Temporal Bag-of-Features Learning for Predicting Mid Price Movements using High Frequency Limit Order Book Data

Nikolaos Passalis*, Anastasios Tefas*, *Member, IEEE*, Juho Kanniainen†, Moncef Gabbouj‡, *Fellow, IEEE*, and Alexandros Iosifidis§, *Senior Member, IEEE*
*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
†Laboratory of Industrial and Information Management, Tampere University of Technology, Tampere, Finland
‡Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland
§Department of Engineering, Electrical and Computer Engineering, Aarhus University, Denmark
passalis@csd.auth.gr, tefas@aiia.csd.auth.gr,
{juho.kanniainen, moncef.gabbouj}@tut.fi, alexandros.iosifidis@eng.au.dk

Abstract

Time-series forecasting has various applications in a wide range of domains, e.g., forecasting stock markets using limit order book data. Limit order book data provide much richer information about the behavior of stocks than its price alone, but also bear several challenges, such as dealing with multiple price depths and processing very large amounts of data of high-dimensionality, velocity and variety. A well-known approach for efficiently handling large amounts of high-dimensional data is the Bag-of-Features (BoF) model. However, the BoF method was designed to handle multimedia data, such as images. In this work, a novel temporal-aware neural BoF model is proposed tailored to the needs of time-series forecasting using high frequency limit order book data. Two separate sets of Radial Basis Function (RBF) and accumulation layers are used in the Temporal Bag-of-Features to capture both the short-term behavior and the long-term dynamics of time-series. This allows for modeling complex temporal phenomena that occur in time-series data and further increase the forecasting ability of the model. Any other neural layer, such as feature transformation layers, or classifiers, such as Multilayer Perceptrons, can be combined with the proposed deep learning approach, which can be trained end-to-end using the back-propagation algorithm. The effectiveness of the proposed method is validated using a large-scale limit order book dataset, containing over 4.5 million limit orders, and it is demonstrated that it greatly outperforms all the other evaluated methods.

Index Terms

Limit Order Book, Neural Networks, Bag-of-Features

I. Introduction

Time-series forecasting has various applications in a wide range of domains, e.g., forecasting stock markets [1], energy load prediction [2], etc. The main focus of this paper is forecasting the mid price movement of stocks using high frequency limit order book data. Note that several useful stock market forecasting tasks can be defined, e.g., forecasting the price of a stock based on its behavior or predicting possible stock market crashes, thus protecting the investors. Limit order book data [3], provide much richer information about the behavior of stocks than its price alone, but also bear several challenges, such as dealing with multiple price depths and processing very large amounts of data of high-dimensionality, velocity and variety. An order to sell or buy an amount of shares at a specific price or better is called *limit order*. This can be better understood through the following example. A buyer who intends to buy 10 shares for at most \$1 each will place a buy limit order with volume of 10 and price \$1. On the other hand, a seller who intends to sell 10 shares for at least \$1 will place a sell limit order with volume of 10 and price \$1. Note that both the price and the volume are needed to define a limit order. All the ask and bid prices and their corresponding volumes are kept in the *order book*. The bid orders are sorted in descending order of their prices, while the ask orders are sorted in increasing order of their prices. When a bid order price exceeds an ask order price, then they are executed by exchanging the corresponding assets. If the orders have different volumes, then the order with the largest volume remains in the order book (after subtracting the volume used in the aforementioned transaction). The *mid price* of a stock is the average between the highest bid and the lowest ask prices.

Many different methods have been proposed in the literature to classify time-series data [4]–[10]. In [4] and [5], distance metrics, e.g., the Dynamic Time Wrapping, were developed to measure the distance between time-series. Then, the kNN algorithm can be used to classify the data. More advanced techniques, such as [6], [7], [11], employ recurrent neural networks or hidden Markov models to perform time-series forecasting. In contrast to these approaches, in this work the Bag-of-Features model [8], [9], [12], is adapted towards efficiently processing large amounts of complex and high-dimensional limit order book time-series.

The Bag-of-Features model (BoF) was initially proposed for efficiently extracting compact histogram representations from images [13]. However, it was then successfully used for many different tasks, such as audio [10] and video [14] representation. The BoF model involves the following steps: First, multiple features are extracted from each object (*feature extraction* step). Each object can be represented as a set of features in the *feature space* formed by this process. Then, the *dictionary learning* step follows, where the extracted feature vectors are used to form a compact dictionary that contains the most representative features. These feature vectors are also called *codewords*. Finally, each feature vector is quantized using the codebook and the corresponding object is represented by a histogram vector. During this step, which is called *feature quantization and encoding* step, the histogram space is formed.

Using the BoF model it is possible to extract a constant length representation of time-series regardless their actual length. The extracted representation captures part of the behavior of the time-series and it can be then used efficiently for various classification tasks without having to directly process the raw data. However, a feature extractor must be first employed to extract multiple feature vectors from a time-series before using the BoF model. Several options exist for this step. Perhaps the simplest one is to consider each multi-dimensional time-series point as a separate feature vector [9]. More advanced techniques involve using short intervals of various lengths to handle time warping [8], or even employing handcrafted feature extractors for extracting more complex features [15], [16].

The representation power of the extracted histogram depends on the dictionary learning step. Early BoF methods, e.g,. [13], [17], employed simple clustering algorithms, e.g., k-means, to learn a generic dictionary that minimizes the reconstruction loss of the quantized vectors. However, supervised dictionary learning, e.g., [9], [18], [19], which learns a codebook tailored towards a specific task, improves the classification accuracy. Even though supervised dictionary learning allows for the optimized BoF model to perform significantly better than its unsupervised counterpart (this was also experimentally verified in Section IV-C), it suffers from a major drawback. The process of aggregating the extracted feature vectors discards a great deal of temporal information contained in the original time-series. The method proposed in this paper mitigates this problem by separately modeling the short-term behavior and the long-term dynamics of the time-series significantly increasing the forecasting accuracy.

The contributions of this work are summarized bellow. First, the BoF model is expressed as a neural layer, which includes a Radial Basis Function (RBF) layer and an accumulation layer. The proposed layer receives the feature vectors extracted from a time-series and extracts its representation. Two separate sets of RBF and accumulation layers are used to describe the short-term behavior as well as the long-term dynamics of the time-series. This allows for capturing complex temporal phenomena that occur in time-series data and further increase the forecasting abilities of the proposed method (as it is demonstrated in Section IV). Any other neural layer, such as feature transformation layers, or classifiers, such as Multilayer Perceptrons, can be combined with the proposed deep learning approach, which is called Temporal Bag-of-Features (Temporal BoF) and can be trained end-to-end using the back-propagation algorithm. To the best of our knowledge this is the first BoF-based neural formulation that can model both the short-term behavior and the long-term dynamics of time-series data. The effectiveness of the proposed method is validated using a large-scale limit order book dataset that contains over 4.5 million limit orders.

The rest of the paper is structured as follows. The related work is introduced and discussed in Section II, while in Section III the proposed method is presented in detail. Next, in Section IV, the Temporal BoF is evaluated using a large-scale limit order book dataset. Finally, Section V concludes the paper and introduces several interesting future work directions.

II. Related Work

This work concerns both stock forecasting using limit order book data, as well as, supervised dictionary learning for time-series classification using the BoF model. There are several recently proposed machine learning methods for predicting various aspects of the financial markets using limit order book data [11], [15], [20]–[24]. In [15], a set of hand-crafted features are extracted and then a Support Vector Machine (SVM) classifier is utilized to predict whether the mid price of a stock stays stationary, increases or decreases, while a similar task is tackled using a deep learning approach in [11], [20], [24], [25]. A different methodology is used in [21], and [22], where reinforcement learning is used to learn the optimal way to perform trading. In contrast with these works, the proposed method is the first that utilizes a neural formulation of the BoF model, that is capable of both handling large amounts of limit order book data and modeling the short-term, as well as the long-term behavior of a stock.

Many dictionary learning methods for the BoF model were also proposed. In [26], the mutual information between each codeword and the corresponding features labels is maximized, while in [27], multiple maximum margin hyperplanes are employed and the corresponding codebook is adjusted to maximize these margins. However, this approach requires a quadratic number of codebooks with respect to the number of the training classes. Later works, such as [18], and [28], where multi-class SVMs are used, addressed this problem. A supervised dictionary learning approach that employs both an MLP layer and a codebook layer was proposed in [29], while an end-to-end neural formulation was proposed in [19]. In [30], the logistic regression loss was used to form the optimization objective, while the representation is learned using an LDA-based criterion in [31]. The aforementioned approaches were developed to handle image classification tasks instead of time-series classification.

