Time-series Classification Using Neural Bag-of-Features

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

***Abstract*—Classification of time-series data is a challenging problem with many real-world applications, ranging from identifying medical conditions from electroencephalography (EEG) measurements to forecasting the stock market. The well known Bag-of-Features (BoF) model was recently adapted towards time-series representation. In this work, a neural generalization of the BoF model, composed of an RBF layer and an accumulation layer, is proposed as a neural layer that receives the features extracted from a time-series and gradually builds its representation. The proposed method can be combined with any other layer or classifier, such as fully connected layers or feature transformation layers, to form deep neural networks for time-series classification. The resulting networks are end-to-end differentiable and they can be trained using regular back-propagation. It is demonstrated, using two time-series datasets, including a large-scale financial dataset, that the proposed approach can significantly increase the classification metrics over other baseline and state-of-the-art techniques.**

I. Introduction

Classification of time-series data is a challenging problem with many real-word applications, ranging from identifying medical conditions from electroencephalography (EEG) measurements [1], to forecasting the stock market [2]. Time-series data are sequences of observations (usually real-valued numbers) sampled at specific (uniform or non-uniform) intervals. Note that the observations can be either one-dimensional, e.g., the price of a stock, or multi-dimensional, e.g., the measurements of the multiple electrodes of an EEG device.

A wide range of techniques have been proposed to classify time-series data [3]–[8]. Some of them rely on a distance metric, such as the Dynamic Time Wrapping, to measure the similarity between different time-series and then use simple machine learning models, such as the k-nn, to classify the data [3]. Other techniques use recurrent neural networks [4], [5], and hidden Markov models [6], to tackle the problem of time-series classification. In this work we focus on a recently proposed model for time-series representation and classification, the Bag-of-Features model [7]–[9].

The Bag-of-Features model (BoF), also known as Bag-of-Visual Words (BoVW), was originally proposed for image representations tasks [10]. However, it was established that it can be used for a wide range of tasks, such as video [11], and time-series representation [7]–[9]. The BoF pipeline can be summarized as follows. First, multiple features, e.g., SIFT descriptors, are extracted from each object, e.g., an image. This step is called *feature extraction*. That way, the feature space is formed where each object is represented as a set of features. Then, in the *dictionary learning* step, the extracted features are used to learn a dictionary/codebook of representative features (also called codewords). Finally, the *feature quantization and encoding* step follows, in which each feature vector is represented using a codeword from the learned codebook and a histogram is extracted for each object. That way, the histogram space is formed where each object is represented by a constant dimensionality histogram vector.

The BoF model allows for creating constant length representations of time-series regardless their length. This representation captures the dynamics of the time-series behavior and it can be used with any classifier without having to deal with the raw data. However, to use the BoF model for time-series representation a feature extractor must be utilized for extracting feature vectors from each time-series and proceeding with the aforementioned BoF pipeline. Several options exists for the feature extraction step. For example, each raw (multi-dimensional) time-series point can be considered as a feature vector [8], short intervals of various lengths can be used to allow for handling time warping [7], or handcrafted features can be extracted from several points of the time-series [12].

The learned BoF representation also critically relies on the dictionary learning step. The early BoF approaches (e.g., [10]) used clustering algorithms, such as k-means, to learn generic dictionaries that minimize the reconstruction loss of the quantized features. These dictionaries were not optimized for a specific task. However, it was later established that supervised dictionary learning (e.g., [8], [13]–[15]), which optimizes the dictionary towards a specific task, performs significantly better.

The contributions of this work are briefly summarized

ISBN 978-0-9928626-7-1

bellow. A neural generalization of the BoF model, composed of an Radial Basis Function (RBF) layer and an accumulation layer, is proposed as a neural layer that receives the features extracted from a time-series and gradually builds its representation. This layer, called Neural Bag-of-Features, can be combined with any other layer or classifier, such as fully connected layers or feature transformation layers, to form deep neural networks for time-series classification. It is demonstrated, using two time-series datasets, including a large-scale financial dataset, that the proposed approach can significantly increase the classification metrics, as well as, reduce the size of the extracted representation compared to the regular BoF model.

