Evaluating data augmentation for financial time series classification

Elizabeth Fons*, Paula Dawson†, Xiao-jun Zeng*, John Keane*, Alexandros Iosifidis‡
*Department of Computer Science, University of Manchester, Manchester, UK
†AllianceBernstein, London, UK
‡Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark.
elizabeth.fons@postgrad.manchester.ac.uk, x.zeng@manchester.ac.uk , john.keane@manchester.ac.uk
paula.dawson@alliancebernstein.com ai@ece.au.dk

***Abstract*—Data augmentation methods in combination with deep neural networks have been used extensively in computer vision on classification tasks, achieving great success; however, their use in time series classification is still at an early stage. This is even more so in the field of financial prediction, where data tends to be small, noisy and non-stationary. In this paper we evaluate several augmentation methods applied to stocks datasets using two state-of-the-art deep learning models. The results show that several augmentation methods significantly improve financial performance when used in combination with a trading strategy. For a relatively small dataset ($\approx 30K$ samples), augmentation methods achieve up to 400% improvement in risk adjusted return performance; for a larger stock dataset ($\approx 300K$ samples), results show up to 40% improvement.**

I. Introduction

Time series classification is an important and challenging problem, that has garnered much attention as time series data is found across a wide range of fields, such as weather prediction, financial markets, medical records, etc. Recently, given the success of deep learning methods in areas such as computer vision and natural language processing, deep neural networks have been increasingly used for time series classification tasks. However, unlike in the case of image or text datasets, (annotated) time series datasets tend to be smaller in comparison, which often leads to poor performance on the classification task [1]. This is especially true of financial data, where a year-long of stock price data may consist of only 250 daily prices. [2]. Therefore, in order to be able to leverage the full potential of deep learning methods for time series classification, more labeled data is needed.

A common strategy to address this problem is use of data augmentation techniques to generate new sequences that cover unexplored regions of input space while maintaining correct labels, thus preventing over-fitting and improving model generalization [3]. This practice has been shown to be very effective in other areas, but it is not an established procedure for time series classification [4] [5]. Moreover, most of the methods used are just adaptations of image-based augmentation methods that rely on simple transformations, such as scaling, rotation, adding noise, etc. While a few data augmentation methods have been specifically developed for time series [1], [6], their effectiveness in the classification of financial time series has not been systematically studied.

Stock classification is a challenging task due to the high volatility and noise from the influence of external factors, such as global economy and investor's behaviour [7]. An additional challenge is that financial datasets tend to be small; ten years of daily stock prices would include around 2500 samples, which would be insufficient to train even a small neural network (e.g. a single-layer LSTM network with 25 neurons has approximately 2700 parameters). In this work we perform a systematic analysis of multiple individual data augmentation methods on stock classification. To compare the different data augmentation methods, we evaluate them using two state-of-the-art neural network models that have been used for financial tasks. As the usual purpose of stock classification tasks is to build portfolios, we compare the results of each method and each architecture by building simple rule-based portfolios and calculating the main financial metrics to assess performance of each portfolio. Finally, we analyse the combination of multiple data augmentation methods, by focusing on the best performing ones.

The contributions of the paper are as follows:

- We provide the first, to the best of our knowledge, thorough evaluation of popular data augmentation methods for time series on the stock classification problem; we perform an in-depth analysis of a number of methods on two state-of-the-art neural network architectures using daily stock returns datasets.
- We evaluate performance using traditional classification metrics. In addition, we build portfolios using a simple rule-based strategy and evaluate performance based on financial metrics.

The remainder of the paper is organized as follows: Section II overviews previous work on data augmentation; Section III describes the data augmentation methods used in our evaluations; Section IV describes the experimental setup; Section V provides the experimental results; conclusions and future work are presented in Section VI.

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement no. 675044 (http://bigdatafinance.eu/), Training for Big Data in Financial Research and Risk Management. A. Iosifidis acknowledges funding from the Independent Research Fund Denmark project DISPA (Project Number: 9041-00004).

