REM WORKING PAPER SERIES

A deep learning test of the martingale difference hypothesis

João A. Bastos

REM Working Paper 0374-2025

March 2025

REM – Research in Economics and Mathematics
Rua Miguel Lúpi 20,
1249-078 Lisboa,
Portugal

ISSN 2184-108X

fct Fundação para a Ciência e a Tecnologia
U LISBOA | UNIVERSIDADE DE LISBOA
ISEG Lisbon School of Economics & Management Universidade de Lisboa
REM RESEARCH IN ECONOMICS AND MATHEMATICS

REM – Research in Economics and Mathematics

Rua Miguel Lupi, 20
1249-078 LISBOA
Portugal

Telephone: +351 - 213 925 912
E-mail: rem@iseg.ulisboa.pt

https://rem.rc.iseg.ulisboa.pt/

https://twitter.com/ResearchRem

https://www.linkedin.com/company/researchrem/

https://www.facebook.com/researchrem/

A deep learning test of the martingale difference hypothesis

João A. Bastos*

Lisbon School of Economics & Management (ISEG)
Universidade de Lisboa. Portugal

Abstract

A deep learning binary classifier is proposed to test if asset returns follow martingale difference sequences. The Neyman-Pearson classification paradigm is applied to control the type I error of the test. In Monte Carlo simulations, I find that this approach has better power properties than variance ratio and portmanteau tests against several alternative processes. I apply this procedure to a large set of exchange rate returns and find that it detects several potential deviations from the martingale difference hypothesis that the conventional statistical tests fail to capture.

Keywords: Martingale difference hypothesis; Convolutional network; Variance ratio test; Portmanteau test; Exchange rates

1 Introduction

The martingale difference hypothesis posits that variations in asset prices are mean-independent at all leads and lags or, equivalently, that they follow a martingale difference sequence. A weaker version asserts that these variations are just uncorrelated at all leads and lags. If this hypothesis holds, the forecasts for future asset prices that minimize the expected squared error are the last observed price. The most straightforward strategy to test this hypothesis is to look for statistically significant autocorrelations in the returns. For instance, we can detect departures from zero in either direction using a portmanteau test based on the sum of the squared autocorrelations up to a given lag (e.g., Ljung and Box, 1978). A test statistic with better power properties against several alternative

*E-mail address: jbastos@iseg.ulisboa.pt

hypotheses is the variance ratio, the asymptotic properties of which were first derived by Lo and MacKinlay (1988). This test is based on the fact that under the martingale difference hypothesis the variance of the k-period returns must be k times the variance of the one-period returns. Since the foundational work of Lo and MacKinlay (1988), the variance ratio test emerged as a valuable tool for testing whether asset returns are serially uncorrelated, and several extensions of the original statistic were proposed in the literature, such as the multiple variance ratio test of Chow and Denning (1993), the automatic variance ratio of Choi (1999), the ranks and signs test of Wright (2001), the wild bootstrap test of Kim (2006), and the panel data test of Kim and Shamsuddin (2015). Tests based on nonlinear measures of dependence were also proposed by several authors, such as the Dominguez and Lobato (2003) test based on the Cramer-von Mises and Kolmogorov-Smirnov statistics, and the generalized spectral test of Escanciano and Velasco (2006). A comprehensive review of tests for the martingale difference hypothesis can be found in Escanciano and Lobato (2009a).

Judging whether a return series follows a martingale difference sequence is a binary decision problem. An alternative to statistical hypothesis tests to handle this problem is machine learning classification – a tool for assigning new observations to one of several categories or *classes*, using other observations with known class labels. In particular, a *binary* classifier is a statistical model that categorizes new observations into two classes, usually labeled as 0 and 1. This classifier can generate two types of errors: classify a class 0 observation as class 1, producing a type I error; or classify a class 1 observation as class 0, leading to a type II error. Classification algorithms are usually trained to minimize the expected number of misclassified observations, thereby giving equal importance to type I and II errors. This contrasts with the Neyman-Pearson framework of hypothesis testing in which these errors have different priorities. In this framework, a level of significance α (the *size* of the test) is imposed on the probability of a type I error. Then, one looks for a test that satisfies this constraint while minimizing the probability of type II errors, or equivalently, maximizing the probability of detecting class 1 events (the *power* of the test).

In this paper, I propose testing the martingale difference hypothesis using a binary classifier. Because this hypothesis is a cornerstone of the efficient-market hypothesis in financial economics (Fama, 1970), I want to be on the conservative side with respect to rejecting it. Therefore, I use the *Neyman-Pearson approach* to classification (Scott and Nowak, 2005; Tong and Rigollet, 2011; Tong, 2013; Zhao et al, 2016), which, as the name implies, is related to the conventional Neyman-Pearson framework for statistical hypothesis testing. Neyman-Pearson classification allows us to incorporate in classification models asymmetric type I and type II errors and contain the population type I

errors just below a nominal test size α with high probability. In particular, I use the 'umbrella' approach of Tong et al (2018) that provides a model-agnostic implementation of the Neyman-Pearson paradigm. The classification model is a convolutional neural network or *convnet*. This is a specialized deep neural network originally developed for image recognition. However, convnets can process any data with a grid-like topology and therefore can be 'downgraded' to process temporal sequences sampled at regular intervals. Because they learn local, translation equivariant features in the data, they are highly efficient on perceptual problems.

