DATA MINING 2
Time Series Classification

Riccardo Guidotti

a.a. 2021/2022

UNIVERSITÀ DI PISA

Time Series Classification

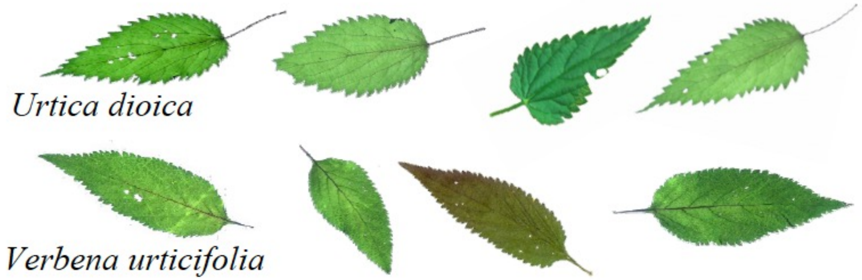
- Given a set X of n time series, $X = \{x_1, x_2, \ldots, x_n\}$, each time series has m ordered values $x_i = < x_{t1}, x_{t2}, \ldots, x_{tm} >$ and a class value c_i.
- The objective is to find a function f that maps from the space of possible time series to the space of possible class values.
- Generally, it is assumed that all the TS have the same length m.

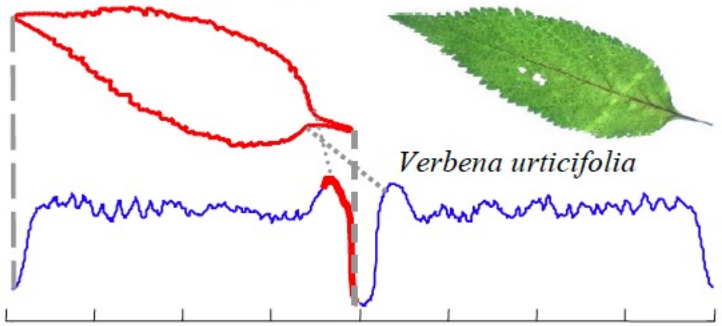
KNN Classification

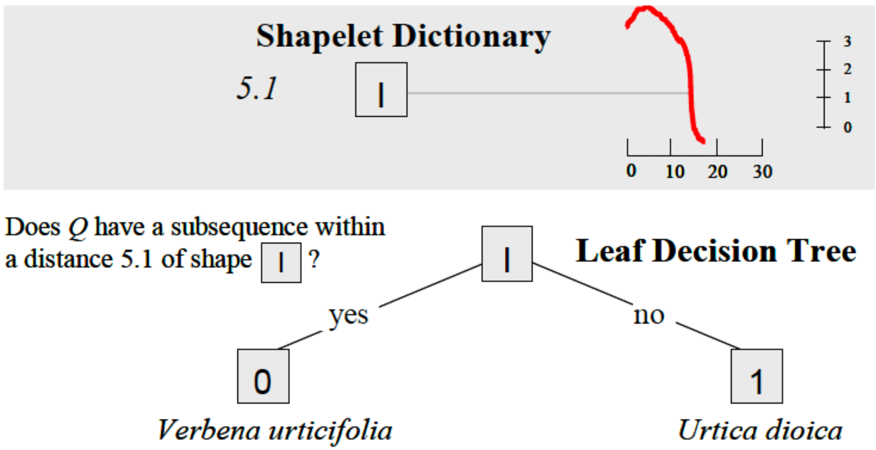
- The most widely used and effective approach for TSC consists in using KNN on the raw time series.
- Pros:
  - Simple
  - Dynamic Time Warping gives much better results than Euclidean distance on many problems.
- Cons:
  - KNN is a lazy classifier and computationally expensive on its own
  - Dynamic Time Warping is very very slow to calculate

Shapelet-based Classification

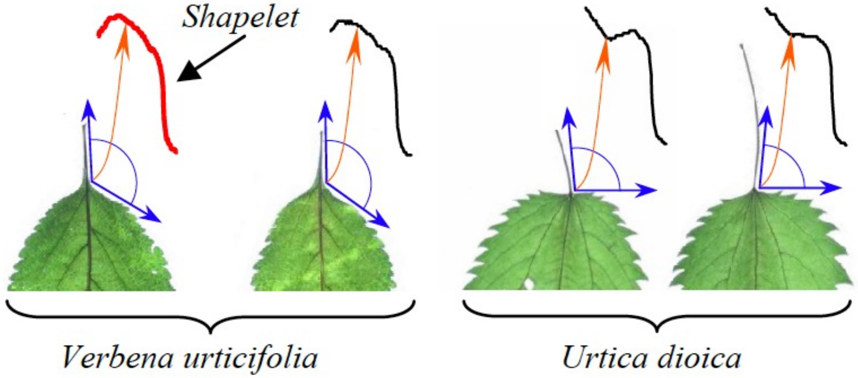
1. Represent a TS as a vector of distances with representative subsequences, namely shapelets.
2. Use it to as input for machine learning classifiers.



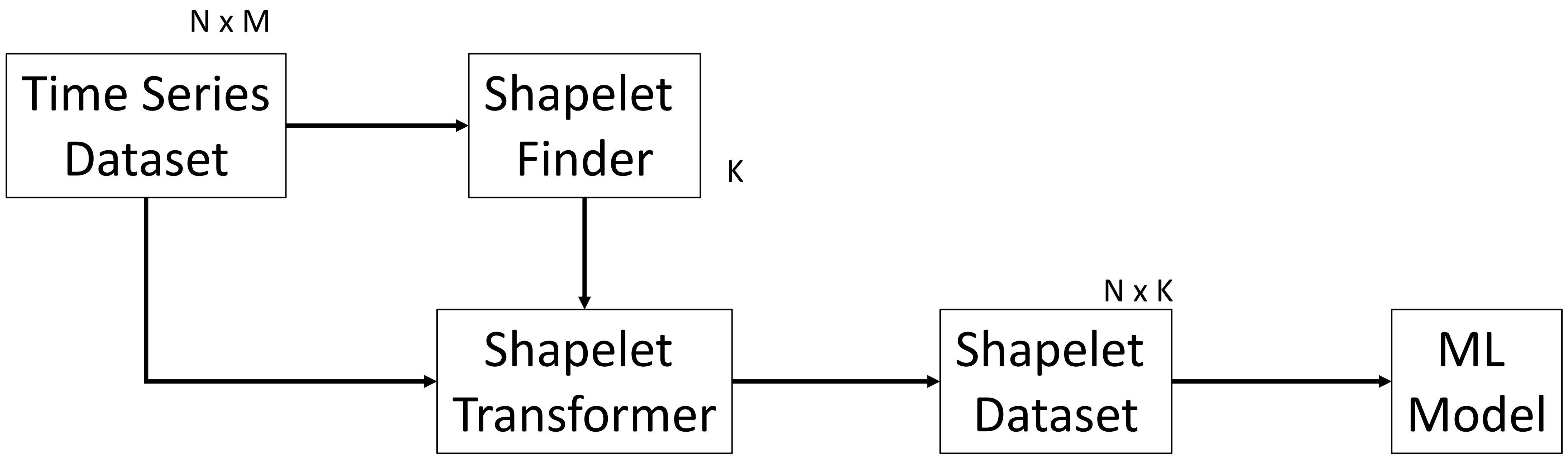




3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4



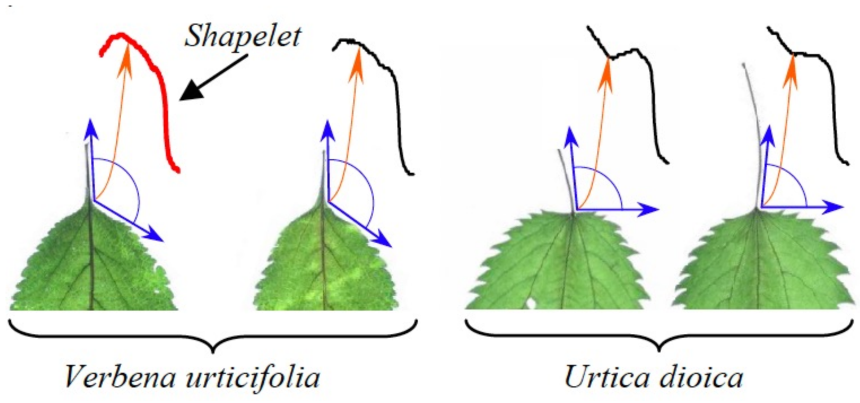
Shapelet-based Classifiers



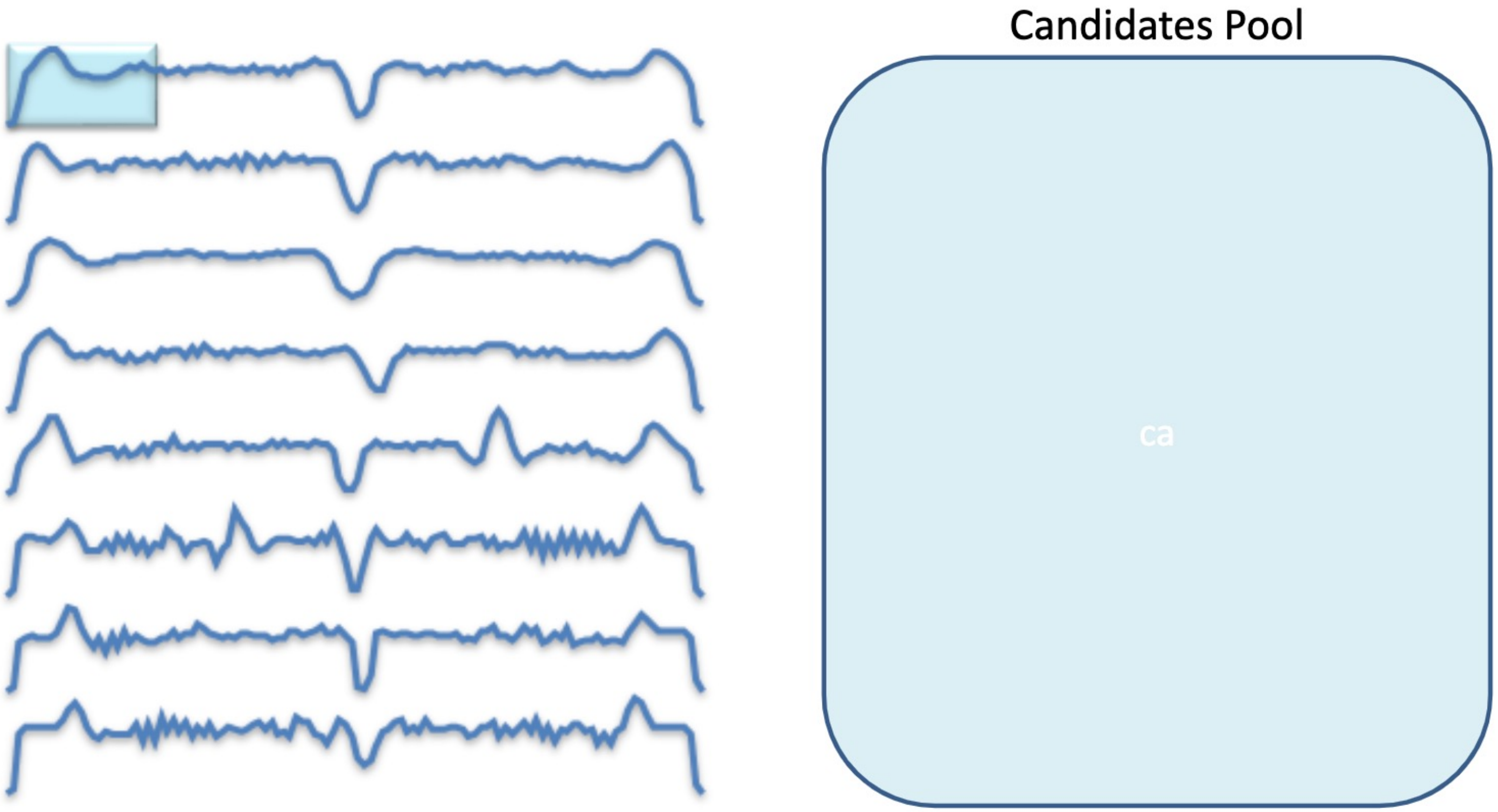
Time Series Shapelets

- Shapelets are TS subsequences which are maximally representative of a class.
- Shapelets can provide interpretable results, which may help domain practitioners better understand their data.
- Shapelets can be significantly more accurate/robust because they are *local features*, whereas most other state-of-the-art TS classifiers consider *global features*.

