Signal/Background Classification of Time Series for Biological Virus Detection*

Dominic Siedhoff, Hendrik Fichtenberger, Pascal Libuschewski, Frank Weichert, Christian Sohler, and Heinrich Müller

TU Dortmund, Germany

This work proposes translation-invariant features based on a wavelet transform that are used to classify time series as containing either relevant signals or noisy background. Due to the translation-invariant property, signals appearing at arbitrary locations in time have similar representations in feature space. Classification is carried out by a condensed k-Nearest-Neighbors classifier trained on these features, i.e. the training set is reduced for faster classification. This reduction is conducted by a k-means clustering of the original training set and using the obtained cluster centers as a new training set. The coreset-technique BICO is employed to accelerate this initial clustering for big datasets. The resulting feature extraction and classification pipeline is applied successfully in the context of biological virus detection. Data from Plasmon Assisted Microscopy of Nano-size Objects (PAMONO) is classified, achieving accuracy 0.999 for the most important classification task.

1 Introduction

Methods for reliable detection of biological viruses by means of inexpensive sensor devices have gained an increasing interest in research. The PAMONO method (Plasmon Assisted Microscopy of Nano-size Objects) [18] enables construction of a biosensor which is capable of indirectly detecting nano-sized particles, including fine dust and biological viruses, using inexpensive components from microscopy. It has potential applications in diagnostics as well as pharmaceutical research because virus-antibody-bindings can be detected, enabling to investigate the ability of antibodies to bind certain viruses.

*The final authenticated version is available online at https://doi.org/10.1007/978-3-319-11752-2_31.

The sensor produces a time series of 2D images, constituting a large spatio-temporal volume of data (e.g. 4000 images with 1080×145 pixels in one measurement), hence automatic analysis is desirable. One crucial step in this analysis is the separation of pixels that are affected by a nano-particle adhesion (signal) from those that are not (background). This separation can be carried out by examining the time series of intensities measured at the pixels because they exhibit characteristic patterns (cf. Figure 1): When a particle appears, pixels in the center of the particle exhibit an up signal in the time series and pixels around it exhibit a down signal. These two signals indicate a particle adhesion. Regions not affected by particles contain only background noise.

This work proposes wavelet-based features of the time series from which a fast condensed k-Nearest-Neighbors (k-NN) [9] classifier is learned, to separate particle signals from background. Since particles may appear at any point in time, the features need to be translation-invariant, such that signals are similar in feature space, independent of the time the particle appears.

While translation-invariant (TI) wavelets have been developed for denoising in [4], their application for feature computation is limited [14]. In face recognition, non-redundant Discrete Wavelet Transform (DWT) was used as the basis for computing estimates of the power spectrum in local windows. These estimates served as approximately TI features [14]. In contrast, [11] use raw coefficient values as features and ensure the TI property by redundant DWT and signal registration. Our approach combines the use of TI coefficients with feature extraction. This serves the purposes of dimensionality reduction and increases robustness to noise.

Furthermore, to speed up learning and classification, the idea of condensing the training set by removing redundancy and similarities [7, 1, 2] is explored. We extend an approach from text classification [17] that clusters the training data and uses cluster centers as a condensed training set. In order to reduce the time taken for clustering, we apply the BICO algorithm [6] to reduce the input to a smaller coreset, to which weighted clustering is applied. Finally the cluster centers serve as the training set of a k-NN learner.

The remainder of this work is organized as follows: Section 2 describes the extraction of translation-invariant features from wavelet coefficients. Section 3 covers the training procedure for accelerated k-NN by means of a fast coreset-based clustering that condenses the training set. Section 4 demonstrates empirical results of the presented methods as attained for the PAMONO biosensor application. Section 5 provides discussion and future work.

