SEMARAK ILMU PUBLISHING

Semarak Engineering Journal

Journal homepage:
https://semarakilmu.my/index.php/sej/index
ISSN: 3036-0145



Unifying Sample, Feature, and Temporal Attention with KNN, SVM, and Sparse Coding

Mohamed Waleed Fakhr[1,*]

[1] School of Computing, Coventry University TKH Branch, New Administrative Capital, Cairo, Egypt

ARTICLE INFO

Article history:
Received 4 July 2026
Received in revised form 10 September 2026
Accepted 10 October 2026
Available online 22 November 2026

Keywords:

Attention mechanisms; Sparsemax; Time series classification; Support vector machines; Sparse coding

ABSTRACT

This paper presents a unified framework demonstrating that k-Nearest Neighbors (KNN), Sparse Coding, Support Vector Machines (SVM), and attention mechanisms all compute the same weighted voting equation, differing only in how the weights are calculated. A Triple-Attention architecture is introduced, combining temporal, feature, and sample attention with both softmax and sparsemax normalization for multivariate time series classification. Two output modes are proposed: Vote mode, which performs direct label voting, and Head mode, which uses a learned projection. Experiments on seven benchmark datasets spanning six domains show that attention is competitive with classical and neural baselines on larger datasets and that KNN, SVM, and Sparse Coding are special cases of sample attention under increasingly restrictive constraints. Sparsemax attention selects only 1.8 to 4.5 active samples per query, while attention degrades much more gracefully than SVM under severe input noise.

1. Introduction

Time series classification underpins applications including human activity recognition, medical diagnosis, and gesture recognition. Deep learning methods such as LSTMs achieve strong performance on these tasks but often lack interpretability and a clear theoretical connection to classical pattern-recognition methods.

This paper shows that classical methods, namely k-Nearest Neighbors (KNN), Sparse Coding, and Support Vector Machines (SVM), together with modern attention mechanisms, are all instances of the same computational primitive: weighted voting over training labels. Sample attention is shown to be the most general form of this primitive, with KNN, SVM, and Sparse Coding arising as special cases obtained by imposing progressively stronger constraints on the attention weight vector.

The main contributions of this paper are as follows. First, a unified framework is presented showing that KNN, SVM, Sparse Coding, and Attention all compute the same weighted-voting equation, with sample attention as the general case and the classical methods as constrained special

* *Corresponding author.*
E-mail address: waleed.fakhr@tkh.edu.eg

https://doi.org/10.37934/sej.17.1.116132

cases. Second, the V=Y principle is introduced: setting attention values equal to the training labels yields direct voting, recovering the master equation without any learned output projection. Third, a Triple-Attention architecture is proposed, combining temporal, feature, and sample attention. Fourth, comprehensive experiments are conducted on seven datasets from six domains, together with controlled studies of noise robustness, memory bank size, and training set size.

Beyond its theoretical interest, this unification has practical value for practitioners choosing among competing time series classifiers. In applied settings such as medical diagnosis, financial decision-making, and safety-critical sensing, the choice of classifier is rarely determined by accuracy alone; the availability of training data, the need for interpretability, and computational budget all play a role. A framework in which KNN, SVM, Sparse Coding, and Attention are recognized as points along a single constraint spectrum, rather than unrelated algorithm families, allows a practitioner to reason about that choice in terms of the constraints each method imposes on the weight vector.

1.1 Related Work

KNN assigns uniform weights to the k closest training samples. Sparse Coding finds reconstruction-based weights through an L1-regularized optimization. SVM learns kernel-weighted combinations of a subset of training samples known as support vectors. Transformer-style attention uses scaled dot-product similarity to compute dense weights, while sparsemax normalization produces sparse attention weights with exact zeros for low-scoring samples. Within time series analysis specifically, Crossformer models cross-dimension dependencies, iTransformer inverts the standard architecture design, and memory-augmented approaches incorporate global memory banks alongside local processing.

Beyond these individual method families, several recent works have explored pairwise connections between sparse coding, attention, and SVM. One line of work shows that self-attention is functionally equivalent to a form of sparse coding, integrating a sparse-attention module with an LSTM for medical image segmentation. A separate line of work merges attention with the margin-maximization principle of SVM to improve classification robustness on noisy EEG signals, addressing the limitation that standard attention does not by itself guarantee optimal class separation. These works share a common strategy of combining sparse coding, attention, and SVM as separate components within a processing pipeline. The present work differs fundamentally: rather than combining these methods as pipeline stages, it shows that they are all instances of the same underlying equation, differing only in the constraints placed on the weight vector.

Standard deep learning baselines for time series classification include fully convolutional networks, residual networks, and LSTM variants. Classical methods including DTW-based KNN, symbolic representations such as BOSS, and random-convolution methods such as ROCKET remain competitive on many benchmarks. None of these standard baselines are themselves framed within the weighted-voting equation used in this paper; LSTM-based and convolutional classifiers, for instance, typically produce a prediction through a final dense layer over a learned global representation rather than through an explicit weighted vote over labeled training examples. Such methods are included in the experiments as external reference points rather than as members of the same theoretical family.

2. Methods and Materials

2.1 Unified Theoretical Framework

All sample-based classifiers can be written as a weighted vote over training labels:

$$\hat{y}(x) = \sum_{i=1}^{N} \alpha_i(x) \cdot y_i$$

where *x* is the query, the training set consists of *N* labelled pairs, and the weight assigned to training sample *i* depends on the query. Sample attention, which learns these weights end-to-end through parameterized projections, is the most general instantiation of this equation. Classical methods are obtained by imposing specific constraints on how the weights are computed, ordered below from the most general to the most constrained.

Softmax attention, the most general case, computes the weights as:

$$\alpha_i = \mathrm{softmax}\left(\frac{q^\top k_i}{\sqrt{d}}\right)$$

producing dense weights in which every weight is strictly positive regardless of relevance. The similarity metric is fully learned through the query and key projections, the effective neighborhood size is unbounded, and all parameters are optimized end-to-end. No constraint is placed on the weight vector beyond normalization to a probability distribution.

Sparsemax attention adds a single constraint to softmax, replacing the normalization with a projection onto the probability simplex:

$$\alpha_i = \mathrm{sparsemax}\left(\frac{q^\top k_i}{\sqrt{d}}\right)$$

The weights are now sparse, with exact zeros for low-scoring samples. Like softmax, the similarity metric and all parameters are learned end-to-end; unlike softmax, sparsemax selects only the most relevant atoms from the sample dictionary, adaptively determining the effective number of neighbors per query. This sparsity constraint is what connects sparsemax attention to the three classical methods described next, each of which also produces sparse weight vectors but under additional constraints.