Using Monte Carlo simulations, I benchmark the performance of this procedure against the automatic variance ratio (AVR) test of Kim (2009), which is based on wild bootstrapping the AVR statistic of Choi (1999), and against the portmanteau test with automatic lag selection (AQ) of Escanciano and Lobato (2009b). Charles et al. (2011) find that these tests have excellent power properties against a wide range of linear and nonlinear processes, with no size distortions. Also, they do not require the somewhat arbitrary selection of the k-period used for calculating returns. I find that a convnet carefully designed to detect deviations from the martingale difference hypothesis has greater power than the AVR and AQ tests against several alternatives. An empirical application to exchange rates shows that the deep learning approach detects potential deviations from the martingale difference hypothesis that the statistical tests fail to capture. Therefore, this model is a more efficient approach to learn patterns in financial returns than conventional statistical tests. While convnets may be regarded as 'black-box' models, we can easily visualize how they respond to different return series and understand which patterns have been learned.

In the following section, I provide an overview of the classifier designed to test the martingale difference hypothesis in financial returns. I present the Neyman-Pearson classification framework and the architecture of the convolutional neural network. Section 3 shows the results of the Monte Carlo simulations. Section 4 contains an empirical application to two sets of exchange rates. Section 5 provides the conclusions.

2 A classifier for testing the martingale difference hypothesis

2.1 Neyman-Pearson classification

Consider a set of observations described by a d-dimensional vector of regressors $\boldsymbol{X} \in \mathcal{X} \subset \mathbb{R}^d$. These observations belong to two classes, labeled as $Y \in \{0, 1\}$. A binary classifier is a data-dependent mapping $\phi : \mathcal{X} \to \{0, 1\}$ that assigns an observation to

one of these classes. Classification algorithms typically produce a function $f(\boldsymbol{X})$ that provides a numerical value called the *score*. The higher the value of the score for a given observation, the higher is the likelihood of belonging to the class labeled as 1. Therefore, a binary classifier is obtained by applying a cutoff $c \in \mathbb{R}$ on the scores:

$$\phi(X) = I(f(\boldsymbol{X}) \geq c), \tag{1}$$

where $I(\cdot)$ is the indicator function.

A common approach to measure performance on a classification task is in terms of the expected error rate

$$R(\phi) = E\left[I\left(\phi(X) \neq Y\right)\right] = P\left[I\left(\phi(X) \neq Y\right)\right], \tag{2}$$

This error rate can be decomposed as

$$R(\phi) = P(Y = 0)R_0(\phi) + P(Y = 1)R_1(\phi), \tag{3}$$

where

$$\begin{aligned} R_0(\phi) =& P\left[I\left(\phi(X) \neq Y\right)|Y = 0\right] \text{ is the type I error probability, and} \\ R_1(\phi) =& P\left[I\left(\phi(X) \neq Y\right)|Y = 1\right] \text{ is the type II error probability.} \end{aligned} \tag{4}$$

These probabilities are not observable and must be estimated from the data. In classification problems one typically tries to find the classifier that minimizes the overall error due to misclassified observations,

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\, R(\phi), \tag{5}$$

and, therefore, the same importance is given to type I and type II errors. However, in many domains the misclassification errors made by a classifier are not equally important. For instance, in credit risk management it is more costly for a bank to provide credit to a client that defaults (false negative), than to deny credit to a client that would not default (false positive).

Here, we should also be on the conservative side with respect to rejecting the martingale difference hypothesis since it corresponds to the null hypothesis in the statistical hypothesis tests. If the martingale difference hypothesis is labeled as 0, we seek a classifier that constrains the probability of type I errors below the significance level α with high confidence, while trying to minimize the probability of type II errors. To achieve this goal we use Neyman-Pearson classification (Scott and Nowak, 2005; Tong and Rigollet, 2011; Tong, 2013; Zhao et al, 2016). This approach seeks a classifier that satisfies as well as possible the problem

$$\phi^*_{NP} = \underset{\phi: R_0(\phi) < \alpha}{\operatorname{argmin}}\, R_1(\phi). \tag{6}$$

To implement a Neyman-Pearson classifier we could find the model that best discriminates the two classes, and choose the smallest cutoff value c_α that results in an empirical type I error rate not greater than α in an independent validation sample,

$$c_\alpha = \inf \left\{ c : \frac{\sum_{i=1}^{n} I(f(\boldsymbol{x}_i) \geq c, y_i = 0)}{\sum_{i=1}^{n} I(y_i = 0)} \leq \alpha \right\}, \tag{7}$$

where n is the size of this sample. The classification rule would then be

$$\phi_\alpha(\boldsymbol{X}) = I\left(f(\boldsymbol{X}) \geq c_\alpha\right). \tag{8}$$

However, Tong et al (2018) showed that this approach results in about half of the classifiers having type I error rates above α. Then, they propose a model-agnostic approach to control the type I error under α with high probability. This approach can be applied to any classification model that provides numerical scores. The idea is to choose the smallest threshold on the classification scores such that the probability that the population type I error rate is larger than α is smaller than a given tolerance level δ.