2 Translation-invariant Feature Extraction

Translation-invariant (TI) features are desirable in PAMONO time series classification because they make the feature space representation of a series independent of the point in time where the signal (up or down, cf. Figure 1) appears. PAMONO time series approximately fulfil the circularity condition assumed for the TI wavelets proposed by Coifman and Donoho [4] utilized here: The signal to be detected has finite support at a

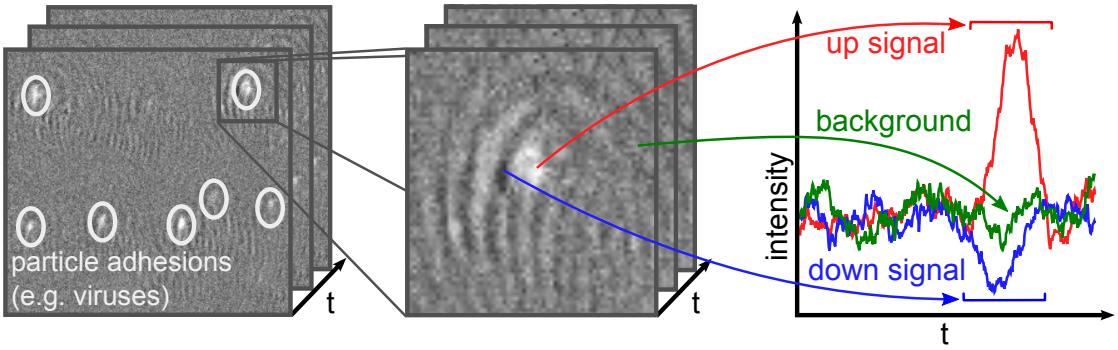
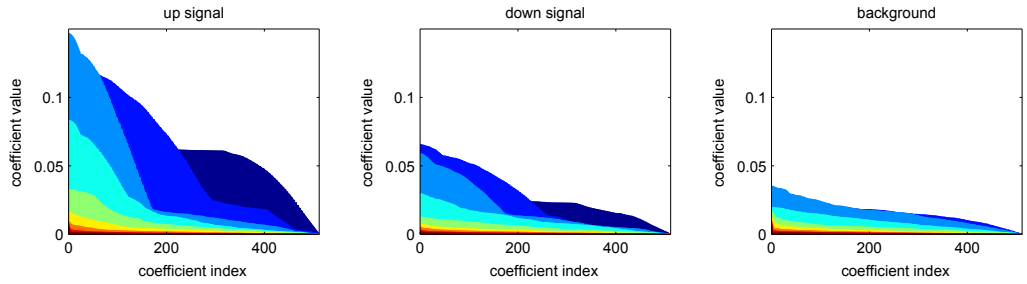


Figure 1: Time series of PAMONO images (*left*) showing seven nano-particles. Magnification of a single particle (*center*) and per pixel time series (*right*), characterizing pixels as belonging to a particle (up and down signal) or to background noise



(a) (b) (c)

Figure 2: Examples of $\mathbf{W}$ for time series from the three classes. Scales (i.e. rows of $\mathbf{W}$) are ordered coarsest (back) to finest (front). Positions on the x axis are column indices in $\mathbf{W}$. Classes differ in coefficient magnitude and slope, especially for coarser scales, which is exploited by the features in equations (1)–(8)

random, irrelevant location, preceded and followed by uniform noise. Hence series with different locations of the signal can be approximately regarded as circular shifts of each other if we regard the uniform noise as 'basically the same' everywhere in the series.

For extracting TI features from a time series, the following preprocessing is applied: To remove constant offsets in the values, the series mean is subtracted from each value. Let $\vec{v} = [v_t]_{t=1\ldots T}$ denote the values of a single series after this subtraction. Let $\widehat{\mathbf{W}}$ denote the transform table of the TI wavelet transform from [4], using the maximum level of decomposition and e.g. the Haar basis (used throughout this paper, as argued for in the supplemental material [15]). The $S \times T$ matrix $\widehat{\mathbf{W}}$ is a non-orthogonal decomposition of the series $\vec{v}$ into $S = \log(T)$ scales with T coefficients per scale. The scales with smaller index s represent coarser structures of $\vec{v}$, while those with larger s represent finer details. The notion of translation-invariance that applies to $\widehat{\mathbf{W}}$ is that a circular shift of the series