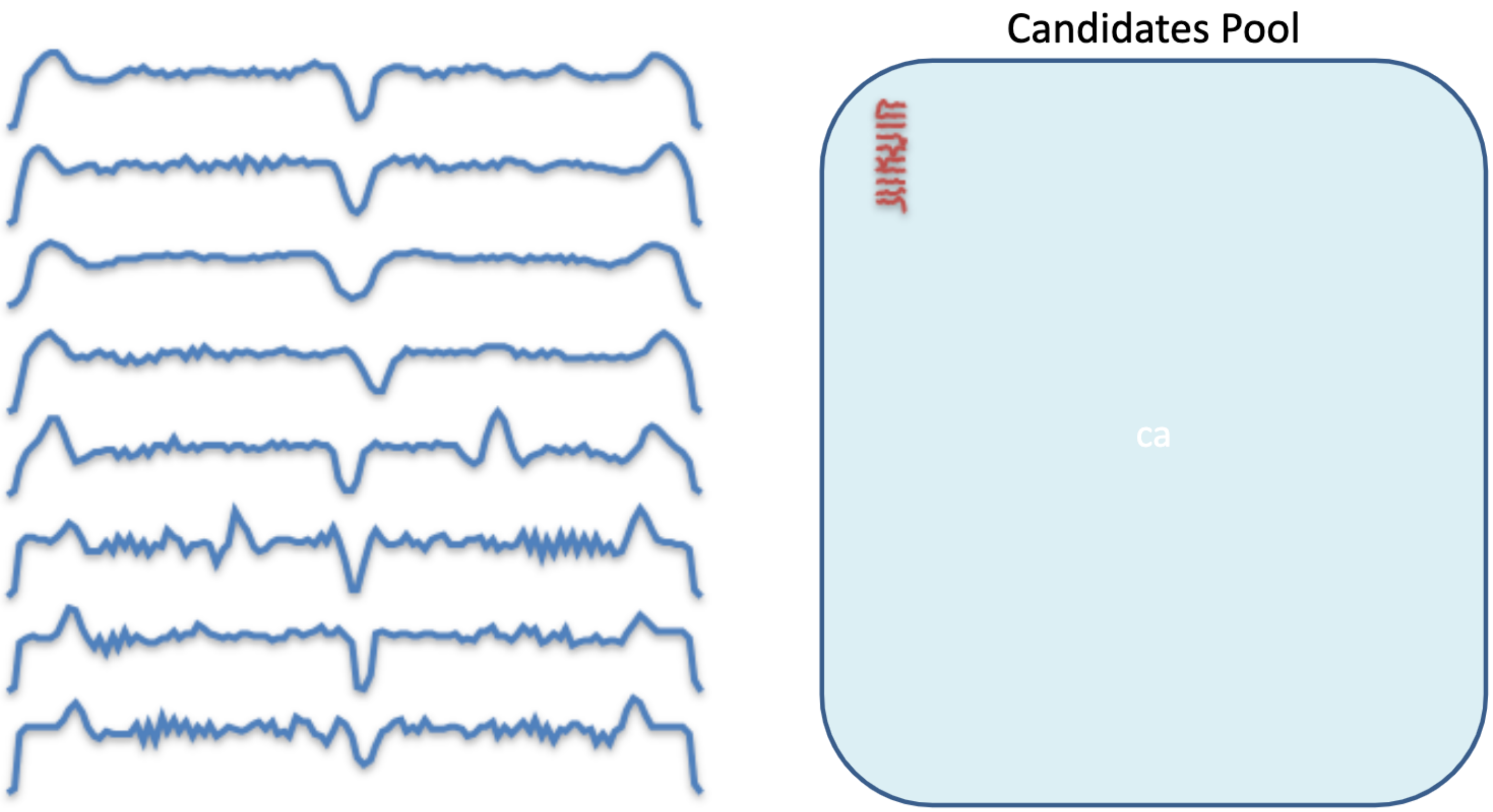
Verbena 0.87
Urtica 0.34



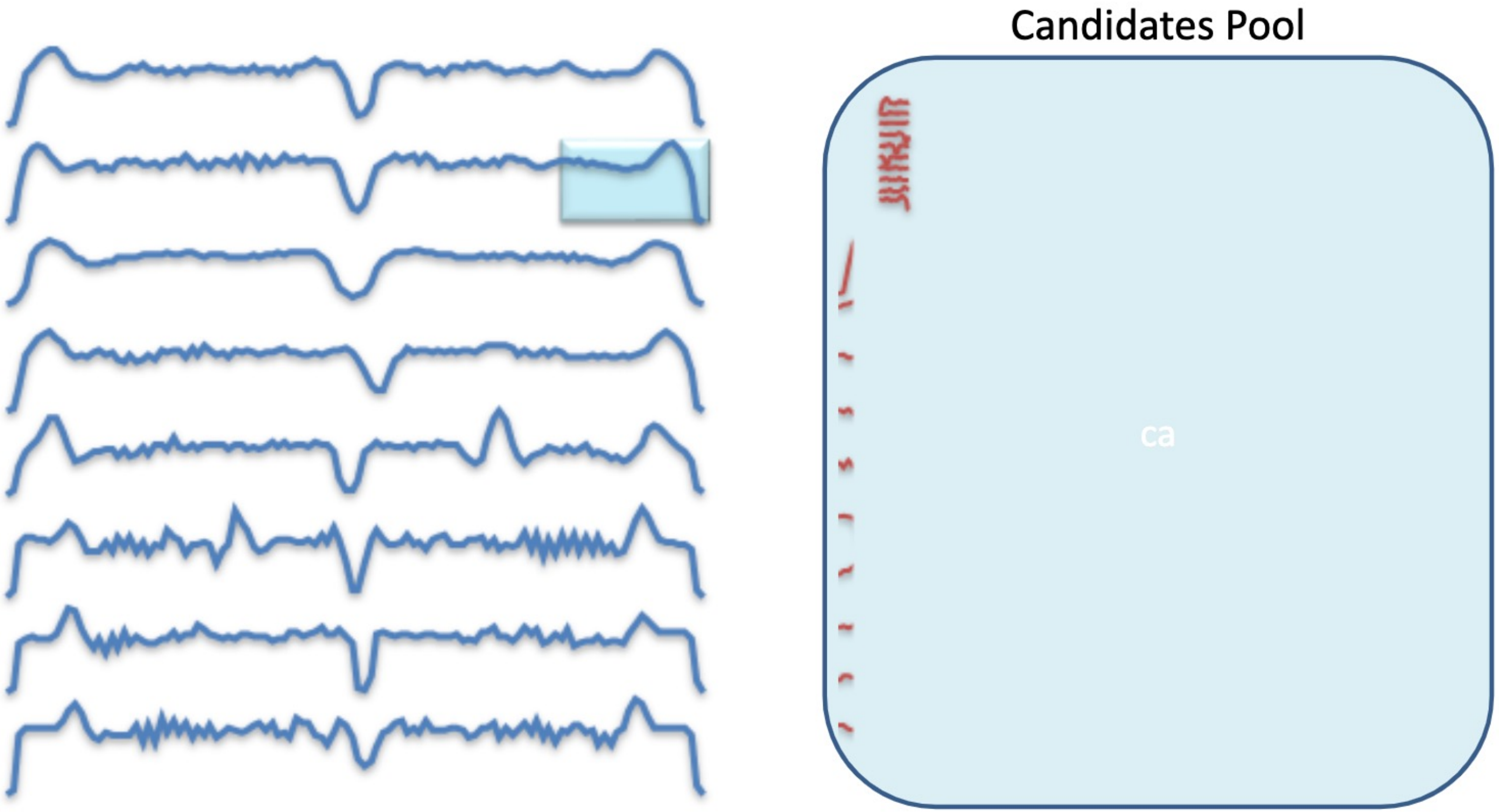
Extract Subsequences of all Possible Lengths



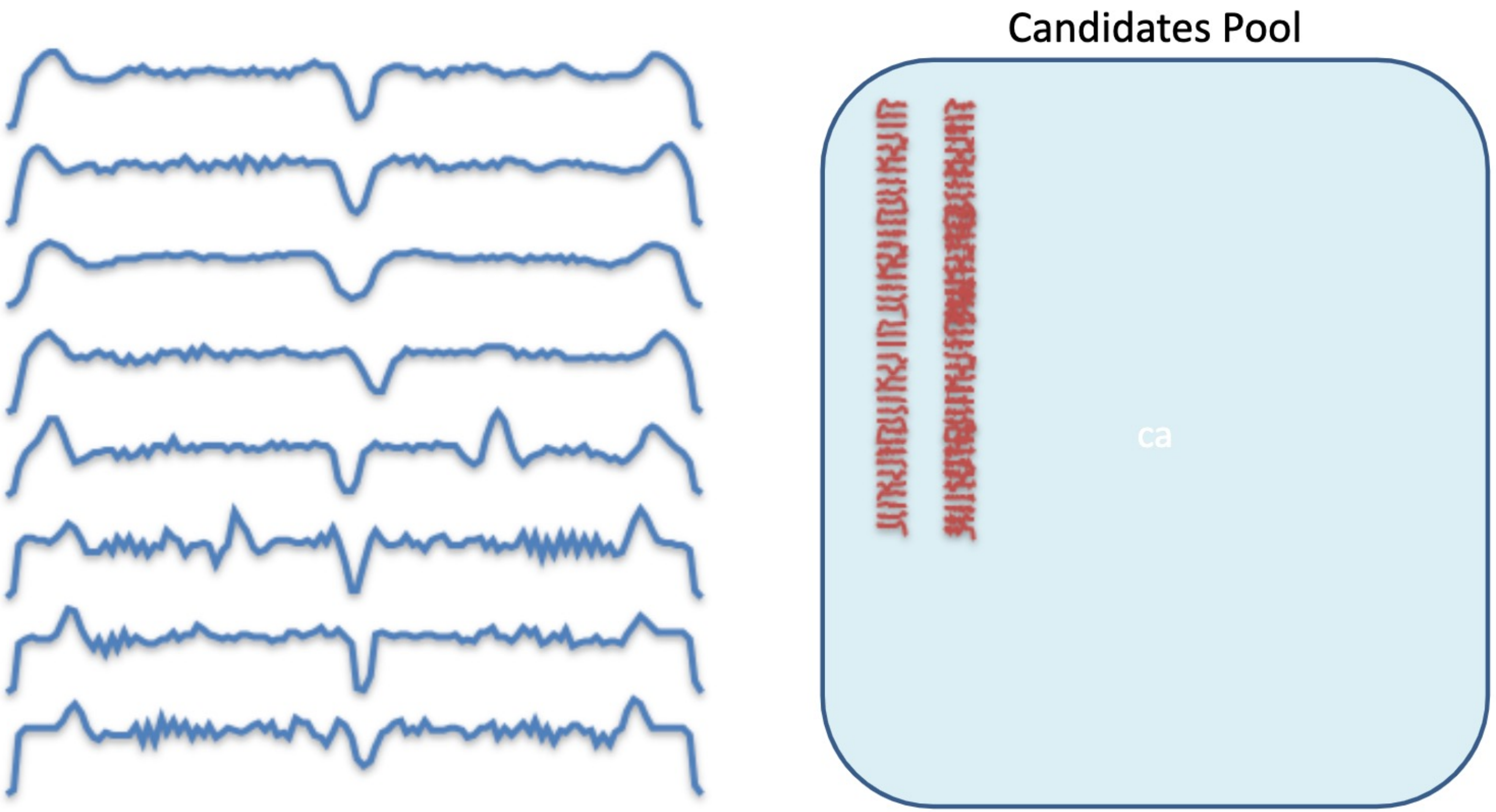
Extract Subsequences of all Possible Lengths