Suppose that we train a classification algorithm and obtain the scoring function $f(\boldsymbol{X})$. Furthermore, suppose that we have an independent validation sample of size n containing observations belonging to the class labeled as 0. Applying $f(\cdot)$ to these observations we obtain a set of n scores, $\hat{Y}_1, \ldots, \hat{Y}_n$. Let $\hat{Y}_{(k)}$ denote the kth order statistic of the set $\{\hat{Y}_i\}_{i=1}^n$, and let $\phi_k(\boldsymbol{X})$ denote the classifier

$$\phi_k(\boldsymbol{X}) = I\left(f(\boldsymbol{X}) \geq \hat{Y}_{(k)}\right). \tag{9}$$

Tong et al (2018) show that the population type I error rate of $\phi_k(X)$, $R_0(\phi_k)$, satisfies

$$\Pr(R_0(\phi_k) > \alpha) \leq \nu(k) := \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}, \tag{10}$$

where $\nu(k)$ is the 'violation rate'. Note that as k decreases or, equivalently, the threshold value $\hat{Y}_{(k)}$ decreases the violation rate $\nu(k)$ increases. To obtain a Neyman-Pearson classifier we select the smallest value of k than gives a violation rate within the chosen tolerance level,

$$k^* = \min\left\{k \in \{1, \ldots, n\} : \nu(k) \leq \delta\right\}. \tag{11}$$

The critical value of the test is $\hat{Y}_{(k^*)}$.

2.2 A convnet for testing the martingale difference hypothesis

The binary classifier is a convolutional neural network or convnet (LeCun, 1989). To conserve space only the most relevant ideas of convolutional neural networks are provided

here.[1] Figure 1 shows a diagram of a convnet architecture for testing the martingale difference hypothesis. Like many deep learning architectures, a convnet has a layered structure where information flows through consecutive layers from the inputs to the outputs. Because we simply want to discriminate two types of stochastic processes – martingale difference sequences (MDS) from models with serial dependence – the convnet has just a single output. If we encode the MDS as 0 and the models with serial dependence as 1, a well-trained network will provide an output value close to 1 if the time series presented at its input has serial dependence, and a value close to 0 otherwise. So far, a convnet is not very different from the binary choice models used in applied econometrics.

In principle, we could feed the binary choice model with the raw values of the time series. However, this does not work well in practice – a better approach would be to feed the model with a low dimensional set of *features*. For instance, these features could be obtained from the distributional properties of the returns, short-term serial dependence, conditional volatility, and so on (see, e.g. Bastos and Caiado, 2021). However, instead of preprocessing the data to derive specific features, we can use a convnet. The convnet takes the raw values of the time series and automatically "learns" how to extract meaningful features, better predicting the class to which a time series belongs.

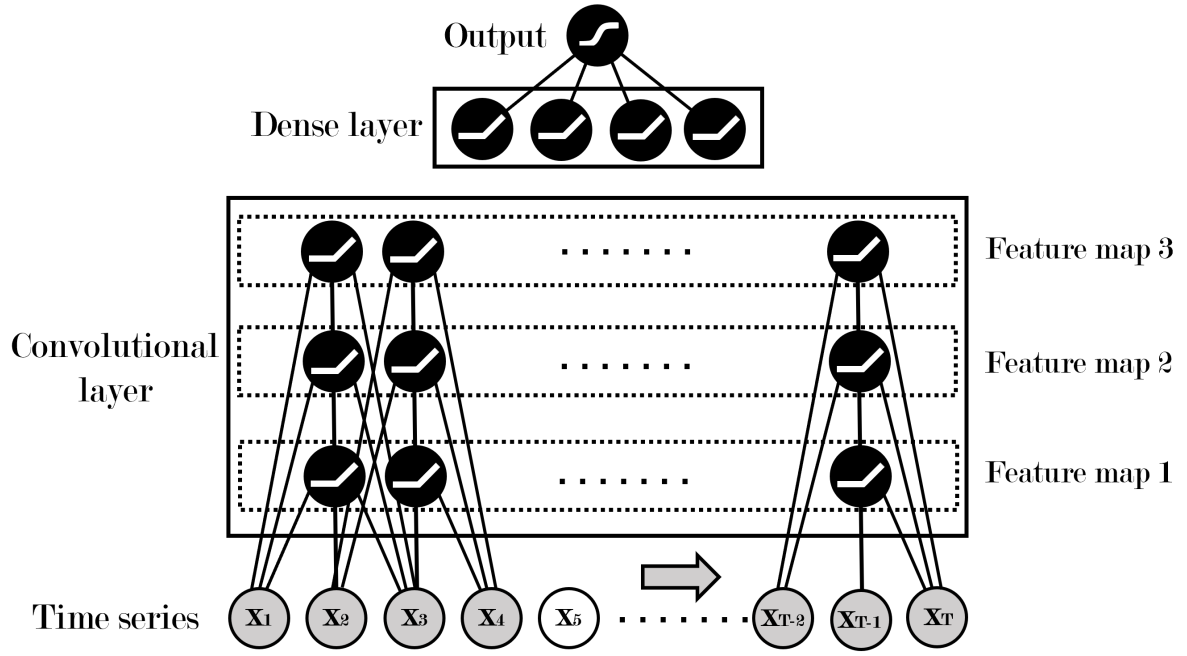

Figure 1: Scheme of a convolutional neural network for testing the martingale difference hypothesis.