The rest of the paper is structured as follows. In Section 2 the related work is introduced and compared to the proposed Neural BoF model. The proposed method is presented in detail in Section 3. Next, the Neural BoF is evaluated using two different datasets (Section 4). Finally, conclusions are drawn in Section 5.

II. Related Work

This work concerns mainly dictionary learning for the BoF model, as well as time-series representation and classification using the BoF model.

There is a rich literature on dictionary learning approaches for the BoF model. In [16], multiple maximum margin hyperplanes are learned and at the same time the codebooks are adjusted to maximize the corresponding margins. A simple method for supervised codebook learning that incorporates both a traditional MLP layer and a codebook layer is proposed in [17], while a full neural formulation is presented in [15]. In [18], the optimization aims to minimize the logistic regression loss, while the representation is optimized using an LDA-based criterion in [13]. In [19], multiple dictionaries with complementary discriminative information are learned by adjusting the weights used during the clustering process using the predictions of a classifier. However, these approaches are oriented towards image classification instead of time-series classification.

In [8], the BoF model is used for time-series representation and the dictionary is optimized using a discriminative loss function. In [9], a similar BoF-based approach is combined with a retrieval-oriented loss function to optimize the representation towards retrieval. Also, in [7], time-series segments of various lengths are used during the feature extraction, which allows for handling warping, and the output of a Random Forest classifier is used to alter the codebook. To the best of our knowledge the proposed method is the first that allows for the end-to-end optimization of a Neural BoF model towards time-series classification.

III. Proposed Method

The proposed method for classifying time-series data using the Neural BoF model (also abbreviated as N-BoF) is presented in this Section. First, a feature extractor is used to extract multiple feature vectors from each time-series. Then, these features are fed to the Neural BoF layer that extracts a constant-length representation for each time-series. This representation is subsequently fed to the used classifier. End-to-end-training is used to learn all the parameters of the proposed unified architecture.

Several options exist for the feature extraction step depending on the nature of the time-series. For example, each raw multi-dimensional point of the time-series can be considered as a separate feature vector, e.g., the raw multi-channel measurements from an EEG device can be used. On the other hand, hand-crafted features can be also utilized. For example, if limit order book data are used, then the feature extraction procedure proposed in [12], can be used to extract a 144-dimensional feature vector from each block of 100 order book records. It should be also noted that it is straightforward to combine the proposed approach with trainable recurrent feature extractors, such as Gated Recurrent Units (GRUs) [5], to further increase the ability of the proposed method to capture complex dynamic phenomena. In this work, all the extracted feature vectors were normalized to have unit variance and zero mean (z-score standardization).

After the feature extraction the i-th time-series is represented by a set of N_i feature vectors $\mathbf{x}_{ij} \in \mathbb{R}^D$ $(j = 1...N_i)$, where D is the dimensionality of the extracted features. Note that the length of the extracted histogram vector does not depend on the number of available feature vectors, which allows the network to handle time-series of any length without any modification.

The BoF model is formulated as a neural layer that is composed of two sublayers, i.e., an RBF layer that measures the similarity of the input features to the RBF centers and an accumulation layer that builds the histogram of the quantized feature vectors. The proposed Neural BoF layer can be thought as a unified processing layer that feeds the extracted representation to a subsequent classifier.

The output of the k-th RBF neuron is defined as:

$$[\boldsymbol{\phi}(\mathbf{x}_{ij})]_k = \exp(-||(\mathbf{x}_{ij} - \mathbf{v}_k) \odot \mathbf{w}_k||_2) \tag{1}$$

where $\odot$ is the element-wise multiplication operator. The RBF neurons behave somewhat like the codewords in the BoF model, i.e., they are used to measure the similarity of the input vectors to a set of predefined vectors. Each RBF neuron is also equipped with a weight vector $\mathbf{w}_k \in \mathbb{R}^D$ that adjusts the width of its Gaussian function per each dimension. That allows for better modeling of the input distribution, since the distribution modeled by each RBF can be independently adjusted. Also, note that by zeroing some of the input weights the length of each codeword can be reduced. The number of the RBF neurons used is denoted by N_K. The size of the extracted representation can be adjusted by using a different number of RBF neurons (N_K) in the Neural BoF layer.