Sparse Coding computes weights by solving a reconstruction objective under an L1 penalty:

$$\alpha^{SC} = \arg\min_{\alpha} \left\| x - \sum_i \alpha_i x_i \right\|^2 + \lambda \|\alpha\|_1$$

This adds two constraints beyond sparsemax: sparsity is enforced through L1 regularization rather than simplex projection, and the weights must minimize a reconstruction objective, meaning the selected samples must be able to reconstruct the query. Unlike attention, weights are computed through iterative convex optimization at inference time rather than through learned projections, making sparse coding substantially slower.

SVM produces weights of the following form, where *K* is a kernel function:

$$\alpha_i^{SVM}(x) = \alpha_i \cdot K(x_i, x)$$

This adds further constraints: the similarity metric is a fixed kernel rather than a learned projection; the dual variables are optimized during training and remain fixed thereafter, with only the kernel evaluation varying per query; and sparsity arises from the KKT conditions, which enforce non-zero weights only for support vectors.

KNN, the most constrained case, applies uniform weights over a fixed neighborhood:

$$\alpha_i^{KNN}(x) = \frac{1}{k} \cdot \mathbb{1}[x_i \in N_k(x)]$$

Here the similarity metric is a fixed distance function with no learning, the neighborhood size (k) is a fixed hyperparameter, weights are uniform across selected neighbors, and there is no training phase at all. KNN is thus the zero-parameter extreme of the master equation, where every design choice is fixed *a priori* rather than learned from data.

Table 1
Generalization Hierarchy from Softmax Attention (Most General) to KNN (Most Constrained)

Method	Weights	k_eff	Similarity	Differentiable
Softmax Attention	Dense (all > 0)	N (all)	Learned	Yes
Sparsemax Attention	Sparse (adaptive)	Adaptive	Learned	Yes
Sparse Coding	Sparse (L1)	Adaptive	Reconstruction	Per-query
SVM-RBF	Sparse (support vectors)	Fixed after training	Kernel	Partial
KNN	Uniform (k)	Fixed	Distance	No

This hierarchy makes a concrete empirical prediction: more general methods should benefit from additional training data because they must learn more parameters, while constrained methods should be more robust when data is scarce, since their built-in constraints act as implicit regularization. Softmax and sparsemax attention, with roughly 120K learned parameters, sit at the general end of this spectrum; KNN, with zero parameters, and SVM, with N dual variables, sit at the constrained end. Section 3 examines this prediction directly.

The decision function of an SVM can be rewritten to reveal an attention-like structure:

$$f^{SVM}(x) = \sum_i \alpha_i \cdot K(x_i, x) \cdot y_i$$

where the kernel plays the role of query-key similarity and the labels serve as value vectors, consistent with the V=Y principle introduced below. The inference step of sparse coding likewise instantiates the master equation, with the reconstruction-based weights subsequently used to vote over labels as in Sparse Representation-based Classification:

$$\hat{x} = \sum_i \alpha_i^{SC} x_i \,, \qquad \hat{y} = \sum_i \alpha_i^{SC} y_i$$

In both cases, the classical method can be read as sample attention subject to an additional, method-specific constraint on how the weights are computed.

Setting the attention value matrix equal to the training labels yields direct weighted voting, which is precisely Equation (1) and requires no output projection. This V=Y principle makes explicit what was always implicit in the classical methods: KNN votes uniformly over neighbor labels, SVM votes with kernel-weighted support-vector labels, and sparse coding votes with L1-regularized reconstruction weights over labels. All are instances of V=Y, differing only in how the weights are computed. Theoretically, this shows that the distinction between classical and deep-learning methods is one of degree of constraint rather than of kind.

2.2 Proposed Method: Triple-Attention Machine

The Triple-Attention Machine processes a multivariate time series X with T timesteps and F features through three attention stages. Temporal attention aggregates information across timesteps:

$$h_{temp} = \sum_t \alpha_t^{temp} \cdot x_t$$

Feature attention models cross-feature interactions:

$$\tilde{x}_f = \sum_{f'} A_{f,f'} \cdot x_{f'}$$

Sample attention compares the query representation to memory-bank samples and implements the core weighted voting of the master equation:

$$\alpha_i^{sample} = \mathrm{norm}\left(\frac{(W_q h)^\top (W_k x_i)}{\sqrt{d}} \right)$$

where norm is either softmax, giving dense weights, or sparsemax, giving the sparsity constraint that connects to the classical methods.

Two output modes are supported. Vote mode implements the master equation directly, with no learned output projection:

$$\hat{y} = \sum_{i=1}^{M} \alpha_i^{sample} \cdot y_i$$

where M is the memory-bank size. The prediction is a weighted average of one-hot label vectors, yielding class probabilities with no additional trainable parameters at the output stage, which makes Vote mode robust to overfitting on small datasets. Head mode instead aggregates learned hidden representations and passes them through a two-layer MLP classifier with a ReLU nonlinearity and dropout:

$$h = \sum_i \alpha_i^{sample} \cdot W_v \,\mathrm{enc}(x_i)\,, \qquad \hat{y} = \mathrm{MLP}(h)$$

Head mode adds roughly 18K parameters over Vote mode for the settings used here, which is beneficial with sufficient training data but can lead to overfitting on small datasets.

Figure 1 summarizes the architecture. The input is patch-embedded, passed through temporal and feature attention to produce a query representation, and compared against an encoded memory bank via sample attention. The resulting weights either vote directly over labels in Vote mode or aggregate learned value vectors followed by an MLP head in Head mode.