II. Related work

Data augmentation has proven to be an effective approach to reduce over-fitting and improve generalization in neural networks [8]. While there are several methods to reduce over-fitting in neural networks, such as regularization, dropout and transfer learning, data augmentation tackles the issue from the root, i.e., by enriching the information related to the class distributions in the training dataset. Therefore, by assuming that more information can be extracted from the dataset through augmentations, it further has the advantage that it is a model-independent solution [3].

In tasks such as image recognition, data augmentation is a common practice, and may be seen as a way of pre-processing the training set only [9]. For instance Krizhevsky *et al*. [10] used random cropping, flipping and changing image intensity in AlexNet, Simonyan *et al*. used scale jittering and flipping [11] on the VGG network. However, such augmentation strategies are not easily extended to time-series data in general, due to the non i.i.d. property of the measurements forming each time-series. Data augmentation has been applied to domain-specific time series data encoding information of natural phenomena with great success. Cui *et al*. [8] use stochastic feature mapping as a label preserving transformation for automatic speech recognition. Um *et al*. [12] test a series of transformation-based methods (many inspired directly by computer vision) on sensor data for Parkinson's disease and show that rotations, permutations and time warping of the data, as well as combinations of those methods, improve test accuracy. Le Guennec *et al*. [6] propose two augmentation methods, window slicing and window warping to use in combination with a convolutional neural network and test it on the UCR dataset archive, improving the overall classification performance. Most recently, SpecAugment was proposed for end-to-end speech recognition, where an augmentation policy consisting of three augmentation methods, time warp, frequency masking and time masking acts on the log mel spectrogram directly [13].

To date, little work has been done on studying the effect of data augmentation methods for financial data or developing methods specialized on financial time-series. For regression tasks, Teng *et al*. [2] use a time-sensitive data augmentation method for stock trend prediction, where data is augmented by corrupting high-frequency patterns of original stock price data as well as preserving low-frequency ones in the frame of wavelet transformation. For stock market index forecasting, Yujin *et al*. [14] propose ModAugNet, a framework consisting of two modules: an over-fitting prevention LSTM module and a prediction LSTM module.

III. Time Series Augmentation

Most cases of time series data augmentation correspond to random transformations in the magnitude and time domain, such as jittering (adding noise), slicing, permutation (rearranging slices) and magnitude warping (smooth element-wise magnitude change). In our analysis, the following methods were used for evaluation, and examples of these transformations are shown in Figure 1:

- **Magnify:** we propose a variation of the window slicing method presented by Le Guennec et al [6]. In window slicing, a window of 90% of the original time series is selected at random. Instead, we randomly slice windows between 40% and 80% of the original time series, but always from the fixed end of the time series (*i.e*. we slice the beginning of the time series by a random factor). Randomly selecting the starting point of the slicing would make sense in an anomaly detection framework, but not on a trend prediction as is our case. We interpolate the resulting time series to the original size in order to make it comparable to the other augmentation methods.
- **Reverse:** it transforms the time series from $\{t_1, t_2, \ldots, t_{n-1}, t_n\}$ to $\{t_n, t_{n-1}, \ldots, t_2, t_1\}$; thus, the time series is reversed. This method is based on the data augmentation method from computer vision flipping.
- **Scaling:** It multiplies the time series with a random scalar. Each time series in the training set is multiplied by a scalar drawn from a Gaussian distribution with mean $\mu = 1$ and $\sigma = 0.2$ [12].
- **Window warp:** Following Le Guennec et al [6], a window consisting of 10% of the original time series length is randomly selected and then the time dimension is warped by either 0.5 times or 2 times.
- **Magnitude warp:** the time series is multiplied by a warping magnitude controlled by a smooth curve with knots at random locations and magnitudes. In this work we use four knots, and the curve varies around a mean $\mu = 1$ and with $\sigma = 0.2$ [12].
- **Jittering:** it adds Gaussian noise to each time step of the time series. In this work we use mean $\mu = 0$ and standard deviation $\sigma = 0.01$ [12].
- **Pool:** Reduces the temporal resolution without changing the length of the time series by averaging a pooling window. We use a window of size 3. This method is inspired by the resizing data augmentation process followed in computer vision.
- **Quantise:** the time series is quantised to a level set n, therefore the difference between the maximum and minimum values of the time series is divided into levels, and the values in the time series are rounded to the nearest level [15]. We used $n = 25$.
- **Convolve:** the time series is convolved with a kernel window. The size of the kernel is 7 and the type of window is Hann.
- **Time Warp:** the time intervals between samples are distorted based on a random smooth warping curve by cubic spline with four knots at random magnitudes [12].
- **Sub-optimal warped time series generator (SPAWNER):** SPAWNER [16] creates a time series by averaging two random sub-optimally aligned patterns that belong to the same class. Following Iwana *et al*. [5],