The box at the bottom is called the 'convolutional layer' – the perceptual component

[1]The reader may find an authoritative exposition of convolutional neural networks in Goodfellow et al. (2016).

of the model. A convnet for image recognition typically has several convolutional layers stacked on top of each other, with the layers closer to the output learning increasingly abstract patterns in the image. Here, a single convolutional layer is sufficient, given the absence of pattern hierarchies and the low signal-to-noise ratio in financial returns. The convolutional layer consists of several sublayers called 'feature maps'. Each feature map learns a different translation equivariant pattern in the time series, such as low order autocorrelations, nonlinear dependence, and long memory. Figure 1 shows a convolutional layer with three response maps, but this number is typically larger depending on the number of learnable patterns in the data.

Each feature map is composed of 'units', represented by dark circles in Figure 1. Each unit is connected to the values of the input data that are covered by its 'receptive field'. In image recognition, the size of the two-dimensional receptive fields is typically small, say 3×3 pixels. For analyzing asset returns, the size of the one-dimensional receptive field should be large enough to detect the presence of long-range serial dependence. I use receptive fields of size 12, corresponding to 3 months for weekly returns or 1 year for monthly returns. Each unit computes the result of applying an 'activation function' $g(\cdot)$ to a weighted sum of the inputs covered by its receptive field. The weights define the strength of the connections. The activation function used here is the *rectified linear unit* (Glorot et al, 2011) or ReLU, $g(z) = \max(0, z)$.

Because within a given feature map the weights of the linear combinations are constrained to be equal across all units, a pattern learned at a certain position of the return series may be recognized at a different position. On top of the convolutional layer sits a 'dense layer'. Each unit in the dense layer is connected to all units in the convolution layer (these connections are not represented in Figure 1). The task of the convolutional layer is to recognize patterns in the data to be processed by the dense layer. The output unit computes a linear combination of the outputs from the units in the dense layer and applies a logistic function to it, yielding an output value bounded to (0,1) which is the convnet score.

The convnet learns using the backpropagation algorithm (Rumelhart et al, 1986). First, the network weights are initialized using a random initialization scheme (Glorot and Bengio, 2010). Then, a batch of data randomly drawn from the training dataset is passed through the layers. After processing this batch, the convnet calculates the value of a 'cost function' based on how far the scores are from 0, if the returns are martingale difference sequences, or from 1, if the returns have serial dependence. Then, the network weights are adjusted to minimize the cost function using a gradient descent algorithm with adaptive learning rate (Kingma and Ba, 2015). This process is repeated for different batches of data, until all observations in the training dataset are used. Finally, the

training process executes many iterations through the training set, called 'epochs'. The optimal number of epochs is chosen by looking at the accuracy in an independent dataset. The pseudocode for this network can be found in the appendix. The R source code with the full architecture of the convnet, datasets, and trained models can be found in the supplementary material.

3 Monte Carlo results

To evaluate the power properties of the deep learning approach and compare them to those of alternative hypothesis tests, I consider the models in the Monte Carlo simulation of Kim (2009) that are shown in Table 1. Let $\{x_t\}_{t=1}^T$ denote a series of log returns. The first models correspond to three alternative specifications of the null hypothesis that log returns are MDS. Model 1 is a GARCH(1,1) model, Model 2 is a stochastic volatility (SV) model, and Model 3 is an IGARCH(1,1) model. The remaining models specify two alternative hypotheses. Model 4 is an AR(1) process, whereas Model 5 is an ARFIMA(0, 0.1, 0) or long memory process. The innovations ε_t are generated according to $N(0,1)$, whereas ξ_t is generated according to $N(0, 0.1)$ and is independent of ε_t. The processes u_t are used as innovations for models 4 and 5.

Model 1	$x_t = u_t, \;\; u_t = \sqrt{h_t}\varepsilon_t, \;\; h_t = 0.5 + 0.75h_{t-1} + 0.1\varepsilon_{t-1}^2$
Model 2	$x_t = u_t, \;\; u_t = \exp(0.5h_t)\varepsilon_t, \;\; h_t = 0.95h_{t-1} + \xi_t$
Model 3	$x_t = u_t, \;\; u_t = \sqrt{h_t}\varepsilon_t, \;\; h_t = 0.5 + 0.90h_{t-1} + 0.1\varepsilon_{t-1}^2$
Model 4	$x_t = 0.1x_{t-1} + u_t$
Model 5	$(1-B)^{0.1}x_t = u_t$

Table 1: Models used in the Monte Carlo simulation.

For each scenario, I generated 50,000 MDS and 50,000 series with linear dependence. Then, a convnet was trained to discriminate the two processes (for instance, GARCH(1,1) against AR(1) with GARCH(1,1) innovations). Then, I generated an independent validation dataset containing two additional groups of 50,000 MDS and 50,00 linearly dependent returns which were processed by the trained model. The scores given by the convnet for the MDS in the validation data provide the null distribution. The scores for the linearly dependent returns in the validation data yield the alternative distribution. Because the MDS are coded as 0, and the alternative hypotheses are coded as 1, rejections are in the right tail of the null distribution.