$\vec{v}$ manifests solely as per-scale permutations in $\widehat{\mathbf{W}}$, i.e. for each of the S rows of $\widehat{\mathbf{W}}$, the T entries are permuted. Note that there are neither permutations between coefficients of *different* scales, nor is there energy-transfer between *any* coefficients. The per-scale permutations are the way in which circular shifts of the input series are encoded by the transform table. Since the goal is computing features that are invariant to shifts of the input series – or synonymously invariant to the location of the finite support of the up or down signal to be detected (cf. Figure 1) – this locational information must be eliminated. This is done by sorting each scale in $\widehat{\mathbf{W}}$ separately by descending absolute coefficient values. The resulting matrix of sorted absolute values is called $\mathbf{W}$. Figure 2 shows examples of $\mathbf{W}$ for series $\vec{v}$ from different classes. For any permutation of coefficients in $\widehat{\mathbf{W}}$, the same $\mathbf{W}$ will result, and hence the locational information has been removed. Feature computation is carried out with respect to $\mathbf{W}$ as detailed in the following section.

2.1 Features of Translation-invariant Wavelet Coefficients

Let $\mathbf{W} = [w_{s,t}]_{s=1...S,t=1...T}$ be the table from section 2 with scale index s and coefficient index t. Furthermore, let $\mu(w_{s,\circ}) = \frac{1}{T}\sum_{t=1}^{T} w_{s,t}$ yield the mean value over the variable indicated by the wildcard symbol $\circ$ for a fixed value of the symbol s. The according standard deviation is defined analogously as $\sigma(w_{s,\circ}) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(w_{s,t} - \mu(w_{s,\circ}))^2}$. In the following equations superscript indices are feature names while the subscript s indicates the scale on which a feature is computed. Feature f_s^1 is the mean value of the coefficients on scale s:

$$f_s^1 = \mu(w_{s,\circ}) \ . \tag{1}$$

Feature f_s^2 is the ratio of f_s^1 and the mean coefficient value over all scales:

$$f_s^2 = \frac{f_s^1}{\mu(w_{\circ,\circ})} \ . \tag{2}$$

Feature f_s^3 is the standard deviation of the coefficients on scale s,

$$f_s^3 = \sigma(w_{s,\circ}) \ , \tag{3}$$

while feature f_s^4 is the ratio of f_s^3 and the coefficient standard deviation accumulated over all scales:

$$f_s^4 = \frac{f_s^3}{\sum_r \sigma(w_{r,\circ})} \ . \tag{4}$$

For defining the last four features, let $\vec{r}_s = [r_{s,t}]_{t=1...T}$ denote T discrete samples of a regression line approximating the coefficients $w_{s,t}$ for a fixed scale s. The line is defined as $a_s x + b_s$, where a_s and b_s are computed as $\text{argmin}_{a_s,b_s} = \sum_t (w_{s,t} - (a_s x_t + b_s))^2$, and the sampling points x_t are chosen at the locations of the coefficients $w_{s,t}$. Hence $r_{s,t} = a_s x_t + b_s$ is the linear approximation of coefficient $w_{s,t}$. Given these prerequisites, f_s^5 is the slope of the regression line,

$$f_s^5 = a_s \ , \tag{5}$$

and f_s^6 is the linear approximation of the largest coefficient $w_{s,1}$

$$f_s^6 = a_s x_1 + b_s \ . \tag{6}$$

Feature f_s^7 is the accumulated absolute deviation between coefficients and their linear approximation, normalized by the accumulated coefficient set:

$$f_s^7 = \frac{\sum_t |w_{s,t} - r_{s,t}|}{\sum_t w_{s,t}} \ . \tag{7}$$

Feature f_s^8 is defined analogously to f_s^7 but with sums replaced by standard deviations:

$$f_s^8 = \frac{\sigma\left(w_{s,\circ} - r_{s,\circ}\right)}{\sigma\left(w_{s,\circ}\right)} \ . \tag{8}$$

Note that the differences in the numerator of f_s^8 are taken only between coefficients $w_{s,t}$ and approximations $r_{s,t}$ with the same index t. To prevent numerical issues, a small constant ϵ on the order of computational working precision is added to each denominator.

In order to balance the impact of the different features during the distance computations occuring in subsequent processing stages, the features need to be normalized accordingly. The employed normalization method is shifting and scaling the range of each feature to the unit interval [0, 1].