A BoF-based time-series representation approach was proposed in [9], where a discriminative objective was employed for the optimization of the codebook. Also, in [12] a BoF-based method was combined with a retrieval-oriented loss function to

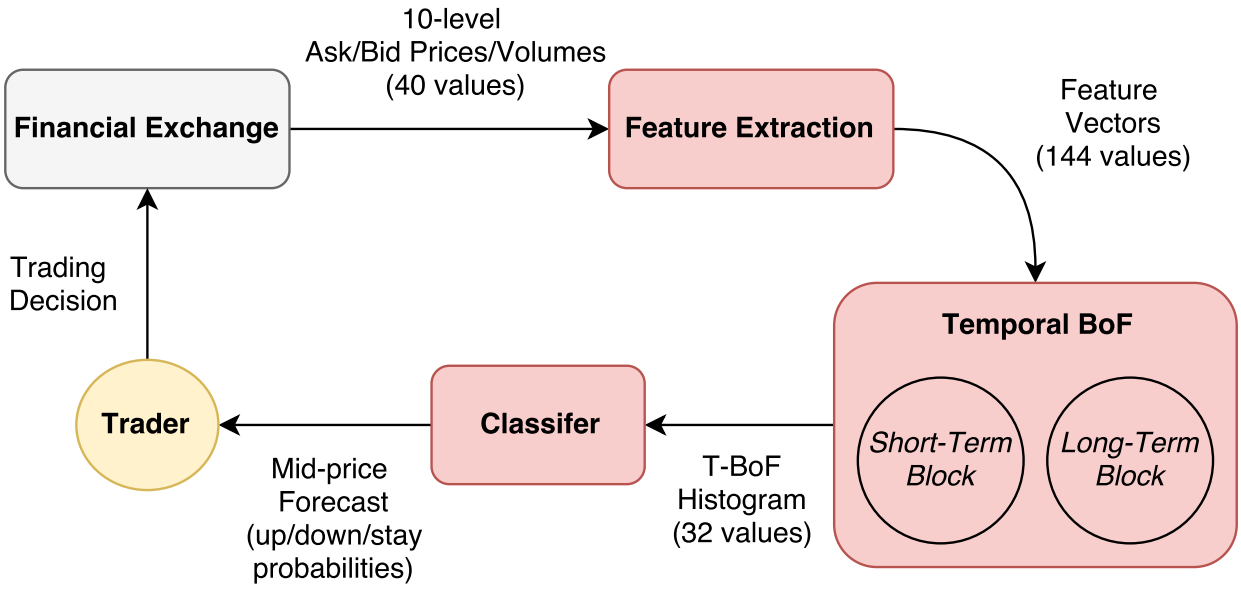


Fig. 1: Pipeline of the proposed financial forecasting model

learn a retrieval-oriented representation. On the other hand, time-series segments of various lengths were employed in [8], to allow for handling warping. A simple temporal modeling approach was also used in [16], while a neural BoF formulation was used in [32] to classify time-series data. To the best of our knowledge, the proposed Temporal BoF approach is the first that is capable of efficiently modeling both short-term behavior and the long-term dynamics of time-series data using a fully neural architecture that can be optimized end-to-end towards forecasting.

III. Proposed Method

The pipeline of the proposed financial forecasting model is shown in Figure 1. First, the raw time-series stream is fed to the employed feature extractor that extracts multiple feature vectors, following the procedure described in Subsection III-A. Then, the extracted features are fed to the proposed Temporal BoF model to extract one constant-length representation for each time-series that models both its long-term and short-term behavior using two separate codebooks (Subsection III-B). The extracted histogram is then received by a classifier to predict the mid price movement (Subsection III-C). Note that end-to-end training is employed to simultaneously learn all the parameters of the proposed model (Subsection III-D). The predicted mid price movements (forecast) are then evaluated by a domain expert (trader), who makes the final trading decision.

A. Feature Extraction

As discussed in Section 1, several different methods can be used to extract multiple feature vectors from a time-series. In this work, the feature extraction method proposed in [15] was employed to extract various features from the limit order book data. The raw order book data consist of the k highest bid and the k lowest ask prices and volumes:

1) $v_1^{(bid)}, v_2^{(bid)}, ..., v_k^{(bid)}$,
2) $p_1^{(bid)}, p_2^{(bid)}, ..., p_k^{(bid)}$,
3) $v_1^{(ask)}, v_2^{(ask)}, ..., v_k^{(ask)}$, and
4) $p_1^{(ask)}, p_2^{(ask)}, ..., p_k^{(ask)}$,

where $v_i^{(bid)}$ and $p_i^{(bid)}$ is the volume and the price at the i-th level on the bid side, while $v_i^{(ask)}$ and $p_i^{(ask)}$ are the corresponding values for the ask side. The number of levels k was set to 10. Therefore, for each time-step 40 values are used as input to the feature extractor. Following the feature extraction process described in [15], the following features were extracted from each raw time-series: a) the raw 10-level order book data (volumes and prices for both the bid and ask sides of each level), b) time-insensitive features (spread and mid prices, price differences, price and volume means and accumulated price differences) and c) time-sensitive features (price and volume derivatives, average intensity and intensity comparisons, as well as limit activity accelerations). These statistics are calculated every 10 time steps, effectively sub-sampling the data by a factor of 10. The dimensionality of the extracted feature vectors, denoted by $\mathbf{x}_{ij}$, is 144 and each feature vector carries information for the short-term behavior of the corresponding stock during the last 10 orders. The interested reader is referred to [15] for a detailed description of the extracted features. Each of the 144 values were normalized using z-score standardization, i.e., the data were

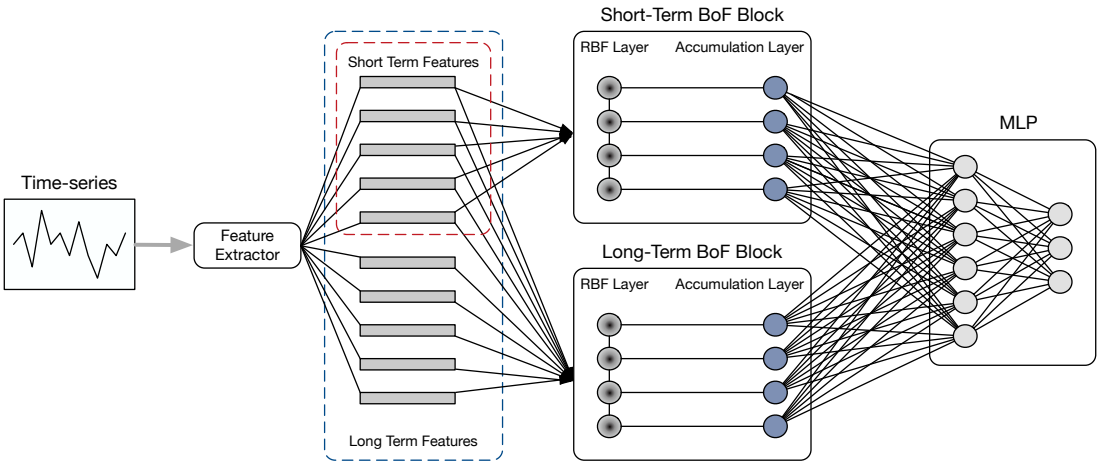


Fig. 2: The architecture of the proposed Temporal BoF model

normalized to have unit variance and zero mean. Note that only the training set statistics were used for the normalization process to ensure a fair evaluation of the proposed method (the test set statistics cannot be known beforehand).

B. Temporal BoF Quantization

The i-th time-series can be represented by a set of N_i feature vectors: $\{\mathbf{x}_{ij} \in \mathbb{R}^D,\ j = 1, \ldots, N_i\}$, where D denotes the dimensionality of the extracted feature vectors ($D = 144$ if the feature extraction process described in the previous subsection is used). The proposed method is capable of modeling both the long-term dynamics and the short-term behavior of a time-series using two different quantization layers with separate codebooks. For each extracted feature vector $\mathbf{x}_{ij}$, two binary variables are introduced, $s_{ij}^{(long)}$ and $s_{ij}^{(short)}$, to indicate whether the corresponding feature vector is used for compiling the long-term or the short-term representation. The intuition behind the proposed Temporal BoF quantization is to use a longer horizon N_L to model the long-term dynamics of a stock, while employing a small window N_S to model its short-term dynamics. That is, for the i-th time-series, the binary variables are set as:

$$s_{ij}^{(long)} = \begin{cases} 1, \text{if } i \geq N_i - N_L \ (N_L \text{ most recent time-steps}) \\ 0, \text{otherwise} \end{cases}, \tag{1}$$

and

$$s_{ij}^{(short)} = \begin{cases} 1, \text{if } i \geq N_i - N_S \ (N_S \text{ most recent time-steps}) \\ 0, \text{otherwise} \end{cases}, \tag{2}$$

assuming that a total of N_i feature vectors have been extracted and the feature vectors are sorted from the earliest to the most recent one. The N_S most recently extracted feature vectors are used to calculate the short-term histogram, while N_L feature vectors from a longer horizon are used for the calculation of the long-term histogram that models the long-term behavior of each time-series. Therefore N_L is set to a value larger than N_S, i.e., $N_L > N_S$. Even though N_L can be set to N_i, i.e., all the available feature vectors can be used to model the long-term behavior, N_L is set to a smaller value due to memory constraints. Also, note that, by definition, the feature vectors used for the short-term histogram are always used for compiling the long-term representation, but not vice-versa.

The Temporal BoF model is composed of two blocks, the short-term block and the long-term block, as shown in Figure 2. Each of these blocks is further composed of two sub-layers. First, an RBF layer is employed to measure the similarity of the input feature vector to the RBF centers. Then, an accumulation layer is used to build the final histogram. Therefore, the proposed method can be thought as a neural layer that feeds the extracted histogram to a subsequent classifier, as it is demonstrated in Figure 1.

