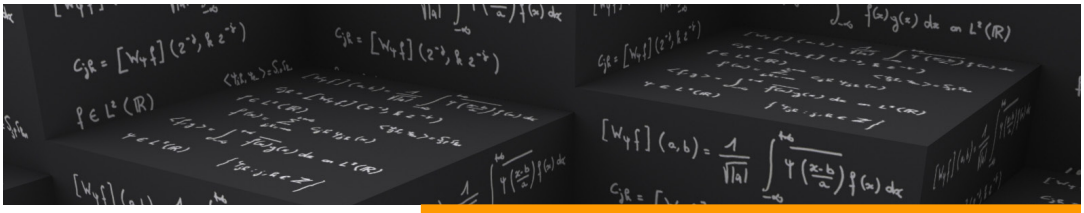
VIVIENNE
INVESTISSEMENT



Vivienne investissement

“Artificial Intelligence for innovative investment solutions”

Research report extracted from the article:

“Deep learning abilities to classify intricate variations in temporal dynamics of multivariate time series”

Published for ICASSP 2020, Barcelona, Spain

VIVIENNE INVESTISSEMENT

contact@vivienne-im.com

6 quai Jules-Courmont - 69002 Lyon, France

AMF : GP10000029 | 484 288 428 RCS Lyon

DEEP LEARNING ABILITIES TO CLASSIFY INTRICATE VARIATIONS IN TEMPORAL DYNAMICS OF MULTIVARIATE TIME SERIES

P. Liotet[1,4]*, P. Abry*[2]*, R. Leonarduzzi*[3]*, M. Senneret*[4]*, L. Jaffrès*[4]*, G. Perrin*[4]

[1] ENS Paris-Saclay, France.
[2] Univ Lyon, ENS de Lyon, Univ Claude Bernard, CNRS, Laboratoire de Physique, Lyon, France.
[3]ENS, PSL, Paris, France.
[4]Vivienne Investissement, Lyon, France.

ABSTRACT

The aim of this work is to investigate the ability of deep learning (DL) architectures to learn temporal dynamics in multivariate time series. The methodology consists in using well known synthetic stochastic processes for which changes in joint temporal dynamics can be controlled. This permits to compare deep learning against classical machine learning techniques relying on documented hand-crafted wavelet-based features. First, we assess the performance of several different DL architectures and show the relevance of convolutional neural networks (CNN). Second, we test the robustness of CNN performance in classifying subtle changes in multivariate temporal dynamics with respect to learning conditions (dataset size, time series sample size, transfer learning).

***Index Terms*—** Deep Learning, CNN, Time Series, Multivariate, Multifractal.

1. INTRODUCTION

Context. In a wide range of domains, deep learning (DL) is sprawling and reaching state-of-the-art results. It is notably well adapted for classification tasks and began to get attention after unprecedented performance in recognition of images with strong geometric contents [1–3]. More difficult and less addressed challenges are related to texture classification, where the available information is statistic in nature rather than geometric [4]. Further, most research on texture classification is focused on images, and thus the literature on 1D texture (time series) classification is even more sparse. Despite successes in specific applied contexts [5–8], the ability of DL to capture temporal dynamics in time series remains rarely investigated at a generic level, and constitutes a broad issue that we aim to address here.

Related Works. Novel neural networks are proposed at an increasing pace, resulting in a rich variety of very different architectures—multilayer perceptrons [9], Long-Short Term Memory networks [10], convolutional networks, residual networks [3], Wavenet [11], amongst others—making performance and complexity comparisons difficult to achieve. Comparisons are further impaired by different architectures being applied to different tasks. Even when focusing on the specific task of interest here—classification of temporal dynamics—the available literature, e.g. [5, 8, 12–15], remains too diverse and spread in settings and contexts to permit comparisons or to gain a global understanding in the ability of DL architectures to discover/model temporal dynamics.