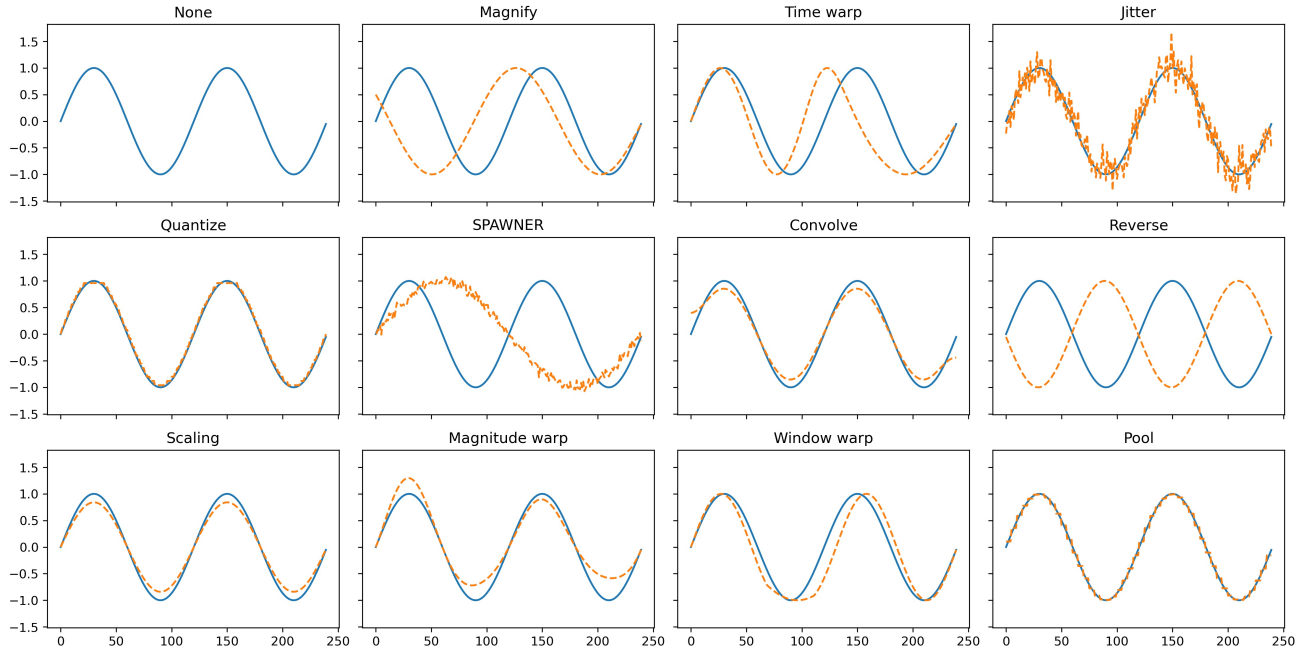

Fig. 1: Examples of time-series data augmentation methods on a sine wave. The blue line corresponds to the original time-series and the dotted orange lines correspond to the generated time-series patterns.

noise is added to the average with $\sigma = 0.5$ in order to avoid cases where there is little change.