The activation of the k-th RBF neuron of the m-th block (either $m = 0$ for the long-term block, or $m = 1$ for the short-term block) is calculated as:

$$[\boldsymbol{\phi}(\mathbf{x})]_{mk} = \exp(-||(\mathbf{x} - \mathbf{v}_{mk}) \odot \mathbf{w}_{mk}||_2) \in \mathbb{R}, \tag{3}$$

where $\mathbf{x}$ is a feature vector, $\mathbf{v}_{mk}$ is the center of the k-th RBF neuron of the m-th layer block and $\odot$ is the element-wise multiplication operator. An additional weight vector $\mathbf{w}_{mk} \in \mathbb{R}^D$ also exists for each RBF neuron. In that way, the width of the

Gaussian function of each RBF neuron can be independently finetuned towards the task at hand. In this work, a normalized RBF architecture is employed to ensure that the output of each neuron is bounded:

$$[\boldsymbol{\phi}(\mathbf{x})]_{mk} = \frac{\exp(-||(\mathbf{x} - \mathbf{v}_{mk}) \odot \mathbf{w}_{m\kappa}||_2)}{\sum_{\kappa=1}^{N_K} \exp(-||(\mathbf{x} - \mathbf{v}_{m\kappa}) \odot \mathbf{w}_{m\kappa}||_2)} \in \mathbb{R}. \tag{4}$$

Note that the center of each RBF neuron acts similarly to the codewords used in the regular BoF model. Therefore, after feeding a feature vector into the (normalized) RBF neurons, the output of the RBF layer is a membership vector that expresses the similarity between the current feature vector and the codewords/RBF centers. It can be easily shown that this formulation is equivalent to the BoF model that uses soft quantization, as proved in [19]. Hard quantization is also supported, as described in [19]. However, hard quantization introduces non-continuities that make the optimization intractable.

The extracted membership vectors are then accumulated (averaged) in the next layer of the first (long-term) block, compiling the final long-term histogram representation of each time-series, similarity to the plain BoF model [19], as:

$$\mathbf{h}_{0i} = \frac{1}{N_i^{(long)}} \sum_{j=1}^{N_i} \boldsymbol{\phi}_0(\mathbf{x}_{ij}) \odot s_{ij}^{(long)} \in \mathbb{R}^{N_K}, \tag{5}$$

where $\boldsymbol{\phi}_m(\mathbf{x}) = ([\boldsymbol{\phi}(\mathbf{x})]_{m1}, \ldots, [\boldsymbol{\phi}(\mathbf{x})]_{mN_K})^T \in \mathbb{R}^{N_K}$ is the output of the first layer of the m-th block and $N_i^{(long)} = \sum_{j=1}^{N_i} s_{ij}^{(long)}$ is the number of features fed to the long-term block. Each $\mathbf{h}_{mi}$ has unit l^1 norm, defines a histogram and describes the long-term behavior of each time-series. Note that only the N_L most recent feature vectors contribute to the histogram $\mathbf{h}_1$. Similarly, the short-term histogram is calculated as:

$$\mathbf{h}_{1i} = \frac{1}{N_i^{(short)}} \sum_{j=1}^{N_i} \boldsymbol{\phi}_1(\mathbf{x}_{ij}) \odot s_{ij}^{(short)} \in \mathbb{R}^{N_K}, \tag{6}$$

where $N_i^{(short)} = \sum_{j=1}^{N_i} s_{ij}^{(short)}$ is the number of feature vectors fed to the short-term block. Again, only the N_S most recent feature vectors contribute to the short-term histogram vector. The final histogram representation $\mathbf{h}_i$ of the i-th time-series is extracted by concatenating the long-term histogram $\mathbf{h}_{0i}$ and the short-term histogram $\mathbf{h}_{1i}$:

$$\mathbf{h}_i = \begin{pmatrix} \mathbf{h}_{0i} \\ \mathbf{h}_{1i} \end{pmatrix} \in \mathbb{R}^{2N_K}. \tag{7}$$

The same number of N_K RBF neurons are used for both the long-term and the short-term histograms, leading to a representation with length $2N_K$. The vector $\mathbf{h}_i$ is then used for the subsequent forecasting tasks.

C. Temporal BoF Forecasting

The proposed Temporal BoF layer receives the time-series features and extracts its histogram $\mathbf{h}_i$, which is composed of the long-term histogram $\mathbf{h}_{0i}$ and the short-term histogram $\mathbf{h}_{1i}$. Then a classifier is used to predict the future behavior of the time-series using the extracted histogram vector. A multilayer perceptron (MLP) is used in this work, even though any other differentiable classifier can be used.

For each time-series i a label t_i is used to denote its class, while the number of possible categories is denoted by N_C. Also, let $\mathbf{W}_H \in \mathbb{R}^{N_H \times (2N_K)}$ denote the weight matrix of the hidden layer weights and $\mathbf{W}_O \in \mathbb{R}^{N_C \times N_H}$ denote the corresponding weight matrix of the output layer, where N_H is the number of hidden neurons. The activations of the hidden layer for a given histogram $\mathbf{h}_i$ are computed as:

$$\mathbf{p}_i = \phi^{(elu)}(\mathbf{W}_H \mathbf{h}_i + \mathbf{b}_H) \in \mathbb{R}^{N_H}, \tag{8}$$

where $\phi^{(elu)}(x)$ denotes the *elu* activation function [33], and $\mathbf{b}_H \in \mathbb{R}^{N_H}$ is the bias vector of the hidden layer. The parameter α_{elu} is set to 1, following the suggestions of [33]. The output of the final classification layer is calculated as:

$$\mathbf{y}_i = \phi^{(softmax)}(\mathbf{W}_O \mathbf{p}_i + \mathbf{b}_O) \in \mathbb{R}^{N_C}, \tag{9}$$

where each neuron corresponds to a label, $\mathbf{b}_O \in \mathbb{R}^{N_C}$ is the bias vector of the output layer and $\phi^{(softmax)}$ is the softmax activation function.

D. Temporal Bag-of-Features Learning

To train the network, the categorical cross entropy loss function is used:

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{N_C} [\mathbf{t}_i]_j \log([\mathbf{y}_i]_j). \tag{10}$$

Input: A set $\mathcal{X} = \{x_1, \ldots, x_N\}$ of N training time-series and the ground-truth prediction labels $\mathcal{L} = \{t_1, \ldots, t_N\}$
Hyper-parameters: g, N_K, N_{iters}, N_H, N_{piters}, N_{batch}, η_{MLP}, η_V, η_W
Output: The trained neural network

1: **procedure** OPTIMIZENETWORK
2: Use random orthogonal initialization to initialize the MLP weights
3: Initialize the RBF centers $\mathbf{v}_{0k}$ and $\mathbf{v}_{1k}$ (for each k) using the k-means algorithm
4: Initialize both $\mathbf{w}_{0k}$ and $\mathbf{w}_{1k}$ to $1/g$
5: Use the ADAM algorithm the pre-train the classification layer (MLP)
6: **for** $i \leftarrow 1; i \leq N_{iters}; i++$ **do**
7: **for** every batch $B \in \mathcal{X}$ **do**
8: Feedforward the extracted feature vectors
9: Backpropagate the network and compute the corresponding gradients
10: Apply the Adam algorithm and optimize the network
return the optimized parameters $\mathbf{v}_{mk}$, $\boldsymbol{w}_{mk}$ and $\mathbf{W}_{MLP}$

Fig. 3: Temporal BoF Learning Algorithm

The target output vector is denoted by $\mathbf{t}_i \in \mathbb{R}^{N_C}$ and depends on the label (t_i) of the input time-series. Therefore, the target vectors is defined as: $[\mathbf{t}_i]_j = 1$, if $j = t_i$, or $[\mathbf{t}_i]_j = 0$, otherwise. Since the employed Temporal BoF formulation is differentiable, the back-propagation method is employed for learning the parameters of the network using gradient descent:

$$\Delta(\mathbf{W}_{MLP}, \mathbf{v}_{mk}, \boldsymbol{w}_{mk}) = -(\eta_{MLP}\frac{\partial L}{\partial \mathbf{W}_{MLP}}, \eta_V \frac{\partial L}{\partial \mathbf{v}_{mk}}, \eta_W \frac{\partial L}{\partial \boldsymbol{w}_{mk}}), \tag{11}$$

where $\mathbf{W}_{MLP}$ denotes all the parameters of the classification layer. The derivatives defined in Equation (11) are calculated in Appendix A. Note that only the feature vectors that actually contribute to the calculation of each histogram are used to back-propagate the gradients to the corresponding RBF neurons. In other words, during the learning process, the short-term neurons are adapted to the features that model the short-term behavior of a time-series, while the RBF neurons of the long-term block are trained to model its long-term behavior.

The Adam (Adaptive Moment Estimation) algorithm [34], is employed for training the proposed model. The magnitude of the gradients of different parameters of the network varies largely. Therefore, to ensure a smooth convergence a different learning rate must be used for each parameter to account for the differences in the magnitude of the gradients. The largest learning rates that provided smooth convergence were used. Therefore, for the conducted experiments the following learning rates were employed: $\eta_{MLP} = \eta_V = 0.001$ and $\eta_W = 0.01$. Using lower learning rates is not expected to harm the classification performance, but more optimization iterations will be required, since the convergence speed is proportional to the used learning rate.