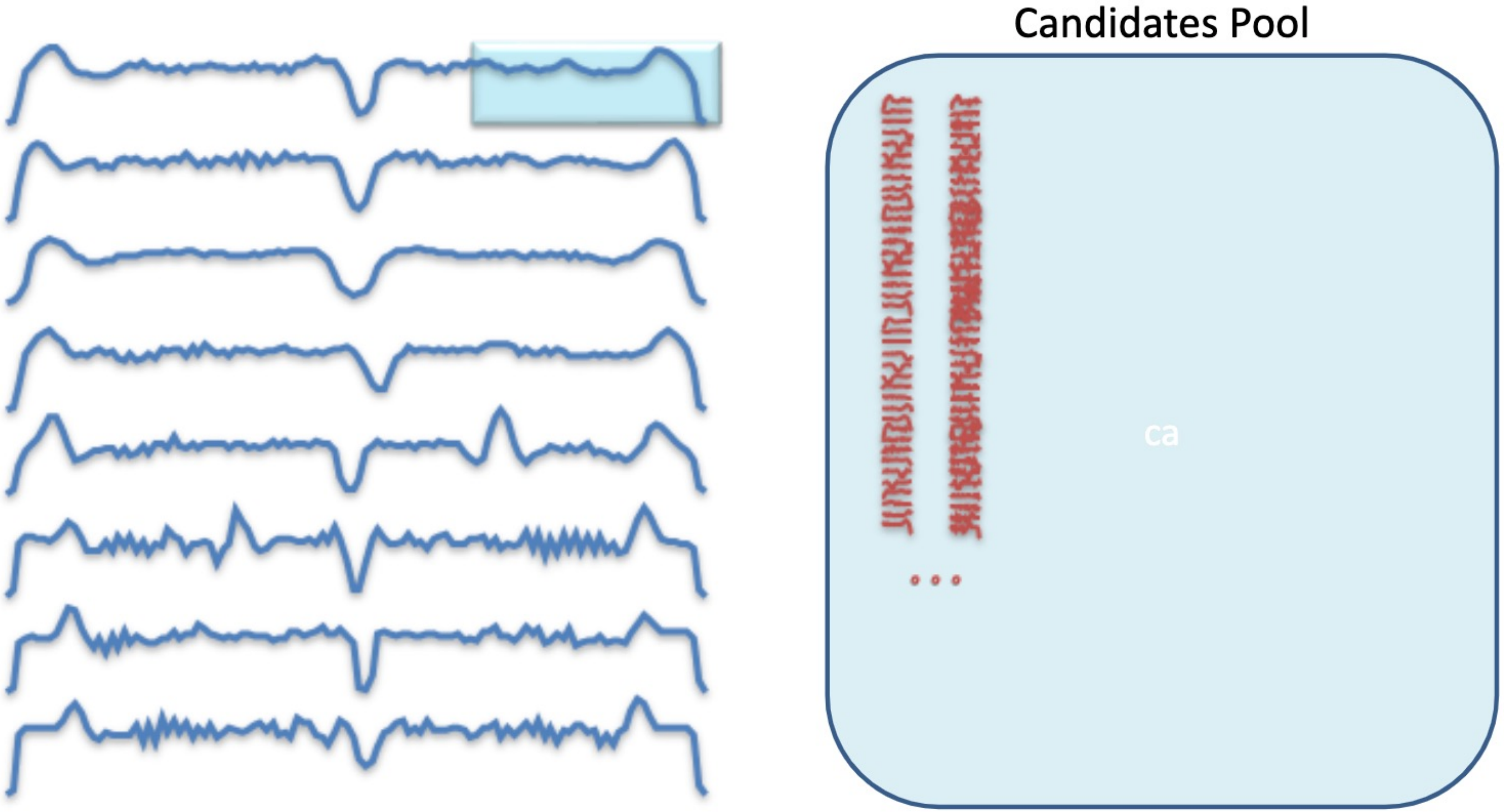
Extract Subsequences of all Possible Lengths



Extract Subsequences of all Possible Lengths

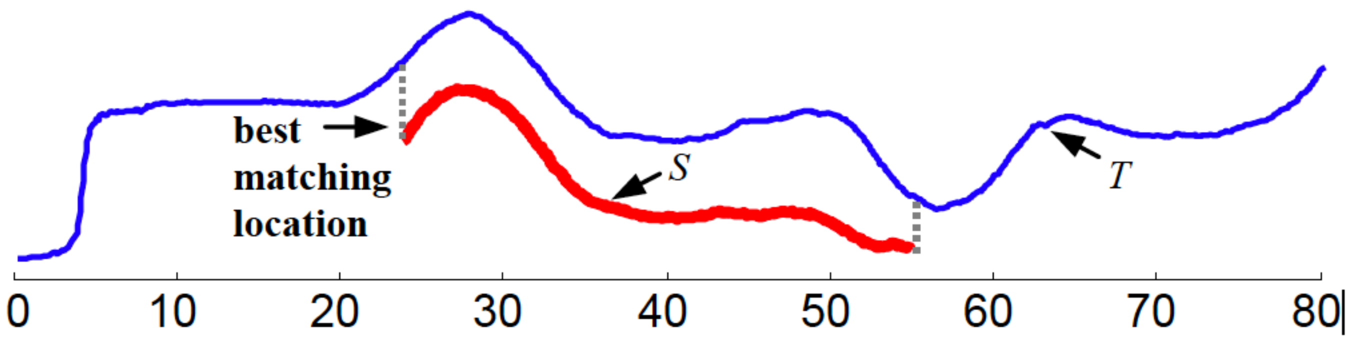


Extract Subsequences of all Possible Lengths



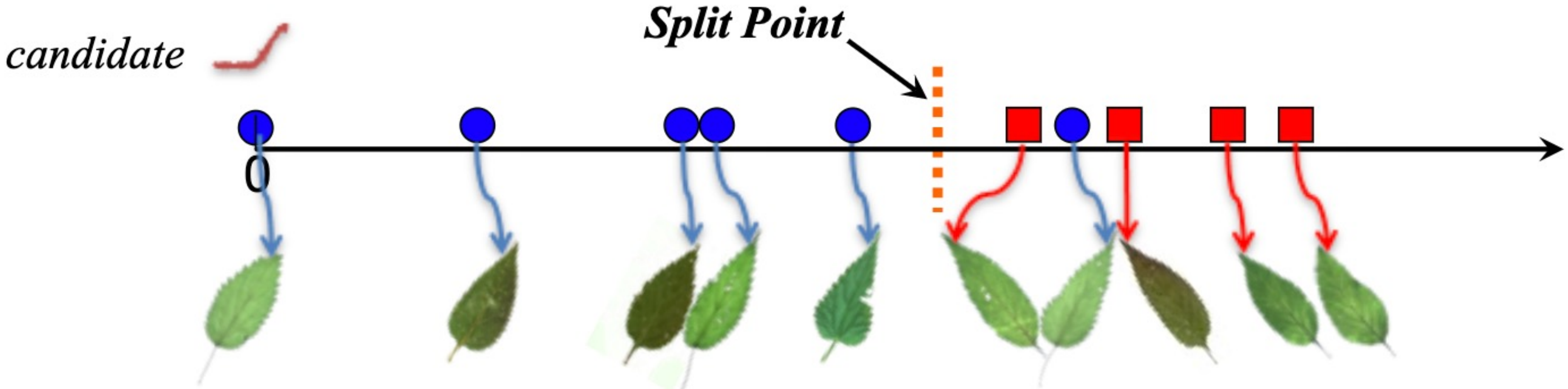
Distance with a Subsequence

- Distance from the TS to the subsequence *SubsequenceDist(T, S)* is a distance function that takes time series *T* and subsequence *S* as inputs and returns a nonnegative value *d*, which is the distance from *T* to *S*.
- $SubsequenceDist(T, S) = min(Dist(S, S')), for\ S' \in S_T^{|S|}$
- where $S_T^{|S|}$ is the set of all possible subsequences of *T*
- Intuitively, it is the distance between *S* and its best matching location in *T*.

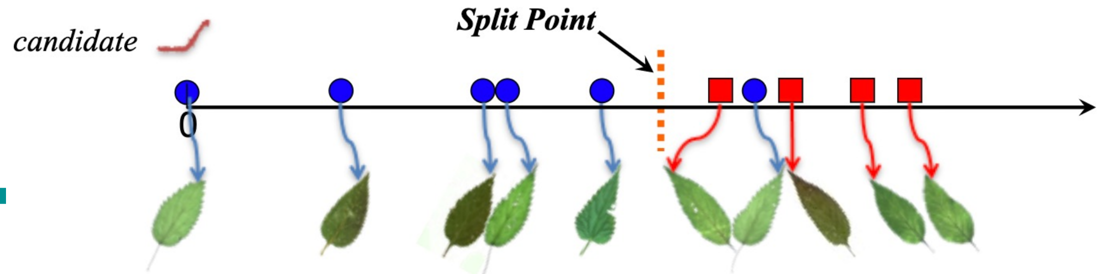


Testing The Utility of a Candidate Shapelet

- Arrange the TSs in the dataset *D* based on the distance from the candidate.
- Find the optimal split point that maximizes the information gain (same as for Decision Tree classifiers)
- Pick the candidate achieving best utility as the shapelet

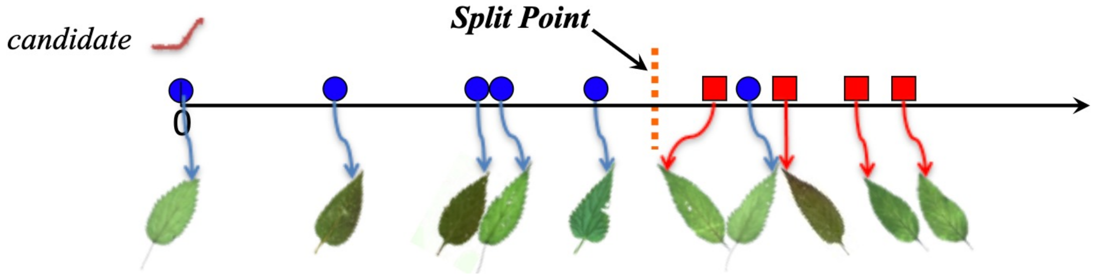


Entropy



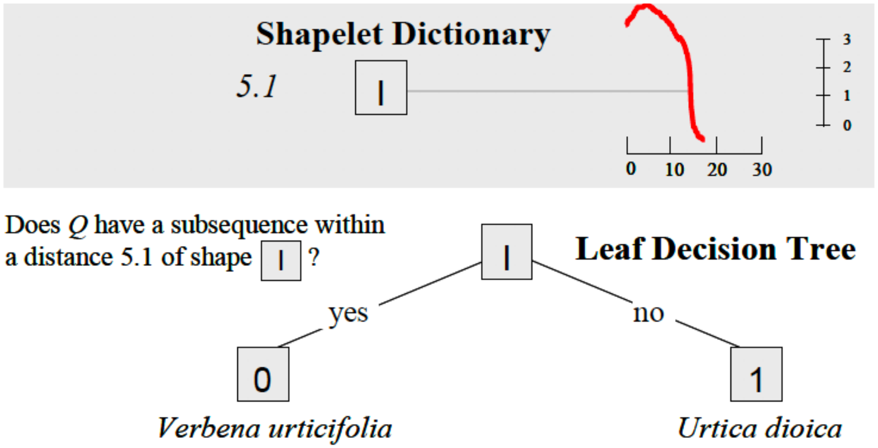
- A TS dataset *D* consists of two classes, A and B.
- Given that the proportion of objects in class A is *p(A)* and the proportion of objects in class B is *p(B)*,
- The **Entropy** of D is: $I(D) = -p(A)\log(p(A)) - p(B)\log(p(B))$.
- Given a strategy that divides the *D* into two subsets D_1 and D_2, the information remaining in the dataset after splitting is defined by the weighted average entropy of each subset.
- If the fraction of objects in D_1 is $f(D_1)$ and in D_2 is $f(D_2)$,
- The total entropy of *D* after splitting is $\hat{I}(D) = f(D_1)I(D_1) + f(D_2)I(D_2)$.

Information Gain



Split point distance from shapelet = 5.1

- Given a certain split strategy *sp* which divides D into two subsets D_1 and D_2, the entropy before and after splitting is $I(D)$ and $\hat{I}(D)$.
- The **information gain** for this splitting rule is:
- $Gain(sp) = I(D) - \hat{I}(D) =$
- $\quad\quad\quad\quad = I(D) - f(D_1)I(D_1) + f(D_2)I(D_2).$



- We use the distance from T to a shapelet S as the splitting rule *sp*.