For the methods Pool, Quantise, Convolve and Time warping we used the code from Arundo [17][1] and for the methods Window warp and Magnitude warp we used an adapted version of the code provided by Iwana *et al.* [5].

IV. Methodology

A. Datasets

The data used is this work consists of the daily returns of constituent stocks of the S&P500 index, from 1990 to 2018 and it comprises 7000 trading days. We follow the data pre-processing scheme proposed by Krauss *et al.* [18], where we divide the data into splits of 1000 trading days using a sliding window of 250 days. Therefore, each data split overlaps with the previous one by 750 points and results in 25 splits of data in total. One model is trained on each data split.

Within each of the 25 splits, for a stock s the returns are segmented into sequences of 240 times steps $\{\tilde{R}^s_{t-239}, \ldots, \tilde{R}^s_t\}$ using a sliding window of one day. Figure 2 shows the data segmentation scheme. We use the first 750 days of the data split for training and the remaining 250 days for testing. The data is then standardised using the mean (μ_{train}) and the standard deviation (σ_{train}) of the training set in the following way:

$$\tilde{R}^s_t = \frac{R^s_t - \mu_{train}}{\sigma_{train}}$$

where R^s_t corresponds to the return of stock s at time t. We frame the problem as a binary classification task, where the target variable Y^s_{t+1} for stock s and date t can take the value 1 if the returns are above the daily median (*trend up*) or 0 if the returns are below the daily median (*trend down*). Because we use the daily median return to build the target, this leads to a balanced dataset.

[1]https://arundo-tsaug.readthedocs-hosted.com/en/stable/

Stock s_1

Date	1	2	3	...	239	240	241	242	243	...
$\tilde{R}^s_t$	0.655	0.143	1.476	...	0.622	-0.024	-0.980	-0.512	-0.353	...

1	2	3	...	239	240
0.655	0.143	1.476	...	0.622	-0.024

2	3	...	239	240	241
0.143	1.476	...	0.622	-0.024	-0.980

Fig. 2: Construction of input sequences, segmented in 240 time steps, with a moving window of one day.

1) Full S&P500 dataset: We use all the constituents of the S&P500 index, therefore, the training set contains approximately 255K samples ((750-240)*500), and the test set has approximately 125K samples.

2) 50 stocks dataset: In order to have a smaller dataset, we use the same pre-processing scheme but only for the largest 50 stocks on the S&P500 measured by market capitalization. Therefore, on the day 750 (the cut date between training and test separation) on each data split, we measure the market capitalization of all stocks in the S&P500 index, rank them and select the largest 50. This leads to 25500 samples for training and 12500 for testing.

B. Augmentation

The training data (750 days) is divided into training and validation with a proportion 80/20. Before splitting the data,

TABLE I: Performance of the $k = 10$ long-only portfolios after transaction costs for the TLo-NBoF model and small dataset.