The codebook is initialized using the k-means algorithm to cluster the feature vectors $\mathcal{S} = \{\mathbf{x}_{ij} | i = 1, \ldots, N, j = 1, \ldots, N_i\}$. Then, the corresponding centroids can be used to initialize the neurons. Note that the centers of the RBF neurons are not fixed. Instead, they are learned using the back-propagation algorithm. The weight vectors of the RBF neurons are initially set to $1/g$, where g is a positive number. Again, the RBF input weights ($\mathbf{w}_k$) are learned during the training process. The initialization process is appropriately repeated for the two separate sets of RBF neurons that are used, i.e., for the short-term block and the long-term block.

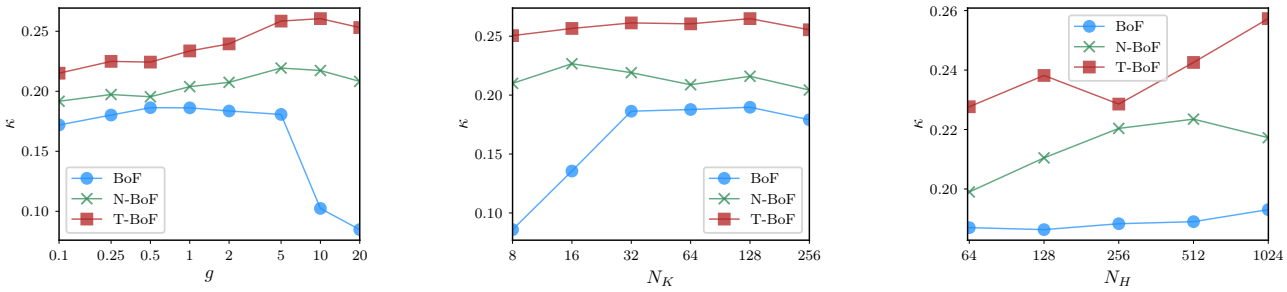
The complete learning algorithm for the Temporal BoF model is shown in Figure 3. First, the parameters of the MLP are initialized using random orthogonal initialization [35] (line 2). Then, the RBF centers of the short-term and the long-term blocks are initialized by running k-means on a subsample of the extracted feature vectors and the input weights of the RBF neurons are set to $1/g$ (lines 3-4). The output layers (MLP) are pre-trained for N_{piters} to avoid back-propagating gradients from a randomly initialized output layer to the BoF layers (line 5). Finally, the training time-series are fed to the network in batches of N_{batch} and all the parameters of the network are optimized using the Adam algorithm (lines 6-10). For all the conducted experiments the network was trained for $N_{iter} = 5000$ iterations using batch size of $N_{batch} = 32$. For pre-training the MLP $N_{piter} = 500$ iterations were used. Note that the used financial dataset is highly unbalanced, since most of the time the mid price remains stationary. Therefore, time-series from the less common classes were sampled with a higher probability, i.e., in each batch the same number of samples of each class were used. The architecture of the network, i.e., the number of RBF neurons, the number of hidden neurons, etc., was selected using experiments on a validation set (Section IV-C).

IV. EXPERIMENTS

In this Section the proposed method was evaluated using a large-scale limit-order book dataset. First, the used dataset is briefly described and the utilized pre-processing and feature extraction procedures are introduced. Then, the evaluated metrics and the experimental evaluation under different evaluation setups are presented and discussed.

TABLE I: ISIN (International Securities Identification Number), market capitalization and average number of shares traded for each of the used stocks (Statistics for June 2010)

Stock	ISIN	Market Cap.	# traded shares
Kesko Oyj	FI0009000202	$1.8 billions	179,625
Outokumpu Oyj	FI0009002422	$2.3 billions	1,387,391
Sampo Oyj	FI0009003305	$9.7 billions	1,265,170
Rautaruukki	FI0009003552	$1.7 billions	746,385
Wartsila Oyj	FI0009003727	$3.7 billions	338,595



(a) Varying the scaling parameters g (b) Varying the number of codewords / RBF neurons (N_K) (c) Varying the number of hidden neurons (N_H)

Fig. 4: Effect of varying various hyper-parameters

A. Dataset Acquisition and Preprocessing

To evaluate the proposed method a limit order book dataset, called FI-2010 in this paper, was collected [36]. The FI-2010 is a large-scale dataset that consists of limit order book data that were collected from 5 Finish companies traded in Helsinki Exchange (operated by Nasdaq Nordic): Kesko Oyj (retail industry), Outokumpu Oyj (steel industry), Sampo Oyj (insurance industry), Rautaruukki (steel industry), and Wartsila Oyj (manufacturing industry). The ISIN (International Securities Identification Number), the market capitalization, and the average number of the shares traded for each stock during the data collection period (July 2010) is shown in Table I. For each time-step the 10 highest bid and 10 lowest ask orders prices and volumes were collected. The data were gathered over a period of 10 business days (1st June 2010 to 14th June 2010) and a total number of 4.5 million limit orders were collected. The feature extraction process described in Section III-A was used leading to 453,975 extracted feature vectors. The performance of the algorithms was evaluated using the following setup: The 15 most recently extracted feature vectors were used to predict the direction (up, stationary or down) of the mean mid price after k time-steps. A moving average filter (window size 9) was used to filter the mid prices. The proposed method was evaluated for the prediction of the mid price's direction for the next 10, 50 and 100 time-steps. For each of them, the threshold for considering the stock stationary was set to 0.01%, 0.02%, and 0.03% respectively. Note that as the prediction horizon increases, then the changes in the mid price are becoming larger. An anchored setup was used to evaluate the methods [37]. First, the data that were extracted from the first day were used to train the model, while the data from the second day were used for the evaluation. Then, the two first days were used for the training and the next day was used for the evaluation, etc. For all the evaluated metrics, the mean and the standard deviation are reported.

B. Evaluation Metrics

To evaluate the performance of the proposed method the following metrics were employed: a) accuracy, b) macro-precision, c) macro-recall d) macro-F1 per class [38], and e) Cohen's κ metric [39]. Accuracy is defined as $acc = N_{correct}/N_{total}$, where $N_{correct}$ is the number of predicted labels that match the ground truth labels and N_{total} is the total number of samples. Let TP_c, FP_c, TN_c and FN_c denote the true positives, false positives, true negatives and false negatives (respectively) for a class c. Then, the precision of a class is defined as $prec_c = TP_c/(TP_c+FP_c)$, while the recall as $recall_c = TP_c/(TP_c+FN_c)$. The F1 score for a class c is calculated as the harmonic mean of the recall and the precision, i.e., $F1_c = 2\cdot(prec_c \cdot recall_c)/(prec_c+recall_c)$. Macro-averaging is used, i.e., the metrics are separately calculated for each class and then averaged together. The Cohen's κ allows for evaluating the agreement between two different sets of annotations, while accounting for the possible random agreements. Cohen's κ is defined as $\kappa = (p_0 - p_e)/(1 - p_e)$, where p_0 is the probability of agreement between the predicted labels and the ground truth, while p_e is the probability of having random agreement after taking into account the statistics of the classes [39]. For all the evaluated metrics, higher values indicate better classification performance.

TABLE II: Ablation Study

Method	**F1** (%)	**Cohen's κ**
Long-term	41.63 ± 1.90	0.1724 ± 0.0212
Short-term	43.19 ± 2.34	0.1876 ± 0.0250
Long-term + Short-Term	$\mathbf{43.96 \pm 1.59}$	$\mathbf{0.1992 \pm 0.0201}$

C. Hyper-parameter Selection and Stability Analysis

Before evaluating the proposed method using the FI-2010 dataset, the effect of hyper-parameters is evaluated. To this end, a validation set that consists of the last two days of the FI-2010 data was used. The proposed Temporal BoF model, abbreviated as T-BoF, is compared to two other BoF-based methods: the regular unsupervised BoF model [13], and a recently proposed neural formulation of the BoF model [19], which is called Neural BoF (abbreviated as N-BoF). The same classifier, i.e., an MLP with 512 hidden units, is used for all the evaluated methods.

The effect of varying the scaling parameter g on the Cohen's κ metric is shown in Figure 4a (32 codewords/RBF neurons were used for the conducted experiments). Note that the value g remain fixed through the training process for the BoF model, while for the N-BoF and the T-BoF methods the value of g is only used to initialize the input weights of the RBF neurons that are then learned during the training. Using large values, i.e., $g > 1$, seems to significantly improve the performance of the N-BoF and the T-BoF methods that peak around $g = 5$ and $g = 10$ respectively. Note that in other domains, e.g., image classification, harder assignments $g < 0.1$ seem to work better [19]. However, due to the relatively small number of extracted feature vectors (15 feature vectors are extracted from each time-series) softer assignments leads to significantly improved forecasting accuracy. On the other hand, for the BoF model large values for the scaling parameter harm the forecasting abilities of the model. When the simple BoF model is used the best results are obtained for $g = 0.5$. Note that the proposed combined short-term and long-term modeling of the time-series used in the Temporal BoF always improves the κ metric, regardless selected scaling factor g, highlighting the importance of capturing both the short-term and long-term time-series dynamics. For all the conducted experiments the feature vectors extracted from the last 15 time-steps were used for compiling the long-term representation, while the feature vectors extracted from the last 5 time-steps were used for compiling the short-term representation.

The effect of varying the number of codewords is shown in Figure 4b. The performance of both N-BoF and the T-BoF models are relatively stable, given that at least 16 neurons are used. For the regular BoF model, the forecasting performance stabilizes when 32 RBF neurons are used. The best performance for the BoF model is achieved when 128 RBF neurons are used, while for the N-BoF model is achieved for 16 neurons. For the proposed T-BoF model 16 RBF neurons are used in each of the two temporal BoF blocks leading to a total number of 32 RBF neurons (which is the same as in the N-BoF model). Again, the proposed Temporal BoF model outperforms all the other evaluated models for any number of used codewords/RBF neurons.