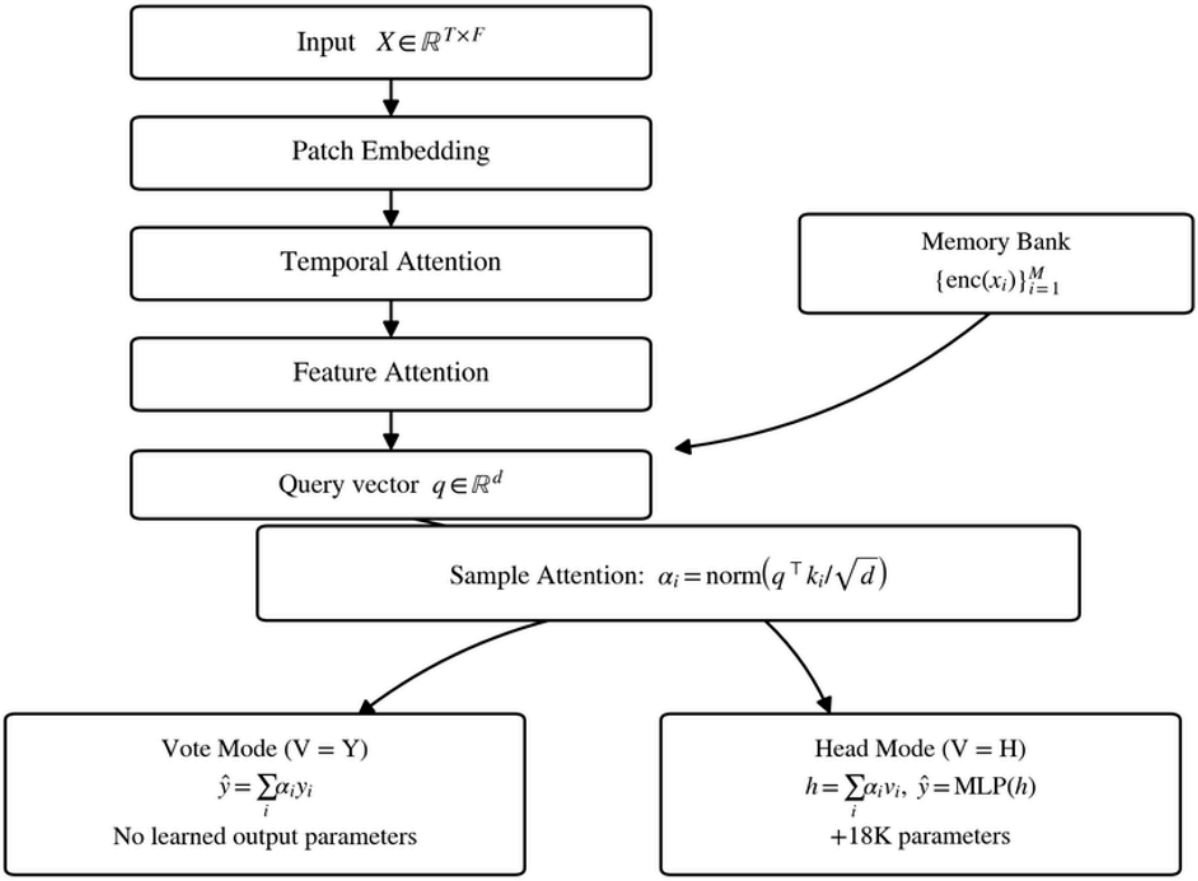


Fig. 1. Triple-Attention architecture. Vote mode directly implements the master equation; Head mode adds a learned nonlinear projection

Three attention combinations are evaluated to isolate the contribution of each axis. FS applies feature attention before sample attention, bypassing temporal modeling. TS applies temporal self-attention over patches before sample attention, without explicit feature weighting. FTS applies all three attention stages. Sample attention is always present, since it implements the core weighted voting of the master equation.

2.3 Experimental Setup

Seven multivariate time series classification benchmarks spanning six application domains are used, as summarized in Table 2. The first five, namely HAR, PenDigits, WISDM, Epilepsy, and NATOPS, are standard benchmarks drawn from the UEA and UCI archives. LSST is a 14-class astronomical light-curve classification problem from the Photometric LSST Challenge, providing a medium-large dataset that fills the gap between the small and large regimes. EURUSD is a financial time series dataset

constructed from daily EUR/USD exchange-rate data with 82 technical-analysis features, where each sample is a sliding 20-trading-day window and the binary target indicates whether the next-day high price increases or decreases. Unlike the other datasets, EURUSD uses a chronological 70/10/20 percent split to avoid future data leakage.

Table 2
Dataset Characteristics (T = Timesteps, F = Features, C = Classes, N = Training Samples)

Dataset	N	Test	T	F	C	Domain
HAR	7,352	2,947	128	9	6	Wearable
PenDigits	7,494	3,498	8	2	10	Handwriting
WISDM	6,822	2,566	200	3	6	Wearable
EURUSD	3,110	889	20	82	2	Finance
LSST	2,459	2,466	36	6	14	Astronomy
Epilepsy	137	138	206	3	4	Medical
NATOPS	180	180	51	24	6	Gesture

Baselines include KNN, Random Forest, XGBoost, Sparse Coding, SVM with an RBF kernel, LSTM, and a multilayer perceptron. ROCKET, a random convolutional kernel transform followed by a ridge classifier, is included as a strong recent reference method from outside the weighted-voting family. All attention experiments use PyTorch with the hyperparameters listed in Table 3. Attention models use a patch embedding that divides the input time series into non-overlapping patches of size P, projecting each patch to a d-dimensional embedding with d equal to 128. The memory bank stores M representative training samples selected via class-stratified k-means clustering. Training uses a temperature warmup schedule that starts with soft attention and gradually sharpens toward the target sparsity.

Table 3
Hyperparameter Settings and Justification

Parameter	Value	Justification
Optimizer	Adam	Standard for attention models
Learning rate	0.001	Common default for Adam
Batch size	32	Balances memory usage and gradient noise
Epochs	100	Early stopping with a patience of 15 epochs
Embedding dimension	128	Sufficient capacity
Memory bank	200	Representatives selected via k-means
Patch sizes	2 to 20	Divides T evenly for each dataset
Attention normalization	Softmax / Sparsemax	Both evaluated for each dataset
Temperature warmup	2.0 to 0.1	Stability for sparse attention

Table 4 compares parameter counts across models. The Triple-Attention architecture occupies a middle ground between lightweight classical methods and larger neural models. Its parameter count is significantly larger than the N dual variables of an SVM, which is relevant to the data-efficiency questions examined in Section 3.

Table 4
Parameter Count Comparison

Model	Parameters	Type
KNN	0	Instance-based
SVM-RBF	N	Dual variables
Sparse Coding	N per query	L1 optimization
XGBoost	~12K	Tree-based
LSTM	~67 to 79K	End-to-end
Triple Attention	~107 to 140K	End-to-end
MLP	~3 to 158K	End-to-end
Random Forest	~205K	Tree-based

All attention experiments were implemented in PyTorch and trained on a single GPU. Classical baselines used standard scikit-learn implementations with hyperparameters tuned via cross-validation on a held-out validation split. Dataset splits followed the original benchmark protocols to allow direct comparison with previously reported results, with the exception of EURUSD, as described above. The complete experimental notebook is publicly available, as stated in the data and code availability section.

3. Results and Discussion

3.1 Main Results

Table 5 presents classification accuracy across all seven datasets, reporting the best result achieved by Triple Attention under both softmax and sparsemax normalization, together with the classical, neural, and ROCKET baselines.