Problem

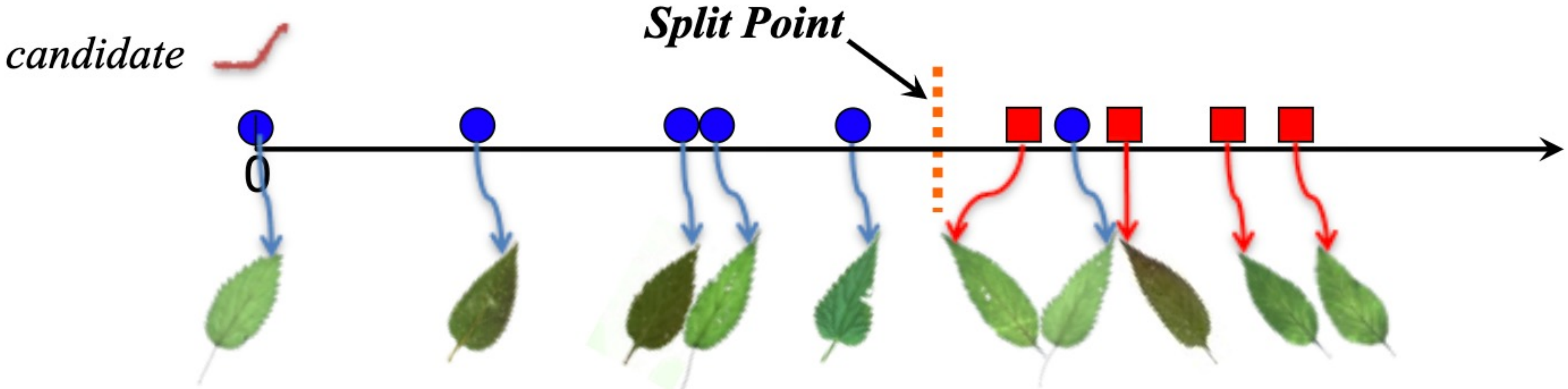
- The total number of candidate is

$$\sum_{l=MINLEN}^{MAXLEN} \sum_{T_i \in D} (|T_i| - l + 1)$$

- For each candidate you have to compute the distance between this candidate and each training sample
- For instance
  - 200 instances with length 275
  - 7,480,200 shapelet candidates

Speedup

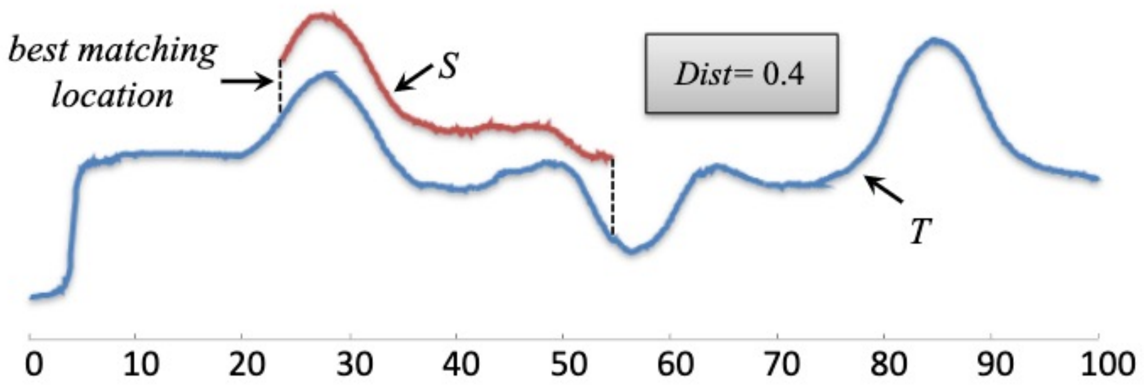
- Distance calculations form TSs to shapelet candidates is expensive.
- Reduce the time in two ways
- Distance Early Abandon
  - reduce the distance computation time between two TS
- Admissible Entropy Pruning
  - reduce the number of distance calculations

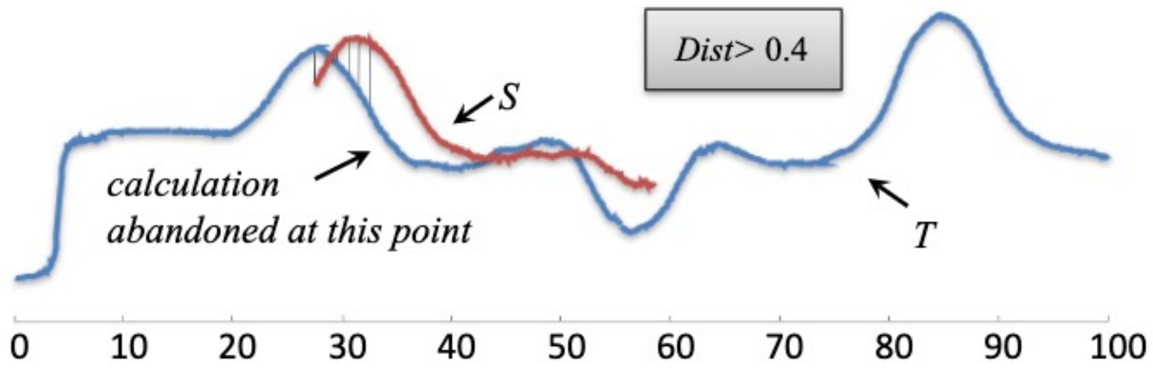


Distance Early Abandon

- We only need the minimum distance.

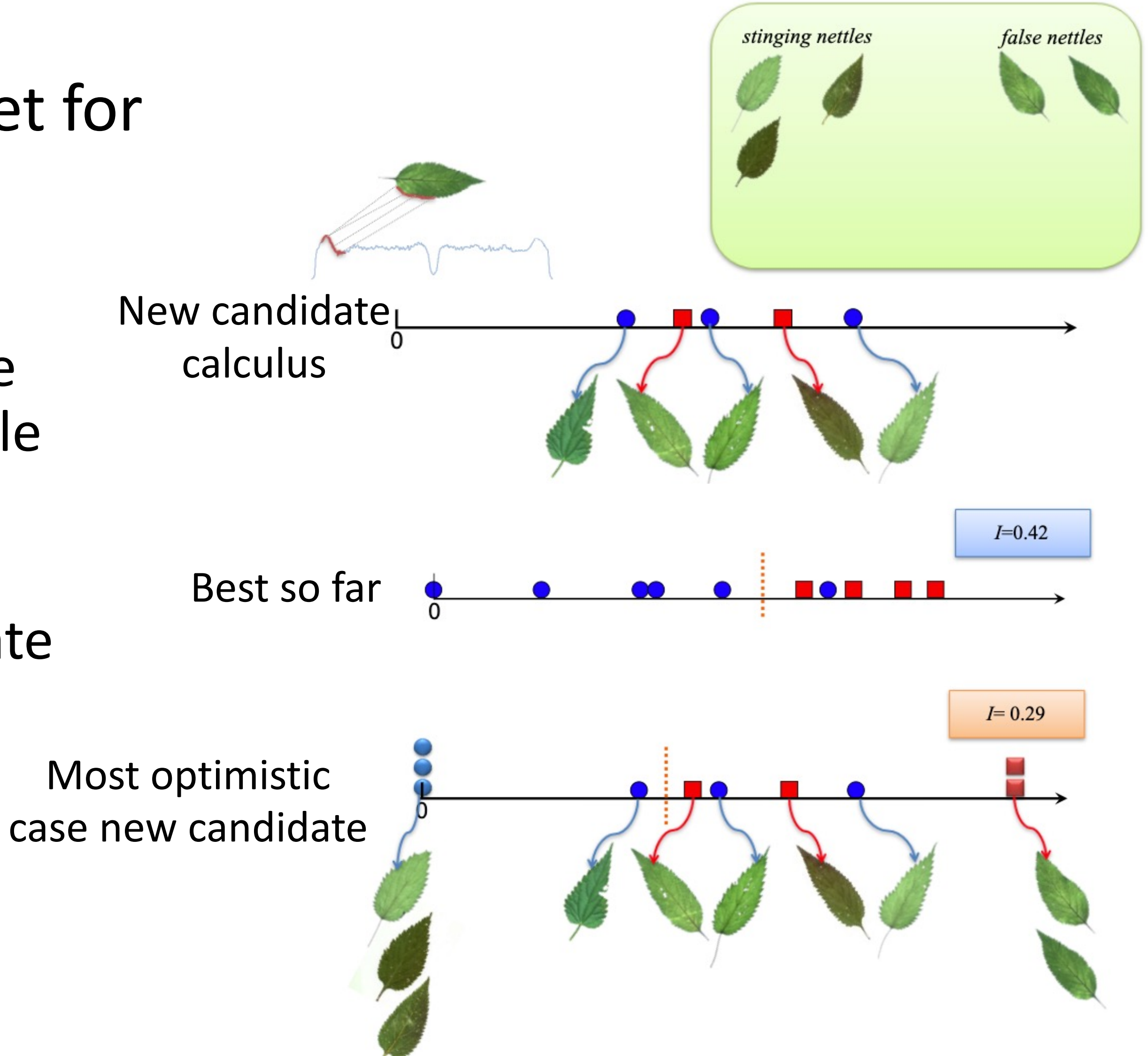
- Method
  - Keep the best-so-far distance
  - Abandon the calculation if the current distance is larger than best-so-far.





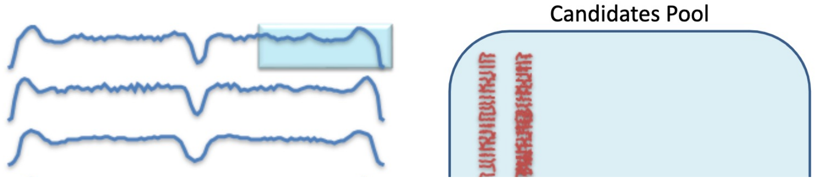
Admissible Entropy Pruning

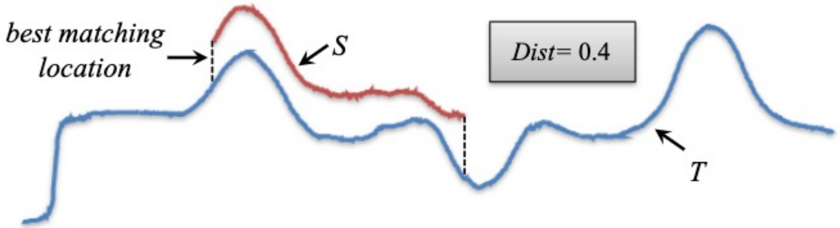
- We only need the best shapelet for each class
- For a candidate shapelet
  - We do not need to calculate the distance for each training sample
  - After calculating some training samples, the upper bound of information gain < best candidate shapelet
  - Stop calculation
  - Try next candidate

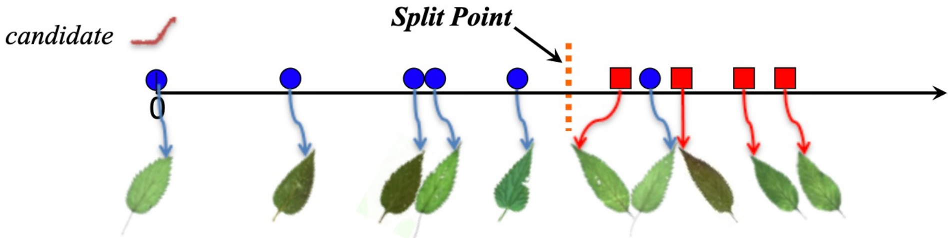


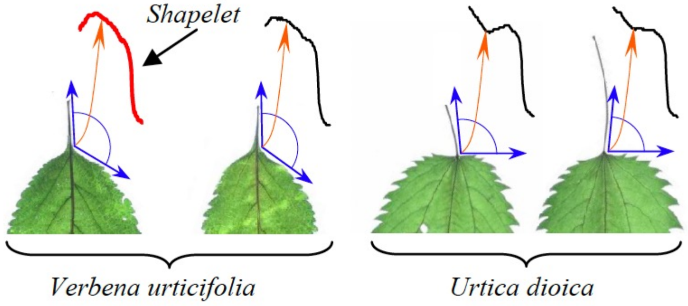
Shapelet Summary

1. Extract all possible subsequences of a set given lengths (candidate shapelets)
2. For each candidate shapelet
    1. Calculate the distance with each time series keeping the minimum distance (best alignment)
    2. Evaluate the discriminatory effect of the shapelet through the Information Gain
3. Return the *k* best shapelets with the highest Information Gain.
4. Transform a dataset and train a ML model.









An Alternative Way for Extracting Shapelets

- The minimum distances (M) between Ts and Shapelets can be used as predictors to approximate the TSs label (Y) using a linear model (W):

$$\hat{Y}_i = W_0 + \sum_{k=1}^{K} M_{i,k} W_k, \quad \forall i \in \{1, \ldots, I\}$$

- A logistic regression loss can measure the quality of the prediction:

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln \left(1 - \sigma(\hat{Y})\right)$$

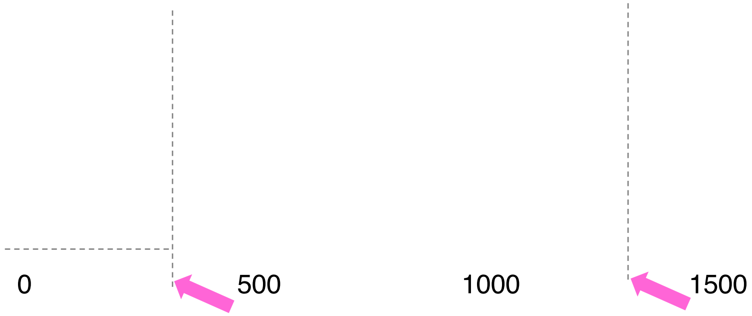
- The objective is to minimize a regularized loss function across all the instances (I) :

$$\underset{S,W}{\operatorname{argmin}} \; \mathcal{F}(S, W) = \underset{S,W}{\operatorname{argmin}} \sum_{i=1}^{I} \mathcal{L}(Y_i, \hat{Y}_i) + \lambda_W ||W||^2$$

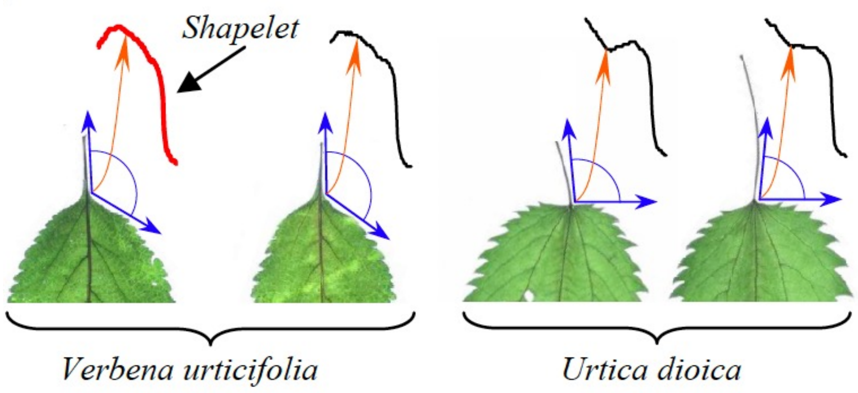
- We can find the optimal shapelet for the objective function in a NN fashion by updating the shapelets in the minimum direction of the objective, hence the first gradient. Similarly, the weights can be jointly updated towards minimizing the objective function.