Goals, contributions and outline. The overall goal of the present work is to quantify the ability of deep learning architectures to discriminate mild but intricate differences in complex multivariate temporal dynamics. To that end, use is made of specific and well controlled stochastic models with scale-free temporal dynamics: multivariate multifractal random walks, widely used in applications and of intrinsic interest [16,17]. However, the focus is not on multifractality per se and those processes are only chosen because they enable an easy design of classes of processes with mild differences in their higher order cross (multivariate) temporal dynamics [16, 17], thus permitting to asses the ability of neural networks to *discover* such subtle differences in multivariate temporal dynamics. Further, scale-free temporal dynamics are known to be well analyzed by wavelet representations, which can hence be used as expert benchmarks. These stochastic models and analysis tools are described in Section 2. Contributions of the present work are threefold. First, several neural network architectures, described in Section 3, are compared with respect to classification performance in Section 4, showing the relevance of convolutional neural networks (CNN) for temporal dynamics assessment. Also, their complexity is discussed and quantified in terms of Vapnik-Chervonenkis dimension. Second, the performance of several CNN with different complexities are compared against each other and against support vector machines using hand-crafted (wavelet-based) features, constructed from explicit *expert prior knowledge* (cf. Section 5). Finally, the impact on classification performance of learning conditions—such as time series length, number of training data, or departures from the statistics of training data (transfer learning)—are investigated, quantified and discussed in Section 5.

2. MULTIVARIATE TEMPORAL DYNAMICS

Scale-free temporal dynamics. Multifractal random walks (MRW) constitute well defined processes with rich scale-free temporal dynamics, widely used in real-world applications [18], introduced in the univariate setting in [16] and recently extended in a bivariate setting [17]. The present work further proposes to define Multivariate MRW (MMRW) $\underline{X}(t) = \{X_1(t), \ldots, X_M(t)\}$ as $\forall k = 1, \ldots, M$, $X_k(t) = \sum_{n=1}^{t} G_k(n) e^{\omega_k(n)}$, with $\underline{G}(t) = \{G_1(t), \ldots, G_M(t)\}$ and $\underline{\omega}(t) = \{\omega_1(t), \ldots, \omega_M(t)\}$ two independent multivariate Gaussian processes, with zero-mean and prescribed covariance functions. $\underline{G}(t)$ is defined as the multivariate extension of fractional Gaussian noise (fGn), the increment process of fractional Brownian motion—the reference Gaussian self-similar process [19]. $\underline{G}(t)$ is thus fully defined by a vector of self-similarity exponents $\underline{H} = (H_1, \ldots, H_M)$ and a $M \times M$ pointwise covariance matrix Σ [20, 21] . $\underline{\omega}(t)$ is defined via pairwise covariance functions, chosen to induce multifractality in temporal dynamics, $\Sigma_{mf}(k,l) = \rho_{mf}(k,l)\lambda_k\lambda_l \log\left(\frac{T}{|k-l|+1}\right)$, if $|k-l| \leq T-1$ and 0 otherwise, with T an arbitrary integral scale. $\underline{\omega}(t)$ is thus fully defined by a vector of multifractality $\underline{\Lambda} = (\lambda_1, \ldots, \lambda_M)$ and a $M \times M$ multifractal correlation matrix ρ_{mf}.

Extending bivariate calculations [17], it can be shown that the (cross) second-order statistics governing the temporal dynamics of MMRW are driven mostly by $\underline{H}$ and Σ and marginally by $\underline{\Lambda}$. Furthermore, the part of temporal and cross-temporal dynamics not already encoded by cross-correlations are controlled by $\underline{\Lambda}$ and ρ_{mf}. Classes of temporal dynamics with subtle differences in cross higher-order statistics (i.e., beyond cross-correlation) can thus be constructed by tuning the off-diagonal entries of Matrix ρ_{mf}.

Wavelet based scale-free analysis. It is now well-documented that scale-free temporal dynamics are well analyzed using multiscale wavelet representations [21]. Let $d_X(j,k)$ denote the discrete wavelet transform coefficients of X, defined as $d_X(j,k) = \langle 2^{-j}\psi(2^{-j}\cdot - k)|X(\cdot)\rangle$. It can be shown that for any pair of components $(X_k(t), X_l(t))$, the wavelet cross-correlations satisfy $S_{k,l}(j) = \sum_n d_{X_k}(j,n) d_{X_l}(j,n) \simeq \Sigma_{k,l} C_\psi(H_k, H_l, \lambda_k, \lambda_l) 2^{j(H_k+H_l-(\lambda_k^2+\lambda_l^2)/2}$. Wavelet coefficients thus only probe the second-order statistics of MMRW.