Also, the effect of varying the number of hidden neurons is evaluated in Figure 4c (the curves were smoothed using a moving average filter with window size 2 to reduce the fluctuations). The classification performance tends to increase as the number of hidden neurons increases for all the evaluated models. However, using more hidden neurons also increases the complexity and the training time of a model. Therefore, for all the conducted experiments the number of hidden neurons was set to $N_H = 512$, which provided the best trade-off between classification performance and complexity.

Also, an ablation study is provided in Table II, where the effect of the long-term, short-term and the combined long-term/short-term representation is evaluated (the prediction target was set to the next 10 time-steps, while the best selected hyper-parameters were used for the conducted experiments). The short-term representation seems to have a larger influence on the forecasting accuracy. Nonetheless, the proposed T-BoF method, which combines both the short-term and the long-term representations, significantly outperforms the rest of the representations. In Table III the performance of the proposed T-BoF method is also evaluated for different number of feature vectors used to compile the short-term representation (N_S). Using the 5 most recent features vectors leads to the best classification performance. Note that using more feature vectors decreases the evaluated metrics since the short-term representation starts overlapping with the long-term representation, for which the last 15 feature vectors were used. Using more feature vectors for the long-term representation can have a positive effect on the classification performance, but also requires more memory for storing the vectors. Therefore, in the conducted experiments the number of feature vectors used for compiling the long-term representation was set to $N_L = 15$. The selected hyper-parameters are summarized in Table IV. The total representation length is also reported in the last line of Table IV.

Finally, the effect of the dataset size on the forecasting accuracy is evaluated in Table V. The prediction target was set to the next 50 time-steps, the proposed T-BoF method was used for all the conducted experiments, while the last test split (day) was used for the evaluation. Therefore, the proposed method was evaluated using the most recent k days for the training process. Using more data seems to steadily increase the classification accuracy, which peaks when the 8 most recent days were used for training. Using training data over a longer horizon can possibly negatively impact the forecasting accuracy, due to concept drift issues [40], as demonstrated in Table V when the last 9 days were used: the κ metric drops from 0.30 (training with data

TABLE III: Effect of varying the number of feature vectors (N_S) used for compiling the short-term representation

Short Term Horizon	Precision	Recall	F1	Cohen's κ
1	46.24	57.24	46.84	0.2379
2	46.82	58.74	47.49	0.2469
5	**47.34**	58.68	**48.21**	**0.2558**
7	45.71	**59.14**	43.21	0.2101
10	46.01	56.21	45.46	0.2222

TABLE IV: Hyper-parameters selected for the conducted experiments

Hyper-parameters	BoF	N-BoF	T-BoF
Learning rate η_{MLP}		0.001	
Learning rate η_V		0.001	
Learning rate η_Q		0.01	
Short-term horizon N_S		5	
Long-term horizon N_L		15	
# hidden neurons N_H		512	
# codewords N_K	128	16	16
Scaling factor g	0.5	5	10
Representation length	128	16	32

from 8 days) to 0.26 (training with data from 9 days).

D. Experimental Evaluation

The experimental results using the anchored evaluation protocol for different prediction targets are shown in Table VI. The mid price movement was predicted for the next 10, 50 and 100 time-steps (prediction targets). An MLP with 512 hidden units, that utilizes the last feature vector extracted from the time-series, was used as baseline method for the conducted experiments. Apart from the MLP, two unsupervised BoF-based models were employed: a) a simple BoF model that uses all the extracted feature vectors and b) a BoF model that uses two independent (unsupervised) codebooks, one for short-term modeling and one for long-term modeling. The first approach is denoted by "BoF", while the latter by "BoF-2T". The proposed method is also compared to a powerful state-of-the-art supervised BoF method, the Neural BoF (N-BoF) [19], [32].

Several conclusions can be drawn from the results reported in Table VI. First, learning all the parameters of the BoF model using the Neural BoF method allows for significantly increasing the forecasting performance for all the evaluated metrics, confirming the importance of the supervised dictionary learning. Also, the temporal modeling involved in the "BoF-2T" approach increases the classification metrics over the plain BoF approach. Nonetheless, the proposed Temporal BoF method further increases the evaluated metrics for every prediction target both over the baseline MLP and the other BoF formulations. For example, the Cohen's κ metric increases over 65% for predicting the mean mid price of the next 10 time-steps, over 70% for predicting the mean mid price of the next 50 time-steps, and over 60% for predicting the mean mid price of the next 100 time-steps. Note that predicting the mid-term behavior, i.e., the movement of the average mid price for the next 50 time-steps, is easier than the more short-term predictions, i.e., predicting the movement of the average mid price for the next 10 time-steps. This behavior can be understood if the noisy and high frequency nature of the short term mid price movements is considered. Therefore, it is easier to predict the more stable and noise-free mid-term behavior rather than the very short-term noisy movements. The precision, recall and F1 score for each class are also reported individually for the best performing model (predication target 100) in Table VII. It is worth noting that approximately 74% of the data belong to the stationary class, while the rest 26% of the data belong to the up (13%) and down (13%) classes. Therefore, an appropriately biased random classifier (according to the aforementioned class probabilities) would achieve around 13% precision/recall for the up and down classes and a mean precision/recall/F1 score of 33%.

TABLE V: Effect of the dataset on the forecasting accuracy (prediction target: 50)

Training Days (# samples)	F1	Cohen's κ
1 (57,536)	40.93	0.1736
2 (117,094)	45.76	0.2323
3 (161,778)	43.92	0.2163
4 (208,616)	49.35	0.2592
5 (249,372)	48.98	0.2622
6 (292,209)	50.58	0.2818
7 (325,514)	48.93	0.2654
8 (370,213)	**52.70**	**0.3039**
9 (417,140)	48.83	0.2638

TABLE VI: Evaluation results using the FI-2010 dataset

Method	Predict Target	Accuracy	Precision	Recall	F1	Cohen's κ
MLP	10	55.81 ± 6.98	40.20 ± 0.50	56.25 ± 2.20	36.91 ± 1.81	0.1281 ± 0.0137
PCA	10	55.91 ± 6.98	39.69 ± 0.83	53.98 ± 1.89	36.32 ± 2.26	0.1209 ± 0.0188
LDA	10	56.00 ± 6.47	39.56 ± 0.91	53.60 ± 2.29	36.17 ± 1.76	0.1187 ± 0.0146
AE	10	57.41 ± 6.49	40.07 ± 1.15	54.24 ± 1.63	37.17 ± 2.71	0.1263 ± 0.0231
BoF	10	57.59 ± 7.34	39.26 ± 0.94	51.44 ± 2.53	36.28 ± 2.85	0.1182 ± 0.0246
BoF-2T	10	59.09 ± 6.33	40.02 ± 0.90	54.05 ± 2.08	37.64 ± 2.14	0.1345 ± 0.0195
N-BoF	10	62.70 ± 6.73	42.28 ± 0.87	61.41 ± 3.68	41.63 ± 1.90	0.1724 ± 0.0212
T-BoF	10	$\mathbf{63.96 \pm 4.94}$	$\mathbf{43.85 \pm 1.11}$	$\mathbf{66.66 \pm 3.40}$	$\mathbf{43.96 \pm 1.59}$	$\mathbf{0.1992 \pm 0.0201}$
MLP	50	52.25 ± 5.48	44.03 ± 1.25	52.67 ± 1.56	41.91 ± 2.33	0.1787 ± 0.0229
PCA	50	52.59 ± 5.59	43.39 ± 1.34	51.20 ± 1.94	41.34 ± 1.81	0.1730 ± 0.0202
LDA	50	53.60 ± 3.72	43.12 ± 1.52	50.46 ± 2.01	41.48 ± 1.83	0.1726 ± 0.0205
AE	50	52.76 ± 4.00	43.48 ± 1.31	50.99 ± 1.62	41.32 ± 1.94	0.1722 ± 0.0193
BoF	50	50.21 ± 5.59	42.56 ± 1.26	49.57 ± 2.28	39.56 ± 2.36	0.1576 ± 0.0254
BoF-2T	50	53.34 ± 3.92	43.70 ± 1.19	51.51 ± 1.75	41.79 ± 2.05	0.1803 ± 0.0239
N-BoF	50	56.52 ± 8.67	47.20 ± 1.80	58.17 ± 2.61	46.15 ± 4.07	0.2285 ± 0.0419
T-BoF	50	$\mathbf{59.43 \pm 5.28}$	$\mathbf{49.58 \pm 2.10}$	$\mathbf{63.50 \pm 2.54}$	$\mathbf{49.82 \pm 3.18}$	$\mathbf{0.2723 \pm 0.0348}$
MLP	100	52.26 ± 6.06	44.76 ± 1.71	51.65 ± 2.15	43.29 ± 2.87	0.1904 ± 0.0301
PCA	100	51.70 ± 5.53	43.64 ± 1.47	49.16 ± 1.39	41.85 ± 2.83	0.1747 ± 0.0273
LDA	100	53.96 ± 5.69	43.52 ± 1.81	48.28 ± 2.22	42.27 ± 2.95	0.1761 ± 0.0295
AE	100	51.96 ± 7.13	43.75 ± 1.66	48.76 ± 1.76	41.78 ± 3.37	0.1725 ± 0.0329
BoF	100	50.97 ± 5.62	42.89 ± 1.46	47.84 ± 2.08	40.84 ± 2.78	0.1641 ± 0.0300
BoF-2T	100	54.34 ± 5.44	43.51 ± 1.34	48.42 ± 1.83	42.58 ± 2.19	0.1811 ± 0.0275
N-BoF	100	56.43 ± 6.86	47.27 ± 1.72	54.99 ± 2.19	46.86 ± 2.87	0.2300 ± 0.0338
T-BoF	100	$\mathbf{57.94 \pm 5.42}$	$\mathbf{49.52 \pm 2.17}$	$\mathbf{59.35 \pm 2.49}$	$\mathbf{49.65 \pm 3.62}$	$\mathbf{0.2647 \pm 0.0414}$