Motif/Shapelet Summary

- A **motif** is a repeated pattern/subsequence in a given TS.



- A **shapelet** is a pattern/subsequence which is maximally representative of a class with respect to a given dataset of TSs.

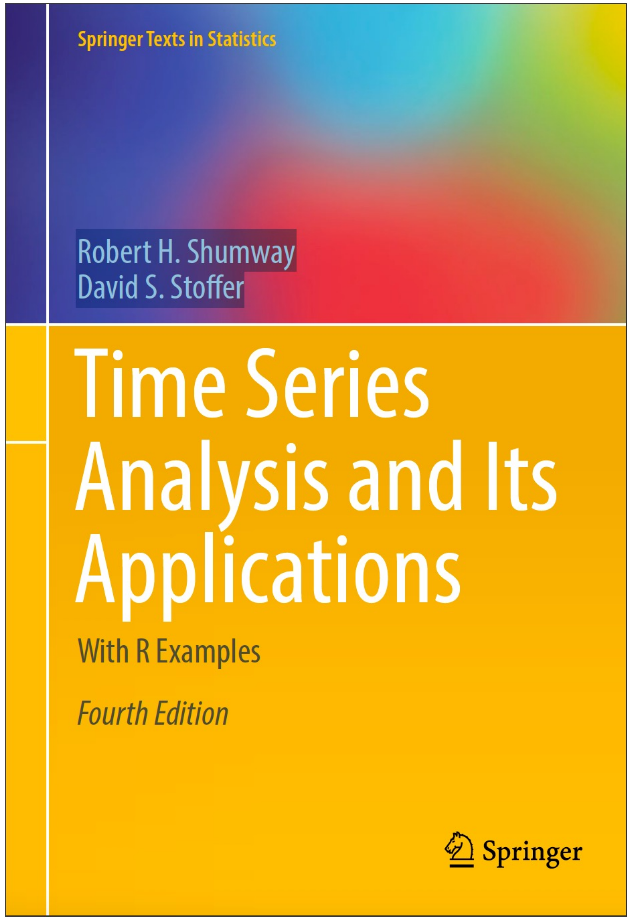


References

- Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Chin-Chia Michael Yeh et al. 1997
- Time Series Shapelets: A New Primitive for Data Mining. Lexiang Ye and Eamonn Keogh. 2016.
- Josif Grabocka, Nicolas Schilling, Martin Wistuba, Lars Schmidt-Thieme (2014): Learning Time-Series Shapelets, in Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2014
- Deep learning for time series classication: a review. Hassan Ismail Fawaz et al. 2019.

References

- Selective review of offline change point detection methods. Truong, C., Oudre, L., & Vayatis, N. (2020). Signal Processing, 167, 107299.
- Time Series Analysis and Its Applications. Robert H. Shumway and David S. Stoffer. 4th edition.(https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf)
- Mining Time Series Data. Chotirat Ann Ratanamahatana et al. 2010. (https://www.researchgate.net/publication/227001229_Mining_Time_Series_Data)
- Dynamic Programming Algorithm Optimization for Spoken Word Recognition. Hiroaki Sakode et al. 1978.
- Experiencing SAX: a Novel Symbolic Representation of Time Series. Jessica Line et al. 2009
- Compression-based data mining of sequential data. Eamonn Keogh et al. 2007.



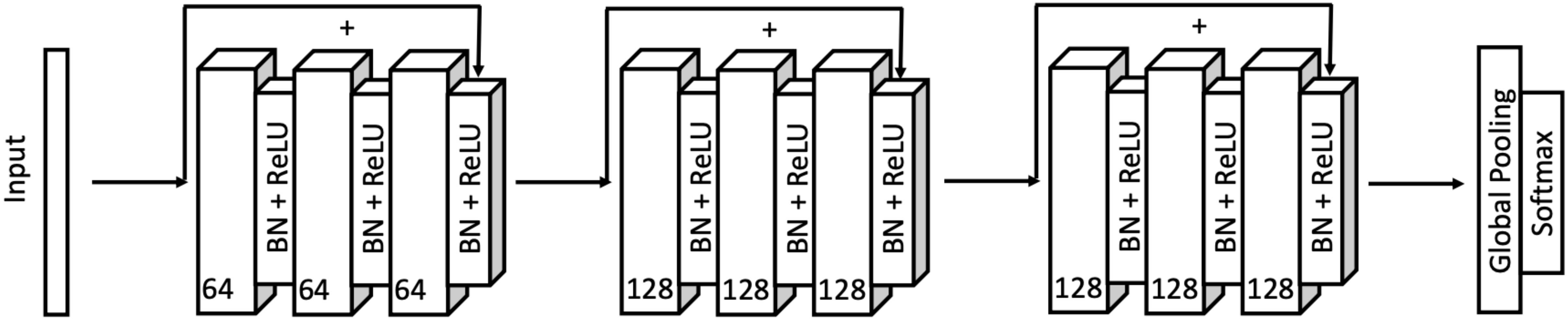
TSC State-of-The-Art

A special thank to Francesco Spinnato for the slides

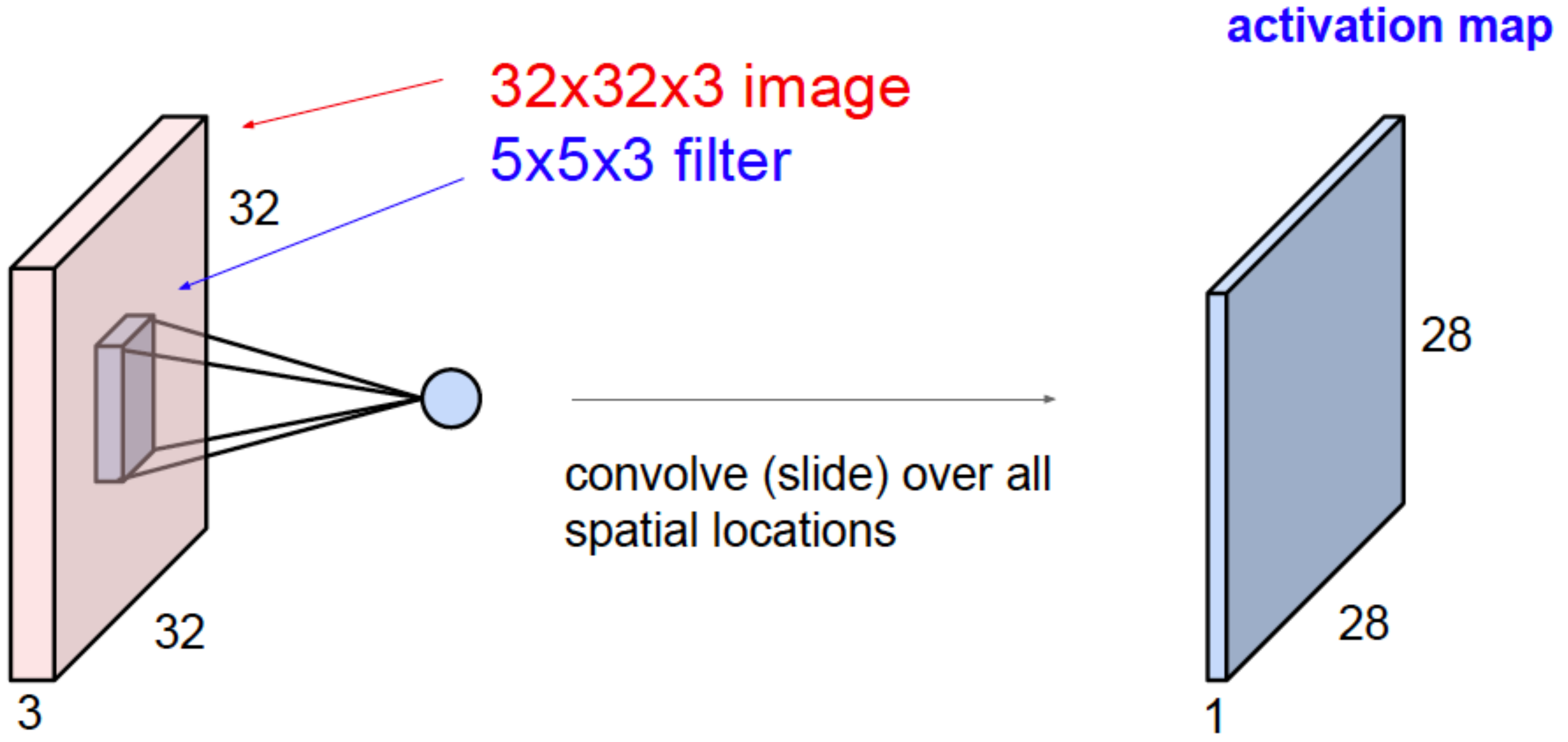
UNIVERSITÀ DI PISA

ResNet

- Three consecutive blocks, comprised of three convolutional layers, connected by residual 'shortcut' connections.
- The blocks are followed by global average pooling and softmax layers to form features and subsequent predictions.