To measure higher-order temporal dynamics, wavelet leaders $L(j,k)$ need to be further constructed as local suprema of wavelet coefficients, taken over finer scales and within a short temporal neighborhood [22]. It can then be shown that the first- and second-order cumulants, $C_1^{(k)}(2^j)$ and $C_2^{(k,l)}(2^j)$, of $(\ln L_{X_k})_{k=1,\ldots,M}$, behave as $C_1^{(k)}(2^j) = c_1^{(0,k)} + c_1^{(k)} \ln 2^j$, with $c_1^{(k)} = H_k + \lambda_k^2/2$, and $C_2^{(k,l)}(2^j) = c_2^{(0,k,l)} + c_2^{(k,l)} \ln 2^j$, with $c_2^{(k,l)} = \rho_{mf}(k,l)\lambda_k\lambda_l$. The $c_2^{(k,l)}$, controlled by $\underline{\Lambda}$ and ρ_{mf} only, measure fluctuations of temporal and cross-temporal dynamics not already encoded in the covariance functions, referred to here as multifractality.

3. NEURAL NETWORKS - SUPERVISED LEARNING

Neural Networks. Neural networks (NN) consist of parametrized functions $f(X)$ cascading affine mappings and pointwise nonlinear functions, referred to as activation functions (such as sigmoid, ReLU,...). They are inspired from simple models of neurons, which make a linear combination of inputs and produce an output only if it exceeds a certain threshold. Numerous architectures were proposed across the years, designed either to match specific tasks or mimic *natural* mechanisms (cf. e.g., [23]) and based on different elementary layers.

Layers. In a fully-connected (FC) layer [9], the affine mapping is dense: it is applied to all outputs of the previous layer, resulting in large numbers of parameters to learn. In convolutional (C) layers the affine map is a convolution with a linear filter, where the same weights are applied to different regions of the input; the number of parameters is thus largely reduced. Often, several filters, or *channels*, are applied in parallel. Long-Short Term Memory (LSTM) layers [10] are based on neurons that track the time evolution of inputs; they combine current inputs with previous outputs through an internal state, and thus directly measure temporal evolutions.

Architectures. In this work, four architectures are compared. They are sketched in Fig.1 and their complexities are compared in Table 1. All architectures share the last two FC layers—which use Leaky-ReLU and sigmoid activations. We detail below the rest of their layers.

The fully-connected networks (FCNN) concatenate two FC layers with leaky-ReLU activations (slope of 0.02). Wide (a1) and narrow (a2) FCNNs are considered; (a2) is chosen to have a similar size to the convolutional network (c). The LSTM network (LSTMNN) uses two LSTM layers of 8 neurons. The Convolutional network (CNN) uses two convolutional layers with Leaky-ReLUs. A residual network (ResNet) [3] elaborates on CNN to allow the training of deeper networks. ResNet layers combine the outputs of previous layers to prevent the gradient from vanishing and stopping training [3].

4. ARCHITECTURE COMPARISONS

Neural Network Complexity. To ensure meaningful performance comparisons, tools assessing architecture complexities must be used. A direct comparison of the number of parameters or layers is not meaningful because of their different roles in different architectures. As an alterative, the Vapnik-Chervonenkis dimension [24]—which measures the expressive power of a parametrized family of functions—has been proposed to compare network architectures [25, 26]. Though it is hard to compute in general, an upper bound is available for NN in binary classification problems [27]. It is based on the input length d and the number of nodes m of a computational graph that represents the NN—expressing, to that end, activation functions as Pfaffian chains. The largest Pfaf-

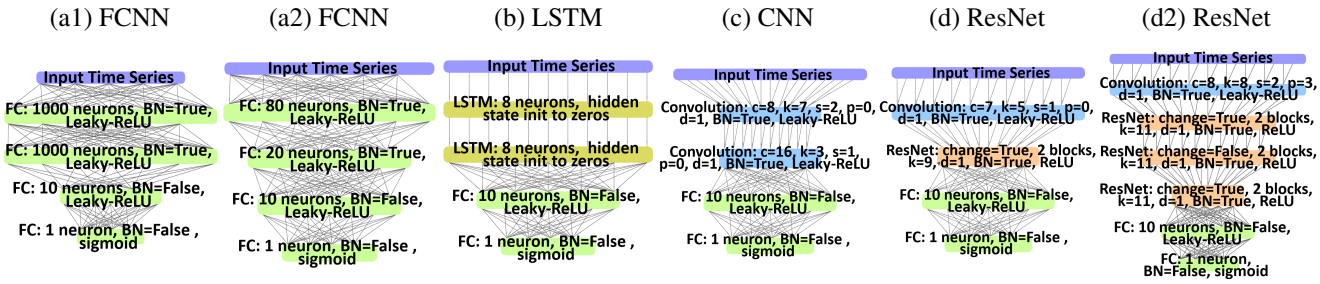


Fig. 1. Summary of Neural Network architectures.