	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	F1
None	10.28	22.62	0.07	15.53	0.10	50.49±0.46	40.06±6.44
Convolve	12.29	22.35	0.24	15.04	0.35	**50.62±0.60**	**42.98±6.5**
Jitter	9.2	22.32	-0.02	15.32	-0.02	50.43±0.59	42.5±6.71
Magnify	13.33	21.98	0.31	14.71	0.47	50.55±0.5	40.35±6.35
Pool	12.76	21.9	0.28	14.8	0.41	50.51±0.6	41.33±6.52
Quantize	12.69	20.23	0.27	13.83	0.38	50.45±0.63	40.67±6.51
Reverse	7.28	22.08	-0.18	15.03	-0.27	50.52±0.59	40.28±6.17
Time warp	12.81	22.41	0.27	14.89	0.42	50.44±0.61	41.64±5.64
Magnitude-warp	11.08	22.39	0.12	15.17	0.18	50.51±0.61	40.32±7.45
Window-warp	13.31	20.5	0.28	13.77	0.4	50.57±0.55	40.45±6.97
Scaling	**15.78**	21.59	**0.48**	14.52	**0.71**	50.50±0.55	39.9±6.45
Spawner	11.93	21.99	0.20	14.89	0.29	50.46±0.53	41.63±6.94
Mag-Pool	9.24	22.58	-0.01	15.18	-0.02	50.52±0.44	40.2±6.68
Mag-Quant	11.52	21.43	0.16	14.48	0.24	50.43±0.55	39.63±6.13
Mag-TW	10.4	21.5	0.08	14.75	0.11	50.46±0.56	40.15±6.41
Quant-Pool	11.52	**20.15**	0.15	**13.69**	0.21	50.54±0.53	41.51±6.62
Quant-TW	12.06	20.7	0.20	14.09	0.29	50.54±0.46	41.09±6.7

all samples are shuffled in order to make sure that all stocks and time steps are randomly assigned to train or validation. Each train set is augmented with 1X the original size.

C. Network architectures and training

We use two neural network architectures proposed in previous financial studies, optimizing the cross entropy loss:

1) LSTM: Following Krauss *et al*. [18], we use a neural network with a single layer LSTM with 25 neurons and a fully connected two-neuron output. We train the network using a learning rate of 0.001, batch size 128 and early stopping with patience 10 with RMSProp as optimizer.

2) Temporal Logistic Neural Bag-of-Features (TLo-NBoF): we adapt the network architecture proposed by Passalis *et al*. [19] to forecast limit order book data. The original network was used on data samples of 15 time steps and 144 features so we adapt it for our univariate data of 240 time steps. It comprises an 1D-convolution (25 filters, kernel size 81), a TLo-NBoF layer ($N_K = 12$, $N_T = 3$), a fully-connected layer (50 neurons) and a fully-connected output layer of 2 neurons. The initial learning rate is set to 0.0005, the learning rate is decreased on plateau of the validation loss, batch size is 256 and the optimizer is Adam.

D. Rule-based portfolio strategy and evaluation

In order to evaluate if data augmentation provides an improvement in asset allocation, we propose a simple trading strategy, following the conclusions of Krauss et al [18]. The trading rule on the full S&P500 dataset is as follows: stocks in both classes are ranked daily by their predicted probability of belonging to that class, we then take the top 10 and bottom 10 stocks and build a long-short portfolio by equally weighting the stocks. Portfolios are analysed after transaction costs of 5 bps per trade.

On the 50-stocks dataset, building a long-short portfolio would not be profitable as it consists of the 50 largest US market cap stocks. So we only build a portfolio by going long on the top 10 stocks [20]. In order to compare our methods with the performance of their stocks universe, we build a benchmark that consists of all 50 stocks weighted by their market cap. All portfolios are built including transaction costs.

We evaluate portfolio performance by calculating the Information ratio (IR), the ratio between excess return (portfolio returns minus benchmark returns) and tracking error (standard deviation of excess returns) [21]. Additionally, we calculate the downside information ratio, the ratio between excess return and the downside risk (variability of underperformance below the benchmark), that differentiates harmful volatility from total overall volatility.

V. Results

Tables I and II present the results obtained for each individual augmentation method and the combination of the most successful individual methods for the small 50 stock dataset using the LSTM and the TLo-NBoF networks. For comparison, we also show the results without augmentation.

We also show classification metrics (accuracy and F1) over the 25 data splits expressed by the mean and standard deviation. In both models, the classification accuracy improvement is very small with respect to no augmentation, and for F1 as well. But we see that both the IR and DIR improve using several augmentation methods. Magnify and time warp methods are consistently good performers, as well as spawner. For the TLo-NBoF, IR increases four times with respect of no method, and time warp on the LSTM model doubles the IR. We anticipated that the Reverse method would not be effective - and in both cases it decreases overall performance. Further, we note that he combination of two augmentation methods does not always improve performance.