Table 5
Classification Accuracy (%) Across Seven Datasets

Model	HAR	PenD.	WISDM	EURUSD	LSST	Epil.	NAT.
KNN	70.3	97.2	28.2	53.9	43.0	52.9	78.3
Random Forest	84.5	96.5	73.0	69.4	52.0	74.6	85.6
XGBoost	90.4	96.4	76.4	72.9	52.0	70.3	82.8
Sparse Coding	70.9	86.3	65.5	51.3	31.6	83.3	76.7
SVM-RBF	89.8	98.2	74.2	68.1	35.7	85.5	78.3
LSTM	90.7	98.2	85.3	73.6	56.0	54.4	51.1
MLP	90.1	97.2	76.6	70.4	50.1	58.7	79.4
ROCKET	93.4	97.9	91.6	64.2	64.2	98.6	86.7
Triple Attn (Vote)	91.2	97.2	78.8	74.9	55.3	74.6	70.0
Triple Attn (Head)	92.5	98.1	79.9	72.7	61.1	73.2	75.0

Triple Attention is competitive with the classical and neural baselines on most datasets, achieving the highest accuracy among them on HAR, EURUSD, and LSST, and nearly matching SVM-RBF on PenDigits. It is outperformed by LSTM on WISDM and by SVM-RBF on Epilepsy and NATOPS, the two smallest datasets in the collection.

ROCKET, added in response to reviewer feedback as a strong recent reference method, achieves the highest accuracy on five of the seven datasets: HAR, WISDM, LSST, Epilepsy, and NATOPS. The

margins are substantial on several of these, most notably Epilepsy, where ROCKET reaches 98.6 percent against 74.6 percent for the best attention configuration and 85.5 percent for SVM-RBF. Triple Attention remains ahead on EURUSD by a clear margin of 10.7 points and is marginally ahead on PenDigits. This outcome merits a direct statement: the contribution of this paper is the theoretical unification of KNN, SVM, Sparse Coding, and attention as constrained instances of a single weighted-voting equation, not a claim of state-of-the-art accuracy. ROCKET does not compute a weighted vote over labelled training examples and therefore lies outside the family of methods this framework describes. Its inclusion establishes an honest external reference point and indicates that random convolutional features remain a very strong approach for time series classification, particularly on small datasets where the attention model has insufficient data to learn its similarity metric.

3.2 Key Findings

Attention performance relative to training set size. Attention achieves its best relative results on the five datasets with more than 2,000 training samples and is outperformed by SVM-RBF on the two datasets with fewer than 200 training samples, namely Epilepsy, at 85.5 against 74.6 percent, and NATOPS, where Random Forest reaches 85.6 against 75.0 percent. This pattern is consistent with parameter efficiency: SVM optimizes N dual variables while attention optimizes roughly 120K weights. Section 3.5 examines whether training set size alone accounts for this pattern and finds that it does not.

Temporal attention is consistently the strongest axis. Across all seven datasets, the Temporal plus Sample combination is the best or near-best configuration on four of the seven datasets, namely PenDigits, LSST, Epilepsy, and NATOPS, while the full triple combination wins on HAR, WISDM, and EURUSD. This suggests that for most time series, learning which temporal positions matter is more important than learning which features matter, though adding feature attention provides consistent gains on datasets with rich multi-sensor or multi-indicator inputs.

Head mode scales to many classes; Vote mode suits binary tasks. Head mode outperforms Vote mode on multi-class datasets with sufficient training data, with the gap most pronounced on many-class problems: on LSST with 14 classes, Head mode reaches 61.1 percent while Vote mode reaches only 55.3 percent. On the binary EURUSD dataset, however, Vote mode achieves 74.9 percent compared with 72.7 percent for Head mode, suggesting that for two-class problems, direct label voting provides a cleaner signal. The advantage of Vote mode is also evident on Epilepsy, where direct voting avoids overfitting the MLP projection layer.

Domain-specific insights. The EURUSD financial dataset, with 82 pre-engineered technical-analysis features rather than raw sensor measurements, provides a distinct test case and is the one dataset where Triple Attention leads all baselines, including ROCKET, by a wide margin. The best configuration uses all three attention axes at 74.9 percent, with softmax normalization and Vote mode. The chronological train and test split also introduces distribution shift between time periods, a realistic challenge absent from the randomly split benchmark datasets.

3.3 Unified Framework Validation

Table 6 validates the master equation across methods, contrasting a large dataset, HAR, where learned attention performs well, with a small dataset, Epilepsy, where kernel-based weights are superior. All five rows compute the same weighted vote over training labels and differ only in how the weights are obtained.

Table 6
Unified Framework Validation. All Methods Compute a Weighted Vote Over Training Labels

Method	Weight computation	HAR	Epil.
KNN	Uniform over k neighbors	70.3	52.9
Sparse Coding	L1 optimization	70.9	83.3
SVM-RBF	Kernel similarity	89.8	85.5
Sparsemax (Vote)	Learned sparse	91.2	74.6
Softmax (Head)	Learned dense plus MLP	92.5	73.2

3.4 Empirical Sparsity Analysis

To empirically validate the generalization hierarchy, the effective number of active weights is measured for each method on PenDigits and HAR, using sparsemax attention with the full triple combination. For sparsemax, this is the number of non-zero attention weights per query; for SVM-RBF, it is the number of support vectors; for Sparse Coding, it is the mean number of non-zero coefficients per query. Table 7 presents the results.

Table 7
Empirical Sparsity Comparison. k_eff Is the Effective Number of Active Weights, and Pool Is the Total Number of Candidates Available

Method	k_eff (PenD.)	Pool (PenD.)	Spar.% (PenD.)	k_eff (HAR)	Pool (HAR)	Spar.% (HAR)
Softmax Attn	200	200	0.0	200	200	0.0
Sparsemax Attn	1.8	200	99.1	4.5	200	97.8
KNN (k=5)	5	6,370	99.9	5	6,250	99.9
Sparse Coding	8.3	6,370	99.9	279	6,250	95.5
SVM-RBF	1,174	6,370	81.6	3,248	6,250	48.0

The results confirm the hierarchy quantitatively. Sparsemax attention achieves the highest sparsity of any learned method, attending to fewer than five samples out of 200 on both datasets, which is sparser than the fixed neighborhood of KNN. SVM-RBF is the least sparse learned method, relying on 1,174 support vectors on PenDigits and 3,248 on HAR, reflecting its dependence on a large fraction of training samples to define the decision boundary. Sparse Coding falls between the extremes, with 8.3 active coefficients on PenDigits but 279 on HAR, where the high-dimensional nine-sensor input requires more dictionary atoms for accurate reconstruction.

Figure 2 visualizes the sample-attention behavior on PenDigits. The distribution of active weights shows that sparsemax selects one to two memory samples for the vast majority of queries, achieving KNN-like selectivity with learned rather than fixed similarity. The cross-class attention matrix reveals a near-perfect block-diagonal structure: queries overwhelmingly attend to memory samples of the same class, with a same-class attention ratio of 103.97, empirically confirming that the V=Y voting mechanism drives the model to learn class-discriminative similarity.

