Jump Detection in Financial Time Series using Machine Learning Algorithms

Jay F.K. Au Yeung,[a], Zi-kai Wei,[b], Kit Yan Chan,[c], Henry Y.K. Lau,[a], Ka-Fai Cedric Yiu,[b]

[a] *Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam, Hong Kong*
[b] *Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*
[c] *School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia*

Abstract

In this paper, we develop a new Hybrid method based on machine learning algorithms for jump detection in financial time series. Jump is an important behaviour in financial time series, since it implies a change in volatility. Ones can buy the volatility instrument if ones expect the volatility will bloom up in the future. A jump detection model attempts to detect short term market instability, since it could be jumping up or down, instead of a directional prediction. The directional prediction can be considered as a momentum or trend following, which is not the focus of this paper. A jump detection model is commonly applied in a systematic fast-moving strategy, which reallocates the assets automatically. Also, a systematic opening position protection strategy can be driven by a jump detection model. For example, for a tail risk protection strategy, a pair of long call and put option order could be placed in the same time, in order to protect the open position given a huge change in volatility.

One of the key differentiation of the proposed model with the classical methods of time-series anomaly detection is that, jump threshold parameters are not required to be predefined in our proposed model. Also the model is a combination of a Long-short term memory (LSTM) neural network model and a machine learning pattern recognition model. The LSTM model is applied for time series prediction, which predicts the next data point. The historical prediction errors sequence can be used as the information source or input of the jump detection model/module. The machine learning pattern recognition model is applied for jump detection. The combined model attempts to determine whether the current data point is a jump or not. LSTM neural network is a type of Recurrent Neural Networks (RNNs). LSTM records not only the recent market, but also the historical status. A stacked RNN is trained on a dataset which is mixed with normal and anomalous data. We compare the performance of the proposed Hybrid jump detection model and different pattern classification algorithms, such as k-nearest neighbors algorithm (KNN) identifier, Hampel identifier, and Lee Mykland test. The model is trained and tested using real financial market data, including 11 global stock market in both developed and emerging markets in US, China, Hong Kong, Taiwan, Japan, UK, German, and Israel. The experiment result shows that the proposed Hybrid jump detection model is effective to detect jumps in terms of accuracy, comparing to the other classical jump detection methods.

Keywords: Recurrent Neural Network, Anomaly Detection, Machine Learning, Long-short term memory (LSTM)

Email addresses: jayauyeung@hku.hk (), nathaniel.zk.wei@connect.polyu.hk (), kit.chan@curtin.edu.au (), yklau@hku.hk (), cedric.yiu@polyu.edu.hk ()

1. Introduction

Anomaly detection, which also called outlier detection or jump detection, attempts to identify an unexpected pattern for a given dataset [1]. Assuming the time series $t(x)$ is continuous on $[a,b]$, the spike at a finite number of points is defined as a jump in this paper. The number and the location of the jumps are unknown. It is an important statistical behaviour in the financial market, since the jump detection model can be applied on a volatility trading strategy, tail protection strategy and systematic asset allocation. In the machine learning perspective, an anomaly detection can be viewed as a pattern classification related problem. Machine learning classification models, such as naive Bayesian classifier (NB) [2], support vector machine (SVM) [3], KNN [4; 5], decision tree learning (TREE) [6], and the Hampel filter can be used to identify anomaly data. The applications of anomaly detection using pattern recognition models are included outliers detection and outlier removal, and extensions [7; 8; 9].

Jump is a common anomaly in a financial time series. There are several classical jump detection models. They are included Barndorff-Nielsen Shepherd test [10], Lee Mykland test [11], and Aït-Sahalia Jacod test [12]. Barndorff-Nielsen Shepherd test is applied on a 5-minute time intervals time series jump detection [13]. Ait-Sahalia Jacod test is applied on a high-frequency time series data of which time intervals are from 5 seconds to 30 seconds [12]. Milla Makinen [14] applied the stacked LSTM neural network for jump detection as well. However, the approach is applied on an high frequency intraday limit order book depth data , instead of a time series. Lee Mykland test is applied on a timer series data in 15-minute time interval. Our proposed model focuses on a 15-minute time interval data, instead of high frequency data. Therefore, we compare our proposed jump detection model with Hampel filter and Lee Mykland test.

Regarding the importance of jump detection, it was driven by the increase in investor risk aversion after the 2008 global financial crisis. Institutional investors or pension funds are looking for better ways to limit the risk taking. Downside risk or tail risk protection strategies benefits investors whose investment goal is capital preservation. Tail risk means that a high degree of uncertainty in stock price change lead to greater returns dispersion. Tail risk protection strategy is aimed to protect the position against extreme market movement, given the current portfolio active over-weighted or under-weighted position exposure. The idea of tail risk protection is more or less similar to buying insurance. For example, a portfolio manager may buy a credit default swap for protect the downside risk of bankruptcy, or long a pair of call and put options in the same time for protecting the active weight under extreme market volatility. A jump detection model could be applied for these strategies, in order to determine the volatility change systematically.

We propose a new Hybrid method for financial time series anomaly detection. The model is combined with the RNN model and a machine learning pattern classification model, in order to identify jumps in the time series. Long-short term memory (LSTM) neural network is a recurrent neural network (RNN), which was proposed by Hochreiter

and Schmidhuber [15]. The architecture includes a forget gates which are differentiate with traditional neural network and the architecture attempts to reduce the interference of irrelevant input and output features. This architecture enables the neural network to handle long-term memory storage, i.e. historical time series [16]. A neural network consists of a stacked recurrent hidden layers of sigmoid activation units, which are able to capture the historical pattern of both times series and multiple time intervals [16; 17]. The proposed model outperforms classical methods in terms of accuracy, when performing jump detection in real financial market data. The results show that the proposed Hybrid model is able to detect jump, and possible to help volatility trading and risk management in the financial market.

Milla Makinen [14] proposed a jump prediction model using convolutional neural network and LSTM, by inputting limit order book (LOB) data. The model proposed by Milla Makinen and our proposed Hybrid model both applied deep neural network and LSTM. However, there are a few significant differences between these two models. Firstly, the input data of Milla Makinen's model is less than 1-minute LOB data (high frequency market data), while our proposed Hybrid model inputs time series 15-minutes index price data (low frequency market data). There are a few limitations in using high frequency market data, such as the cost of data collection, the accuracy of data, and limited historical LOB data availability. Regarding the cost of data collection, LOB is required to purchase from the stock market exchange in most of cases. The cost depends on the length of look-back history, and how deep the order book (Level I or Level II) is. Also, the data within 1-minute has a limited accuracy, because the tick-by-tick market data that provided by the exchange is a "snap-shot" in one second, instead of the exact number of trade executed. Some of the institutional investors may use dark pool for their huge block trade. The dark pool is an trade execution service provided by their brokers. Such kind of huge transactions cannot be captured by the stock market exchange, because those buy/sell trades execution are internally crossed inside the dark pool, before executing in the market (the stock market exchange). Our proposed model inputs 15-minutes time series price data. The price data is the final result of all orders being executed in the order book. Also, it is not all the financial time series contains LOB data, such as an index like SPX. It allows applying our proposed model to most global financial market index. Another key benefit of using a lower frequency market data is de-noising. On the other hands, noisy small price up and down are the key components of high frequency data. In addition, low frequency market data could be easily accessed by most exchange participants or students. The cost of data acquisition is low and almost no data cleaning is required before using. Secondly, in terms of model architecture, Milla Makinen [14] proposed a CNN-LSTM-Attention network which implemented one attention layer, one convolutional layer and one LSTM layer. On the other hands, our proposed method is a stacked LSTM with three LSTM layers and combined with one additional detection layer. The detection layer is a classical machine learning pattern classification algorithms.