Figures 3 and 4 show the cumulative profit over time (out of sample) of the models trained with different augmentation methods and the baseline (no augmentation). We focus on the most competitive techniques and for comparison, we add the benchmark calculated by the market weighted returns of the 50 constituent stocks. The top plots show the full history while the

TABLE II: Performance of the $k = 10$ long-only portfolios after transaction costs for the LSTM model and small dataset.

	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	F1
None	12.24	**24.05**	0.22	**15.89**	0.33	50.8±0.75	47.69±4.81
Convolve	12.33	25.91	0.21	16.95	0.33	50.74±0.81	48.43±3.39
Jitter	11.75	24.35	0.18	16.49	0.27	50.89±0.73	**48.86±2.87**
Magnify	14.16	25.44	0.32	16.58	0.51	**50.94±0.68**	48.57±3.2
Pool	11.81	26.15	0.18	17.15	0.27	50.86±0.77	48.5±3.49
Quantize	12.80	24.41	0.26	16.46	0.38	50.93±0.79	48.6±2.82
Reverse	6.12	24.12	-0.22	16.27	-0.33	50.76±0.78	45.96±4.74
Time Warp	**15.60**	24.38	**0.43**	16.12	**0.67**	50.85±0.74	48.24±3.48
Magnitude-warp	12.44	24.68	0.23	16.38	0.35	50.87±0.68	49.00±3.03
Window-warp	13.43	24.97	0.28	16.56	0.44	50.83±0.73	48.92±2.83
Scaling	11.40	24.47	0.15	16.49	0.23	50.86±0.75	48.79±2.76
Spawner	14.58	24.49	0.38	16.02	0.60	50.95±0.74	48.36±3.66
Mag-Quant	13.70	25.82	0.29	16.74	0.47	50.92±0.67	48.43±3.29
Mag-TW	14.00	25.66	0.31	16.63	0.49	50.88±0.67	48.45±3.06

TABLE III: Performance of the $k = 10$ long-short portfolios after transaction costs for the LSTM model and large dataset.

	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	F1
None	34.64	28.43	1.22	18.78	1.84	51.0 ± 1.0	48.5 ± 2.1
Convolve	32.60	25.99	1.25	17.49	1.86	51.1 ± 0.9	49.2 ± 2.0
Jitter	34.35	25.3	1.36	16.69	2.06	51.0 ± 1.0	$\mathbf{50.0 \pm 1.0}$
Magnify	$\mathbf{46.56}$	29.41	1.58	19.56	2.38	$\mathbf{51.2 \pm 0.9}$	48.8 ± 2.7
Pool	36.18	26.16	1.38	17.15	2.11	51.1 ± 0.9	49.4 ± 2.2
Quantize	29.42	**25.48**	1.15	**16.62**	1.77	51.0 ± 1.0	48.8 ± 2.1
Reverse	33.03	26.34	1.25	16.9	1.95	51.0 ± 1.0	47.3 ± 4.1
Time warp	47.01	29.26	1.61	19.17	2.45	51.2 ± 0.9	49.8 ± 1.6
Scaling	36.23	26.08	1.39	17.00	2.13	51.0 ± 1.0	49.4 ± 1.7
Window warp	26.77	26.49	1.01	17.51	1.53	51.0 ± 1.0	49.8 ± 1.1
Magnitude warp	20.06	26.96	0.74	18.33	1.09	51.0 ± 1.0	49.7 ± 1.3
Spawner	38.08	27.85	1.37	18.05	2.11	51.1 ± 1.0	49.1 ± 2.2
Mag-Jit	30.03	27.59	1.09	18.62	1.61	51.1 ± 1.0	49.4 ± 2.0
Mag-TW	44.03	27.41	1.61	17.66	2.49	51.1 ± 1.0	49.6 ± 1.4
TW-Pool	44.98	26.21	**1.72**	16.82	**2.67**	51.1 ± 0.9	49.3 ± 2.0
TW-Jitter	22.47	25.94	0.87	17.71	1.27	51.1 ± 1.0	49.3 ± 1.8