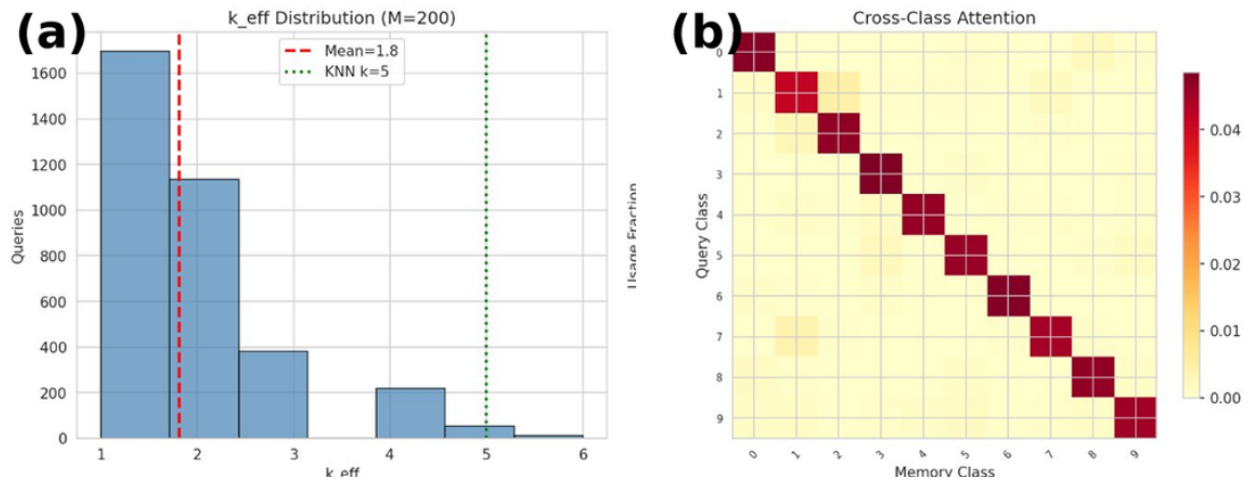


Fig. 2. Sample attention analysis on PenDigits (sparsemax, M = 200). (a) Distribution of the effective number of active weights across test queries, with the fixed k = 5 of KNN shown for reference. (b) Cross-class attention matrix showing mean attention from each query class to each memory class; the strong diagonal confirms same-class preference

To confirm that this behavior is not specific to PenDigits, the same analysis was repeated on HAR, which has six activity classes and nine sensor channels rather than two-dimensional pen trajectories. Sparsemax attention on HAR selects 4.2 samples per query on average, corresponding to 97.9 percent sparsity, and yields a same-class attention ratio of 45.8. This is lower than the 103.97 measured on PenDigits but remains far above the value of 1.0 that would indicate no class preference. The same-class preference underlying the V=Y mechanism therefore generalizes across datasets, while its magnitude is dataset-dependent: the clean and well-separated two-dimensional trajectories of PenDigits admit sharper class boundaries than the overlapping accelerometer and gyroscope signatures of HAR, where the seated and standing activities in particular are known to be difficult to separate.

Figure 3 presents the feature and temporal attention patterns learned on HAR. The feature attention heatmap reveals interpretable class-specific patterns: the laying activity relies almost exclusively on the total accelerometer channels, while dynamic activities distribute attention more broadly across body accelerometer and gyroscope channels. The class-discriminative ranking confirms that these two channels carry the highest cross-class variance. The temporal patch-importance analysis shows that laying concentrates attention on early patches, consistent with rapid sensor stabilization when a subject lies down, while walking activities attend more uniformly across the full timestep window.

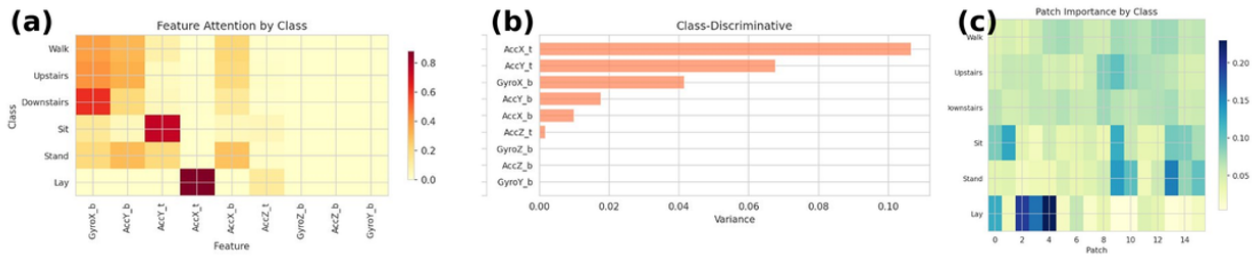


Fig. 3. Feature and temporal attention on HAR (T = 128, F = 9, C = 6). (a) Feature attention by class, showing mean attention weight per sensor for each activity. (b) Class-discriminative features ranked by cross-class variance. (c) Temporal patch importance by class

3.5 Training Set Size and the Absence of a Within-Dataset Crossover

The pattern reported in Section 3.2, in which attention performs well on the five larger datasets and is outperformed by SVM-RBF on the two smallest, invites a causal interpretation in terms of training set size. To test that interpretation directly, a controlled experiment was carried out in which the HAR training set was subsampled to 500, 1,000, 2,000, and 4,000 examples using class-stratified sampling, and Triple Attention and SVM-RBF were retrained at each size. Results are averaged over three random subsamples per size, and the memory bank was capped at one-third of the available training data so that it never exceeded the training set. Figure 4(a) presents the outcome.

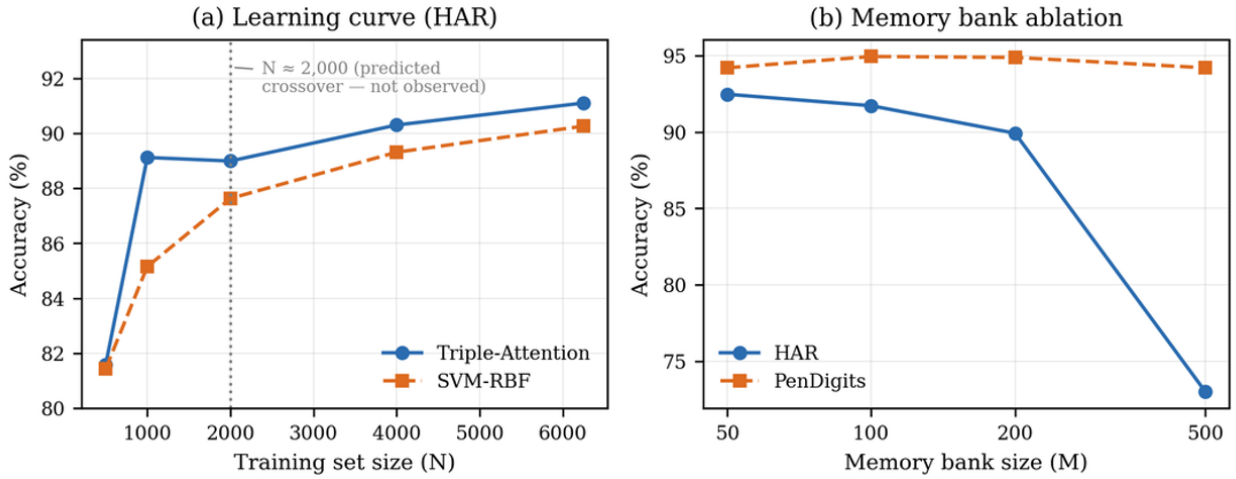


Fig. 4. (a) Learning curve on HAR. Triple Attention matches or exceeds SVM-RBF at every training set size tested, including sizes well below the predicted crossover point. (b) Effect of memory bank size M on accuracy for HAR and PenDigits