To ensure that the output of each RBF neuron is bounded, a normalized RBF architecture is used. This normalization is equivalent to the l^1 scaling that is utilized in the BoF model

ISBN 978-0-9928626-7-1 © EURASIP 2017

that uses soft-assignments [9]. Thus, the output of the RBF neurons is re-defined as:

$$[\boldsymbol{\phi}(\mathbf{x_{ij}})]_k = \frac{\exp(-||(\mathbf{x}_{ij} - \mathbf{v}_k) \odot \mathbf{w}_k||_2)}{\sum_{m=1}^{N_K} \exp(-||(\mathbf{x}_{ij} - \mathbf{v}_m) \odot \mathbf{w}_m||_2)} \quad (2)$$

The output of the RBF neurons is accumulated in the next layer, compiling the final representation of each time-series:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{\phi}(\mathbf{x}_{ij}) \quad (3)$$

where $\boldsymbol{\phi}(\mathbf{x}_{ij}) = ([\boldsymbol{\phi}(\mathbf{x}_{ij})]_1, ..., [\boldsymbol{\phi}(\mathbf{x}_{ij})]_{N_K})^T \in \mathbb{R}^{N_K}$ is the output vector of the RBF layer. Note that each $\mathbf{s}_i$ has unit l^1 norm, defines a histogram distribution over the RBF neurons and describes the dynamics of each time-series.

The Neural BoF layer receives the feature vectors of a time-series and compiles its histogram representation. Then, this histogram must be fed to a classifier that decides the class of the time-series. In this work a multilayer perceptron (MLP) with one hidden layer is used for this purpose, although any other classifier with differentiable loss function can be used.

Let t_i be the label of the i-the training time-series sample and N_C be the number of the different labels. Also, let $\mathbf{W}_H \in \mathbb{R}^{N_H \times N_K}$ be the hidden layer weights and $\mathbf{W}_O \in \mathbb{R}^{N_C \times N_H}$ be the output layer weights, where N_H is the number of hidden neurons. Then, the hidden layer activations for the input histogram $\mathbf{s}_i$ of the i-th time-series are computed as $\mathbf{h}_i = \phi^{(elu)}(\mathbf{W}_H \mathbf{s}_i + \mathbf{b}_H) \in \mathbb{R}^{N_H}$, where $\phi^{(elu)}(x)$ is the *elu* activation function [20]:

$$\phi^{(elu)}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha_{elu}(exp(x) - 1), & \text{if } x \leq 0 \end{cases} \quad (4)$$

which is applied element-wise and $\mathbf{b}_H \in \mathbb{R}^{N_H}$ is the hidden layer bias vector. For all the conducted experiments, a hidden layer with 512 neurons was used. Similarly, the output of the MLP is calculated as:

$$\mathbf{y}_i = \phi^{(softmax)}(\mathbf{W}_O \mathbf{h}_i + \mathbf{b}_O) \in \mathbb{R}^{N_C} \quad (5)$$

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

The categorical cross entropy loss is used for training the network:

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{N_C} [\mathbf{t}_i]_j \log([\mathbf{y}_i]_j) \quad (6)$$

where $\mathbf{t}_i \in \mathbb{R}^{N_C}$ is the target output vector, which depends on the label (t_i) of the input time-series and it is defined as: $[\mathbf{t}_i]_j = 1$, if $j = t_i$, or $[\mathbf{t}_i]_j = 0$, otherwise. All the layers of the proposed network can be trained using back-propagation and gradient descent:

$$\Delta(\mathbf{W}_{MLP}, \mathbf{v}_k, \boldsymbol{w}_k) = -(\eta_{MLP} \frac{\partial L}{\partial \mathbf{W}_{MLP}}, \eta_V \frac{\partial L}{\partial \mathbf{v}_k}, \eta_W \frac{\partial L}{\partial \boldsymbol{w}_k}) \quad (7)$$