bottom plots show the last 10 years. Both models perform well over time, and cumulative profits of the models trained with augmentation are higher when compared to not using augmentation; however, only TLo-NBoF is competitive on the most recent testing period (2007-2017), along with several of the augmentation methods. The LSTM model fluctuates around zero and does not improve with regards to the benchmark. Krauss *et al*. [18] observes that the edge of the LSTM method seems to have been arbitraged away in the latter years.

Table III presents the results obtained for each individual augmentation method and the combination of the most successful methods for the large S&P500 dataset trained on the LSTM network. As the portfolios are long-short, they are market-neutral (therefore, the performance of the portfolio in independent of the performance of the market and no benchmark has to be subtracted). As with the small dataset, Magnify and Time warp show a strong performance in IR and DIR, as well as their combination. Jitter performs well in this dataset, but in the models trained on the small dataset, performance decreased so maybe in a larger dataset, the added noise helps with generalization, while in smaller data, diminishes the signal. The changes to the classification metrics are not significant.

VI. Conclusions

Data augmentation is a ubiquitous technique to improve generalization in supervised learning. In this work, we have studied the impact of various data augmentation methods for time series on the stock classification problem. We have shown that even with very noisy datasets such as stocks returns, it is beneficial to use data augmentation to improve generalization. Magnify, Time warp and Spawner consistently improve both the Information ratio and downside information ratio on all models and datasets. On the small datasets, augmentation achieves up to four-times (TLo-NBoF) and two-times (LSTM) performance improvement on IR compared to no augmentation. On a larger dataset, as espected, improvement is not that sharp, but still it achieves an increment in IR of up to 40%.

We tested the TLo-NBoF network that has not previously been used on low-freq stock data, and this network shows consistent positive returns over the last ten years of data, therefore, unlike the LSTM architecture, the profit has not been leveraged away.

References

[1] Brian Kenji Iwana and Seiichi Uchida, "Time series data augmentation for neural networks by time warping with a discriminative teacher,"

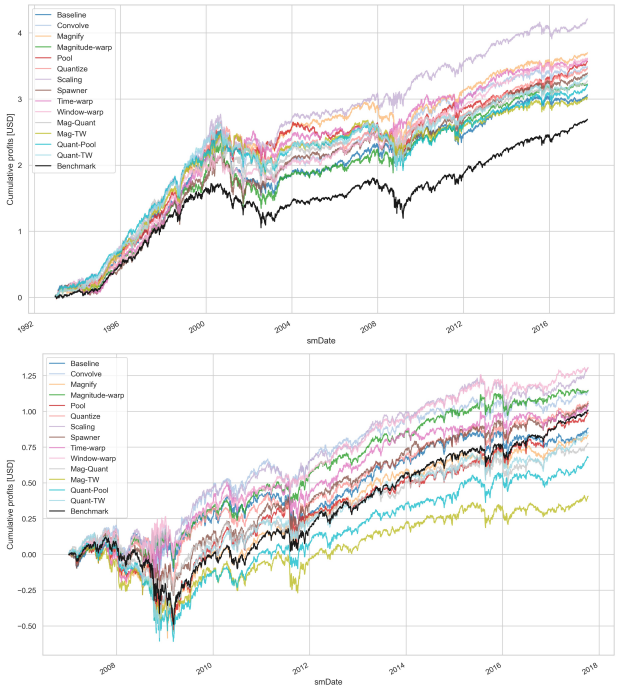

Fig. 3: Performance of the TLo-NBoF models trained with and without augmentation and the benchmark (in black) measured as cumulative profits on 1USD average investment per day. Top corresponds to full testing history and bottom corresponds to the last 10 years.