Regarding the classical time series model for predicting volatility, the parametric model inputs the most recent data time point features only. For example, the Autoregressive integrated moving average (ARIMA) model only addresses a 2-time-step. In practice, a systematic volatility strategy takes a basket of factors into account. Therefore, an ideal model should be possible to handle a large number of factors input. A large feature space can be inputted into the

RNN model. The proposed jump detection model in this paper takes historical time series pattern into account, which is relatively flexible than the currently used jump detection models, such as KNN, Hample identifier, and Lee Mykland test. The paper is organized as the following: Section 2 discusses the architecture of the Hybrid method. Section 3 presents the jump detection experiment results, using real global stock market data. In Section 4, a conclusion is given.

2. Hybrid machine learning model for anomaly detection

2.1. The LSTM

Similar to the typical recurrent network, each LSTM cell shares the same inputs and outputs. The key distinction of LSTM [15] is that, the gate function controls the information flow. Figure 1 illustrates the inner structure of a LSTM cell. The state unit is colored in orange, which is the most important component of the LSTM. The state unit is capable for both long and short memory. A state unit $s_i^{(t)}$ of cell i at time t has a linear self-loop weight which is controlled by a forget gate unit $f_i^{(t)}$. The forget gate helps the model to remove the irrelevant memory. A sigmoid unit attempts to ensure that the weight is in a range between 0 and 1 [18]:

$$s_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f + x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}), \tag{1}$$

where $x_j^{(t)}$ is the j^{th} element of the input vector to cell i at time t and $h_j^{(t)}$ is the j^{th} element of the hidden layer vector at time t, which are used to contain the outputs of all the LSTM cells; b_i^f , $U_{i,j}^f$, and $\mathbf{W}_{i,j}^f$, are the biases, input weights and recurrent weights for the forget gates [18; 15].

Based on the conditional self-loop weight $f_i^{(t)}$, the state unit is updated as follows:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}), \tag{2}$$

where b_i, $U_{i,j}$ and $W_{i,j}$ denote the biases, input weights and recurrent weights of the LSTM cell respectively. The external input gate unit $g_i^{(t)}$ is computed as

$$g_i^{(t)} = \sigma(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}), \tag{3}$$

where b_i^g, $U_{i,j}^g$, $W_{i,j}^g$ are the biases, input weights and recurrent weights for the input gate respectively. The input gate is used to select the relevant information and memorize such valuable information. The output $h_i^{(t)}$ of the LSTM cell can be controlled by the output gate $q_i^{(t)}$ of which a sigmoid unit is used for gating:

$$\begin{aligned} h_i^{(t)} &= tanh(s_i^{(t)}) q_i^{(t)} \\ q_i^{(t)} &= \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}) \end{aligned} \tag{4}$$

where the parameters b_i^o, $U_{i,j}^o$ and $W_{i,j}^o$ are denoted as the biases, input weights and recurrent wights for the output gate, respectively. In our proposed anomaly detection model, a stacked LSTM is a prediction unit for the next data point.

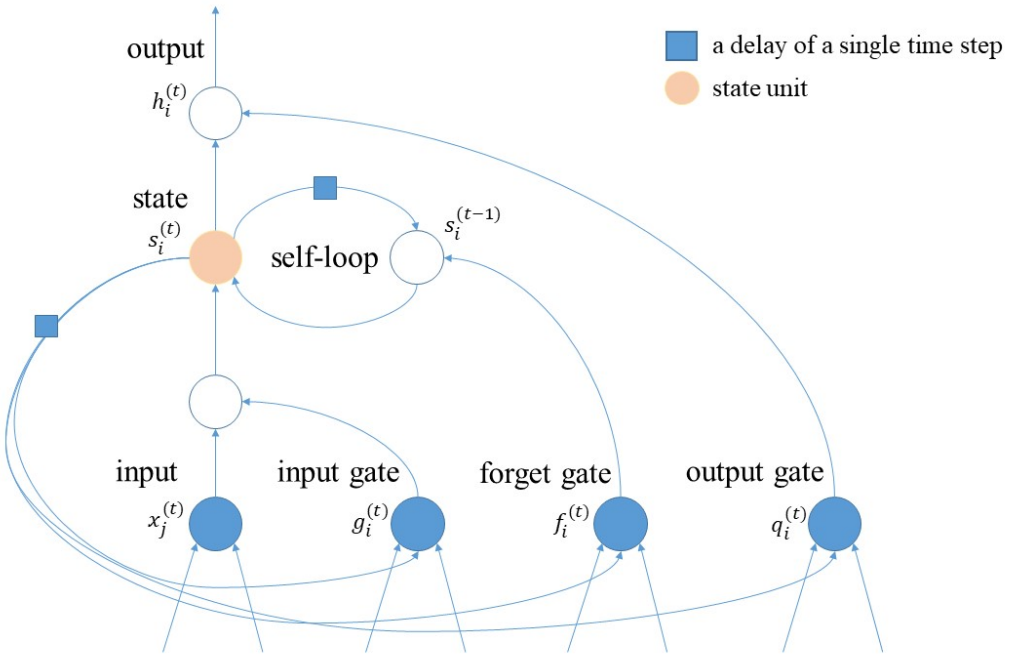


Figure 1: LSTM recurrent network cell

Figure 2 shows the architecture of the proposed stacked LSTM, which consists of one input layer, three hidden layers, and one output layer. Only one unit exits in either the input layer or the output layer. 64 LSTM units, 256 LSTM units, and 100 LSTM units exist in the three hidden layers respectively.

2.2. Hybrid Model Architecture

The flowchart of the proposed Hybrid method is illustrated in Figure 3. The Hybrid model is a combination of a stacked LSTM neural network mode, and a machine leaning pattern recognition layer which is also called a jump detection layer. The jump detection layer can be applied with either the Naive Bayesian classifier (NB), support vector machine (SVM), KNN, or decision tree learning (TREE).

2.2.1. To Label the Targets (Jumps) Systematically

The targets are those anomalous data points or jumps, which can be labelled by using the structural-break model. The structural-break model fits the whole time series, including both testing and training data sets, in order to label all the jumps. However, the structural-break model is aimed for target labelling purpose only. The model cannot be applied for jump detection in our proposed Hybrid model. Therefore, it is impossible to have a data forward looking in prediction or model testing phase.

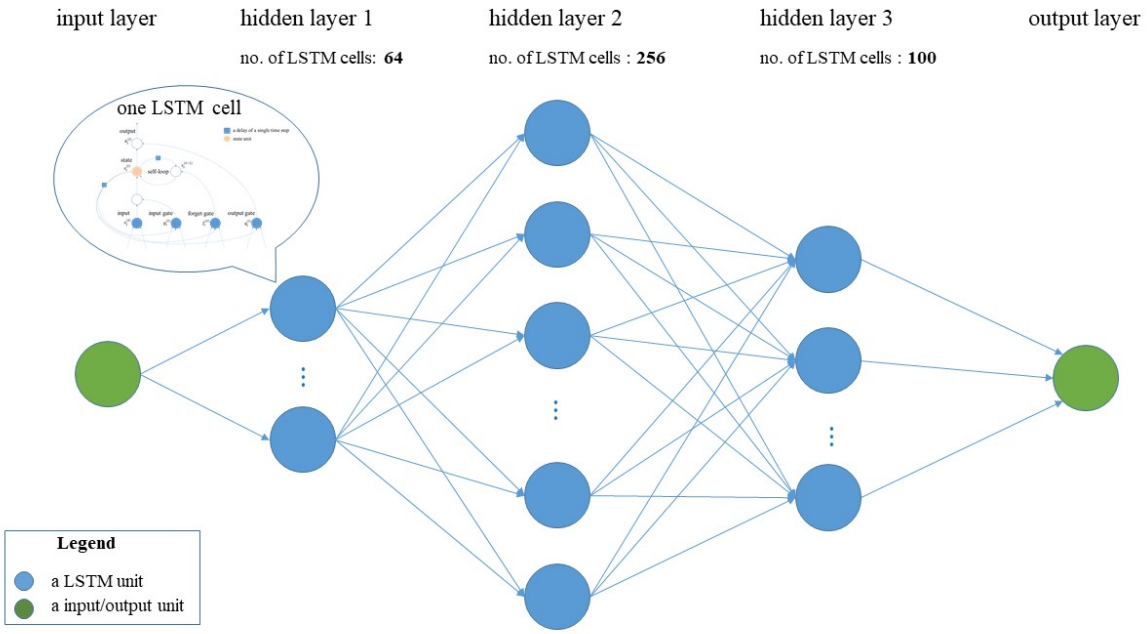


Figure 2: The architecture of a stacked LSTM

2.2.2. *Hybrid Model Training*

This section discusses how a Hybrid model is trained:

a) The first part of the Hybrid model, the LSTM neural network, attempts to forecast the data point next step ahead to the current time. A vector, $\{x_{t-d+1}, x_{t-d+2}, \ldots, x_{t-1}\}$, is feed into the input layer, and the output layer generates a future value $\hat{x}_t$, which is one time step ahead prediction of the previous time stamp. The prediction error is defined as $e_t = x_t - \hat{x}_t$. We label e_t as l_t, such that $l_t = 1$, if x_t is the anomalous data or detected jump; otherwise, $l_t = 0$. The idea is more or less similar to plotting a new data point using extrapolation, using a regression and historical time series.

The second part of the Hybrid model is a jump detection model, which map the errors $\mathbf{e} = \{e_1, \ldots, e_n\}$ to the labels $\mathbf{l} = \{l_1, \ldots, l_n\}$, where $\mathbf{l}$ is used to determine whether this is a jump or not. The jump is detected based on the historical errors sequence. Assuming the time series $t(x)$ is continuous on $[a,b]$, spike existing at a finite number of points is defined as a jump. The number and the location of the jumps are unknown. As the continuity of a time series is corrupted by market noise, the prediction error which has a huge deviation from its historical time series pattern is likely to be a potential jump point. The detection layer in Figure 3 illustrates the mechanism of the mapping.

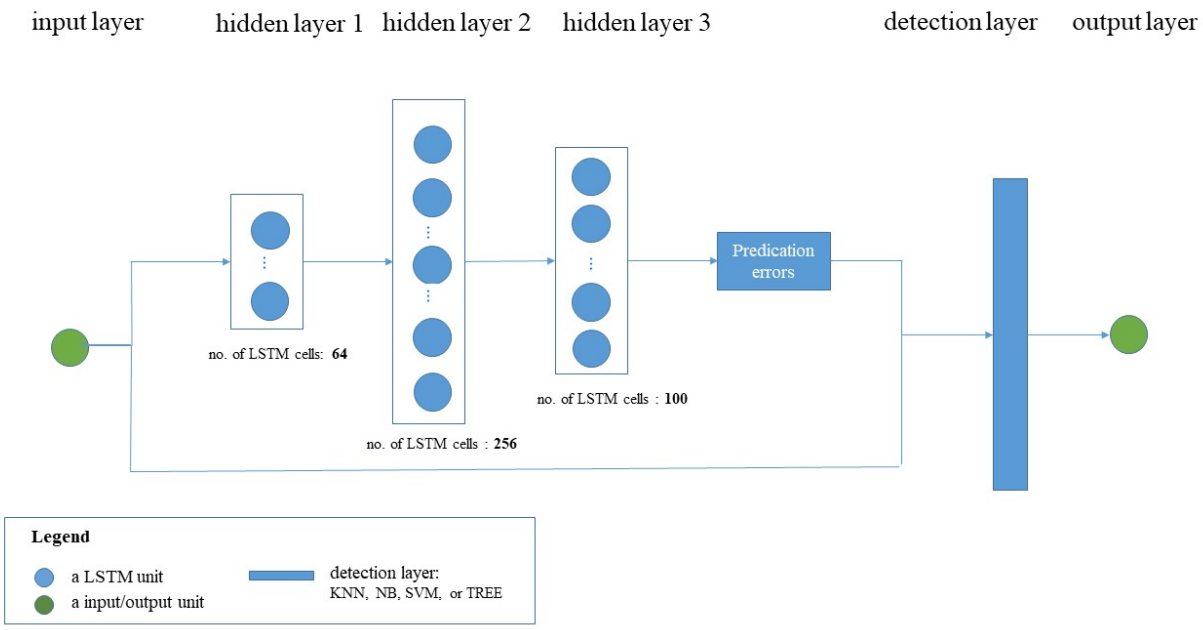


Figure 3: the architecture of the Hybrid method

b) The LSTM networks includes three hidden layer. For each hidden layer, we stack layers of LSTM neural network.

The dimensions of the three hidden layers are 64, 256 and 100 respectively. All units in the lower layers are fully connected to the units in the higher layer. In this paper, we applied RBF kernel for the RSVM model, which is a default setting for a typical SVM. 64 (i.e. 2^6) units and 256 (i.e. 2^8) units are used in the first and second layers, as they are the commonly used setting [11].The last layer has 100 units to mimic the setting of choice of $K = 100$ [11]. The data set S_1 is applied to train the stacked LSTM neural network part.

The second part of the Hybrid model, jump detection model, is developed by the following steps:

a) When the trained LSTM neural networks and the data set S_3 with anomalous and normal data are given, the errors vector $\mathbf{e}$ and specify the labels vector $\mathbf{l}$ can be computed. The errors vector, $\mathbf{e}$, is the difference between the real $\mathbf{x}$ and $\hat{\mathbf{x}}$, where $\mathbf{x} = \{x_{t_1}, \ldots, x_{t_2}\}$ and $\hat{\mathbf{x}} = \{\hat{x}_{t_1}, \ldots, \hat{x}_{t_2}\}$. Corresponding to the error vector $\mathbf{e}$, which is illustrated as prediction errors in Figure 3, a label vector can be defined as $\mathbf{l} = \{l_{t_1}, \ldots, l_{t_2}\}$, where $l_i \in \{0, 1\}$, $i = t_1, \ldots, t_2$. If the i^{th} item is anomalous, $l_i = 1$; otherwise, $l_i = 0$.

b) The classification model is described as

$$\hat{z}_t = l_t = \begin{cases} 1, \ anomalous \\ 0, \ normal \end{cases},$$

where $\hat{z}_t$ denotes a classification result, which is generated by the jump detection model at the time index t; when $\hat{z}_t = 1$, the anomalous data is classified as true, and when $\hat{z}_t = 0$, the anomalous data is classified as false. Here, z is a specific classification model. The detection layer in Fig 3 can be embedded by either the Naive Bayesian classifier (NB), SVM, KNN, and decision tree learning (TREE):

– RKNN stands for the Hybrid model with a jump detection layer, using a KNN classifier. KNN is an unsupervised pattern recognition algorithm, which involved two major steps: a) determining the nearest neighbors, and b) making classification of those neighbors. Given a dataset $\mathbf{X}$ in the training phase, a set of features F and any numeric features can be normalized to a range between 0 and 1. Each training example is labeled with a class label $y_j \in Y$. Given an unknown example $\mathbf{q}$, the distance between $\mathbf{q}$ and $\mathbf{x}_i$ [19] is given as

$$d(\mathbf{q}, \mathbf{x}_i) = \sum_{f \in F} \delta(\mathbf{q}_f, \mathbf{x}_{if}).$$

The continuous values are converted to a discrete feature, in order to reduce the complexity of the model, i.e

$$\delta(\mathbf{q}_f, \mathbf{x}_{if}) = \begin{cases} 1 & , \ f \ discrete \ and \ \mathbf{q}_f \neq \mathbf{x}_{if} \\ 0 & , \ f \ discrete \ and \ \mathbf{q}_f = \mathbf{x}_{if} \\ |\mathbf{q}_f - \mathbf{x}_{if}|, & f \ continuous \end{cases}.$$

The distance weighted voting is the neighbors which vote on the class of the query case. The votes are weighted by the inverse of their distance to the query:

$$Vote(y_i) = \sum_{c=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{X_c})^n} I(y_j, y_c),$$

where

$$I(y_j, y_c) = \begin{cases} 1, & y_j = y_c, \\ 0, & y_j \neq y_c, \end{cases},$$

and n is chosen to be 1 at usual. The influence of more distant neighbors can be reduced when $n > 1$ [19].