The expected crossover does not appear. Triple Attention matches SVM-RBF at 500 training samples, 81.6 against 81.4 percent, and is ahead at every larger size, with its widest margin of 4.0 points occurring at 1,000 samples, which is below rather than above the anticipated threshold. Reducing the HAR training set to a size comparable with the medium-sized datasets in the collection therefore does not reproduce the advantage that SVM-RBF holds on Epilepsy and NATOPS.

The correct conclusion is that training set size alone does not account for the pattern observed across datasets. Epilepsy and NATOPS differ from a subsampled HAR in more than sample count: both are smaller still, at 137 and 180 training samples, respectively, both come from different application domains, and both have different class structures and signal characteristics. Training set size is correlated with the advantage of the constrained methods in these experiments, but the present results do not establish it as the cause. Identifying which properties of the small datasets actually favour kernel methods, whether intrinsic dimensionality, class separability, or signal-to-noise characteristics, requires a study that varies these properties independently and is left to future work.

Figure 4(b) reports the effect of the memory bank size M, which was fixed at 200 for all main experiments. Values of 50, 100, 200, and 500 were evaluated on HAR and PenDigits using the best configuration for each dataset. Accuracy on PenDigits is essentially flat across the range, varying by less than one point between 94.2 and 94.9 percent, with a shallow optimum at M equal to 100. On HAR, accuracy decreases monotonically as the memory bank grows, from 92.5 percent at M equal to 50 to 89.9 percent at M equal to 200 and 73.0 percent at M equal to 500. The choice of 200 used throughout this paper is therefore conservative rather than tuned, and a smaller memory bank would have performed at least as well on both datasets. The degradation on HAR at the largest setting

suggests that as the number of k-means representatives grows, individual representatives become less class-pure and the sparse attention mechanism has more opportunity to select a mixed-class neighborhood, though establishing this mechanism would require a dedicated analysis.

3.6 Noise Robustness

Practical deployment of a time series classifier rarely involves clean signals. To assess robustness to input corruption, additive white Gaussian noise was applied to the HAR test set at three signal-to-noise ratios, while training was performed on clean data throughout. This corresponds to the realistic scenario in which a model trained under controlled conditions is deployed on noisier hardware or in a less controlled environment. The signal-to-noise ratio is defined in the usual way:

$$\mathrm{SNR}_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right)$$

Noise was calibrated to the mean signal power of the test set, and each condition was repeated with three random noise draws. Triple Attention in Head mode with the Temporal plus Sample combination was compared against SVM-RBF, the strongest classical competitor on this dataset. Figure 5 presents the results.

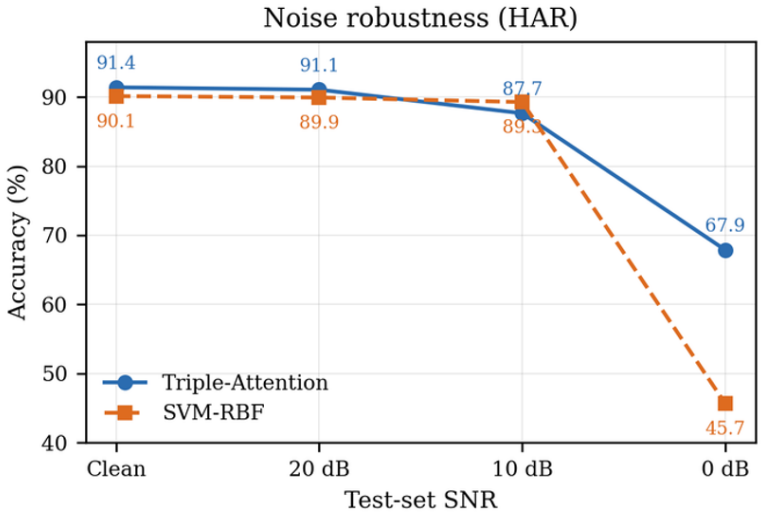


Fig. 5. Noise robustness on HAR. Accuracy under additive Gaussian noise applied to the test set at three signal-to-noise ratios, averaged over three noise draws. Training used clean data in all conditions

The two methods behave very differently as noise increases. At 20 dB, both are essentially unaffected, losing less than half a point relative to clean data. At 10 dB, the ordering reverses slightly, with SVM-RBF at 89.3 percent marginally ahead of attention at 87.7 percent, indicating that attention begins to degrade a little earlier under moderate corruption. At 0 dB, where signal and noise powers are equal, the two diverge sharply: Triple Attention retains 67.9 percent accuracy while SVM-RBF falls to 45.7 percent, a gap of 22.2 points. Expressed as a fraction of clean accuracy, attention retains 74 percent of its performance while SVM-RBF retains only 51 percent.

This difference is consistent with the constraint hierarchy developed in Section 2.1. The similarity metric of SVM-RBF is a fixed Gaussian kernel evaluated on raw Euclidean distances in the input space,

so severe corruption of the inputs directly corrupts every kernel evaluation and the support vector weights cannot adapt. Attention computes its similarity in a learned embedding space produced by patch embedding and temporal attention, and these learned projections appear to attenuate the noise before the similarity comparison takes place. The practical implication is that the additional parameters of the general case purchase robustness as well as accuracy, which is relevant for wearable and clinical sensing applications where sensor quality varies. It should be noted that this experiment corrupts the test set only; robustness to noisy training data is a separate question not addressed here.

3.7 Ablation Study

Softmax versus sparsemax. Figure 6 compares the two normalizations across all seven datasets, reporting the best accuracy achieved under each. Softmax wins on six of the seven datasets, but the margins are typically under 2.5 points, with the exception of NATOPS, where the gap reaches 5.0 points. The notable exception is PenDigits, where sparsemax achieves an improvement of 1.6 points. PenDigits consists of clean two-dimensional handwriting trajectories where focusing sharply on the most similar training samples is beneficial, which is precisely the KNN-like behavior that sparsemax encourages. On NATOPS, the very small training set provides insufficient samples for effective sparse selection, making the dense weighting of softmax preferable, and EURUSD similarly favours softmax, consistent with its high-dimensional correlated feature space where dense aggregation provides stability.