2.2 Feature Ranking and Selection

The employed feature ranking and selection method is based on computing a figure of merit for each feature. Let $g = f_s^j, s \in \{1 \ldots S\}, j \in \{1 \ldots 8\}$ denote the single scalar feature under consideration. Let $g_i^c, i \in \{1 \ldots N^c\}, c \in \{1 \ldots C\}$ denote the values that feature g attains over the N^c examples of class c in the training dataset, where C is the total number of classes.

The basic measure in computing the figure of merit of a feature g is the mean absolute distance d^{c_1,c_2} between the feature values of class c_1 and the mean feature value of class c_2:

$$d^{c_1,c_2} = \mu\left(|g_\circ^{c_1} - \mu\left(g_\circ^{c_2}\right)|\right) \ . \tag{9}$$

Using equation (9), the figure of merit m_g of feature g is computed as the following ratio:

$$m_g = \frac{\sum_{c_2} \sum_{c_1 \neq c_2} d^{c_1,c_2}}{\sum_c d^{c,c}} \ . \tag{10}$$

The numerator of m_g accumulates over all classes c_2, in how far the mean feature value for class c_2 differs from the feature values for all other classes $c_1 \neq c_2$. For features that separate the classes well, this value is large. The denominator accumulates over all

classes, in how far feature values vary within a class, hence smaller is better. The features g are then sorted in the order of descending merit m_g, giving a feature ranking assigning rank 1 to the best feature, rank 2 to the second-best and so forth. For determining the final sequence of features, Kuncheva's ranking idea [10] is applied within a 10-fold cross-validation: The ranks attained by each feature are accumulated over the different folds, and the features are sorted in the order of ascending accumulated ranks. Kuncheva indices are computed during this cross-validation in order to assess feature selection stability. Finally the first F of these features are selected to be used for classification, where F is chosen to maximize the mean classification performance (e.g. accuracy) in a second cross-validation using the rank-sorted features and incrementally increasing the candidate for F.

3 Condensed k-NN Using Fast Coreset Clustering

This section describes how the k-Nearest-Neighbors (k-NN) classifier [9] is accelerated by computing it from cluster centers of the training data. This involves clustering as a preprocessing step, which is accelerated for large input sets by using the coreset-based BICO approach [6]. Section 3.1 explains how coresets can be used for clustering, while section 3.2 depicts the training procedure and the application of a condensed k-NN classifier making use of the clustering results.

3.1 Fast Clustering with Coresets

Clustering is often defined as partitioning a set of objects into groups, such that objects in the same group are similar and objects in different groups are dissimilar. The k-means problem is a well-studied clustering problem defining similarity via Euclidean distance. For two points $\vec{p} = (p_1, \ldots, p_F), \vec{q} = (q_1, \ldots, q_F) \in \mathbb{R}^F$, let $||\vec{p} - \vec{q}|| := \sqrt{\sum_{i=1}^{F} (p_i - q_i)^2}$ denote their Euclidean distance. Given a matrix $\mathbf{P} = [\vec{p}_1; \ldots; \vec{p}_N]$ where each point $\vec{p}_i \in \mathbb{R}^F$ is a row vector, the k-means problem asks for a matrix of K_m centers $\mathbf{Q} = [\vec{q}_1; \ldots; \vec{q}_{K_m}]$ that minimizes the sum of squared distances of all points in $\mathbf{P}$ to their nearest center in $\mathbf{Q}$, i.e.

$$\min_{\mathbf{Q} \in \mathbb{R}^{K_m \times F}} \text{cost}(\mathbf{P}, \mathbf{Q}) := \min_{\mathbf{Q} \in \mathbb{R}^{K_m \times F}} \sum_{\vec{p}_i \in \mathbf{P}} \min_{\vec{q}_j \in \mathbf{Q}} ||\vec{p}_i - \vec{q}_j||^2 \,. \tag{11}$$

This is a special case of the weighted k-means problem (with weights all 1), where each point can be weighted by a function $w : \mathbb{R}^F \to \mathbb{R}^+$, i.e. $\text{cost}_w(\mathbf{P}, \mathbf{Q}) = \sum_{\vec{p}_i \in \mathbf{P}} w(\vec{p}_i) \min_{\vec{q}_j \in \mathbf{Q}} ||\vec{p}_i - \vec{q}_j||^2$ is minimized.