TABLE VII: Per class precision, recall and F1 score for the best performing model (T-BoF)

Class	Precision	Recall	F1
$\uparrow$	31.76 ± 4.14	57.46 ± 5.43	40.56 ± 3.32
$-$	86.84 ± 4.89	57.49 ± 7.92	68.79 ± 6.03
$\downarrow$	29.97 ± 6.58	62.95 ± 7.82	39.66 ± 5.14

The proposed method was also compared with three other feature learning baselines in Table VI: a) the Principal Component Analysis (PCA) method [41], the Linear Discriminant Analysis (LDA) method [42], and an Autoencoder (AE) [43]. All methods received the same input as the MLP (a 144-valued feature vector) and reduced its dimensionality into 64 values (2 values for the LDA method). The autoencoder was trained for 2000 iterations and the same setup (activation function, input scaling, etc.) as the MLP was used. After extracting the new representation, an MLP was trained to classify the samples (as in the other methods). Even though these feature learning methods perform better than the plain BoF model, the proposed T-BoF approach significantly outperform them.

Next, the average feed-forward (forecasting) time for the various BoF-based models was evaluated. The results are shown in Table VIII. Both the N-BoF and the proposed T-BoF methods significantly reduce the classification time, while also requiring the least amount of memory. Therefore, the proposed method can accurately and promptly classify new data samples, providing an efficient and useful tool for high-frequency trading applications. Note that the best hyper-parameters, as reported in Table IV, were used for the conducted experiments.

The distribution of the activations of the RBF neurons/codewords are visualized in Figures 5, 6 and 7 to provide better insight and further evaluate the proposed methods. The mean value, the extrema (minimum and maximum values), as well as the density of the distribution are shown in the corresponding figures. In this way, it is possible to perform a qualitative comparison between the evaluated methods and compare their discriminative power. The last prediction target (100 time-steps ahead) is used for the conducted experiments. For the BoF and the N-BoF method 8 codewords are used, while for the proposed method 4 codewords are used for each temporal block leading to a total of 8 codewords. In the BoF model (Figure 5) two winning codewords (4 and 6) almost dominate over all the others restricting the representation ability of the model. In contrast,

TABLE VIII: Average feed-forward time for various BoF-based models and total number of parameters (average over 100 runs, a 4-core 3.2GHz CPU workstation with 16GB of RAM was used for the conduced experiments)

Model	Time	Number of parameters
BoF	0.972 msec	86k
BoF-2T	1.632 msec	170k
N-BoF	0.524 msec	12k
T-BoF	0.543 msec	23k

both the Neural BoF and the proposed Temporal BoF models lead to a much more rich distribution over the activations of the RBF neurons, increasing the representations power of the models, which is also indicated by the better forecasting performance (Table VI). Note that specific neurons are associated with specific mid price movements. For example the 3rd neuron is mostly activated for a downward movement, the 6th neuron fires when the mid price stays stationary, while the 5th neuron is more often activated when the mid price increases. For the Temporal BoF model the 5th neuron is more strongly associated with the stationary class, while the 3rd neuron is mostly activated with the upward movements. Similar conclusions can be draw for the other neurons as well.

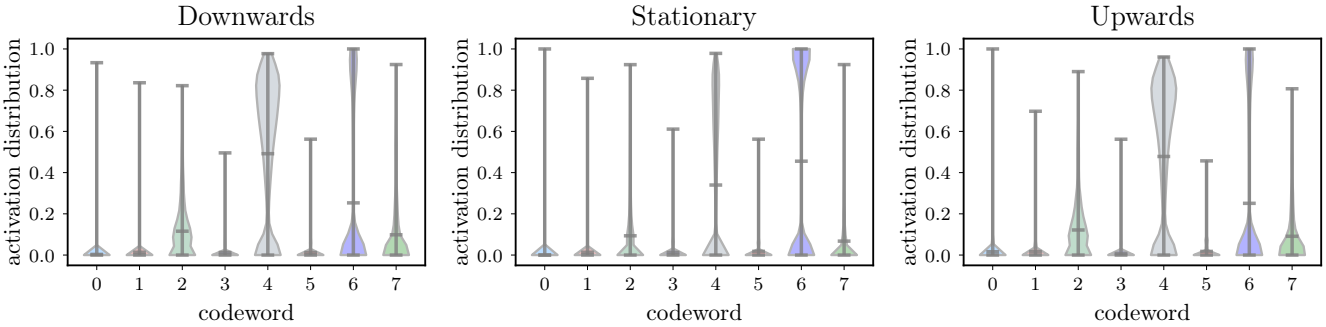


Fig. 5: BoF: Histogram distributions for three different mid price movements

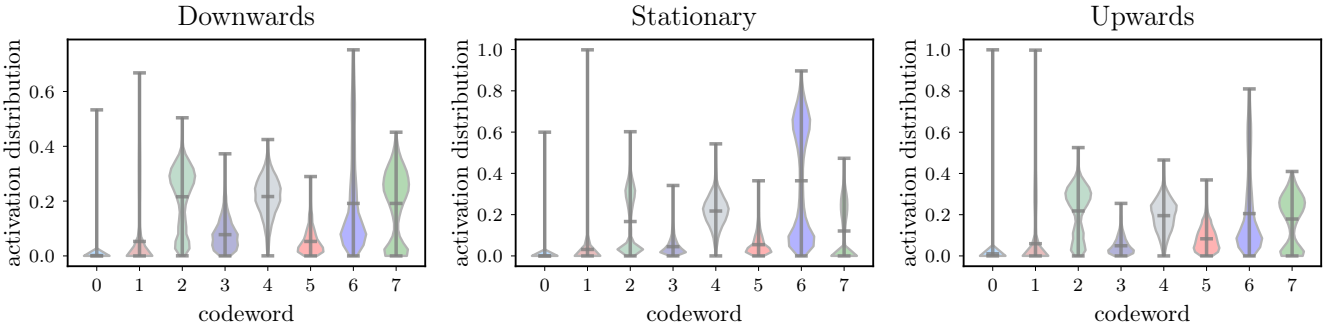


Fig. 6: Neural BoF: Histogram distributions for three different mid price movements

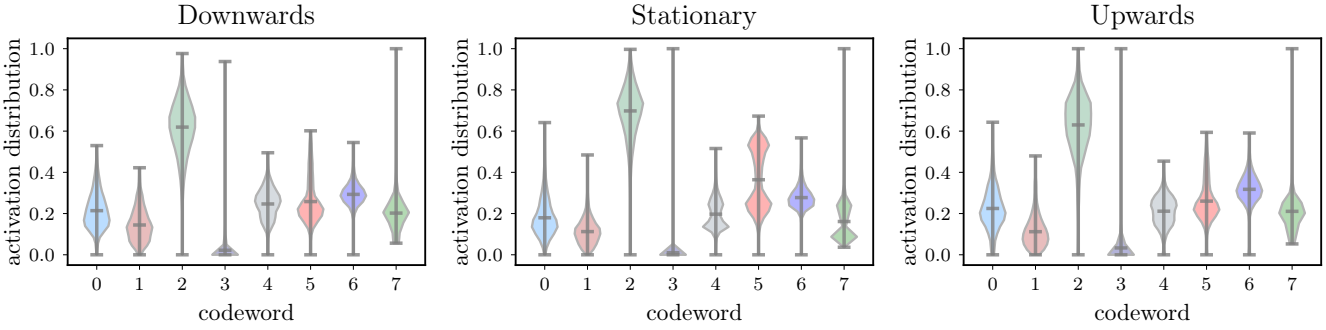


Fig. 7: Temporal BoF: Histogram distributions for three different mid price movements

The histograms of the learned models can be also used to visualize the time-series data and provide a new analysis tool that can be used in combination with visualization methods. In this work, the t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] is used to visualize the data, demonstrating the ability of the proposed method to actually learn the structure of the financial market. The t-SNE algorithm is a powerful data visualization method that works by learning a nonlinear low-dimensional embedding that accurately models the neighbor interactions between the data points. The resulting data points are learned in such way that the local relationships between the data points are maintained. Even though t-SNE is not an essential part of the proposed method, it allows for performing exploratory analysis using the learned representation, as well as comparing the representations learned using different methods. The scatter plot of the 2-dimensional data for three different methods (BoF, N-BoF and T-BoF) are provided in Figure 8. The perplexity of the t-SNE algorithm was set to 20, $N_K = 8$ codewords were used for all the evaluated methods and 1000 randomly selected training time-steps were visualized. Each