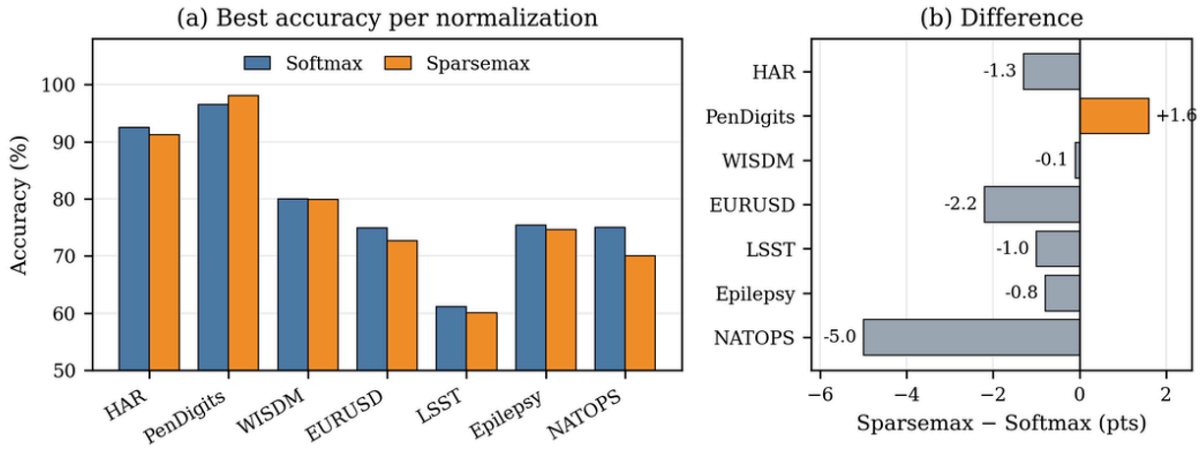


Fig. 6. Softmax versus sparsemax normalization. (a) Best accuracy per dataset under each normalization. (b) Difference in percentage points, with positive values indicating that sparsemax is preferred

Attention combination ablation. Figure 7 shows the contribution of each attention axis, using the best normalization for each dataset. The Temporal plus Sample combination is the single strongest, achieving the best results on three of six datasets, namely PenDigits, LSST, and Epilepsy. The full triple combination provides the best results on HAR, WISDM, and EURUSD, where the input features benefit from learned weighting. The Feature plus Sample combination alone is consistently the weakest, confirming that temporal attention is the primary driver across domains.

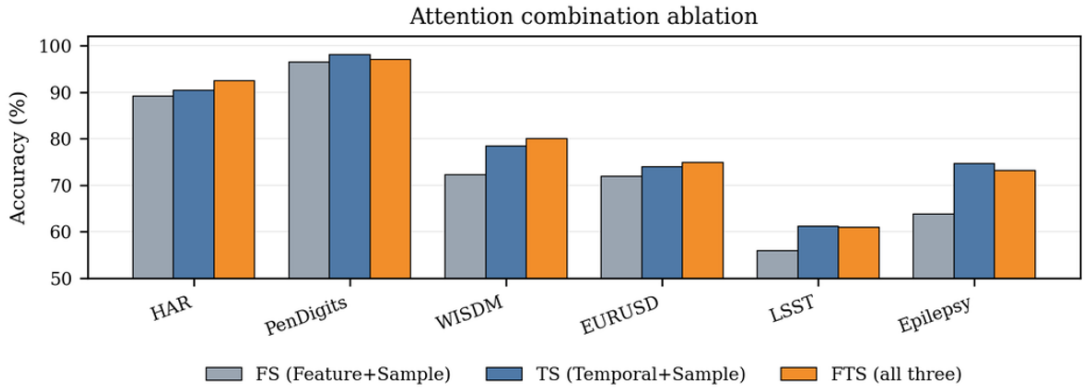


Fig. 7. Attention combination ablation. FS uses feature and sample attention, TS uses temporal and sample attention, and FTS uses all three axes

Vote versus Head mode. Vote mode performs better on small datasets, reaching 74.6 percent on Epilepsy against 73.2 percent for Head mode. Head mode dominates on multi-class datasets with sufficient data, with the gap widening as the number of classes grows: on LSST with 14 classes, Head mode achieves 61.1 percent while Vote mode reaches only 55.3 percent, since direct label voting across 14 classes from a 200-sample memory bank is inherently noisy. On the binary EURUSD dataset, the pattern reverses, with Vote mode achieving 74.9 percent against 72.7 percent for Head mode, suggesting that for binary classification, the direct weighted average of class labels already provides a smooth probability estimate that needs no further nonlinear transformation.

4. Conclusions

This paper presented a unified framework showing that KNN, Sparse Coding, SVM, and Attention all implement the same master equation, a weighted vote over training labels. The central insight is that sample attention is the most general form of this equation, with classical methods arising as progressively constrained special cases: sparsemax attention adds a sparsity constraint, sparse coding adds a reconstruction objective, SVM fixes weights after training using kernel-defined similarity, and KNN applies the strongest constraints of all, using uniform weights over a fixed number of neighbors under a predefined distance metric.

The proposed Triple-Attention architecture combines temporal, feature, and sample attention for multivariate time series classification. Experiments on seven datasets from six domains, together with controlled studies of noise, memory bank size, and training set size, support four main conclusions.

First, the sparsity hierarchy is confirmed quantitatively. Sparsemax attention attends to between 1.8 and 4.5 memory samples per query, which is sparser than the fixed neighborhood of KNN, while SVM-RBF requires between 1,174 and 3,248 support vectors on the same datasets. The same-class attention ratio is 103.97 on PenDigits and 45.8 on HAR, confirming that the V=Y voting mechanism drives class-discriminative similarity on both datasets while its magnitude depends on how cleanly separated the classes are.

Second, attention is more robust than SVM-RBF to severe input noise. At a signal-to-noise ratio of 0 dB on HAR, Triple Attention retains 67.9 percent accuracy against 45.7 percent for SVM-RBF. Computing similarity in a learned embedding space rather than through a fixed kernel on raw inputs appears to provide substantial protection when inputs are heavily corrupted, and the additional parameters of the general case therefore purchase robustness as well as accuracy.

Third, the advantage of attention on the larger datasets is not explained by training set size alone. Subsampling the HAR training set to sizes between 500 and 4,000 examples did not reproduce the advantage that SVM-RBF holds on Epilepsy and NATOPS; attention matched or exceeded SVM-RBF at every size tested. The pattern across datasets is therefore better described as an observation than as a demonstrated consequence of sample count, and identifying the dataset properties actually responsible remains open.