Figure 2 shows the class-conditional densities of the validation data scores given by convolutional neural networks for return series of size $T = 100, 500, 1000$. The dark gray distributions correspond to the null hypothesis of MDS with GARCH(1,1) innovations,

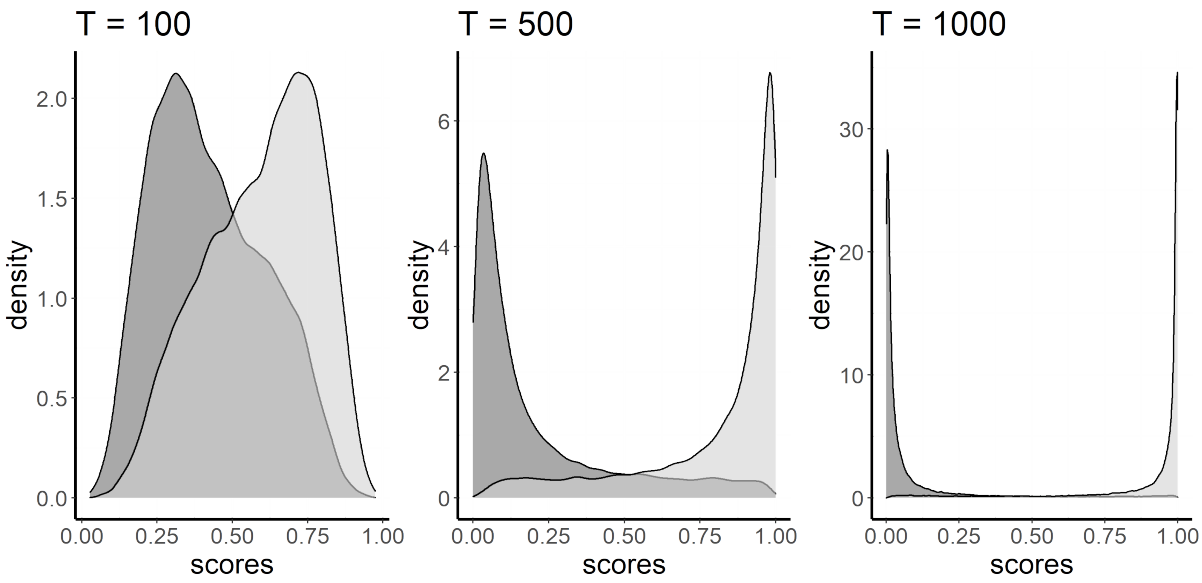

Figure 2: Class-conditional densities of convnet scores for return series of size $T = 100, 500, 1000$. The dark gray distributions correspond to the null hypothesis of MDS with GARCH(1,1) innovations, while the light gray distributions correspond to the alternative hypothesis of AR(1) with GARCH(1,1) innovations.

while the light gray distributions correspond to the alternative hypothesis of AR(1) with GARCH(1,1) innovations. As the size of the time-series increases the discrimination power of the network improves.

The empirical size of the convnets is the proportion of type I errors in the validation data – the proportion of observations in the null distribution of the validation data with score above the critical values $\hat{Y}_{(k^*)}$, where k^* is given in Equation (11). Table 2 shows the empirical size of the convnets for a nominal size $\alpha = 0.05$ and for 3 different levels of the tolerance level δ – the probability that the population type I error exceeds α. I do not report the size properties of the AVR and AQ tests since they show no size distortions. The lower the tolerance with respect to population type I errors, the lower is the empirical size of the test. Still, for all scenarios the empirical size is close to the nominal level.

The empirical power of the convnets is the proportion of observations in the alternative distribution for the validation data with score above the critical value. Using the Neyman-Pearson classification framework, I calculated the critical values $\hat{Y}_{(k^*)}$ for $\alpha = 0.05$ and a tolerance level $\delta = 0.05$. Table 3 reports the power properties of the convnet, the automatic variance ratio test (AVR) of Kim (2009), and the automatic portmanteau test (AQ) of Escanciano and Lobato (2009b), for return series of size $T = 100, 500$ and 1000, and for a significance level of 5%. For the AVR and AQ tests the number of Monte Carlo trials is set to 1000. The number of bootstrap iterations in the AVR test is set to 500. For all scenarios under consideration, the convnets are more powerful than the AVR and

	AR(1) – GARCH			AR(1) – SV			AR(1) – IGARCH		
$T \setminus \delta$	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
100	4.68	4.77	4.86	4.76	4.81	4.85	4.68	4.76	4.81
500	4.74	4.82	4.87	4.77	4.84	4.87	4.68	4.77	4.82
1000	4.77	4.84	4.87	4.77	4.84	4.87	4.68	4.77	4.82
	ARFIMA – GARCH			ARFIMA – SV			ARFIMA – IGARCH		
$T \setminus \delta$	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
100	4.76	4.83	4.85	4.72	4.83	4.87	4.68	4.77	4.82
500	4.77	4.84	4.87	4.77	4.84	4.88	4.68	4.77	4.82
1000	4.77	4.84	4.87	4.77	4.84	4.88	4.68	4.77	4.82

Table 2: Empirical size of the test (%) for a nominal size of 5% and different values of the tolerance parameter δ.

	AR(1) – GARCH			AR(1) – SV			AR(1) – IGARCH		
T	**Convnet**	**AVR**	**AQ**	**Convnet**	**AVR**	**AQ**	**Convnet**	**AVR**	**AQ**
100	21.8	18.7	17.3	22.3	19.2	14.1	20.9	19.6	15.6
500	60.5	55.9	53.6	67.3	52.0	54.6	65.7	55.1	55.7
1000	87.6	81.2	87.2	91.1	81.4	83.5	89.6	88.3	87.7
	ARFIMA – GARCH			ARFIMA – SV			ARFIMA – IGARCH		
T	**Convnet**	**AVR**	**AQ**	**Convnet**	**AVR**	**AQ**	**Convnet**	**AVR**	**AQ**
100	33.8	25.9	21.1	33.6	25.1	18.8	33.0	26.4	18.0
500	84.1	66.2	64.2	83.8	68.5	64.0	84.2	72.5	64.1
1000	97.7	90.8	90.2	98.1	93.7	90.4	97.2	93.9	93.1

Table 3: Power properties of the convnet, and AVR and AQ tests for the alternative processes. The level of significance is 5%.