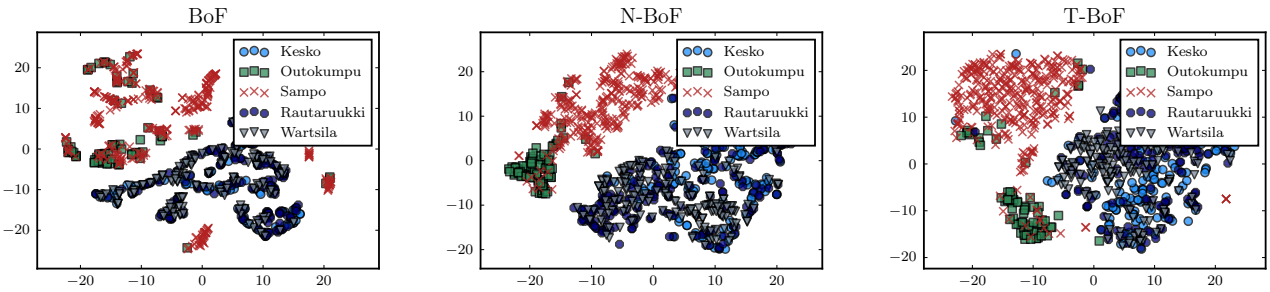


Fig. 8: Visualizing the learned BoF-based representations using the t-SNE algorithm. Different markers correspond to different stocks. The name of each stock is shown in the legend. (Figure is best viewed in color)

TABLE IX: Forecasting performance per stock and day (the first two days were used for training, while the next 5 days were used for evaluating the forecasting performance). The F1 measure and the Cohen's κ are reported.

Stock	**Day 1** (F1 (%), κ)	**Day 2** ((F1 (%), κ)	**Day 3** (F1 (%), κ)	**Day 4** (F1 (%), κ)	**Day 5** (F1 (%), κ)	**Day 6** (F1 (%), κ)	**Day 9** (F1 (%), κ)	**Total** (F1 (%), κ)
Kesko	(47.31, 0.223)	(44.92, 0.184)	(47.11, 0.2219)	(41.39, 0.131)	(45.40, 0.191)	(44.82, 0.179)	(40.55, 0.136)	(44.49, 0.185)
Outokumpu	(39.91, 0.171)	(48.85, 0.257)	(41.92, 0.191)	(39.04, 0.167)	(47.29, 0.247)	(36.57, 0.146)	(41.07, 0.193)	(43.62, 0.206)
Sampo	(45.74, 0.216)	(46.19, 0.199)	(46.44, 0.205)	(47.10, 0.225)	(45.96, 0.197)	(42.78, 0.162)	(48.53, 0.248)	(46.02, 0.208)
Rautaruukki	(42.08, 0.166)	(40.31, 0.146)	(45.23, 0.194)	(38.56, 0.131)	(41.15, 0.149)	(43.32, 0.159)	(40.36, 0.145)	(41.61, 0.156)
Wartsila	(40.46, 0.146)	(40.85, 0.1616)	(41.24, 0.160)	(43.40, 0.191)	(45.19, 0.199)	(40.31, 0.147)	(36.07, 0.099)	(41.38, 0.161)

of the five different stocks is marked with a different color/shape in Figure 8. It is evident that the proposed Temporal BoF leads to a much better separation of the different stocks in the 2-D space than the BoF and the N-BoF models. This behavior is quite interesting since the models were not trained to distinguish between different stocks, but to forecast the mid price movement direction. Therefore, apart from predicting the stock mid price movement, the trained models were capable of learning the structure of the financial market, which can be then further analyzed by the domain experts (e.g., by examining stocks with similar behavior or studying outliers). For example, note that points that correspond to the insurance industry are well separated from the other industries, e.g., steel and energy industries, that are more interconnected. Furthermore, note that the high-frequency patterns can be independent from the long-term behavior of a stock, as they mostly depend on the short-term liquidity properties, e.g., the depth of the limit order book and the trading frequency. This is confirmed in the plots of Figure 8, since the two most frequently traded stocks (based on the average number of shares traded), i.e., the Outukumpu and the Sampo, are well separated from the other three, less frequently traded stocks.

Finally, the forecasting performance of the proposed method was also individually evaluated for the five different stocks and test splits (days). The results are shown in Table IX. Even though there are no significant differences between the different days, the proposed method was able to predict the behavior of some stocks more accurately. More specifically, the model was more accurate for the "Sampo", "Kesko" and "Outokumpu" stocks, which can be also attributed to the nature of the stocks. For example, the "Sampo" ($>$ 26,000,000 shares traded during June 2010) and the "Outokumpu" ($>$ 29,000,000 shares traded during June 2010) are among the most frequently traded stocks allowing the model to more easily learn the market's dynamics regarding these stocks. Therefore, the model was more prone to failure when predicting the behavior of stocks for which there were fewer trades and, as a result, less training data.

V. Conclusions

A temporal extensions of the BoF model was proposed in this paper. The proposed neural layer is composed of a set of RBF and accumulation layers, that can model both the short-term dynamics and the long-term behavior of time-series. The effectiveness of the proposed approach was validated using a large-scale limit order book dataset for predicting mid price movements. It was demonstrated that the proposed approach outperforms all the other evaluated baseline and competitive methods, while reducing the size and the complexity of the network over the regular BoF model. Finally, it was demonstrated that the Temporal BoF representation is capable of learning the dynamics of the financial markets, allowing the Temporal BoF to be used as a novel visualization tool that can provide further insight to domain experts.

There are many future research directions. First, the proposed method provides a powerful data analysis tool that can be also used for other tasks apart from time-series forecasting, e.g., interactive exploratory analysis [45]. Furthermore, the quantization process involved in the proposed BoF-based method has been proven to withstand distribution shifts more successfully than other models. This was also demonstrated in [46], where a BoF formulation was employed for multi-scale image classification using deep convolution neural networks. Using a multi-scale time-series-oriented training scheme will possibly allow the method to handle distribution shifts more effectively. Finally, recurrent models, e.g., [6], [11], can be employed and combined with the

proposed method to further increase the modeling capacity of the short-term and long-term histograms by providing a dynamic approach for accumulating the quantized feature vectors.

ACKNOWLEDGMENTS

The research project leading to these results received funding from the EU Research and Innovation Programme Horizon 2020 under grant agreement No. 675044 (BigDataFinance).

APPENDIX A

The long-term BoF derivatives are calculated as $\frac{\partial L}{\partial \mathbf{v}_{0m}} = \sum_{i=1}^{N}\sum_{k=1}^{N_K}\frac{\partial L}{\partial [\mathbf{h}_i]_{0k}}\frac{\partial [\mathbf{h}_i]_{0k}}{\partial \mathbf{v}_{0m}}$ and $\frac{\partial L}{\partial \mathbf{w}_{0m}} = \sum_{i=1}^{N}\sum_{k=1}^{N_K}\frac{\partial L}{\partial [\mathbf{h}_i]_{0k}}\frac{\partial [\mathbf{h}_i]_{0k}}{\partial \mathbf{w}_{0m}}$..

By defining $[\mathbf{d}_{ij}]_{0k} = exp(-||(\mathbf{x}_{ij} - \mathbf{v}_{0k}) \odot \mathbf{w}_{0k}||)$, the output of the RBF layer of the long-term block can be expressed as:$[\boldsymbol{\phi}(\mathbf{x}_{ij})]_{0k} = \frac{[\mathbf{d}_{ij}]_{0k}}{||[\mathbf{d}_{ij}]_0||_1}$.

The derivative that projects the MLP's gradients into the long-term RBF centers is calculated as: $\frac{\partial [\mathbf{h}_i]_{0k}}{\partial \mathbf{v}_{0m}} = \frac{1}{N_i^{(long)}}\sum_{j=1}^{N_i}\frac{\partial [\boldsymbol{\phi}(\mathbf{x}_{ij})]_k}{\partial [\mathbf{d}_{ij}]_{0m}}\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{v}_{0m}}$ where $N_i^{(long)}$ is the number of features fed to the long-term BoF neuron. Also, the derivative $\frac{\partial [\boldsymbol{\phi}(\mathbf{x}_{ij})]_{0k}}{\partial [\mathbf{d}_{ij}]_{0m}}$ is calculated as:

$$\frac{\partial [\boldsymbol{\phi}(\mathbf{x}_{ij})]_{0k}}{\partial [\mathbf{d}_{ij}]_{0m}} = \frac{\delta_{km}}{||[\mathbf{d}_{ij}]_0||_1} - \frac{[\mathbf{d}_{ij}]_{0k}}{||[\mathbf{d}_{ij}]_0||_1^2}, \tag{12}$$

and $\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{v}_{0m}} = [\mathbf{d}_{ij}]_{0m}\frac{(\mathbf{x}_{ij}-\mathbf{v}_{0m})\odot \mathbf{w}_{0m}}{||(\mathbf{x}_{ij}-\mathbf{v}_{0m})\odot \mathbf{w}_{0m}||_2} \odot \mathbf{w}_{0m}$, where $\delta_{km} = 1$ if $m = k$, or $\delta_{km} = 0$ otherwise.