where the notation $\mathbf{W}_{MLP}$ is used to refer to the parameters of the classification layer. Instead of using simple gradient descent, a recently proposed method for stochastic optimization, the Adam (Adaptive Moment Estimation) algorithm [21], is utilized for learning the parameters of the network . Note that if trainable recurrent feature extractors are used, the gradients can back-propagate to them, further fine-tuning the feature extraction process towards the classification task at hand. The learning rates used for the conducted experiments were set to $\eta_{MLP} = \eta_V = 0.001$ and $\eta_W = 0.01$ and the default hyper-parameters were used for the Adam algorithm [21]. The optimization ran for 5000 iterations using batch size of 32 samples (1000 iterations were used for the EEG dataset). Also, to avoid back-propagating gradients from a randomly initialized MLP to the Neural BoF layer, the MLP was pre-trained for the first 500 iterations (100 iterations for the EEG dataset).

The centers of the RBF neurons can be either randomly chosen or initialized using the k-means algorithm over the set of all feature vectors $\mathcal{S} = \{\mathbf{x}_{ij} | i = 1...N, j = 1...N_i\}$. In this work, the set $\mathcal{S}$ is clustered into N_K clusters and the corresponding centroids (codewords) $\mathbf{v}_k \in \mathbb{R}^D (k = 1...N_K)$ are used to initialize the centers of the RBF neurons. The same approach is used in the BoF model to learn the codebook that is used to quantize the feature vectors. However, in contrast to the BoF model, the Neural BoF model uses this process only to initialize the centers. The RBF weight vectors are also initialized to $1/g$, where g is a small positive number ranging from 0.1 to 10. The value of g is selected using a validation set (by splitting the training data into a new training set and validation set). Both the RBF centers ($\mathbf{v}_k$) and the RBF input weights ($\mathbf{w}_k$) are learned using back-propagation.

IV. Experiments

We evaluated the proposed Neural BoF model in two time series problems, those of high-frequency stock market trading and EEG signal classification. Information related to the datasets and experimental protocols used are provided in the following, along with experimental results comparing the proposed model with existing ones.

A. High-frequency stock market trading

In our first set of experiments, we employed a new large-scale dataset, called FI-2010 in this paper, consisting of limit order book data collected from 5 Finnish stocks. The dataset is provided by Nasdaq Nordic [22], [23]. A limit order in equity markets is an order to buy (sell) a specified about of shares at a specified bid price (ask price) or better. For each time-step the prices and volumes of limit orders at the 10 best bid and 10 best ask levels are collected. After collecting data for 10 business days (from the 1st to the 14th June 2010) and pre-processing and cleaning them (e.g., removing unnecessary event messages) the handcrafted features described in [13] were extracted. The total number of the collected limit order events is 4.5 million, leading to 453,975 144-dimensional extracted feature vectors (one feature vector is extracted every

ISBN 978-0-9928626-7-1 © EURASIP 2017

TABLE I
FI-2010: ANCHORED EVALUATION FOR PREDICTING THE MID-PRICE DIRECTION FOR THE NEXT 10, 50 AND 100 TIMESTEPS.

Method	Predict Target	Accuracy	Precision	Recall	F1	Cohen's κ
BoF	10	57.59 ± 7.34	39.26 ± 0.94	51.44 ± 2.53	36.28 ± 2.85	0.1182 ± 0.0246
N-BoF	10	$\mathbf{62.70 \pm 6.73}$	$\mathbf{42.28 \pm 0.87}$	$\mathbf{61.41 \pm 3.68}$	$\mathbf{41.63 \pm 1.90}$	$\mathbf{0.1724 \pm 0.0212}$
BoF	50	50.21 ± 5.59	42.56 ± 1.26	49.57 ± 2.28	39.56 ± 2.36	0.1576 ± 0.0254
N-BoF	50	$\mathbf{56.52 \pm 8.67}$	$\mathbf{47.20 \pm 1.80}$	$\mathbf{58.17 \pm 2.61}$	$\mathbf{46.15 \pm 4.07}$	$\mathbf{0.2285 \pm 0.0419}$
BoF	100	50.97 ± 5.62	42.89 ± 1.46	47.84 ± 2.08	40.84 ± 2.78	0.1641 ± 0.0300
N-BoF	100	$\mathbf{56.43 \pm 6.86}$	$\mathbf{47.27 \pm 1.72}$	$\mathbf{54.99 \pm 2.19}$	$\mathbf{46.86 \pm 2.87}$	$\mathbf{0.2300 \pm 0.0338}$

