysical Research: Space Physics* 110(A2).

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuremoto, T.; Kimura, S.; Kobayashi, K.; and Obayashi, M. 2014. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing* 137:47–56.

Lee, H.; Pham, P.; Largman, Y.; and Ng, A. Y. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, 1096–1104.

Lin, J.; Keogh, E.; Wei, L.; and Lonardi, S. 2007. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15(2):107–144.

Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; and Wang, F.-Y. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16(2):865–873.

Lyu, Q.; Wu, Z.; and Zhu, J. 2015. Polyphonic music modelling with lstm-rtrbm. In *Proceedings of the 23rd ACM international conference on Multimedia*, 991–994. ACM.

Mesnil, G.; He, X.; Deng, L.; and Bengio, Y. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, 3771–3775.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Schaul, T.; Zhang, S.; and LeCun, Y. 2013. No more pesky learning rates. *ICML (3)* 28:343–351.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

Tieleman, T., and Hinton, G. Lecture 6e-rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; and Zhao, J. L. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, 298–310. Springer.