– RNB stands for the Hybrid model with a jump detection layer, using a naive Bayesian classifier. For the naive Bayesian classifier, a training samples are assumed with k classes, $C_1, C_2, ..., C_k$. Given a set of samples, T, along with their class labels in the training phase, each sample is represented by an n-dimensional vector, $\mathbf{X} = x_1, x_2, ..., x_n$, and the i-th element in X, X_i represents the i-th attribute, A_i, $A_1, A_2, ..., A_n$, respectively. $\mathbf{X}$ is assigned to the class C_i, when

$$P(C_i|X) > P(C_j|X), \text{ for } 1 \leq j \leq m, \ j \neq i.$$

– RSVM stands for the Hybrid model with a jump detection layer, using a SVM layer. The N data points $\{y_k, x_k\}_{k=1}^{N}$ in the training phase, where $x_k \in \mathbb{R}^n$ is the k^{th} input pattern and $y_k \in R$ is the kth output pattern. The SVM classifier can be developed based on the N data points as[20]:

$$y(x) = \text{sign}[\sum_{k=1}^{N} \alpha_k y_k \psi(x, x_k) + b],$$

where α_k are positive real constants and b is a real constant. $\psi(\cdot,\cdot)$ is the kernel function of SVM: $\psi(x, x_k) = x_k^T x$ (linear SVM); $\psi(x, x_k) = (x_k^T + 1)^d$ (polynomial SVM of degree d); $\psi(x, x_k) = \exp\{-||x - x_k||_2^2/\sigma^2\}$ (Gaussian or Radial Basis Function (RBF)); $\psi(x, x_k) = tanh[\kappa x_k^T x + \theta]$ [21; 20]. In this paper, we use RBF kernel for the SVM, since it is a default setting for most of the SVM classifier.

– RTREE stands for the Hybrid model with a jump detection layer, using a decision tree [6].

2.2.3. Hybrid Model Testing

a) After the LSTM and jump detection model are combined , the Hybrid model is successfully constructed. We may start testing the model using real financial time series data. In particular, the 15-minute interval global financial market indices.

b) The trained Hybrid method is validated using the testing dataset. A vector, $\{x_{t-d+1}, x_{t-d+2}, \ldots, x_{t-1}\}$ is inputted to the model for jump detection. The LSTM part of the Hybrid model detects the data point, x_t. The second part of the Hybrid method, pattern recognition model, helps determine whether x_t it is a jump. A confusion matrix is generated as the prediction result.

2.3. Comparison

We compare our proposed Hybrid model with the classical anomaly detection models, i.e. KNN, Hampel filter, and Lee Mykland test:

2.3.1. KNN identifier

KNN can be applied for jump detection given the following steps:

a) The KNN can be trained by the returns series of an index which are labeled with ”anomaly” or ”normal”;

b) The jump classification result of the trained KNN model can be evaluated by a confusion matrix.

2.3.2. Hampel identifier

Each market index is transformed to the returns series and Hampel filter is used to identify which returns are anomaly returns, by using the returns series input.

2.3.3. Lee Mykland test

Lee and Mykland developed a jump test at each individual observation [11]. A ratio of the return at each data point is included to as a measure of "instantaneous volatility" over a period. This data point is identified as a jump if this ratio exceed the threshold, $d = 4.60001$ [11]. This ratio, which is used to test a jump at time t, which is defined as

$$J(i) = \frac{r_i}{\sigma(t_i)}$$

where

$$r_i = log(P_i) - log(P_{i-1}),$$

and

$$\sigma(t_i) = \frac{1}{K-2} \sum_{j=i-K+2}^{i-1} |r_j||r_{j-1}|.$$

K is the window size between $252 \times n$ and $\sqrt{252 \times n}$, where n is the number of data points in a day. For a 15-minute sampling intervals of a market index, 27 data points are in day ($n = 27$) and K is an integer between $[83, 6804]$. Based on the suggestion in [11], K is chosen as 100 in our case.

Let

$$\varepsilon_i = \frac{|L(i)| - C_n}{S_n},$$

where

$$C_n = \frac{(2logn)^{1/2}}{c} - \frac{log(\pi) + log(logn)}{2c(2logn)^{1/2}},$$

$$S_n = \frac{1}{c(2logn)^{1/2}},$$

and

$$c = \sqrt{\frac{2}{\pi}},$$

If ε_i is larger than threshold d, then the observation i is identified as a jump or anomaly.

3. Empirical results

We test the proposed Hybrid model using both simulated and real stock market data:

a) Simulated data is generated based on the real world data where the artificial jumps are added on the original data.

b) Real market data is a 15-minutes time interval time series data. The real market data are collected from the major developed market and emerging market stock market:

   i) US: Dow Jones Industrial Average (DJI), S&P 500 Index (SPX);

ii) China/Hong Kong/Taiwan: Shanghai Stock Exchange Composite Index (SHCOMP), Hang Seng Index (HSI), Taiwan Stock Exchange Weighted Index (TWSE);

iii) Japan: Nikkei 225 (NKY);

iv) UK: FTSE 100 Index (UKX);

v) Germany: DAX Performance Index (DAX);

vi) Israel: Tel Aviv Stock Exchange 35 Index (TA-35);

vii) Global: The MSCI World Index (MXWO), S&P Global 100 Index (OOI).

The jump detection performance is evaluated by using the Sensitivity, Specificity, Precision, Negative Predictive Value, False Positive Rate, False Discovery Rate, False Negative Rate, Accuracy, F1 Score in Table 1. These metrics can be extracted from the confusion matrix.

3.1. Datasets

Each dataset has been separated into 4 parts: S1, data for training the stacked LSTM neural network; S2, data for testing the stacked LSTM neural network; S3, data for training jump detection model; and S4, data for testing jump detection model.

3.1.1. Simulated data

The 15-minutes time interval simulated index data and artificial jumps are applied in this simulation. For the simulated index, the data sequences are divided into four data sets: S_1 consists of 1/3 data sequences for training a stacked LSTM neural network, S_2 consists of 1/3 data sequences of prediction errors, S_3 consists of 1/6 data sequences for the jump detection model training, and S_4 consists of the left 1/6 data sequences for testing the trained jump detection model. S_1 and S_2 only include normal data; S_3 and S_4 mixed with abnormal data. In S_3 and S_4, the abnormal data point is generated by combining the normal data point with jumps. Those jumps are a random number, which is generated at a random time and a random magnitude from a t-distribution. Hence, the abnormal data point, a_t, is given by, $a_t = x_t + 10 \times d_t$, where d_t is a t-distribution with the parameter value 2000 and a_t is the abnormal data.

3.1.2. Real market data

The real market data is in a 15-minutes time interval and collected from S&P TSX Composite (SPX), 30 Industrials (DJI), Shanghai Composite (SHCOMP), HANG SENG INDEX (HSI), Taiwan Weighted (TWSE), Nikkei 225 (N225), FTSE 100 Index (UKX), GDAXI (DAX), and TA-35, MSCI World (MXWO), and S&P Global 100 (OOI). The realized jumps in the collected data are labelled by the structural break model [22] and [23].