100 limit orders). Performance evaluation was performed as follows. For every time-step and given the last 15 feature vectors, we predict the direction (up, stationary or down) of the mean mid price, i.e., the average of the highest bid and the lowest ask prices, after k time steps. The mid prices were filtered using a moving average filter with window size 9. The prediction targets were set for the next 10, 50 and 100 time-steps, which correspond to the next 1, 5 and 10 feature vectors. For each of them, the threshold for considering the stock stationary was set to 0.01%, 0.02%, and 0.03% respectively (as the prediction horizon increases, the changes in the mid-price are becoming larger). Finally, an anchored evaluation setup was used [24], i.e., the first day was used for training the model and the next for testing, then the first two days were used for training the model and the next for testing, etc. The mean and the standard deviation of the evaluated metrics are reported for the 9 anchored evaluation splits.

Note that the FI-2010 dataset is highly unbalanced (most of the time the mid-price remains stationary). Therefore, during the training the data from the less frequent classes were fed to network with an increased to probability to ensure that each class is equally represented. Also, the *accuracy*, the *average precision per class* (macro precision), the *average recall per class* (macro recall), the *average F1 per class* (macro F1) [24], and the *Cohen's* κ metric [25] were used to evaluate the performance of the algorithms. The accuracy measures the percentage of the predicated labels that exactly match the ground truth. The precision is defined as the ratio of the true positives over the sum of the true positives and the false positives, while the recall as the ratio of the true positives over the sum of the true positives and the false negatives. The F1 score is defined as the harmonic mean of the precision and the recall. The Cohen's κ is a metric that allows us to evaluate the agreement between two different sets of annotations, taking into account the expected agreement when a random classifier is used. Note that the F1 and the Cohen's κ allow for evaluating the performance of algorithms when unbalanced datasets, like the FI-2010, are used. For all the evaluated metrics, higher values indicate better performance of the corresponding aspect of the evaluated method.

In Figure 1, the effect of varying the number of RBF neurons on the Cohen's κ metric is evaluated using a validation set (the last two days of the FI-2010 data were used as validation set). The proposed Neural BoF model outperforms the regular BoF model for any number of codewords, allowing us to use significantly smaller representations while increasing the discriminative abilities of the classifier. In Table 1, the regular BoF method is compared to the proposed Neural BoF for different prediction targets. For the regular BoF method 128 codewords are used, while for the Neural BoF 16 RBF neurons are utilized. Using the proposed Neural BoF method significantly increases all the evaluation metrics, while decreasing the size of the extracted representation.

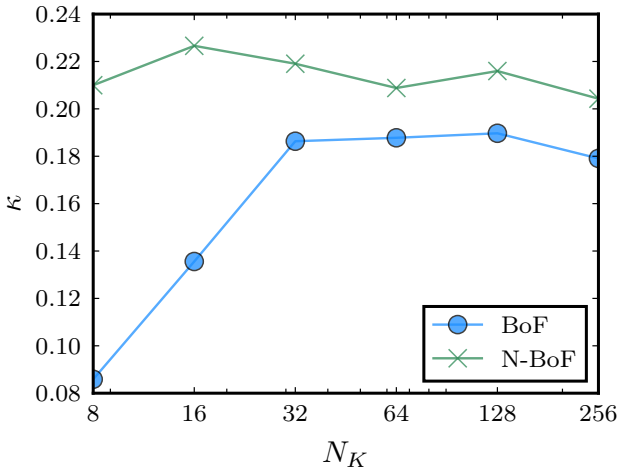

Fig. 1. Effect of varying the number of codewords / RBF neurons (N_K) on the Cohen's κ metric (FI-2010 dataset)

B. EEG signal classification

In our second experiment, we employed the EEG Database [26], consisting of EEG measurements (64 channels) sampled at 256Hz for 1 seconds. The data were collected from two groups, i.e., alcoholics and control. The provided training set consists of 10 alcoholic subjects and 10 control subjects, while the test set consists of 10 out-of-sample runs of the same subjects. More details regarding the data collection protocol can be found in [26].