Fourth, the memory bank size of 200 used throughout is conservative. Reducing it to 50 or 100 leaves accuracy unchanged on PenDigits and improves it on HAR, indicating that the method does not depend on a large memory bank and can be made cheaper at inference time without loss.

Two limitations should be stated plainly. ROCKET, included as an external reference method, achieves higher accuracy than Triple Attention on five of the seven datasets, with a particularly large margin on Epilepsy. Random convolutional features remain a very strong approach to time series classification, and the contribution claimed here is the theoretical unification of the weighted-voting family rather than state-of-the-art accuracy. Separately, all experiments use a single-head formulation of sample attention in order to preserve the direct analogy with KNN and SVM; the reported numbers should be read as characterizing the unified framework rather than as an attempt to maximize accuracy against multi-head transformer architectures.

Several directions remain for future work. Multi-head attention could learn multiple notions of similarity, at the cost of diluting the clean analogy to KNN. Hierarchical memory banks organized by class or cluster could improve retrieval efficiency at larger scale. The V=Y principle could be extended to regression with continuous targets. A controlled study varying intrinsic dimensionality, class separability, and signal-to-noise ratio independently would help identify which dataset properties favour constrained methods over general ones. Finally, robustness to noise in the training data, as distinct from the test-time corruption studied here, remains to be examined.

Acknowledgements
This study was not supported by any grants from funding bodies in the public, private, or not-for-profit sectors. The author thanks the anonymous reviewers, whose comments prompted the noise robustness, memory bank, learning curve, and additional baseline experiments reported in Sections 3.1, 3.5, and 3.6.

Conflict of Interest
The author declares no conflicts of interest.

Author Contribution
M. W. Fakhr: Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Resources; Data curation; Writing, original draft; Writing, review and editing; Visualisation; Supervision; Project administration.

Data and Code Availability
HAR, PenDigits, WISDM, Epilepsy, and NATOPS are standard benchmarks available from the UEA and UCI multivariate time series archives. LSST is available from the Photometric LSST Challenge. The EURUSD dataset is available at https://github.com/MWaleedFakhr-Coventry-TKH/Forex-Slope-Prediction-and-Regime-Change-Detection/. The complete experimental notebook and code for reproducing all results reported in this paper, including the revision experiments of Sections 3.5 and 3.6, are publicly available at https://github.com/Dr-WaleedFakhr/Triple-Attention-Paper.

Reference

[1] Cover, T., and P. Hart. 1967. "Nearest Neighbor Pattern Classification." IEEE Transactions on Information Theory 13 (1): 21–27.
[2] Cortes, C., and V. Vapnik. 1995. "Support-Vector Networks." Machine Learning 20 (3): 273–297.
[3] Vaswani, A., et al. 2017. "Attention Is All You Need." In Advances in Neural Information Processing Systems (NeurIPS), 5998–6008.
[4] Martins, A., and R. Astudillo. 2016. "From Softmax to Sparsemax." In Proceedings of the International Conference on Machine Learning (ICML), 1614–1623.
[5] Tibshirani, R. 1996. "Regression Shrinkage and Selection via the Lasso." Journal of the Royal Statistical Society: Series B 58 (1): 267–288.
[6] Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." Neural Computation 9 (8): 1735–1780.
[7] Chen, T., and C. Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 785–794.
[8] Breiman, L. 2001. "Random Forests." Machine Learning 45 (1): 5–32.
[9] Anguita, D., et al. 2013. "A Public Domain Dataset for Human Activity Recognition Using Smartphones." In Proceedings of the European Symposium on Artificial Neural Networks (ESANN).
[10] Kwapisz, J. R., et al. 2011. "Activity Recognition Using Cell Phone Accelerometers." ACM SIGKDD Explorations 12 (2): 74–82.
[11] Bagnall, A., et al. 2018. "The UEA Multivariate Time Series Classification Archive." arXiv preprint arXiv:1811.00075.
[12] Alimoglu, F., and E. Alpaydin. 1996. "Methods of Combining Multiple Classifiers Based on Different Representations for Pen-Based Handwritten Digit Recognition." In Proceedings of the Turkish Artificial Intelligence Symposium.
[13] Chen, Y., et al. 2015. "The UCR Time Series Classification Archive."
[14] Wright, J., et al. 2009. "Robust Face Recognition via Sparse Representation." IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2): 210–227.
[15] Schölkopf, B., and A. Smola. 2002. Learning with Kernels. Cambridge, MA: MIT Press.
[16] Zhang, Y., and J. Yan. 2023. "Crossformer: Transformer Utilizing Cross-Dimension Dependency." In Proceedings of the International Conference on Learning Representations (ICLR).
[17] Liu, Y., et al. 2024. "iTransformer: Inverted Transformers Are Effective for Time Series Forecasting." In Proceedings of the International Conference on Learning Representations (ICLR).
[18] Cheng, Y., et al. 2024. "A Memory Guided Transformer for Time Series Forecasting." Proceedings of the VLDB Endowment 18 (2): 239–252.
[19] Wang, Z., et al. 2017. "Time Series Classification from Scratch with Deep Neural Networks." In Proceedings of the International Joint Conference on Neural Networks (IJCNN).
[20] Fawaz, H. I., et al. 2019. "Deep Learning for Time Series Classification: A Review." Data Mining and Knowledge Discovery 33: 917–963.
[21] Karim, F., et al. 2018. "LSTM Fully Convolutional Networks for Time Series Classification." IEEE Access 6: 1662–1669.
[22] Rakthanmanon, T., et al. 2012. "Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping." In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).
[23] Schäfer, P. 2015. "The BOSS Is Concerned with Time Series Classification." Data Mining and Knowledge Discovery 29: 1505–1530.
[24] Dempster, A., et al. 2020. "ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels." Data Mining and Knowledge Discovery 34: 1454–1495.
[25] Fakhr, W. 2025. "Forex Slope Prediction and Regime Change Detection." GitHub repository. https://github.com/MWaleedFakhr-Coventry-TKH/Forex-Slope-Prediction-and-Regime-Change-Detection/.
[26] Ji, Z., S. Ye, and X. Ma. 2024. "Sparse Coding Inspired LSTM and Self-Attention Integration for Medical Image Segmentation." IEEE Transactions on Image Processing 33: 6098–6113.
[27] Otarbay, Z., and A. Kyzyrkanov. 2025. "SVM-Enhanced Attention Mechanisms for Motor Imagery EEG Classification in Brain-Computer Interfaces." Frontiers in Neuroscience 19: 1622847.
[28] Nguyen, Q. M., et al. 2024. "Correlated Attention in Transformers for Multivariate Time Series." In Proceedings of the International Conference on Learning Representations (ICLR).