fian chains correspond to the sigmoid activation functions in the output, having length 1 and order 2. The VCD upper bound thus reads: $VCD = (md)(md-1) + 16d + 2d(1+2m)(\log 3 + \log d)$.

The first contribution of the present work is to estimate the VCD upper bound for the NN architectures in Fig. 1. For a FCNN layer with n outputs, $m_{FCNN} = n$. For the LSTM layer used here (with 9 nonlinear funtions) and n neurons applied to an input of size l_{in}, $m_{LSTM} = 9l_{in}n$. For a CNN layer with kernel size k, padding size p, dilation d, stride s, with c_{in} and c_{out} input and output channels, applied to an input of size l_{in}, we have $m_{CNN} = c_{out}\lfloor\frac{l_{in}-2p-d(k-1)-1}{s}+1\rfloor$. For a ResNet layer, we add to m_{CNN} the input additions and reshaping. The final upper bound is obtained by considering the number of layers.

Table 1 compares NN complexities in terms of overall number of parameters, layers and VCD upper bound. The different architectures were designed to have equivalent complexities. For FCNN, it was not possible to achieve both same VCD and same number of parameters compared with other NN, so that two different FCNN are tested.

Classification task. The first goal is to investigate which class of architecture performs better in assessing temporal dynamics. To that end, the classification task consists in discriminating between two classes of (zero mean, unit variance) univariate MRW, $M = 1$. Both classes are designed to have the exact same second-order statistics (covariance functions). Thus NN will have to discover mild differences in high-order statistics temporal dynamics. Parameters $(H_A, \lambda_A) = (0.50, \sqrt{0.03})$ and $(H_B, \lambda_B) = (0.51, \sqrt{0.01})$ are selected.

Train and test sets. The same dataset is used to train all networks, consisting of 8000 series of length 2^{12}, half of them belonging to each class. Performance is estimated with a test set of 2000 series.

Optimization and training. All NN were implemented in PyTorch. Optimization was performed with *Adam* (learning rate of $1e-4$) and default initialisation. Five training epochs were used, and a batch size of 40. Batch Normalization [28] is applied to hidden layers (see Fig. 1). Each NN was trained from scratch 100 times, and average performance was computed.

Table 1. Architecture complexity. Number of parameters, layers and VCD (upper-bound) for architectures tested here.

	Parameters	layers	VCD
a1	5,112,021	4	1.06 E 20
a2	329,601	4	1.34 E 15
b	328,629	4	5.87 E 20
c	327,389	4	2.57 E 20
d	289,338	5	6.18 E 20

Table 2. Classification performance. Training time (in minutes) and performance (averages and std over 100 NN trained independently from scratch) for all architectures.

	a1	a2	b	c	d
Time (m)	0.6	**0.4**	11.7	**0.5**	1.0
Mean, train	**99.8**	**99.5**	51.5	**99.1**	**99.1**
(Std)	(0.1)	(0.4)	(5.0)	(0.2)	(0.4)
Mean, test	49.6	49.9	50.7	**97.4**	**97.3**
(Std)	(0.6)	(0.7)	(3.8)	(0.5)	(0.9)

Classification performance. Table 2 shows average classification performance and training times. Both FCNN perform equally bad despite a significant difference in complexity and despite satisfactory performance on the training set, suggesting overfitting. CNN (c) and ResNet (d) show indisputable superiority in performance, suggesting that linear time invariant filtering as performed by convolutional layers is well adapted to assessing temporal dynamics in time series. LSTMNN is slower to train (by one order of magnitude) and, surprisingly, has poor performance even on the training set, suggesting that its recurrent structure is not well suited to learn temporal dynamics, while the simpler linear filtering of CNN suffices.