The derivative of the RBF weights is similarly calculated as:

$$\frac{\partial [\mathbf{s}_i]_{0k}}{\partial \mathbf{w}_{0m}} = \frac{1}{N_i^{(long)}}\sum_{j=1}^{N_i}\frac{\partial [\boldsymbol{\phi}(\mathbf{x}_{ij})]_{0k}}{\partial [\mathbf{d}_{ij}]_{0m}}\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{w}_{0m}} s_{ij}^{(long)}, \tag{13}$$

where $\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{w}_{0m}} = -[\mathbf{d}_{ij}]_{0m}\frac{(\mathbf{x}_{ij}-\mathbf{v}_{0m})\odot \mathbf{w}_{0m}}{||(\mathbf{x}_{ij}-\mathbf{v}_{0m})\odot \mathbf{w}_{0m}||_2} \odot (\mathbf{x}_{ij} - \mathbf{v}_{0m})$.

Finally, note that the derivatives $\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{v}_{0m}}$ and $\frac{\partial [\mathbf{d}_{ij}]_{0m}}{\partial \mathbf{w}_{0m}}$ do not exist when a feature vector and a codeword coincides. In this case, the corresponding derivatives are set to 0. Also, the derivatives of the short-term BoF neurons can be similarly calculated. The rest of the derivatives, i.e., the MLP derivatives $\frac{\partial L}{\partial \mathbf{W}_{MLP}}$, are straightforward to calculate.

REFERENCES

[1] L.-J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506–1518, 2003.
[2] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, 2001.
[3] R. Cont, "Statistical modeling of high-frequency financial data," *IEEE Signal Processing Magazine*, vol. 28, no. 5, pp. 16–25, 2011.
[4] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining Workshop*, vol. 10, no. 16, 1994, pp. 359–370.
[5] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the International Conference on Machine Learning*, 2006, pp. 1033–1040.
[6] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint 1511.03677*, 2015.
[7] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.
[8] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.
[9] A. Iosifidis, A. Tefas, and I. Pitas, "Multidimensional sequence classification based on fuzzy distances and discriminant analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2564–2575, 2013.
[10] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "Music classification via the bag-of-features approach," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1768–1777, 2011.
[11] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in *Proceedings of the European Signal Processing Conference*, 2017, pp. 2511–2515.
[12] N. Passalis and A. Tefas, "Entropy optimized feature-based bag-of-words representation for information retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1664–1677, 2016.
[13] J. Sivic, A. Zisserman *et al.*, "Video google: A text retrieval approach to object matching in videos." in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, no. 1470, 2003, pp. 1470–1477.
[14] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proceedings of the ACM International Conference on Image and Video Retrieval*, 2007, pp. 494–501.
[15] A. N. Kercheval and Y. Zhang, "Modelling high-frequency limit order book dynamics with support vector machines," *Quantitative Finance*, vol. 15, no. 8, pp. 1315–1329, 2015.
[16] A. Bailly, S. Malinowski, R. Tavenard, T. Guyet, and L. Chapel, "Bag-of-temporal-sift-words for time series classification," in *Proceedings of the ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2015.
[17] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.
[18] X. Wang, B. Wang, X. Bai, W. Liu, and Z. Tu, "Max-Margin Multiple-Instance Dictionary Learning," in *Proceedings of the International Conference on Machine Learning*, 2013, pp. 846–854.
[19] N. Passalis and A. Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.
[20] J. Sirignano, "Deep learning for limit order books," *arXiv preprint 1601.01987*, 2016.
[21] B. Park and B. Van Roy, "Adaptive execution: Exploration and learning of price impact," *Operations Research*, vol. 63, no. 5, pp. 1058–1076, 2015.

[22] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the International Conference on Machine Learning*, 2006, pp. 673–680.

[23] D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2017, pp. 1–7.

[24] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention augmented bilinear network for financial time-series data analysis," *arXiv preprint arXiv:1712.00975*, 2017.

[25] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proceedings of the IEEE Conference on Business Informatics*, vol. 1, 2017, pp. 7–12.

[26] S. Lazebnik and M. Raginsky, "Supervised learning of quantizer codebooks by information loss minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1294–1309, 2009.

[27] X.-C. Lian, Z. Li, B.-L. Lu, and L. Zhang, "Max-margin dictionary learning for multiclass image categorization," in *Proceedings of the European Conference on Computer Vision*, 2010, pp. 157–170.

[28] H. Lobel, R. Vidal, D. Mery, and A. Soto, "Joint dictionary and classifier learning for categorization of images using a max-margin framework," in *Image and Video Technology*, 2014, pp. 87–98.

[29] M. Jiu, C. Wolf, C. Garcia, and A. Baskurt, "Supervised learning and codebook optimization for bag-of-words models," *Cognitive Computation*, vol. 4, no. 4, pp. 409–419, 2012.

[30] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2559–2566.

[31] A. Iosifidis, A. Tefas, and I. Pitas, "Discriminant bag of words based representation for human action recognition," *Pattern Recognition Letters*, vol. 49, pp. 185–192, 2014.

[32] N. Passalis, A. Tsantekidis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Time-series classification using neural bag-of-features," in *Proceedings of the European Signal Processing Conference*, 2017, pp. 301–305.

[33] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint 1511.07289*, 2015.

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

[35] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint 1312.6120*, 2013.

[36] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price prediction of limit order book data," *arXiv preprint arXiv:1705.03233*, 2017.

[37] E. Tomasini and U. Jaekle, *Trading Systems*. Harriman House Limited, 2011.

[38] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[39] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[40] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010.

[41] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[42] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Proceedings of the IEEE Signal Processing Society Workshop*, 1999, pp. 41–48.

[43] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[44] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[45] D. Cook and D. F. Swayne, *Interactive and dynamic graphics for data analysis: with R and GGobi*. New York: Springer Science & Business Media, 2007.

[46] N. Passalis and A. Tefas, "Bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Nikolaos Passalis obtained his B.Sc. in informatics in 2013 and his M.Sc. in information systems in 2015 from Aristotle University of Thessaloniki, Greece. He is currently pursuing his PhD studies in the Artificial Intelligence & Information Analysis Laboratory in the Department of Informatics at the University of Thessaloniki. His research interests include machine learning, computational intelligence and information retrieval.

Anastasios Tefas received the B.Sc. in informatics in 1997 and the Ph.D. degree in informatics in 2002, both from the Aristotle University of Thessaloniki, Greece. Since 2017 he has been an Associate Professor at the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2017, he was a Lecturer, Assistant Professor at the same University. From 2006 to 2008, he was an Assistant Professor at the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. Dr. Tefas participated in 12 research projects financed by national and European funds. He has co-authored 91 journal papers, 203 papers in international conferences and contributed 8 chapters to edited books in his area of expertise. Over 3730 citations have been recorded to his publications and his H-index is 32 according to Google scholar. His current research interests include computational intelligence, deep learning, pattern recognition, statistical machine learning, digital signal and image analysis and retrieval and computer vision.

Juho Kanniainen is Professor of Financial Engineering at the Tampere University of Technology, Finland. His research agenda is focused on derivative pricing, financial econometrics, order book dynamics and liquidity, and financial networks, with emphasis on big data problems. Dr. Kanniainen has published in many journals in Finance and Engineering, including Review of Finance, Journal of Banking and Finance, Scientific Reports, and IEEE Transactions on Neural Networks and Learning Systems. He has been coordinating two international EU projects, BigDataFinance (www.bigdatafinance.eu) and HPCFinance (www.hpcfinance.eu).

Moncef Gabbouj (F'11) received the B.S. degree in electrical engineering from Oklahoma State University, Stillwater, OK, USA, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1986 and 1989, respectively.

He is a Professor of Signal Processing with the Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland. He was an Academy of Finland Professor from 2011 to 2015. He guided 40 Ph.D. students and published 650 papers. His current research interests include multimedia content-based analysis, indexing and retrieval, machine learning, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding.

Dr. Gabbouj is the past Chairman of the IEEE CAS TC on DSP and a Committee Member of the IEEE Fourier Award for Signal Processing. He served as an Associate Editor and a Guest Editor of many IEEE, and international journals and a Distinguished Lecturer for the IEEE CASS. He organized several tutorials and special sessions for major IEEE conferences and EUSIPCO. He is a member of the Academia Europaea and the Finnish Academy of Science and Letters.

Alexandros Iosifidis (SM'16) is an Assistant Professor of Machine Learning and Computer Vision in the Department of Engineering, Electrical and Computer Engineering, at Aarhus University, Denmark. He has held Postdoctoral Researcher positions at Tampere University of Technology, Finland and Aristotle University of Thessaloniki, Greece. He received many prestigious awards, including the Academy of Finland Postdoc Fellowship and the H.C. Ørsted Forskerspirer Prize for research excellence at a young age. He has contributed in many R&D projects financed by EU, Greek, Finnish, and Danish funding agencies and companies. He has co-authored 48 articles in international journals and 70 papers in international conferences proposing novel Machine Learning techniques and their application in a variety of problems. His work has received 1200+ citations with h-index 18+ (Google Scholar).

Dr. Iosifidis is a Senior Member of IEEE, a member of EURASIP, IAPR and INNS. He served as an Officer of the Finnish IEEE Signal Processing-Circuits and Systems Chapter, and he is currently serving as an Area Editor of Signal Processing: Image Communication journal and Associate Editor in IEEE Access and Neurocomputing journals. He has co-organized Special Issues/Sessions in International Journals/Conferences focusing on Machine Learning approaches for Big Data Analysis.