Table 1: The experiment result of the simulations

Measure	LAD	RKNN	RNB	RSVM	RTREE	KNN	Hampel	Lee
Sensitivity	0.67	0.08	0.19	0.00	0.11	0.00	0.03	0.00
Specificity	0.23	0.93	0.84	1.00	0.95	0.98	0.97	1.00
Precision	0.10	0.13	0.13	-	0.22	0.00	0.12	-
Negative Predictive Value	0.84	0.89	0.89	0.89	0.89	0.88	0.89	0.89
False Positive Rate	0.77	0.07	0.16	0.00	0.05	0.02	0.03	0.00
False Discovery Rate	0.90	0.87	0.87	-	0.78	1.00	0.88	-
False Negative Rate	0.33	0.92	0.81	1.00	0.89	1.00	0.97	1.00
Accuracy	0.25	0.83	0.76	0.89	0.85	0.87	0.86	0.89
F_1 Score	0.17	0.10	0.16	0.00	0.15	0.00	0.05	0.00

3.2. Results

3.2.1. Simulated cases

The size of the simulated data is 3600. The data sets are split as the following: S_1 includes the 1st data point to the 1200th data point. S_2 includes the 1201st to the 2400th. S_3 includes the 2401st to the 3100th. S_4 includes the 3101st to the 3600th. Table 1 summarizes the prediction results, including a jump detection method based on LSTM neural network only (LAD), the proposed Hybrid methods using a stacked LSTM neural network which combines different machine learning jump classification algorithms (RKNN, RNB, RSVM, and RTREE), and three classical jump detection models (KNN, Hampel identifier, and Lee Mykland test). Table 1 shows RSVM, KNN, and Lee are not able to detect the jumps in simulations, of which the sensitivities are all equal to zero. However, LAD, RKNN, RNB, RTREE, and Hampel are possible to make jump detection in simulations.

3.2.2. Real cases

The jump detection application for volatility index VIX, is shown in Figure 4 and 5. In the training phase, pseudo jumps characterized by Lee Mykland test is used to initialize the RKNN and RTREE models. In the testing phase, the RKNN and RTREE achieved similar jump detection results. The red squares are the jumps which are detected by the models. Figures 4 and 5 show that the volatility of the VIX continuous to increase after a jump is detected by the proposed Hybrid model. The jump detection for different markets are shown in Figure 6, 7, and 8, and the results are summarized in Table 3. "1" indicates that the method is able to detect the jumps; "0" indicates that the method is not able to detect jumps even exist. Table3 shows that RSVM is unable to detect jumps in any case. Then, we rank the accuracy rates obtained by the rest of the six methods. The accuracy ranks are given in descending order, i.e. Hampel, RTREE, RKNN, KNN, RNB, and Lee Mykland test. However, the accuracy reflects the capability of correctly detecting both the normal data and that of the abnormal data. It is reasonable to remove the data points which have been wrongly detected. The effort of comparison can be reduced by subtracting those methods which failed to