5. CONVOLUTIONS AND TEMPORAL DYNAMICS

Classification task. Focusing on CNN only, the next goal is to compare performance as a function of NN depth and width. To that end, two classes of 4-variate MMRW, $M = 4$, are used, with the same joint second-order statistics: same

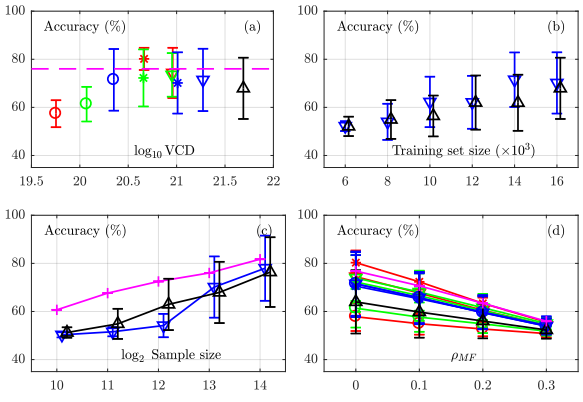


Fig. 2. **CNN performance** as a function of VCD (a), training set size (b), time series length (c), and class differences (d). Colors and symbols for CNNs are indicated in Table 3. Black $\triangle$ indicates ResNet.

Table 3. **VCD for CNN architectures.** The colors and symbols used in Fig. 2 are indicated.

Parameters	**Layers**		
	2 (red)	3 (green)	4 (blue)
$\sim$200k ($\circ$)	$0.6\,E\,20$	$1.2\,E\,20$	$2.3\,E\,20$
$\sim$330k ($*$)	$4.6\,E\,20$	$4.5\,E\,20$	$10\,E\,20$
$\sim$400k ($\triangledown$)	$9.1\,E\,20$	$9.0\,E\,20$	$19\,E\,20$

$H = [0.5, 0.6, 0.7, 0.8]$ and same Σ (for each pair, pairwise correlations of -0.3). Moreover, they all have the same $\lambda = \sqrt{0.03}$ so that their univariate higher order statistics (multifractality) are identical. For one class, all non diagonal entries of ρ_{MF} are set to 0.4. The other class has the same ρ_{MF} but for one entry set to 0. Therefore, the only difference in temporal dynamics between the two classes is subtle and located in cross high-order statistics (joint multifractality) for a single pair of components, resulting in a difficult classification problem.

CNN design. Nine CNN are designed with 3 different depths (2, 3 or 4 layers) , with 3 different numbers of parameters to train ($\simeq 200k$, $\simeq 330k$, $\simeq 400k$), as summarized in Table 3. In addition, the ResNet d2 shown in Fig. 1, designed to have an overall large complexity (9 layers, $\simeq 330k$ parameters and VCD$\simeq 49E20$) is also compared on the same task.

Training. The training setting is identical to the previous section, except that the training set size is now $20,000$ $4-$variate times series, each of length 2^{13}. Each network is trained from scratch 10 times, using 10 epochs.

Performance. Fig. 2(a)reports performance averaged across independent training. It shows that highest performance are obtained for several NN, with different numbers of layers, but with VCD around $\simeq 4.5 \cdot 10^{20}$. Further increasing NN complexity either in numbers of layers, parameters, or architecture (ResNet), slightly decreases performance, suggesting insufficient training. Finally, DL performance are comparable, but not superior, to a classical estimator combining 24 hand-crafted wavelet features (H, c_1, c_2) (see Section 2) with a support vector machine (SVM) (magenta dashed line).

Robustness against training set size. Fig. 2(b) reports classification performance for decreasing training set size (for the largest CNN and ResNet only, for simplicity). The dramatic decrease in performance quantifies the critical need of large datasets in NN training for complex classification tasks.

Robustness against time series length. Fig. 2(c) reports classification performance for decreasing time series length. The significant decrease in performance, far more pronounced than that observed with wavelet-SVM, indicates that CNN assessment of subtle differences in temporal dynamics is conditioned to the observation of long enough time series.

Robustness to departures from the training class. Fig. 2(d) reports performance when training and testing sets differ slightly. New (same size) testing sets are created by increasing progressively the entry of ρ_{mf}, originally set to 0 in the training set, to 0.3. It shows that DL architectures are not more robust than the wavelet-SVM approach when differences between the two classes decrease. Further, all DL architectures show the same performance decrease, suggesting that increasing VCD (and complexity) does not bring robustness (a potentially counter-intuitive outcome).