Using k-means as the objective function for condensing the training set is a natural choice when using k-NN as a classifier because both algorithms decide a point's membership to a cluster/class via Euclidean distance. By assigning points to the same cluster, k-means preserves their spatial proximity.

In practice Lloyd's algorithm [13] is frequently used to optimize equation (11) and its weighted variant. It is an iterative algorithm converging to a local optimum after

a potentially exponential number of steps. The k-means++ algorithm by Arthur and Vassilvitskii [3] is an improvement of Lloyd's algorithm, yielding an $\mathcal{O}(\log K_m)$ approximation guarantee. Its runtime is similar to that of Lloyd's algorithm. Both algorithms do not scale well and are hence time-consuming for large input data sets.

A way to address this problem is to construct a small summary of the input point set first and to cluster that summary instead. Such summaries can be formalized as *coresets*. A (K_m, ϵ)-coreset for K_m sought cluster centers and approximation ratio ϵ is a small weighted set of points $\mathbf{S} \in \mathbb{R}^{N' \times F}$ that ensures that the weighted clustering cost of $\mathbf{S}$ for any set of K_m centers $\mathbf{Q} \in \mathbb{R}^{K_m \times F}$ is a $(1+\epsilon)$-approximation of the cost of the original input $\mathbf{P} \in \mathbb{R}^{N \times F}$ [8]:

$$|\operatorname{cost}_w(\mathbf{S}, \mathbf{Q}) - \operatorname{cost}(\mathbf{P}, \mathbf{Q})| \leq \epsilon \operatorname{cost}(\mathbf{P}, \mathbf{Q}) \; . \tag{12}$$

Here $w : \mathbb{R}^F \to \mathbb{R}^+$ is the weight function, and the number N' of points in the coreset may be considerably smaller than the number N of points in the original dataset $\mathbf{P}$. Since large data sets may not fit into main memory, and short construction times are crucial, coresets are often computed in a streaming setting. The BICO algorithm detailed in [6] is a streaming-capable algorithm for computing coresets. It is used in this work to reduce the time taken for clustering large input sets. The following section describes how BICO is integrated into the training procedure of k-NN, yielding a condensed k-NN.

3.2 Training and Application of Condensed k-NN

The input of the training procedure for condensed k-NN is the $N \times F$ matrix $\mathbf{G}$, where rows denote training examples and columns denote normalized features, cf. section 2.1. F is the number of features selected as according to section 2.2. N is the sum $N = \sum_{c=1}^{C} N^c$ of examples belonging to C different classes. In training, class labels are known and can hence be used to partition $\mathbf{G}$ into C per-class matrices $\mathbf{G}^c$. For each class c separately, the BICO approach is used to compute a coreset $\mathbf{S}^c$ of $\mathbf{G}^c$, allowing for very large input sets. Subsequently weighted k-means++ is used to compute a clustering from each coreset $\mathbf{S}^c$, condensing the examples in $\mathbf{G}^c$ to $K_m \ll N_c$ cluster centers $\mathbf{H}^c$. The union $\mathbf{H} = [\mathbf{H}^1; \ldots; \mathbf{H}^C]$ of the per-class cluster centers is then used as the condensed training set for k-NN. The number K_m of cluster centers per class is chosen as to be tractable in a lazy learning approach like k-NN and can be used for class balancing.

Applying the learned classifier to unlabeled input works as follows: The raw input data is preprocessed like the training set, using the same feature normalization and selection. The resulting feature vectors are classified using k-NN with Euclidean distance on the condensed training set $\mathbf{H}$. The number K_n of nearest neighbors in k-NN is not to be confused with the number K_m of cluster centers in k-means. The output of this k-NN are predicted labels for the unlabeled input.