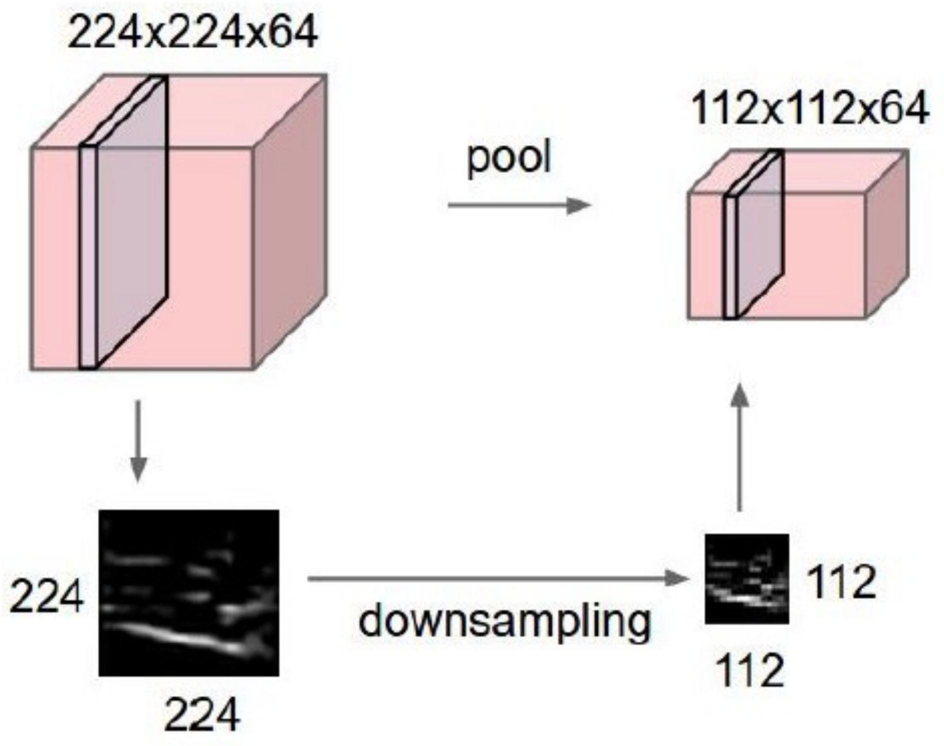


Convolution Layer

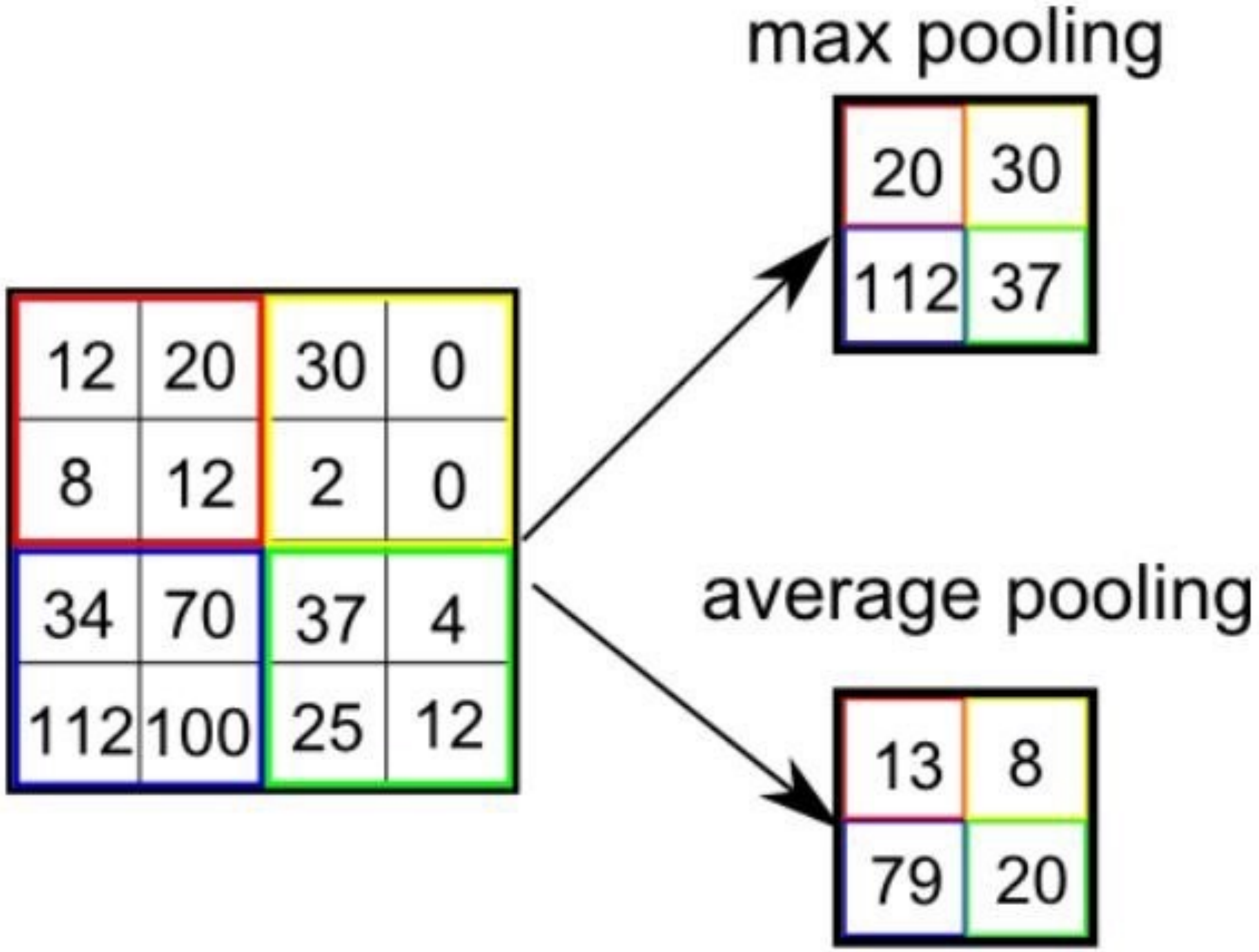


Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



MaxPooling and AvgPoling



InceptionTime

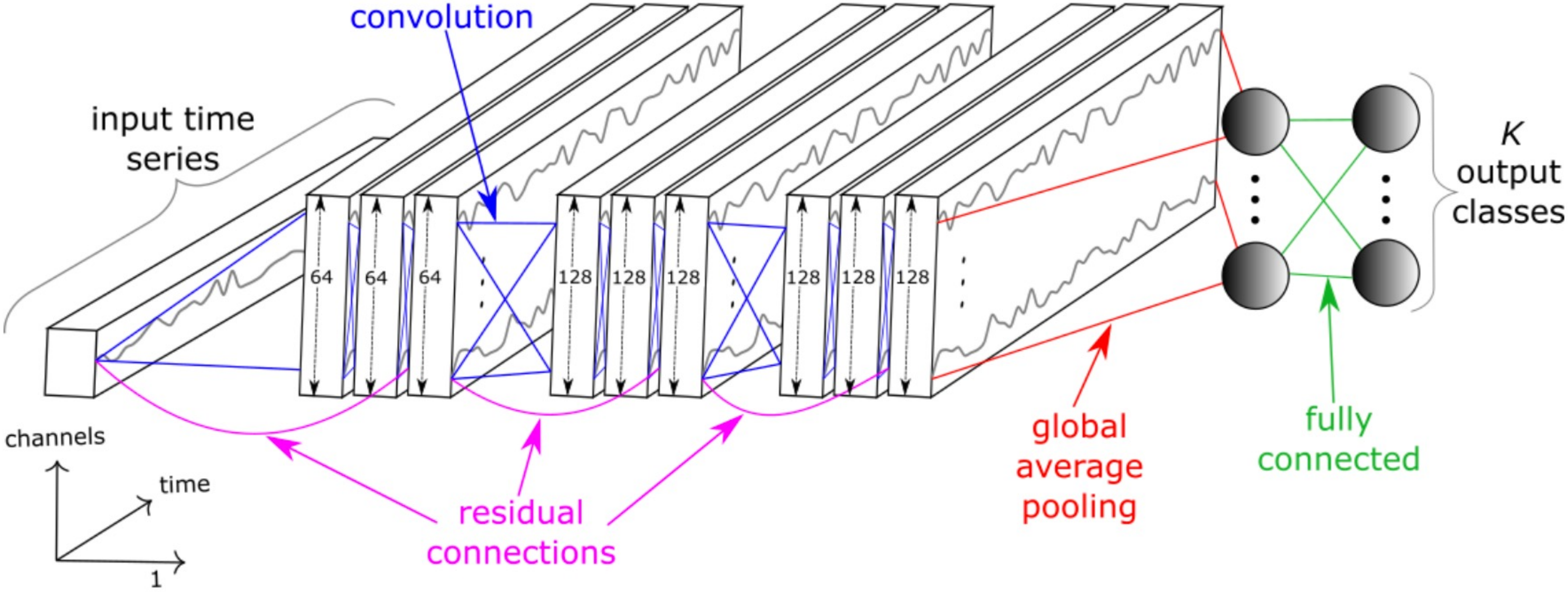
Neural network ensemble consisting of five Inception networks.

For each inception network:

- three Inception modules (6 blocks by default)
- global averaging pooling
- fully-connected layer with the softmax activation function.

Each Inception module consists of convolutions with kernels of several sizes followed by batch normalization and the rectified linear unit activation function.

InceptionTime



TapNet

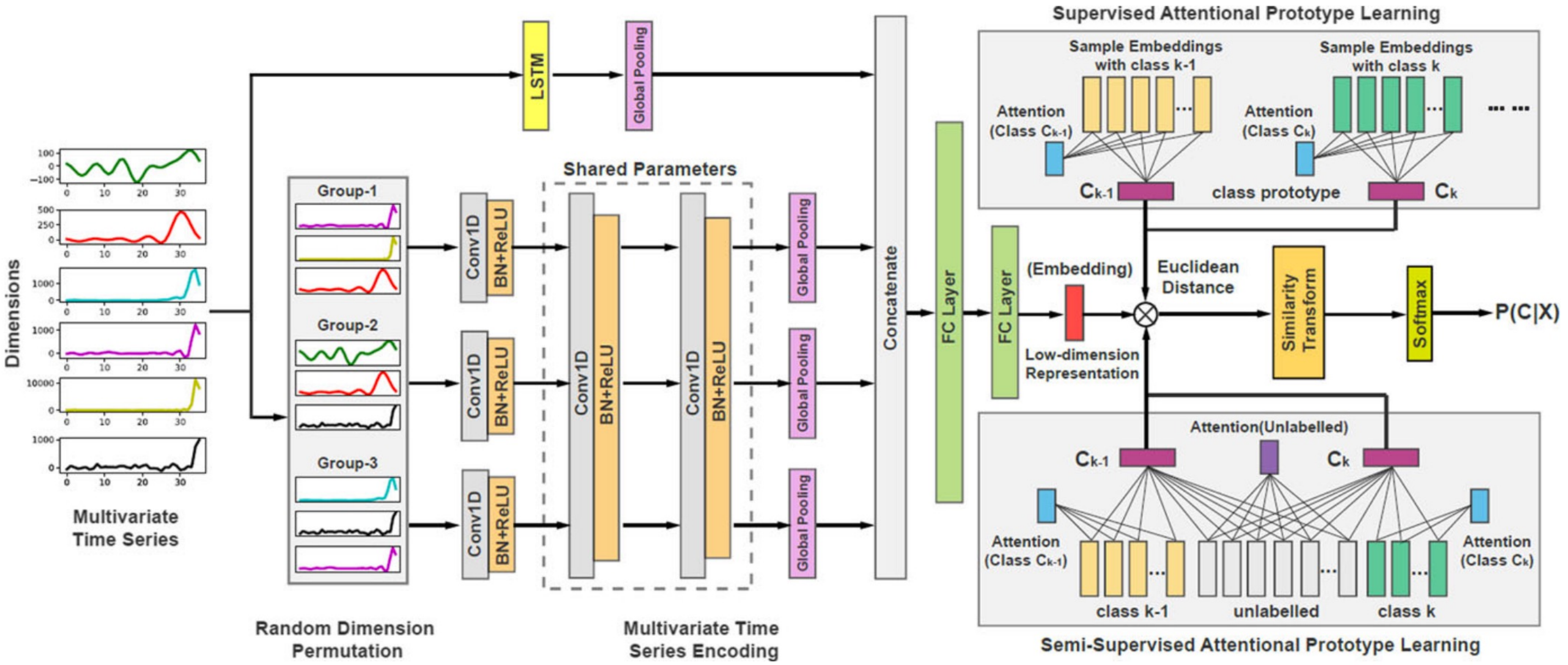
Draws on the strengths of both traditional and deep learning approaches:

- **deep learning approaches** -> excel at learning low dimensional features without the need for embedded domain knowledge, whereas
- **traditional approaches** -> work well on small datasets.

3 distinct modules:

- Random Dimension Permutation: produce groups of randomly selected dimensions with the intention of increasing the likelihood of learning how combinations of dimension values effect class value.
- Multivariate Time Series Encoding:
  - 3 sets of 1d convolutional layers followed by batch normalisation
  - the raw data is also passed through an LSTM and global pooling layer
- Attentional Prototype Learning: used for unlabelled data

TapNet


Supervised Attentional Prototype Learning
Sample Embeddings with class k-1
Sample Embeddings with class k
Attention (Class Ck-1)
Attention (Class Ck)
Ck-1
class prototype
Ck
LSTM
Global Pooling
Shared Parameters
Group-1
Group-2
Group-3
Conv1D
BN+ReLU
Concatenate
FC Layer
(Embedding)
Euclidean Distance
Similarity Transform
Softmax
P(C|X)
Low-dimension Representation
Dimensions
Multivariate Time Series
Attention(Unlabelled)
class k-1
unlabelled
class k
Random Dimension Permutation
Multivariate Time Series Encoding
Semi-Supervised Attentional Prototype Learning

Canonical Interval Forest (CIF)

Ensemble of time series tree classifiers built using the 22 Canonical Time-Series Characteristics (Catch22) features and simple summary statistics (mean, stdev, slope).

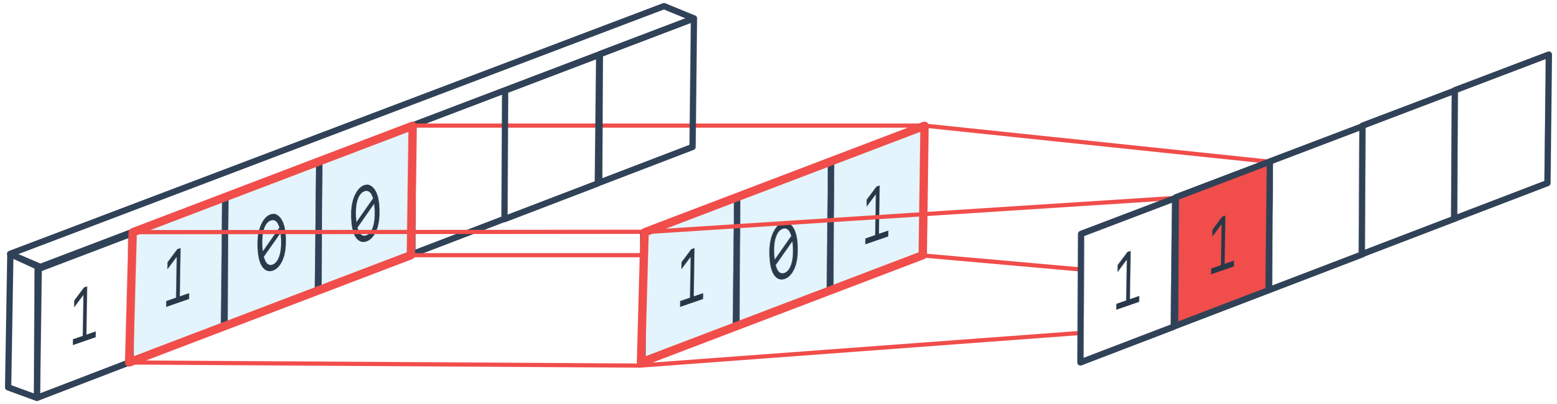
For each tree, CIF:

- samples *k* time series intervals of random position and length;
- subsamples 8 of the 25 features randomly;
- calculates the features for each interval, concatenates them to form a new data set;
- builds a decision tree on the feature-transformed dataset.