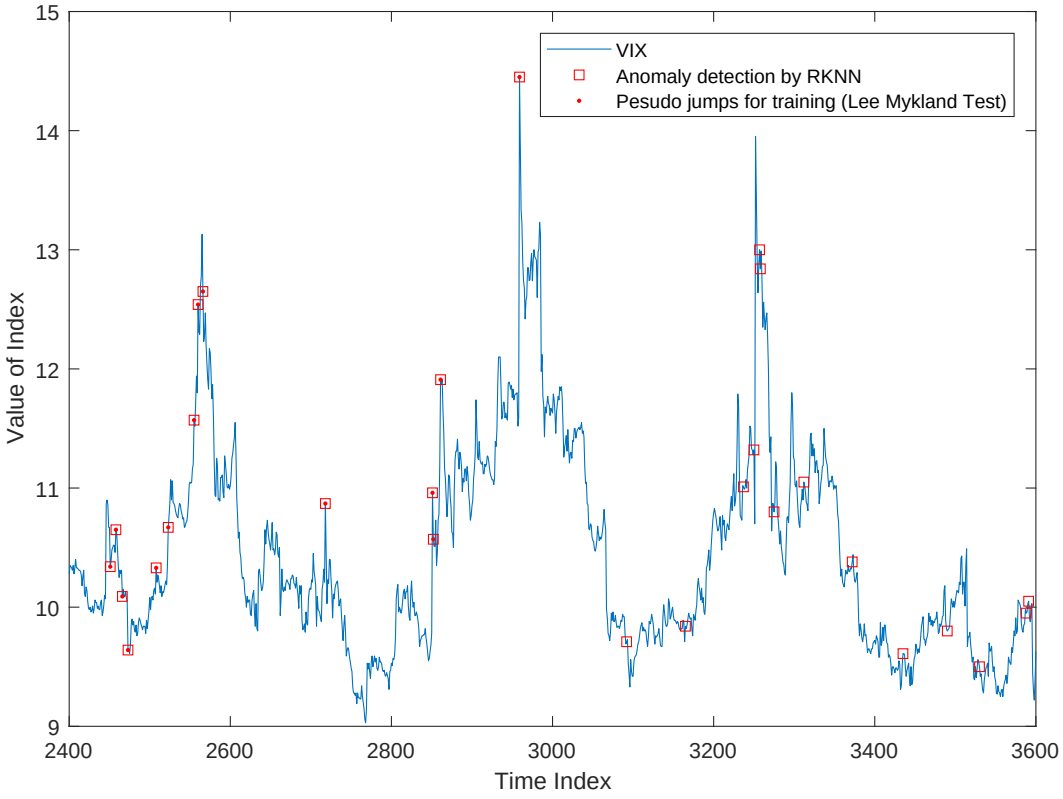


Figure 4: Detecting the jumps on VIX using RKNN

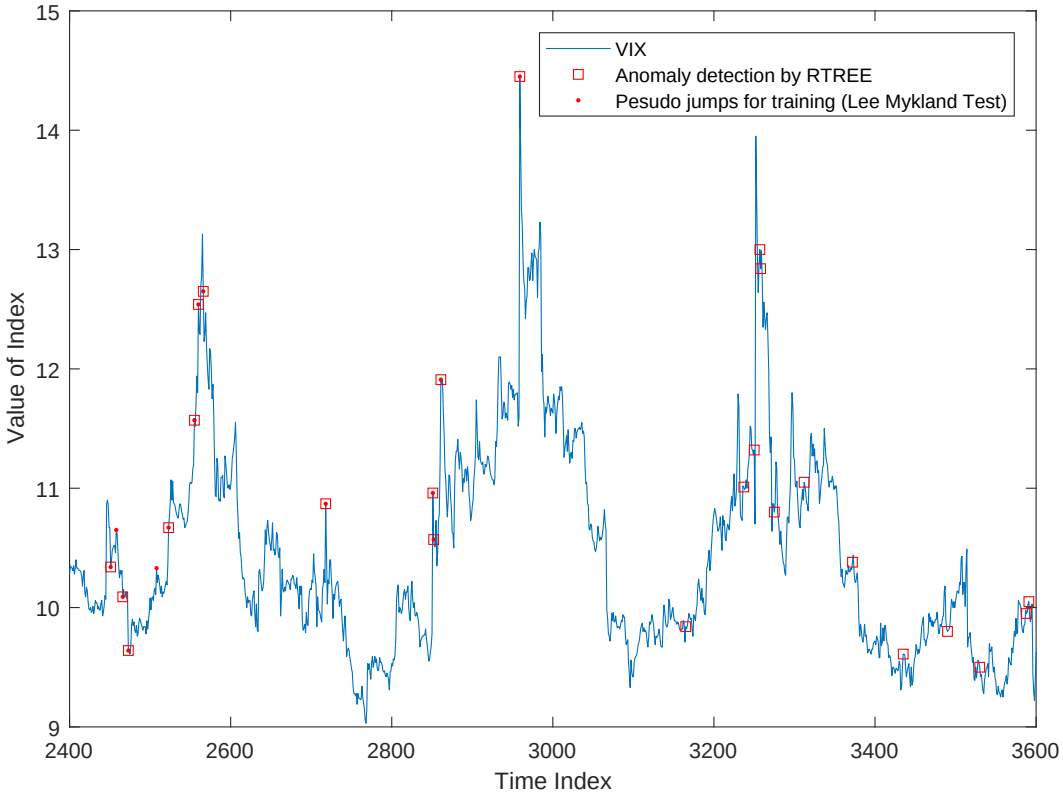


Figure 5: Detecting the jumps on VIX using RTREE

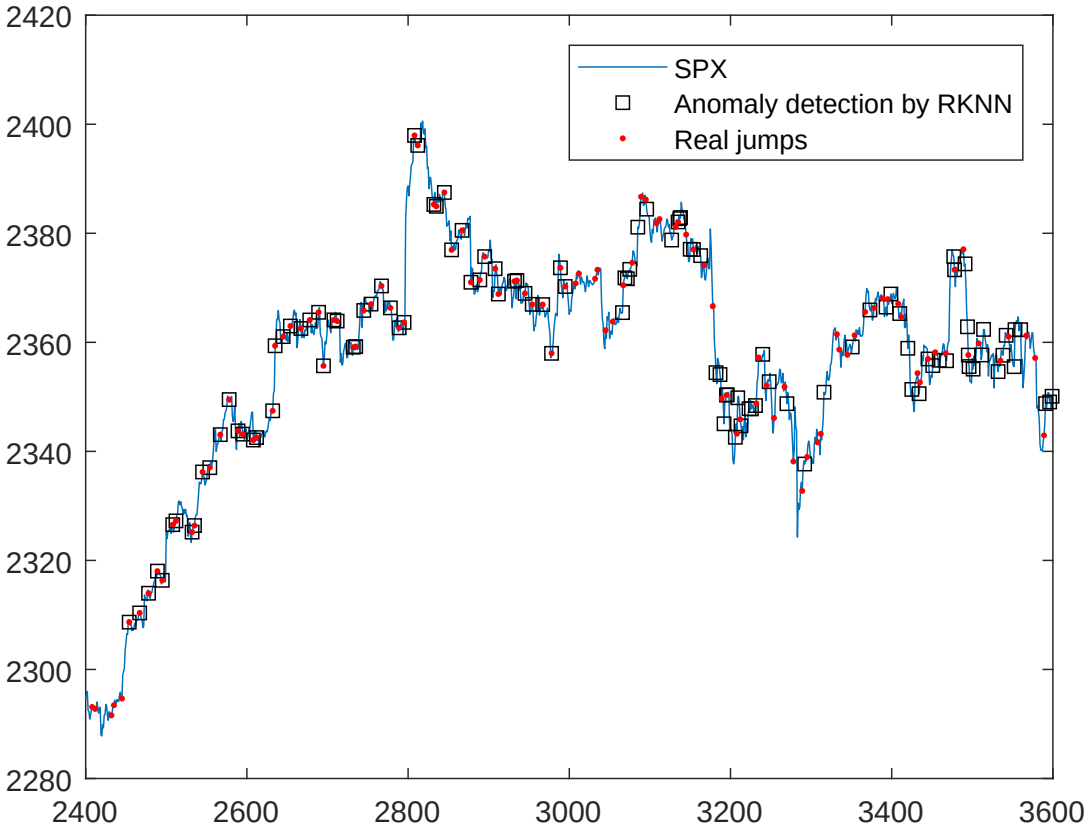
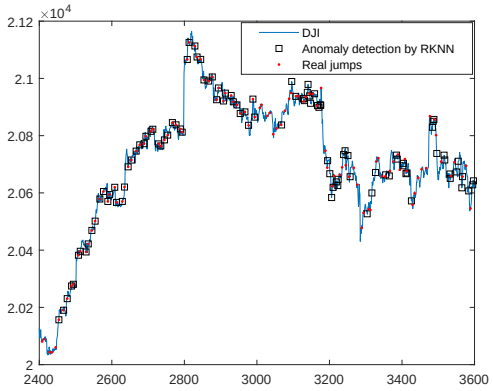


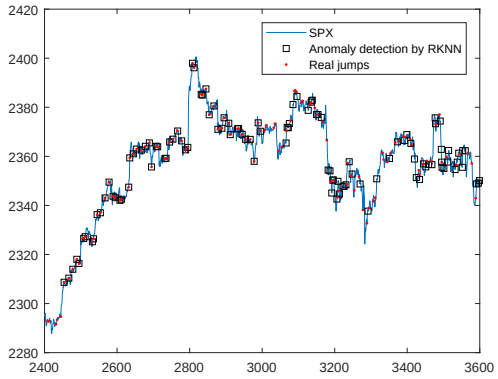
Figure 6: Detecting the jumps on SPX using RKNN

detect jumps on some indices.

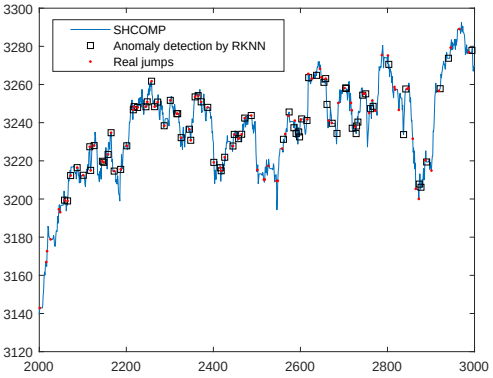
Table 3 shows that only RKNN and Lee Mykland test are possible to detect jumps among all the indices. Other jump detection methods cannot fully detect jumps in all markets. Thus, we only focus the detection results of RKNN and Lee Mykland test. The jump detection results of RKNN and Lee Mykland test are shown in Table 6 and 5 respectively. Refer to Table 4, RKNN achieves a higher accuracy than Lee Mykland test. Although Lee Mykland test can achieve a high True Positive Rate (TPR) mean value (69%), but the True Negative Rate (TNR) mean value is low. This result indicates that Lee Mykland test is not able to to detect jumps, whenever the data is abnormal or not. Since Lee Mykland can only achieve 34% Accuracy (ACC), 69% TPR, and 31% TNR, we may conclude that Lee Mykland test is not able to detect the jumps for real market data. However, RKNN can only achieve 7% TPR, which indicates that poor detection are generated when jumps exist. However, RKNN is able to achieve a high TNR (91%) and ACC (83%). These results indicate that RKNN outperforms Mykland. Thus, Lee Mykland test like a gambler who bets