4 Results

Three variants of the PAMONO time series classification task were examined to validate the proposed methods: Task 1 is the three-class separation of the up signal, down signal

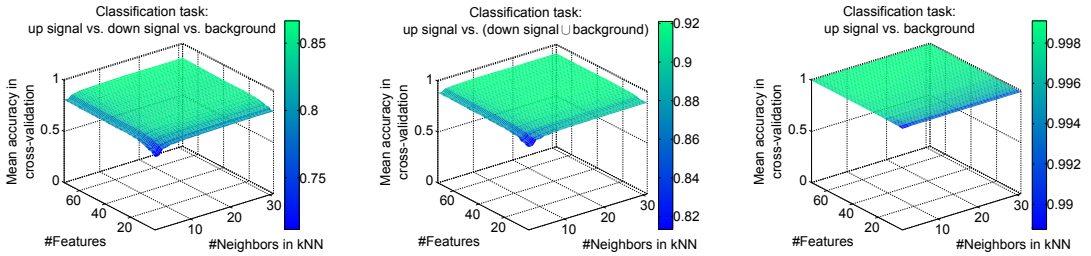
and background classes (cf. Figures 1 and 2). Task 2a is the separation of the up signal class from the union of the down signal and background class. Task 2b is the separation of the up signal from the background class only. The tasks were enumerated in the order of decreasing difficulty and increasing importance: For nano-particle detection it is most important to separate the up signals arising at the centers of particle adhesions from the background arising in regions without particles. Class membership of the small amount of down signals at the fringes of particles can in practice be neglected because down signal pixels can be captured by applying morphological closing to the up signal class mask in image space [5]. This leads to the consideration of task 2b.

Experimental validation was conducted using a total of $N = 315000$ labeled PAMONO time series from $C = 3$ classes as the input. The length was $T = 512$, resulting in 72 features available for selection (8 features on $S = \log(T) = 9$ scales). To obtain 315000 labeled examples, the signal model in [16] was used to create synthetic time series from real background images and particle templates. Three measured datasets were used as the basis for synthesis: 200nm particles on two differently severe levels of noise and one dataset with 100nm particles.

The input examples are partitioned into two disjoint subsets: The cross-validation set contains 2/3 of the examples and is used for feature and parameter selection in 10-fold cross-validation and as the basis to train the condensed k-NN classifier. The test set contains the remaining 1/3 of examples and is used solely for performance assessment. Note that for all classification tasks, class distributions were balanced because accuracy was used as the performance metric, which is sensitive to imbalanced class distributions.

4.0.1 Performance and Parameters.

The number of per-class cluster centers was fixed at $K_m = 1500$ ($\approx 0.5\%$ of the total number of examples) and the coreset size was fixed at $N' = 5K_m$. The following accuracies were attained on the test set: task 1: 0.870, task 2a: 0.920, task 2b: 0.999. For comparison, matching patterns via cosine similarity to T non-cyclic shifts of C ideal model patterns yields the following accuracies: task 1: 0.894, task 2a: 0.996, task 2b: 0.997. Exhaustive matching against all non-cyclic shifts means that TI conditions hold exactly here, not only approximately. The superior accuracy in tasks 1 and 2a comes at the price of requiring model patterns and increased runtime ($\mathcal{O}(T^2)$ per time series for all shifts and distance computations instead of $\mathcal{O}(T \log^2(T))$ for TI table computation and sorting). Increasing the coreset size N' in the proposed method does not increase accuracy, but only run time: The mean (over all tasks) gain in accuracy when using $N' = 200K_m$, which is equivalent to clustering the full input, is $0.0011, \sigma = 0.0016$. Clustering time, on the other hand, increases from a mean value of 72s, $\sigma = 23$s to 2252s, $\sigma = 811$s (CPU: Intel(R) Core(TM) i7-2600 at 3.4 GHz). For determining the number F of best features to be selected and the number K_n of neighbors in k-NN, a grid search was conducted within a 10-fold cross-validation. F and K_n were chosen to maximize mean accuracy over the folds. Figure 3 plots mean accuracy over parameters per task, with task 2b achieving values close to 1. The spread of accuracies over the parameter space decreases with decreasing task difficulty. Accuracies saturate with



(a) (b) (c)

Figure 3: Mean accuracy in 10-fold cross-validation over the number of best features and the number of neighbors to be used in k-NN for the three classification tasks