6. CONCLUSIONS AND PERSPECTIVES

The work proposed here was driven by the idea to apply DL to a problem for which a solution is already well-known. We believe that this methodology permits to obtain modest yet significant conclusions with respect to the ability of DL architectures to learn temporal dynamics, along two lines. First, DL architectures do manage to learn subtle differences in multivariate temporal dynamics blindly—i.e., with no prior knowledge on the temporal dynamics—as their performance compare to classical approaches relying on *informed* features, tailored to match temporal dynamics. Furthermore, among DL architectures of comparable complexities (quantified by VCD), CNN outperform others tested here—even the LSTM which explicitly models temporal evolution—highlighting the deep relation between time invariant convolution and temporal dynamics. Second, designing DL architectures, even within a family (CNN) is complicated, as the optimal architectures depend not only on how different classes are, but also crucially on the richness of the available training data sets (dataset size or time series sample size). We systematically analyzed this issue through the synthesis of training sets of different sizes. Also, often in practice, real data may not always follow exactly the properties of the training set, which may also result in a severe performance decrease. Further work is needed to better design relevant architectures for multivariate temporal dynamics classification with DL.

7. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv Neural Inf Proc Systems*, 2012, pp. 1097–1105.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions," in *Proc IEEE Conf Comp Vis Patt Rec*, 2015, pp. 1–9.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc IEEE Conf Comp Vis And Pattern Recogn*, 2016, pp. 770–778.

[4] S. Basu, M. Karki, S. Mukhopadhyay, S. Ganguly, R. Nemani, R. DiBiano, et al., "A theoretical analysis of deep neural networks for texture classification," in *Int Conf Neural Netw*. IEEE, 2016, pp. 992–999.

[5] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Int Conf Neural Netw*. IEEE, 2017, pp. 1578–1585.

[6] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[7] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Front Comput Sci*, vol. 10, no. 1, pp. 96–112, 2016.

[8] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *J Syst Eng Electron*, vol. 28, no. 1, pp. 162–169, 2017.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[12] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[13] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *29-th AAAI Conference on Artificial Intelligence*, 2015.

[14] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proc 23rd ACM Int Conf Multimedia*. Acm, 2015, pp. 1307–1310.

[15] P. Tanisaro and G. Heidemann, "Time series classification using time warping invariant echo state networks," in *Int Conf Mach Learn App*. IEEE, 2016, pp. 831–836.

[16] E. Bacry, J. Delour, and J.-F. Muzy, "Multifractal random walk," *Phys. Rev. E*, vol. 64: 026103, 2001.

[17] H. Wendt, R. Leonarduzzi, P. Abry, S. Roux, S. Jaffard, and S. Seuret, "Assessing cross-dependencies using bivariate multifractal analysis," in *IEEE Int. Conf. Acoust., Speech, and Signal Proces.*, April 2018.

[18] C. Meneveau, K. Sreenivasan, P. Kailasnath, and M. Fan, "Joint multifractal measures - theory and applications to turbulence," *Phys Rev A*, vol. 41, no. 2, pp. 894–913, 1990.

[19] G. Samorodnitsky and M. Taqqu, *Stable non-Gaussian random processes*, Chapman and Hall, New York, 1994.

[20] G. Didier, V. Pipiras, et al., "Integral representations and properties of operator fractional brownian motions," *Bernoulli*, vol. 17, no. 1, pp. 1–33, 2011.

[21] P. Abry, G. Didier, et al., "Wavelet estimation for operator fractional brownian motion," *Bernoulli*, vol. 24, no. 2, pp. 895–928, 2018.

[22] H. Wendt, P. Abry, and S. Jaffard, "Bootstrap for empirical multifractal analysis," *IEEE Signal Proc. Mag.*, vol. 24, no. 4, pp. 38–48, 2007.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[24] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of complexity*, pp. 11–30. 2015.

[25] E. B. Baum and D. Haussler, "What size net gives valid generalization?," in *Adv Neural Inf Proc Systems*, 1989, pp. 81–90.

[26] G. Friedland and M. Krell, "A capacity scaling law for artificial neural networks," *arXiv preprint arXiv:1708.06019*, 2017.

[27] M. Karpinski and A. Macintyre, "Polynomial bounds for VC dimension of sigmoidal and general neural networks," *J Comp Syst Sci*, vol. 54, no. 1, pp. 169–176, 1997.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.