In this dataset, we compare the proposed approach to other baselines and state-of-the-art methods. The results are shown in Table II. Only the classification accuracy and the κ metric are reported since the dataset is balanced and the classification problem binary. The proposed method is compared to a simple MLP with 512 hidden units that receives an input of 256×64. The method was also compared to a GRU model with 256 hidden units, that is among the state-of-the-art techniques for

ISBN 978-0-9928626-7-1 © EURASIP 2017

TABLE II
EEG DATABASE EVALUATION

Method	Class. Accuracy	Cohen's κ
MLP	71.67%	0.43
GRU	80.50%	0.61
BoF	67.33%	0.35
N-BoF	**86.66%**	**0.73**

time-series classification. For both models the *tanh* activation function was used for the hidden layer and they were trained using the categorical cross-entropy loss. Both the BoF and the Neural BoF model use $N_K = 8$ codewords. As before, the proposed Neural BoF model significantly outperforms all the other evaluated methods.

V. CONCLUSION

In this paper the BoF model was generalized and formulated as a neural layer composed of an RBF layer and an accumulation layer that can be used for classifying time-series data. The proposed method can be combined with any other shallow or deep classifier to form powerful time-series classification machines. Two different datasets were used to evaluate the proposed method. It was demonstrated that the proposed method can greatly increase the classification accuracy, while reducing the size of the extracted representations compared to the regular BoF model.

The proposed method can be combined with trainable recurrent feature extractors, such as GRUs [5], to more accurately model the dynamics of time-series. Also, two different sets of RBF neurons can be used to model the short-term and the long-term behavior of the time-series. Preliminary experiments show that this can further increase the classification metrics, especially for the FI-2010 dataset, where both the long-term and the short term behavior of the time-series is important. Finally, unsupervised manifold-based optimization, e.g,. [27], can be used to learn representatons optimized toward exploratory tasks, such as clustering.

ACKNOWLEDGMENT

The research leading to these results has received funding from the H2020 Project BigDataFinance MSCA-ITN-ETN 675044 (http://bigdatafinance.eu), Training for Big Data in Financial Research and Risk Management. Alexandros Iosifidis was supported from the Academy of Finland Postdoctoral Research Fellowship (No. 295854). He joined Aarhus University on August 2017.

REFERENCES

[1] W. A. Chaovalitwongse, O. A. Prokopyev, and P. M. Pardalos, "Electroencephalogram (eeg) time series classification: Applications in epilepsy," *Annals of Operations Research*, vol. 148, no. 1, pp. 227–250, 2006.

[2] 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.

[3] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *ACM Conference on Knowledge Discovery and Data Mining Workshop*, vol. 10, no. 16, 1994, pp. 359–370.

[4] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[5] R. Józefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 2342–2350.

[6] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] N. Passalis and A. Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.

[16] 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.

[17] 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.

[18] 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.

[19] W. Zhang, A. Surve, X. Fern, and T. Dietterich, "Learning non-redundant codebooks for classifying complex objects," in *Proceedings of the Annual International Conference on Machine Learning*, 2009, pp. 1241–1248.

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

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

[22] M. Siikanen, J. Kanniainen, and J. Valli, "Limit order books and liquidity around scheduled and non-scheduled announcements: Empirical evidence from nasdaq nordic," *Finance Research Letters*, vol. to appear, 2016.

[23] 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.

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

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

[26] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/EEG+Database

[27] N. Passalis and A. Tefas, "Information clustering using manifold-based optimization of the bag-of-features representation," *IEEE Transactions on Cybernetics*, vol. to appear, 2016.