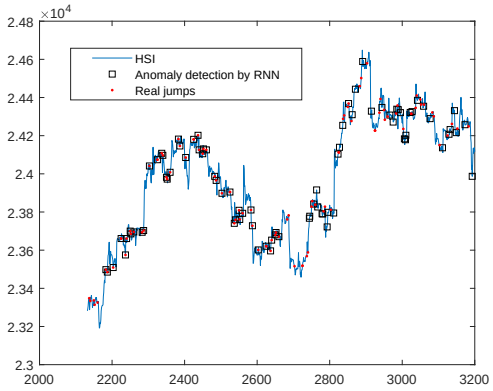
(a) DJI



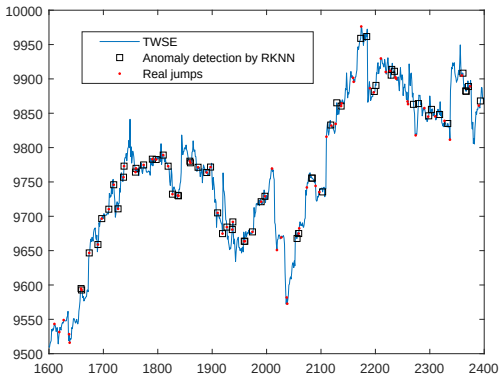
(b) SPX



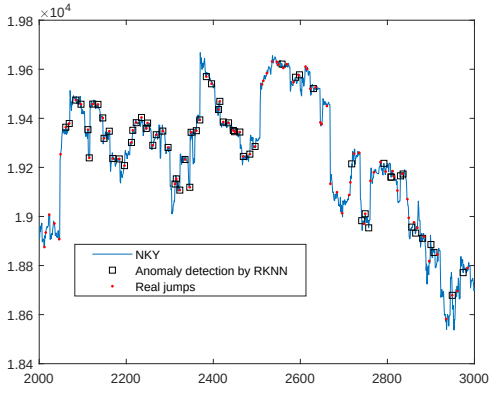
(c) SHCOMP



(d) HSI

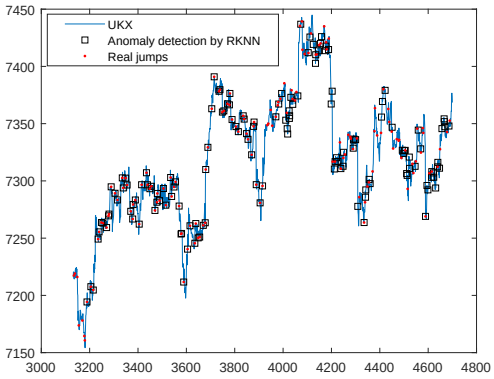


(e) TWSE

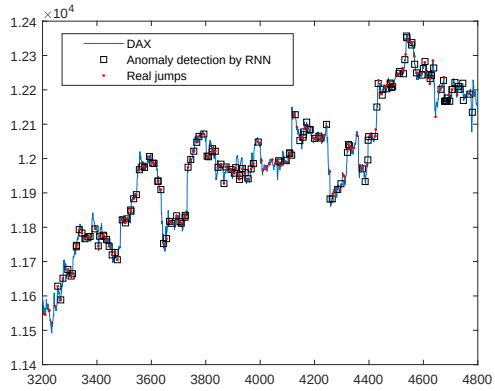


(f) NKY

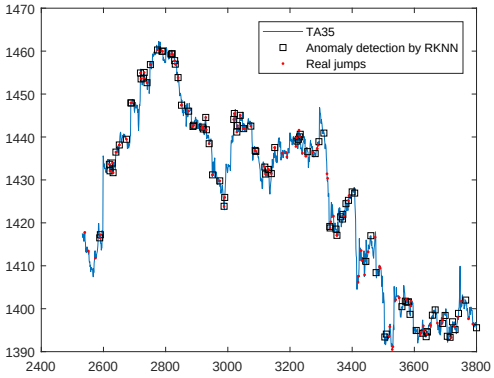
Figure 7: Jumps detection by RKNN on the other indices: Part I



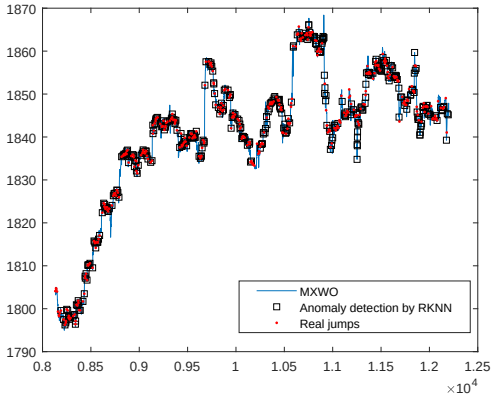
(a) UKX



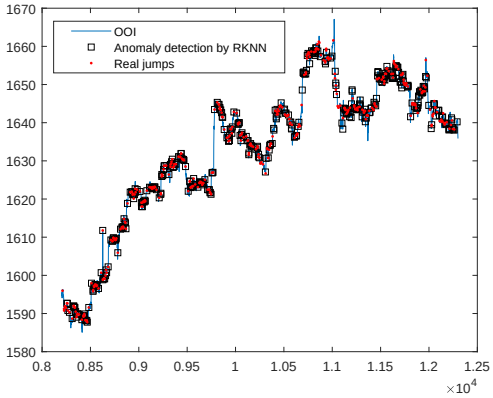
(b) DAX



(c) TA-35



(d) MXWO



(e) OOI

Figure 8: Jumps detection by RKNN on the other indices: Part II

Table 2: The experiment results using real market data implement Hybrid method

mean	LAD	RKNN	RNB	RSVM	RECOC	RTREE	Hampel	KNN	Lee
TPR	0.00	0.07	0.43	0.00	0.00	0.06	0.02	0.18	0.69
TNR	1.00	0.91	0.60	1.00	1.00	0.93	0.97	0.82	0.31
PPV	0.02	0.08	0.11	NaN	NaN	0.08	0.09	0.07	0.09
NPV	0.91	0.91	0.92	0.91	0.91	0.91	0.91	0.91	0.91
FPR	0.00	0.09	0.40	0.00	0.00	0.07	0.03	0.18	0.69
FDR	0.98	0.92	0.89	NaN	NaN	0.92	0.91	0.93	0.91
FNR	1.00	0.93	0.57	1.00	1.00	0.94	0.98	0.82	0.31
ACC	0.91	0.83	0.58	0.91	0.91	0.85	0.89	0.76	0.34
F1	0.00	0.07	0.10	0.00	0.00	0.07	0.04	0.08	0.16

Table 3: The summary of all detection methods

	DJI	SHC	HSI	NKY	TWS	UKX	DAX	TA35	MXW	OOI	SPX	F.
LAD	0	0	0	0	0	0	0	0	0	1	0	10
RKNN	1	1	1	1	1	1	1	1	1	1	1	-
RNB	1	0	1	0	1	1	1	1	1	0	0	4
RSVM	0	0	0	0	0	0	0	0	0	0	0	11
RTREE	1	1	1	1	0	1	1	1	1	1	1	1
Hample	0	0	0	1	1	1	1	1	1	1	1	3
KNN	1	1	1	1	0	1	0	1	1	1	1	2
Lee	1	1	1	1	1	1	1	1	1	1	1	-

'jumps'; RKNN is possible to distinguish jump and not-a-jump cases.

The empirical results show that the Hybrid model which combining the stacked LSTM neural network and KNN jump classifier (or RKNN) is possible to detect jumps in the real market data, comparing with other three classical outliers detection models, i.e. KNN, Hampel identifier and Lee Mykland test. After testing the model using multiples real market data, only the proposed RKNN and Lee Mykland test achieve reasonable results. RKNN significantly outperforms other existing jump detection models. The key reason is that, the RKNN model differentiates the Mykland model by combining a LSTM neural network, which memorizes the historical time series pattern.

However, Lee Mykland test is involved with the control variable as the rolling window length of instantaneous

Table 4: The comparison between RKNN and Lee Mykland test

	TPR	TNR	PPV	NPV	FPR	FDR	FNR	ACC	F1
RKNN	0.07	0.91	0.08	0.91	0.09	0.92	0.93	0.83	0.07
Lee	0.69	0.31	0.09	0.91	0.69	0.91	0.31	0.34	0.16

Table 5: The jump detection result using real market data and implement Lee Mykland test

	DJI	SHC	HSI	NKY	TWS	UKX	DAX	TA35	MXW	OOI	SPX	mean	std
TPR	0.71	0.49	0.58	0.69	0.61	0.71	0.78	0.63	0.78	0.79	0.82	0.69	0.10
TNR	0.26	0.44	0.40	0.45	0.35	0.24	0.26	0.30	0.20	0.20	0.28	0.31	0.09
PPV	0.09	0.08	0.08	0.11	0.08	0.09	0.09	0.08	0.10	0.10	0.11	0.09	0.01
NPV	0.90	0.90	0.91	0.94	0.90	0.89	0.92	0.89	0.89	0.90	0.94	0.91	0.02
FPR	0.74	0.56	0.60	0.55	0.65	0.76	0.74	0.70	0.80	0.80	0.72	0.69	0.09
FDR	0.91	0.92	0.92	0.89	0.92	0.91	0.91	0.92	0.90	0.90	0.90	0.91	0.01
FNR	0.29	0.51	0.43	0.31	0.39	0.29	0.22	0.37	0.22	0.21	0.18	0.31	0.10
ACC	0.31	0.44	0.42	0.48	0.38	0.29	0.31	0.33	0.26	0.26	0.33	0.34	0.07
F1	0.16	0.13	0.14	0.19	0.15	0.16	0.17	0.14	0.17	0.17	0.19	0.16	0.02