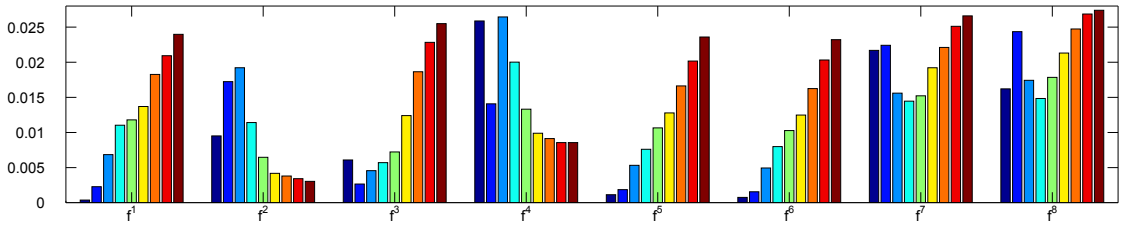
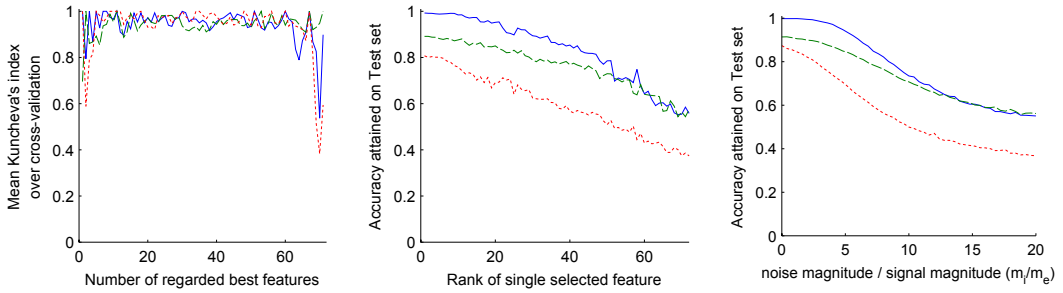


Figure 4: Kuncheva ranks (lower is more relevant) attained by all features on all scales. Scales are ordered coarse (left) to fine (right) for each feature f^i

increasing parameter values, meaning that F and K_n just need to be 'large enough'.

4.0.2 Feature Ranking.

Feature ranking was carried out as described in section 2.2. Figure 4 shows the Kuncheva ranks (lower means 'more relevant'), normalized by the sum of all ranks over all folds. The three most important features are located on the coarsest scale and exhibit strictly decreasing relevance for finer scales. Two of them (f_s^5 and f_s^6) are based on the regression line. The normalized versions f_s^2 and f_s^4 of f_s^1 and f_s^3 favor finer scales. The approximation-error-based features f_s^7 and f_s^8 are comparatively irrelevant on all scales. The figure shows results for classification task 1; for the other tasks, the results are qualitatively the same. Figure 5(a) plots feature selection stability in terms of the mean Kuncheva indices attained over the cross-validation. Kuncheva indices are above 0.88 for selecting between 10 and 60 features for all classification tasks (task 1: *short dashes*, task 2a: *long dashes*, task 2b: *solid line*). Figure 5(b) illustrates that the feature ranking is meaningful in terms of accuracy on the unseen test set: Each feature was used in isolation to classify the test set. The attained accuracy was plotted over the index that feature had in Kuncheva's ranking. Accuracy decreases approximately linearly with increasing Kuncheva rank.



(a) (b) (c)

Figure 5: Kuncheva indices (a), single feature accuracies (b) and robustness to noise (c) for the three classification tasks 'up vs. down vs. background' (*short dashes*), 'up vs. (down ∪ background)' (*long dashes*), 'up vs. background' (*solid line*)

4.0.3 Robustness to Noise.

In order to assess robustness of the features to noise in the input, experiments with artificial noise were conducted. The standard deviation of each input time series was computed, and the mean of these values was used as an estimate m_e of average signal magnitude. Then the interval $[0, 20m_e]$ was equidistantly sampled, and for each sample m_i, zero-mean, unit variance Gaussian noise was scaled by m_i/m_e and added to the original input to give a more noisy input. For each of those noisy inputs a classifier was trained using the proposed method, and test set accuracy was measured for each classification task. Figure 5(c) plots these accuracies over m_i/m_e. The values for zero noise provide a baseline for each classification task. Results deteriorate slowly with increasing noise, which is especially true for task 2b (*solid line*), where noise with up to three times the magnitude of m_e causes only minor loss. From there on, the slope of accuracy loss increases. It decreases again, starting at approximately twelve times more noise than signal, and for each task converges in good approximation to the limit of random guessing for the respective task (1/2 for the two-class tasks and 1/3 for the three-class task; balanced class distributions).