AQ tests.

4 Empirical application

To test the martingale difference hypothesis in real data, we must define two stochastic processes that the convnet will learn to discriminate. The first process, corresponding to the null hypothesis, is that the returns are martingale difference sequences with conditional heteroskedasticity,

$$H_0 : x_t = \sqrt{h_t}\varepsilon_t, \quad h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2, \quad \varepsilon_t \sim N(0,1). \tag{12}$$

The second process corresponds to the alternative hypothesis

$$H_1 : x_t = \phi x_{t-1} + \sqrt{h_t}\varepsilon_t, \quad h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2, \quad \varepsilon_t \sim N(0,1). \tag{13}$$

To obtain a rich variety of stochastic processes, the autoregressive parameter for x is generated according to $\phi \sim U(0.01, 0.15)$, and the heteroskedasticity parameters are generated according to $\alpha_0 \sim U(0, 0.001)$ and $\alpha_1 \sim U(0.85, 0.99)$, where U is the uniform distribution.

First, I trained a convnet using 50,000 temporal sequences generated according to H_0, and 50,000 generated according to H_1. The length of the generated return series is equal to the length of the real return series for which I want to test the martingale difference hypothesis. Again, the null model is encoded as 0 and the alternative model is encoded as 1, and therefore the rejections are in the right tail of the null distribution. After training the convnet, I generated an independent test set comprising 100,000 return series according to the null hypothesis (H_0). This test set yields a set of scores generated by the trained model: $\hat{Y}_1, \ldots, \hat{Y}_n$. Using this distribution of scores, we can calculate the critical value for a given significance level α as $\hat{Y}^{\alpha}_{k^*}$, where k^* is determined by Equation 11. If a real return series obtains a score of $\hat{Y}_{n+1}$, we reject H_0 at level α if $\hat{Y}_{n+1} > \hat{Y}^{\alpha}_{k^*}$. It is important to note that the real data are not used in training the model nor in the generation of the null distribution. The first time the model 'sees' a real return series is when it computes its score.

As an empirical application, I consider two sets of exchange rates. The martingale difference hypothesis in exchange rate returns was studied by many authors, and the evidence against it is not conclusive (Escanciano and Lobato, 2009a). The first empirical application uses the data of Wright (2001). It consists of weekly nominal exchange rates for the Canadian dollar, French franc, German mark, Japanese yen, and British pound against the US dollar, covering the period from August 7, 1974, to May 29, 1996. The returns are given by the first difference of the log-exchange rates. Table 4 shows the convnet scores for the Wright (2001) exchange rates. The critical values for rejection at 10%, 5% and 1% significance levels are 0.617, 0.768, and 0.955, respectively. The rightmost columns report the p-values for the AVR and AQ tests. The convnet and the AVR test reject the martingale difference hypothesis for all exchange rates but the British pound at 10% significance level. The AQ test fails to reject the martingale difference hypothesis for the British pound and the French franc.

In a second empirical illustration, I use the noon buying rates in New York of 23 currencies against the US dollar published by the Federal Reserve.[2] The data cover the period from January 1, 2000, to December 31, 2020, with a total of 5479 observations. To obtain weekly data I considered the rates published on Wednesdays. If on a given Wednesday the market was closed, the rate on the following trading day was used. To obtain monthly data I used the last available rate on each month. The sample sizes are

[2]https://www.federalreserve.org/Release/h10/hist

Currency	**Convnet scores**	**AVR test p-values**	**AQ test p-values**
Canadian dollar	0.982***	0.016	0.040
German mark	0.801**	0.040	0.098
French franc	0.694*	0.096	0.177
British pound	0.386	0.360	0.435
Japanese yen	0.914**	0.010	0.000

Table 4: Convnet scores for the Wright (2001) exchange rates. The critical values for rejection at 10%, 5% and 1% significance levels are 0.617, 0.768, and 0.955, respectively. *, ** and *** indicate rejection of the null at 10%, 5% and 1% levels, respectively. The rightmost columns show the p-values for the AVR and AQ tests.

1096 and 252 for weekly and monthly data, respectively.

Table 5 reports the convnet scores for the weekly and monthly returns of the Federal Reserve of New York exchange rates. For weekly returns, the critical values for rejection at 10%, 5% and 1% significance levels are 0.567, 0.728, 0.937, respectively. For monthly returns, the critical values for rejection at 10%, 5% and 1% significance levels are 0.662, 0.746, and 0.871, respectively. The p-values for the AVR and AQ tests are also shown. Considering a 10% significance level, the three approaches reject the martingale difference hypothesis for weekly and monthly returns of the Chinese yuan and the Taiwanese dollar, and for weekly returns of the Malaysian ringgit. The convnet and the AVR test further reject this hypothesis for monthly returns of the Sri Lankan rupee. Furthermore, the convnet and AVR test rejects it for weekly returns of the Singapore dollar, whereas the AQ test fails the rejection at 10% level. The convnet detects potential deviations from the martingale difference hypothesis that the statistical tests fail to capture. The convnet rejects this hypothesis for weekly returns of the Thai baht and Venezuelan bolivar at 5% level. It is surprising that the statistical tests fail to detect violations of the martingale difference hypothesis in the bolivar given the hyperinflation in Venezuela in recent years. There are indications that this hypothesis may be violated at a 10% significance level for monthly returns of the Danish krone, Swedish krona, and Indian rupee. However, the observed scores for these currencies are very close to the 10% critical value.