Table 6: The jump detection result using real market data and implement RKNN

	DJI	SHC	HSI	NKY	TWS	UKX	DAX	TA35	MXW	OOI	SPX	mean	std
TPR	0.04	0.03	0.03	0.05	0.06	0.14	0.07	0.08	0.10	0.09	0.12	0.07	0.04
TNR	0.90	0.92	0.91	0.96	0.92	0.90	0.90	0.92	0.89	0.89	0.89	0.91	0.02
PPV	0.04	0.03	0.03	0.10	0.08	0.14	0.07	0.09	0.09	0.09	0.10	0.08	0.03
NPV	0.90	0.91	0.91	0.91	0.91	0.91	0.91	0.91	0.90	0.90	0.91	0.91	0.00
FPR	0.10	0.08	0.09	0.04	0.08	0.10	0.10	0.08	0.11	0.11	0.11	0.09	0.02
FDR	0.96	0.97	0.97	0.90	0.92	0.86	0.93	0.91	0.91	0.91	0.90	0.92	0.03
FNR	0.96	0.97	0.98	0.95	0.94	0.86	0.93	0.92	0.90	0.91	0.88	0.93	0.04
ACC	0.82	0.84	0.84	0.88	0.85	0.83	0.83	0.85	0.81	0.82	0.82	0.83	0.02
F1	0.04	0.03	0.03	0.07	0.07	0.14	0.07	0.08	0.09	0.09	0.11	0.07	0.03

volatility arranged from 83 to 6804. Therefore, the optimal value of the control variable in the Lee Mykland test is difficult to be determined. For the rolling window length, RKNN is able to avoid this problem.

4. Conclusion

In this paper, we have proposed a Hybrid method to detect jumps in financial time series. The clustering effect of jumps is able to detect the future market change, which can be used for risk management and volatility trading. The proposed method is a combination of a machine learning classification model and an RNN which uses LSTM for forecasting errors. We have tested the method using both the simulated data and the real market indices. Experimental results have shown that the proposed Hybrid method outperforms other commonly used methods for jump detection. The proposed Hybrid method is capable to achieve a higher true positive rate and more accurate detection rate.

However, there are a few limitations of the proposed model. Firstly, computational complexity of developing a neural network is high, and classical methods are simple. Hence the computational time used by the classical method is much less than that used by the proposed method. A model is required to be trained for each financial time series using historical data, however, it is only required to input a jump threshold or predefined parameters for using classical methods. A trade-off between the detection performance and computational time is required to be considered. Secondly, bias exists in the data, especially for the model which is trained using real historical financial time series data. For example, a black swan may not be happened before. Using a massive time series simulation may overcome this problem, while the cost of simulation, such as computation time and the variations, are required to take into account. It will be one of the future work for this paper. Thirdly, the jump detection model is suitable for tail risk protection trading strategy, instead of a momentum strategy. Regarding the tail risk protection strategy, it is going to limit the risk exposure to the portfolio, targeted investors with high risk aversion and target capital preservation, i.e. retirement fund. If the jump detection model is applied on the momentum trading strategy, the returns would underperform the benchmark. However, an adaptive model could be adopted, i.e using tail protection strategy during the volatility is increasing, and momentum strategy when the volatility is low. It can be achieved by generating the volatility prediction value in every time point, using the first part of the model, the LSTM volatility prediction model.

In the future, we will develop an effective optimization method to determine the optimal RNN parameters and configurations, such as the optimal numbers of units in each layer and the window size of the input index. Models with better generalization capabilities are expected to be developed.

CONFLICT OF INTEREST: AUTHOR, JAY F.K. AU YEUNG, DECLARES THAT HE HAS NO CONFLICT OF INTEREST; AUTHOR, ZI-KAI WEI, DECLARES THAT HE HAS NO CONFLICT OF INTEREST; AUTHOR, KIT YAN CHAN, DECLARES THAT HE HAS NO CONFLICT OF INTEREST; AUTHOR, HENRY Y.K. LAU, DECLARES THAT HE HAS NO CONFLICT OF INTEREST; AUTHOR, KA-FAI CEDRIC YIU, DECLARES THAT HE HAS NO CONFLICT OF INTEREST;.

ETHICAL APPROVAL: THIS ARTICLE DOES NOT CONTAIN ANY STUDIES WITH HUMAN PARTICIPANTS PERFORMED BY ANY OF THE AUTHORS.

FUNDING: THIS STUDY IS NOT FUNDED BY ANY ORGANIZATIONS.

References

[1] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM computing surveys (CSUR) 41 (3) (2009) 15.

[2] Q. Wang, G. M. Garrity, J. M. Tiedje, J. R. Cole, Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy, Applied and environmental microbiology 73 (16) (2007) 5261–5267.

[3] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural processing letters 9 (3) (1999) 293–300.

[4] E. M. Knorr, R. T. Ng, V. Tucakov, Distance-based outliers: algorithms and applications, The International Journal on Very Large Data Bases 8 (3-4) (2000) 237–253.

[5] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: ACM Sigmod Record, Vol. 29, 2000, pp. 427–438.

[6] Y. Freund, L. Mason, The alternating decision tree learning algorithm, in: icml, Vol. 99, 1999, pp. 124–133.

[7] R. K. Pearson, Outliers in process modeling and identification, IEEE Transactions on control systems technology 10 (1) (2002) 55–63.

[8] R. K. Pearson, Y. Neuvo, J. Astola, M. Gabbouj, The class of generalized hampel filters, in: Signal Processing Conference (EUSIPCO), 2015 23rd European, IEEE, 2015, pp. 2501–2505.

[9] Q. Chen, K. Chetty, K. Woodbridge, B. Tan, Signs of life detection using wireless passive radar, in: Radar Conference (RadarConf), 2016 IEEE, IEEE, 2016, pp. 1–5.

[10] O. Barndorff-Nielsen, N. Shephard, Variation, jumps, market frictions and high frequency data in financial econometrics, Economics Papers 2005-W16, Economics Group, Nuffield College, University of Oxford (2005).
URL `https://EconPapers.repec.org/RePEc:nuf:econwp:0516`

[11] S. S. Lee, P. A. Mykland, Jumps in financial markets: A new nonparametric test and jump dynamics, Review of Financial studies 21 (6) (2008) 2535–2563.

[12] Y. Aït-Sahalia, J. Jacod, Testing for jumps in a discretely observed process, The Annals of Statistics 37 (1) (2009) 184–222.
URL http://www.jstor.org/stable/25464746

[13] G. Tauchen, H. Zhou, Realized jumps on financial markets and predicting credit spreads, Journal of Econometrics 160 (1) (2011) 102–118.

[14] M. Makinen, J. Kanniainen, M. Gabbouj, A. Iosifidis, Forecasting of jump arrivals in stock prices: New attention-based network architecture using limit order book data, arXiv preprint arXiv:1810.10845.

[15] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[16] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: Proceedings, Presses universitaires de Louvain, 2015, p. 89.

[17] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Advances in neural information processing systems, 2013, pp. 190–198.

[18] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[19] P. Cunningham, S. J. Delany, k-nearest neighbour classifiers, Multiple Classifier Systems 34 (2007) 1–17.

[20] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, Journal of machine learning research 2 (Nov) (2001) 45–66.

[21] C. Cortes, V. Vapnik, Support vector networks, Machine learning 20 (3) (1995) 273–297.

[22] O. T. Choi, In depth analysis of the dually listed companies in hong kong and china stock markets prior and posterior to the global financial turmoil, International Journal of Economics and Finance 5 (10) (2013) 100–110.

[23] P. Perron, T. Yabu, Testing for shifts in trend with an integrated or stationary noise component, Journal of Business & Economic Statistics 27 (3) (2009) 369–396.