ROCKET

ROCKET (Random Convolutional Kernel Transform) uses a large number of random convolutional kernels to transform the time series:

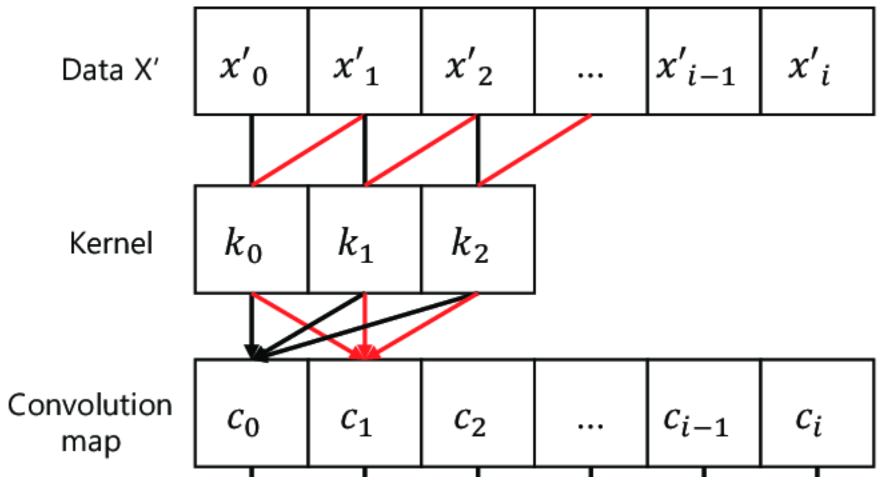
- all the parameters of all the kernels are randomly generated from fixed distributions;
- the transformed features are used to train a linear classifier (Logistic Regression or Ridge Regression Classifier);
- the combination of Rocket and logistic regression forms a single-layer convolution with random kernel weights with a trained softmax layer.



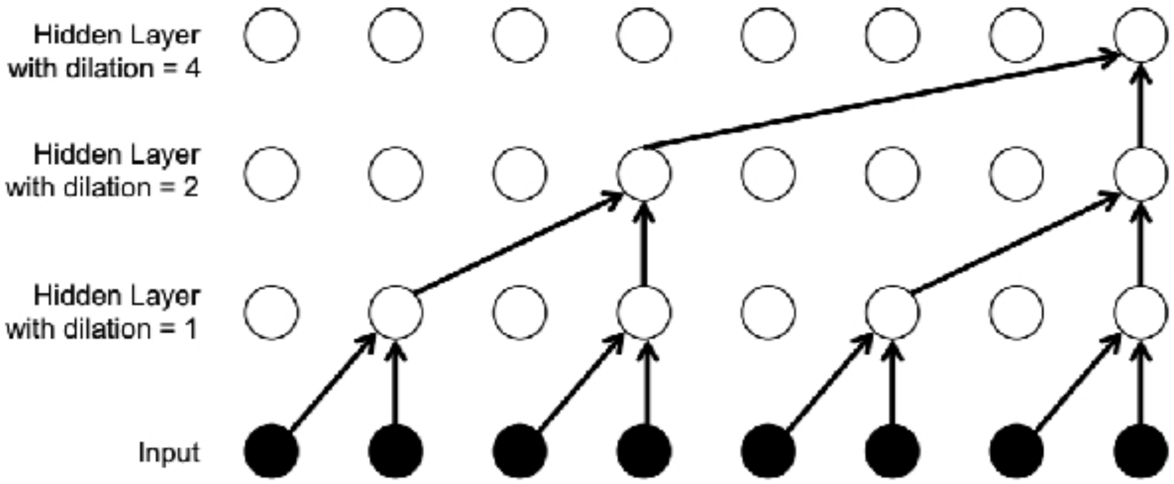
ROCKET vs. CNN

CNNs use <u>trainable</u> filters/kernels optimized by stochastic gradient descent to find patterns in the input data. Rocket differs in the following ways:

- Only a <u>single</u> layer containing a very large number of <u>random</u> kernels.
- Variety of kernels: each kernel has random length, dilation, and padding, weights and biases.

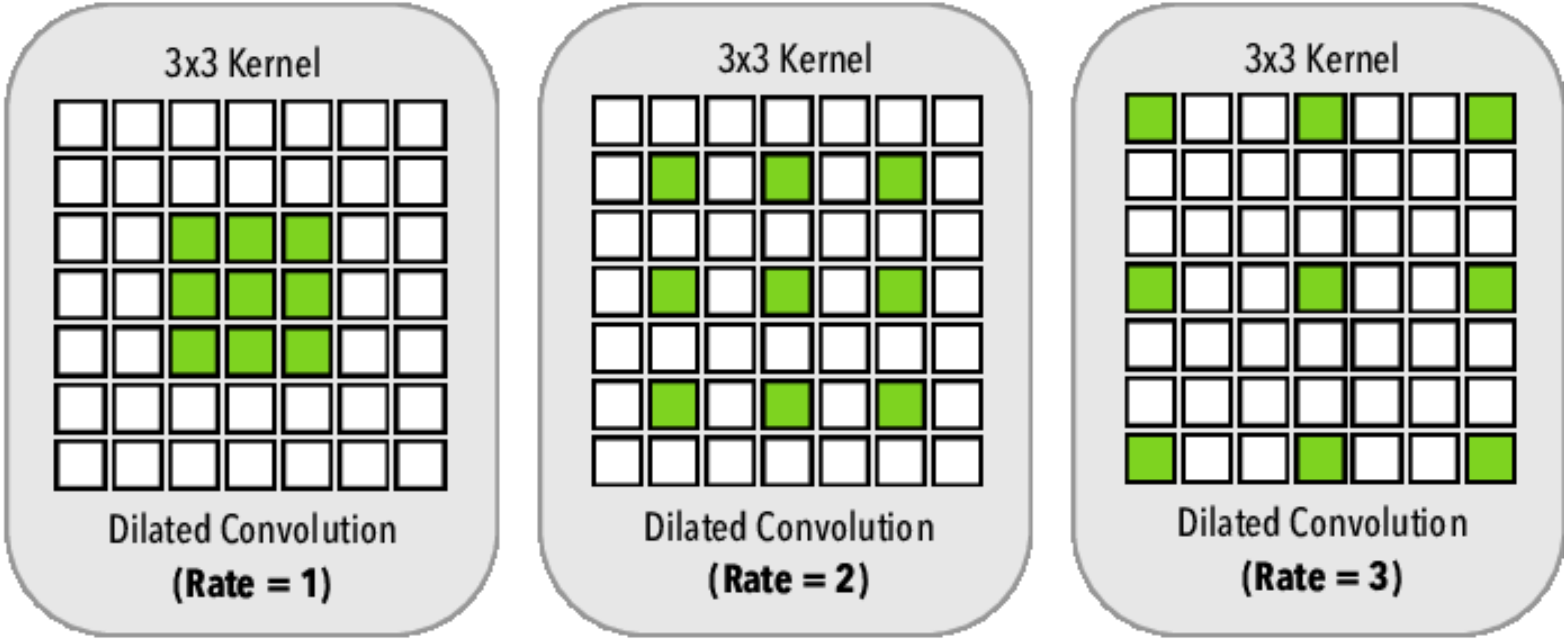


Example of Convolution



Example of Dilated Convolution

Dilated Convolution Kernels



ROCKET vs. CNN

- In CNNs kernel dilation increases exponentially with depth. Rocket sample dilation randomly for each kernel, capturing patterns at different frequencies and scales.
- Rocket uses the maximum value of the resulting feature maps (~global max pooling), and the proportion of positive values (proportion of the input which matches a given pattern).

- The only hyperparameter for Rocket is the number of kernels, *k*.
  - *k* handles the trade-off between classification accuracy and computation time

MINIROCKET

MiniRocket removes almost all randomness from Rocket, and dramatically speeds up the transform.

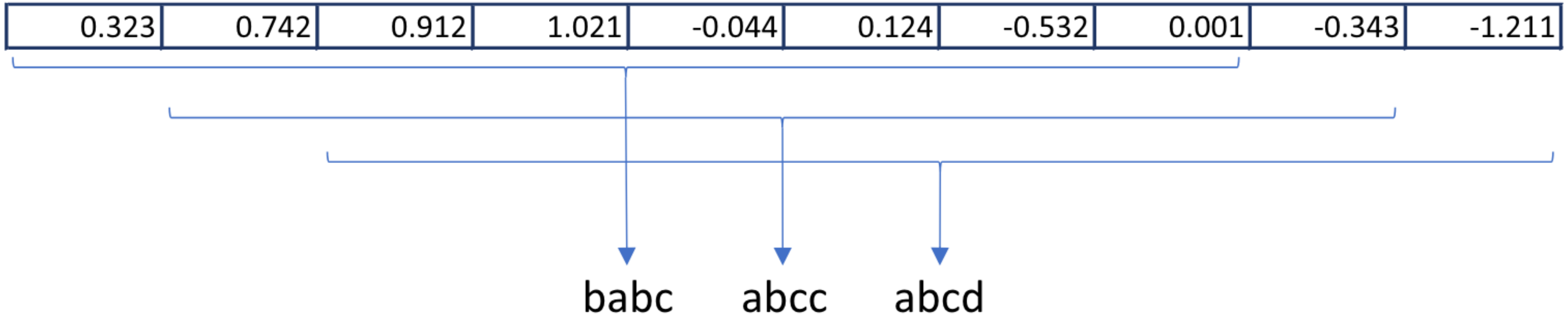
- Length: uses kernels of length 9.
- Weights: restricted to two values, $\alpha = -1$ and $\beta = 2$.
- Kernels: there are 512 possible two-valued kernels of length 9. Only subset of 84 is used.
- Bias: drawn from the quantiles of the convolution output for the entire training set (rather than a single, randomly-selected training example)
- Dilation: Each kernel is assigned the same fixed set of dilations, adjusted to the length of the input time series. The maximum number of dilations per kernel is 32
- Padding: half the kernel/dilation combinations use padding, and half do not.
- Features: only proportion of positive values.

COTE / HIVE-COTE / TS-CHIEF

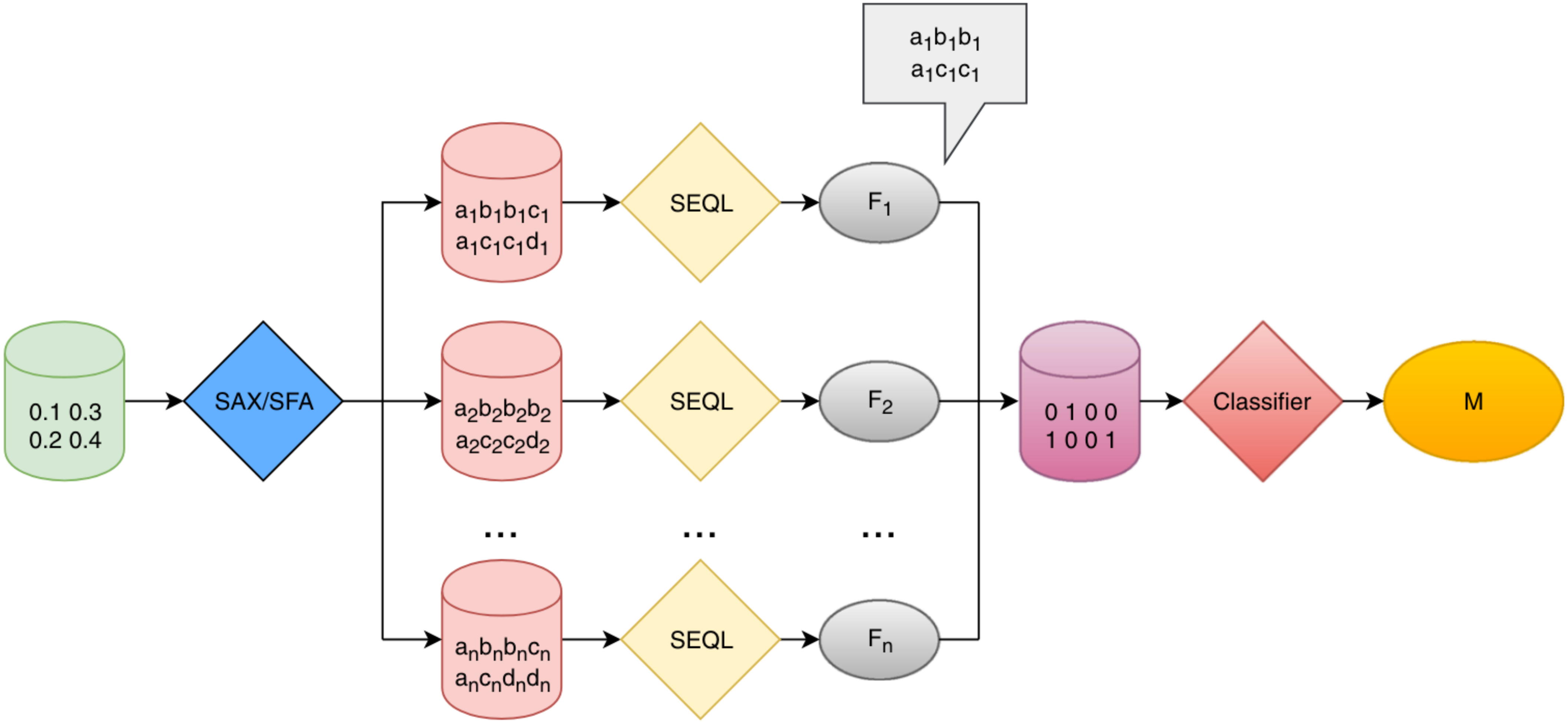
- Collective of Transformation-Based Ensembles (COTE) combines 35 classifiers over four data representations (similarity measures, shapelet-transform, autocorrelation features, power spectrum).
- Hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE) is an extension of COTE including more classifiers and a <u>hierarchical voting procedure</u>.
- Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) builds a random forest of decision trees whose splitting functions are time series specific and based on similarity measures, dictionary (bag-of-words) representations, and interval-based transformations.