5 Conclusions

In this paper, I analyzed the properties of a deep learning approach based on convnets for testing if asset returns follow martingale difference sequences. The Neyman-Pearson classification paradigm was used as a tool to control the type I error of the test. In

Currency	Weekly data			Monthly data		
	Convnet scores	AVR test p-values	AQ test p-values	Convnet scores	AVR test p-values	AQ test p-values
Australian dollar	0.283	0.866	0.985	0.496	0.402	0.378
Brazilian real	0.415	0.354	0.328	0.524	0.564	0.707
British pound	0.121	0.766	0.632	0.503	0.504	0.521
Canadian dollar	0.195	0.922	0.748	0.325	0.558	0.485
Chinese yuan	0.982***	0.022	0.096	0.991***	0.000	0.000
Danish krone	0.336	1.000	0.963	0.711*	0.584	0.635
Euro	0.495	0.984	0.948	0.587	0.588	0.633
Hong Kong dollar	0.246	0.512	0.671	0.485	0.926	0.791
Indian rupee	0.586*	0.226	0.300	0.703*	0.196	0.182
Japanese yen	0.141	0.708	0.750	0.670*	0.840	0.738
Malaysian ringgit	0.978***	0.044	0.050	0.765**	0.640	0.624
Mexican peso	0.219	0.972	0.984	0.668*	0.330	0.332
New Zealand dollar	0.094	0.302	0.442	0.394	0.976	0.916
Norwegian krone	0.167	0.596	0.588	0.608	0.478	0.444
Singapore dollar	0.713*	0.066	0.105	0.703*	0.960	0.786
South African rand	0.280	0.500	0.532	0.466	0.866	0.915
South Korean won	0.296	0.286	0.403	0.371	0.506	0.666
Sri Lankan rupee	0.292	0.916	0.834	0.793**	0.100	0.424
Swedish krona	0.179	0.766	0.746	0.671*	0.380	0.358
Swiss franc	0.628*	0.998	0.977	0.325	0.194	0.275
Taiwan dollar	0.979***	0.012	0.015	0.895***	0.032	0.025
Thai baht	0.948***	0.190	0.284	0.813**	0.480	0.516
Venezuelan bolivar	0.778**	0.622	0.957	0.660	0.656	0.535

Table 5: Convnet scores for weekly and monthly returns of the Federal Reserve of New York exchange rates. For weekly returns, the critical values for rejection at 10%, 5% and 1% significance levels are 0.567, 0.728, 0.937, respectively. For monthly returns, the critical values for rejection at 10%, 5% and 1% significance levels are 0.662, 0.746, and 0.871, respectively. *, ** and *** indicate rejection of the null at 10%, 5% and 1% levels, respectively. The rightmost columns show the p-values for the AVR and AQ tests.

Monte Carlo simulations, I found that this approach has better power properties than automatic variance ratio and portmanteau tests against several alternative processes. I also presented an empirical application to a large set of exchange rates. I showed that this approach can detect potential deviations from the martingale difference hypothesis that the alternative tests failed to capture. Because these models are highly efficient in

perceptual problems, they are a valuable tool to learn patterns in financial returns that are not captured by conventional statistical tests.

Appendix

The pseudocode below shows the algorithm for testing the martingale difference hypothesis with a convolutional neural network. The R source code implementing this network can be found in the supplementary material. In this repository the reader can also find the R source code for the simulation and empirical results, the datasets used in the study, and the trained models in h5 format. Several combinations of *hyper-parameters* (parameters that are not learned in the training process) were evaluated using grid-search and 5-fold cross validation. The following hyper-parameters values were considered: number of feature maps $\in \{8, 16, 32, 64\}$; max pooling size $\in \{2, 4\}$; number of units in dense layer $\in \{4, 8, 16, 32\}$.

INPUTS: Time series of log-returns, $\boldsymbol{x}$; number of Monte Carlo trials, N; significance level α; tolerance level, δ
OUTPUT: decision on null hypothesis that $\boldsymbol{x}$ is MDS

FUNCTION: convnet(data)
    CREATE sequential model
    ADD 1D convolutional layer with 32 filters, kernel size of 12, and ReLU activation function
    ADD 1D max pooling layer with pool size of 4
    ADD flatten layer
    ADD dense layer with 16 units and ReLU activation function
    ADD dropout layer with rate of 0.25
    ADD a dense layer with 1 unit and sigmoid activation function
    TRAIN with data, binary cross-entropy loss, Adam optimizer with learning rate of 1E-5
END FUNCTION