5 Discussion and Future Work

A novel set of translation-invariant wavelet-based features for time series classification was proposed and used in a condensed k-NN classifier with fast coreset-based clustering. The efficacy of the approach was demonstrated with respect to nano-particle detection. For the most important classification task of distinguishing the central parts of particle adhesions from noisy background, accuracy close to 1 was achieved. It was demonstrated that the method is robust to increasing noise in the input signal and insensitive to its main parameters (number of features and k in k-NN), as long they are chosen large enough. Feature selection was shown to be stable, and run time was considerably reduced

by using the coreset-based BICO approach for clustering. For the future it is planned to port the method into a GPU-based detector [12] and assess its impact on quality, as well as validate it on real data, using quality measures like per-class Precision and Recall, that are suitable for the imbalanced real class distributions. Furthermore an examination of classifiers other than k-NN is planned.

Acknowledgments.

Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876. URL: `http://sfb876.tu-dortmund.de/`

References

[1] Alpaydin, E.: Voting over multiple condensed nearest neighbors. Artificial Intelligence Review 11(1-5), 115–132 (1997)

[2] Angiulli, F.: Fast condensed nearest neighbor rule. In: Proceedings of the 22nd International Conference on Machine Learning. pp. 25–32 (2005)

[3] Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the 18th Symposium on Discrete Algorithms (SODA) (2007)

[4] Coifman, R.R., Donoho, D.L.: Translation-invariant de-noising. Springer New York (1995)

[5] Dougherty, E.R.: Introduction to Morphological Image Processing. SPIE Press (1992)

[6] Fichtenberger, H., Gillé, M., Schmidt, M., Schwiegelshohn, C., Sohler, C.: BICO: BIRCH meets coresets for k-means clustering. In: Proceedings of the 21st European Symposium on Algorithms (ESA) (2013)

[7] Gowda, K.C., Krishna, G.: The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. IEEE Transactions on Information Theory 25(4), 488–490 (1979)

[8] Har-Peled, S., Mazumdar, S.: On coresets for k-means and k-median clustering. In: Proceedings of the 36th Symposium on Theory of Computing (STOC). pp. 291–300 (2004)

[9] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. New York: Springer (2009)

[10] Kuncheva, L.I.: A stability index for feature selection. In: Artificial Intelligence and Applications (2007)

[11] Li, D., Luo, H., Shi, Z.: Redundant DWT based translation invariant wavelet feature extraction for face recognition. ICPR (2008)

[12] Libuschewski, P., Siedhoff, D., Timm, C., Gelenberg, A., Weichert, F.: Fuzzy-enhanced, real-time capable detection of biological viruses using a portable biosensor. In: Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSIGNALS) (2013)

[13] Lloyd, S.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28(2) (1982)

[14] Ma, K., Tang, X.: Translation-invariant face feature estimation using discrete wavelet transform. Wavelet Analysis and Its Applications pp. 200 – 210 (2001)

[15] Siedhoff, D., Fichtenberger, H., Libuschewski, P., Weichert, F., Sohler, C., Müller, H.: Signal/background classification of time series for biological virus detection – supplemental material. In: 36th German Conference on Pattern Recognition (GCPR) (2014), supplied along with the Submission of this Paper.

[16] Siedhoff, D., Libuschewski, P., Weichert, F., Zybin, A., Marwedel, P., Müller, H.: Modellierung und Optimierung eines Biosensors zur Detektion viraler Strukturen. In: Bildverarbeitung für die Medizin. pp. 108–113. Springer Berlin Heidelberg (2014)

[17] Yong, Z., Youwen, L., Shixiong, X.: An improved KNN text classification algorithm based on clustering. Journal of computers 4(3) (2009)

[18] Zybin, A., Kuritsyn, Y.A., Gurevich, E.L., Temchura, V.V., Ueberla, K., Niemax, K.: Real-time detection of single immobilized nanoparticles by surface plasmon resonance imaging. Plasmonics 5, 31–35 (2010)