MR-SEQL

- The data is discretized into sequences of words via either Symbolic Aggregate Approximation (SAX) or SFA, using a sliding window.
- The most discriminative symbols are extracted using a SEQuence Learner algorithm.
- The dataset is transformed in presence/absence of subsequences (similar to a shapelet transform)
- A linear (interpretable) model is trained on this new representation



MR-SEQL



0.1 0.3
0.2 0.4
SAX/SFA
a1b1b1c1
a1c1c1d1
a2b2b2b2
a2c2c2d2
...
anbnbncn
ancndndn
SEQL
SEQL
...
SEQL
F1
F2
...
Fn
a1b1b1
a1c1c1
0 1 0 0
1 0 0 1
Classifier
M

Ranking Multivariate TSC algorithms

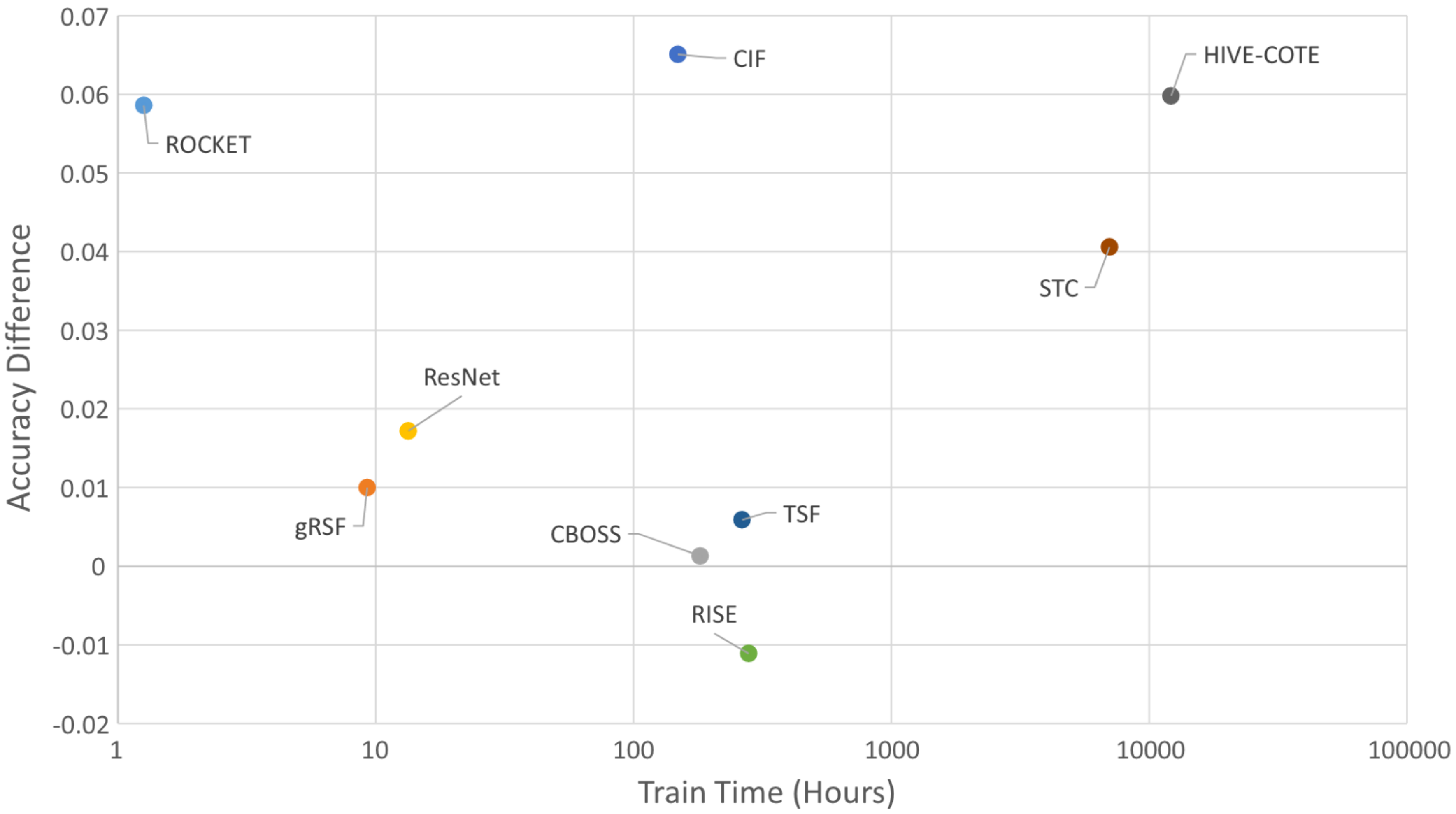
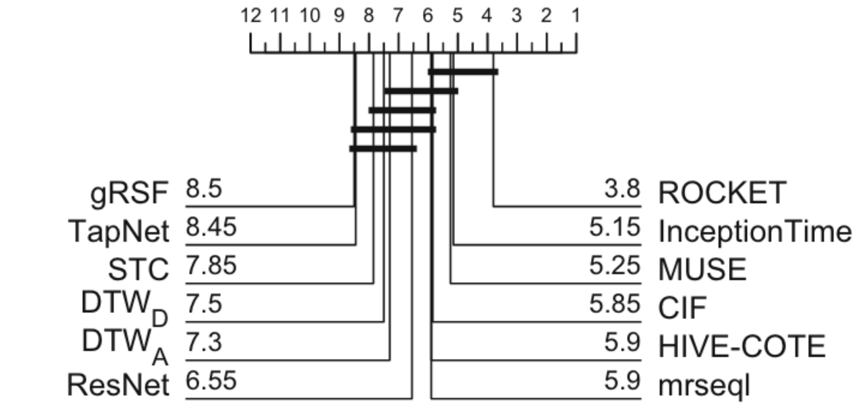
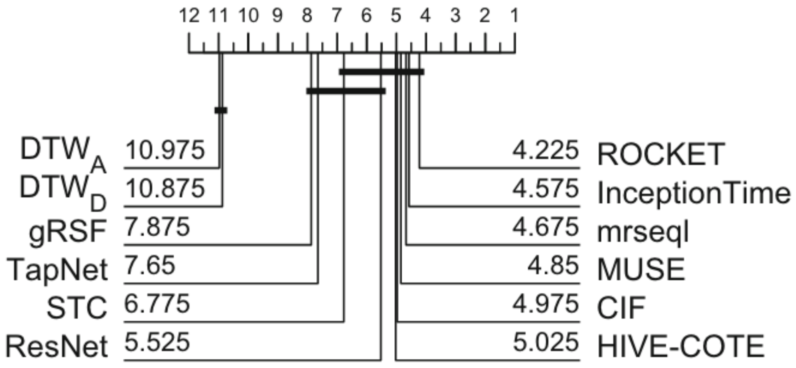


Fig. 10 Average difference in accuracy to DTW_D versus train time for 9 MTSC algorithms

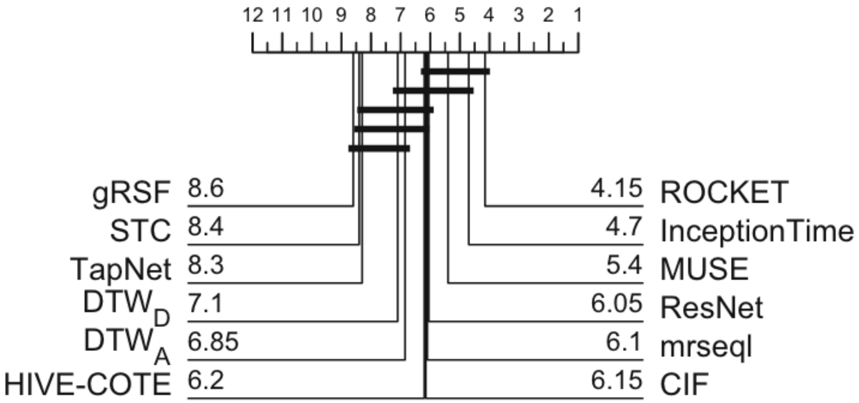
Ranking Multivariate TSC algorithms



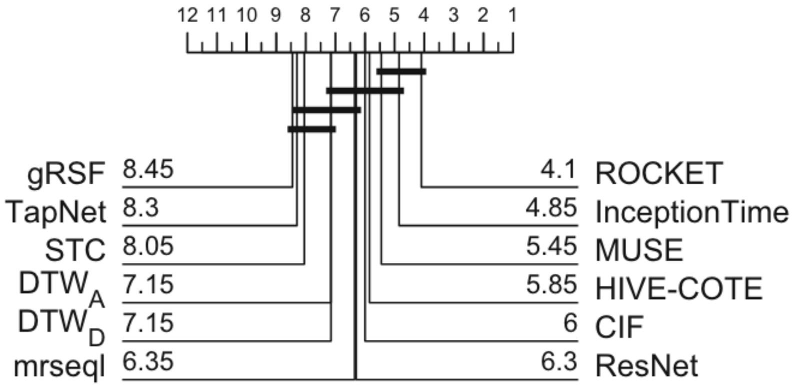
(a) Accuracy



(b) AUROC



(c) Balanced Accuracy



(d) F1

References

- [1] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, “TS-CHIEF: a scalable and accurate forest algorithm for time series classification,” *Data Min Knowl Disc*, vol. 34, no. 3, pp. 742–775, May 2020, doi: 10.1007/s10618-020-00679-8.
- [2] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, “On the Usage and Performance of the Hierarchical Vote Collective of Transformation-Based Ensembles Version 1.0 (HIVE-COTE v1.0),” in *Advanced Analytics and Learning on Temporal Data*, vol. 12588, V. Lemaire, S. Malinowski, A. Bagnall, T. Guyet, R. Tavenard, and G. Ifrim, Eds. Cham: Springer International Publishing, 2020, pp. 3–18. doi: 10.1007/978-3-030-65742-0_1.
- [3] T. Le Nguyen, S. Gsponer, I. Ilie, M. O’Reilly, and G. Ifrim, “Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations,” *Data Min Knowl Disc*, vol. 33, no. 4, pp. 1183–1222, Jul. 2019, doi: 10.1007/s10618-019-00633-3.
- [4] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: A strong baseline,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1578–1585. doi: 10.1109/IJCNN.2017.7966039.
- [5] M. Middlehurst, J. Large, and A. Bagnall, “The Canonical Interval Forest (CIF) Classifier for Time Series Classification,” in *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 188–195. doi: 10.1109/BigData50022.2020.9378424.
- [6] A. Dempster, D. F. Schmidt, and G. I. Webb, “MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification,” *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 248–257, Aug. 2021, doi: 10.1145/3447548.3467231.
- [7] A. Dempster, F. Petitjean, and G. I. Webb, “ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels,” *Data Min Knowl Disc*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020, doi: 10.1007/s10618-020-00701-z.
- [8] J. Faouzi, “Time Series Classification: A review of Algorithms and Implementations,” *Machine Learning*, p. 35.
- [9] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, “The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Min Knowl Disc*, vol. 35, no. 2, pp. 401–449, Mar. 2021, doi: 10.1007/s10618-020-00727-3.
- [10] H. I. Fawaz *et al.*, “InceptionTime: Finding AlexNet for Time Series Classification,” *Data Min Knowl Disc*, vol. 34, no. 6, pp. 1936–1962, Nov. 2020, doi: 10.1007/s10618-020-00710-y.