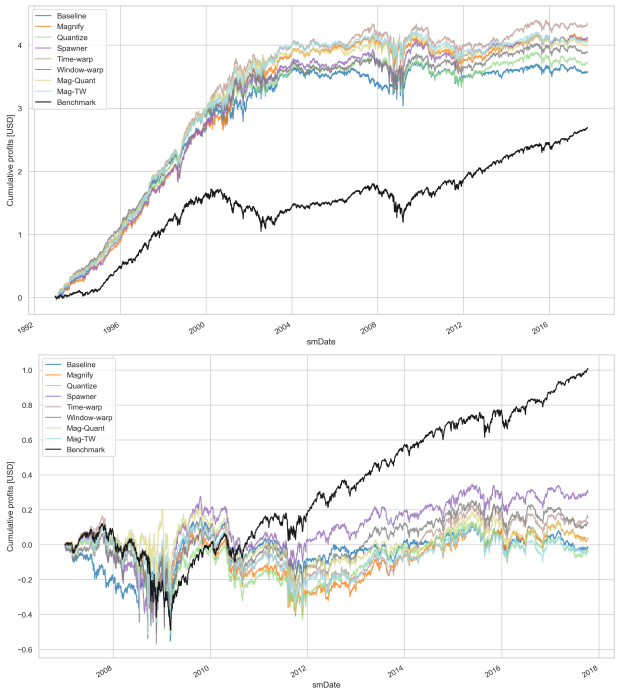

Fig. 4: Performance of the LSTM models trained with and without augmentation and the benchmark (in black) measured as cumulative profits on 1USD average investment per day. Top corresponds to full testing history and bottom corresponds to the last 10 years.

in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2020.

[2] Xiao Teng, Tuo Wang, Xiang Zhang, Long Lan, and Zhigang Luo, "Enhancing stock price trend prediction via a time-sensitive data augmentation method," *Complexity*, 2020.

[3] Connor Shorten and Taghi M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 60, 2019.

[4] Q. Wen, Liang Sun, Xiaomin Song, J. Gao, X. Wang, and Huan Xu, "Time series data augmentation for deep learning: A survey," *ArXiv*, 2020.

[5] Brian Kenji Iwana and Seiichi Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *arXiv preprint arXiv:2007.15951*, 2020.

[6] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[7] Thomas Fischer and Christopher Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[8] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1469–1477, 2015.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. 2012.

[11] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[12] Terry T. Um, Franz M. J. Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 2017, ICMI '17, p. 216–220.

[13] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Dogus Cubuk, and Quoc V. Le, "Specaugment: A simple augmentation method for automatic speech recognition," in *INTERSPEECH*, 2019.

[14] Yujin Baek and Ha Young Kim, "Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module," *Expert Systems with Applications*, vol. 113, pp. 457 – 480, 2018.

[15] Peter Tino, Christian Schittenkopf, and Georg Dorffner, "Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams. lessons learned from financial volatility trading.," Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science" 46, SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, Vienna, 2000.

[16] Krzysztof Kamycki, Tomasz Kapuscinski, and Mariusz Oszust, "Data augmentation with suboptimal warping for time-series classification," *Sensors (Basel, Switzerland)*, vol. 20, no. 1, pp. 98, 12 2019.

[17] Arundo, "Tsaug," https://tsaug.readthedocs.io/en/stable/index.html, 2020.

[18] Christopher Krauss, Xuan Anh Do, and Nicolas Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the *s&p* 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689 – 702, 2017.

[19] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *Pattern Recognition Letters*, 2020.

[20] Jamil Baz, N. Granger, Campbell R. Harvey, Nicolas Le Roux, and Sandy Rattray, "Dissecting investment strategies in the cross section and time series," *Econometric Modeling: Derivatives eJournal*, 2015.

[21] C.R. Bacon, *Practical Risk-Adjusted Performance Measurement*, The Wiley Finance Series. Wiley, 2012.