$T \leftarrow$ size of $\boldsymbol{x}$
$\boldsymbol{X}_0 \leftarrow N$ vectors with MDS of size T
$\boldsymbol{X}_1 \leftarrow N$ vectors with autoregressive series of size T
$Y_0 \leftarrow$ vector of 0's of size N
$Y_1 \leftarrow$ vector of 1's of size N
CALL convnet($\{\boldsymbol{X}_0, Y_0\}$; $\{\boldsymbol{X}_1, Y_1\}$)

$\boldsymbol{X}_{\text{validation}} \leftarrow N$ vectors with MDS of size T
$scores \leftarrow$ **PREDICT** using convnet on $\boldsymbol{X}_{\text{validation}}$
$score_{\boldsymbol{x}} \leftarrow$ **PREDICT** using convnet on $\boldsymbol{x}$
FOR k = 1 **TO** N

$\nu(k) \leftarrow \sum_{j=k}^{N} \binom{N}{j}(1-\alpha)^j \alpha^{N-j}$

END FOR

$k^* \leftarrow \min\{k \in \{1,\ldots,N\} : \nu(k) \leq \delta\}$

$sorted\_scores \leftarrow \textbf{sort}(scores)$

$critical\_value \leftarrow sorted\_scores[k^*]$.

IF $score_{\boldsymbol{x}} > critical\_value$ **THEN**

OUTPUT "Reject the null that $\boldsymbol{x}$ is MDS"

ELSE

OUTPUT "Do not reject the null that $\boldsymbol{x}$ is MDS"

END IF

Data availability statement

The data and code that support the findings of this study are available in the online appendix.

Acknowledgements

I would like to express my gratitude to the reviewers for their valuable comments and suggestions. Additionally, I would like to thank João Nicolau for his valuable feedback on an early draft of this paper. This work was supported by the FCT – Fundação para a Ciência e a Tecnologia [grant number UIDB/05069/2020].

References

Bastos J.A., Caiado, J. (2021). On the classification of financial data with domain agnostic features. International Journal of Approximate Reasoning 138, 1–11.

Charles, A., Darné, O., Kim, J.H. (2011) Small sample properties of alternative tests for martingale difference hypothesis. Economics Letters 110, 151–154.

Chen, W.W., Deo R.S. (2006). The variance ratio statistic at large horizons, Econometric Theory 22, 206-234.

Choi I. (1999). Testing the random walk hypothesis for real exchange rates. Journal of Applied Econometrics, 14, 293-308.

Chow, K.V., Denning, K.C. (1993). A simple multiple variance ratio test. Journal of Econometrics 58, 385-401.

Dominguez, M.A., Lobato, L.N. (2003). Testing the martingale difference hypothesis. Econometric Reviews 22, 351-377.

Escanciano, J.C., Velasco, C. (2006). Generalized spectral tests for the martingale difference hypothesis. Journal of Econometrics 134, 151-185.

Escanciano, J.C., Lobato, I.N. (2009) Testing the martingale hypothesis. In: Mills T.C., Patterson K. (eds) Palgrave Handbook of Econometrics. Palgrave Macmillan, London.

Escanciano, J.C., Lobato, I.N. (2009) An automatic portmanteu test for serial correlation. Journal of Econometrics 151, 140-149.

Fama, E.F. (1970). Efficient capital markets: A review of theory and empirical work. Journal of Finance 25, 383-417.

Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9:249-256.

Glorot, X., Borde, A., Bengio, Y. (2011). Deep sparse rectifier neural networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics.

Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.

Kim, J.H. (2006). Wild bootstrapping variance ratio tests. Economics Letters 92, 38–43.

Kim, J.H. (2009). Automatic variance ratio test under conditional heteroskedasticity. Finance Research Letters 8, 179–185.

Kim, J.H., Shamsuddin, A. (2015). A closer look at return predictability of the US stock market: evidence from new panel variance ratio tests, Quantitative Finance, 15:9, 1501-1514.

Kingma, D.P., Ba, J. (2015). Adam: a Method for Stochastic Optimization. The 3rd International Conference for Learning Representations, San Diego.

LeCun, Y. (1989). Generalization and network design strategies. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto.

Ljung, G.M., Box, G.E.P (1978). On a measure of lack of fit in time series models. Biometrika 65, 297-303.

Lo, A.W., MacKinlay, A.C. (1988) Stock markets prices do not follow random walks: Evidence from a simple specification test. Review of Financial Studies, 1988, vol. 1, issue 1, 41-66

Rumelhart, D.E., Hinton, G.E., Williams R.J. (1986) Learning representations by back-propagating errors, Nature 323, 533-536.

Scott, C., Nowak, R. (2005). A Neyman-Pearson approach to statistical learning. IEEE Transactions on Information Theory, 51, 3806-3819.

Rigollet, P., Tong, X. (2011). Neyman-Pearson classification, convexity and stochastic constraints. Journal of Machine Learning Research, 12, 2831-2855.

Tong, X. (2013). A plug-in approach to Neyman-Pearson classification. Journal of Machine Learning Research, 14, 3011-3040, 2013.

Tong, X., Feng, Y., Li, J. (2018). Neyman-Pearson (NP) Classification algorithms and NP receiver operating characteristics (NP-ROC). Science Advances 4:eaao1659.

Wright, J.H. (2001). Alternative variance-ratio tests using ranks and signs. Journal of Business & Economic Statistics 16, 1-9.

Zhao, A., Feng, Y., Wang, L., Tong, X. (2016). Neyman-Pearson classification under high dimensional settings. Journal of Machine Learning Research, 17, 1-39.