MTSCCleav: a Multivariate Time Series Classification (MTSC)-based method for predicting human Dicer cleavage sites

First Author[1,2*], Second Author[2,3†] and Third Author[1,2†]

[1*]Department, Organization, Street, City, 100190, State, Country.
[2]Department, Organization, Street, City, 10587, State, Country.
[3]Department, Organization, Street, City, 610101, State, Country.

*Corresponding author(s). E-mail(s): iauthor@gmail.com;
Contributing authors: iiauthor@gmail.com; iiiauthor@gmail.com;
†These authors contributed equally to this work.

Abstract

Background: MicroRNAs (miRNAs) are small non-coding RNAs (ncRNAs) that regulate gene expression at the post-transcriptional level, thereby playing essential roles in diverse biological processes. The biogenesis of miRNAs requires dicer to cleave at specific sites on the precursor miRNAs (pre-miRNAs). Several machine learning approaches have been proposed to predict whether an input sequence contains a cleavage site. However, they rely heavily on complex feature engineering or opaque deep neural networks. It results in a lack of generalizability and a long running time. There is a need for an alternative modeling paradigm that is accurate, fast, and simple.
Results: We proposed a novel approach to frame the task as a multivariate time series classification problem. Various encoding methods have been proposed to convert the sequence and the predicted secondary structure into a time series. We also leveraged the probabilities of the base pairs in the predicted secondary structure. Computational experiments demonstrate that our proposed method can achieve better or comparable results using a simpler, more intuitive model and less computational time. It achieves 3.7X $\sim$ 28.8X speedup. Through perturbation experiments, we found that regions close to the center of pre-miRNAs are essential for predicting human dicer cleavage sites.
Conclusion: By transforming the RNA sequence and its secondary structure information into a time series and utilizing simple, state-of-the-art time series classifiers, we achieved comparable or even superior performance in a simpler

and faster manner. Code is available at: https://github.com/cyuab/time-series-classification-cleavage.

Keywords: miRNA, Dicer Cleavage Site, Genomic signal processing (GSP), (Multivariate) time Series Classification (MTSC, TSC)

1 Background

One of the most important theories in molecular biology is the central dogma. It depicts the flow of genetic information [1, 2]. Proteins are the functional units. And the information stored in DNA is used to create them. Genes (segments) in DNA are used as templates for messenger RNAs (mRNAs) synthesis. An mRNA acts as a set of instructions to assemble a chain of amino acids, which form a linear polypeptide. To become biologically active, this chain is folded into a specific 3D structure, a proper configuration that enables it to perform its desired functions. And this folded polypeptide is called a functional protein, or simply a protein. This entire process closely resembles how a computer program runs on a machine. The source code does not function by itself. First, it is translated into assembly code (a lower-level, less human-readable form) and then into an executable file that can actually perform the intended tasks [3].

These mRNAs are called "coding RNAs" because they code for proteins. There are other genes in which the final product is the RNA molecule itself. They are called non-coding RNAs (ncRNAs). Two types of small ncRNAs are particularly important. They are microRNAs (miRNAs) and small interfering RNAs (siRNAs). Their discovery was recognized with the 2006 Nobel Prize in Physiology or Medicine[1], awarded for work completed only eight years prior [1].

In this study, we focus on miRNAs. An miRNA can regulate the expression of several proteins. Hence, understanding the biogenesis of miRNAs is of great value. It involves the processing of primary miRNAs (pri-miRNAs). RNAs are 3D molecules. However, it is hard to measure the 3D structure (tertiary structure) from the experiment and predict it from 1D sequence. We can understand their properties by analyzing their 1D sequence or 2D structure, known as secondary structure. RNA sequence is easily obtained through sequencing. The sequence and its predicted secondary structure of a pri-miRNA "hsa-let-7a-1" is shown in Fig. 1.

Recall that a pri-miRNA contains a hairpin loop, also called a stem loop. A microprocessor complex comprising Drosa and DCGR8 cleaves the pri-miRNA to form a precursor miRNA (pre-miRNA) inside the nucleus. The stem-loop is still preserved, but the two arms become shorter. After that, the pri-miRNA is transported by Exportin 5 from the nucleus to the cytoplasm. It is further cleaved by an enzyme called dicer [4]. Dicer cleaves the stem-loop from the two arms at the two cleavage

[1]The Nobel Prize in Physiology or Medicine 2006 - NobelPrize.org: https://www.nobelprize.org/prizes/medicine/2006/summary/ (Accessed on: 2025-06-13).
[2]Its miRBase entry: https://mirbase.org/hairpin/MI0000060. (Accessed on: 2025-06-12).
[3]RNAfold web server: http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi. (Accessed on: 2025-06-12). The figure is viewed in "forna". This view option can be chosen on the website.

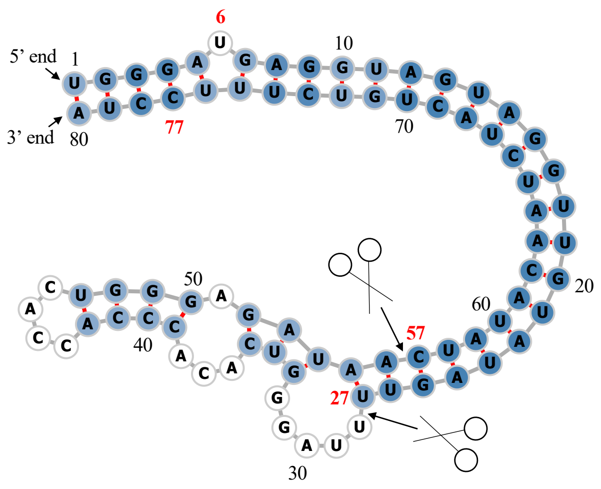


Fig. 1: Predicted secondary structure of the sequence S of pri-miRNA "hsa-let-7a-1"[2]. Experimental evidence suggests that the two deviated mature miRNAs are $UGA \cdots GUU$ and $CUA \cdots UUC$. They are $S(6:27)$ and $S(57:77)$ (Both ends are inclusive.). The ends are highlighted in **bold**. Since $S(6:27)$ ($S(57:77)$) is near the 5' (3') end, we call it "5p (3p) mature miRNA". The two scissors indicate the two cleavage sites. The color intensity of the nodes reflects their base pair probability in this predicted secondary structure. The deeper the color, the higher the probability. The unpaired nodes are uncolored. The raw figure is generated by RNAfold web server[3].

sites, shown as the two scissors in Fig. 1. The stem-loop is removed. It results in a short double-stranded miRNA molecule, known as an miRNA duplex, which consists of the 5p strand and the 3p strand[4]. These molecules may be subjected to additional trimming. The miRNA duplex is loaded into an RNA-induced silencing complex (RISC). RISC unwinds the duplex and tends to retain the strand with the less stable 5' end as the guide strand. The other strand is called the passenger strand. The retained strand guides the RISC to silence the target mRNA. Note that both strands can become the guide strand.

Dicer plays an important role in the biogenesis of miRNAs. It is reasonable to argue that the structure of the pre-miRNAs informs dicer about the cleavage process. It would be of great benefit to understand how dicer selects cleavage sites from the neighborhood information near the cleavage sites. Studies [5–7] revealed that the secondary structures are essential for cleavage site determination. Hence, to predict or classify whether a subsequence, extracted from the sequence of a pri-miRNA, contains a cleavage site, we can make use of both the sequence and secondary structure information. PHDcleav employed support vector machines (SVM), leveraging sequence and structure-based features [8] for the classification. LBSizeCleav improved upon

[4]The 5p strand comes from the 5' arm while the 3p strand comes from the 3' arm. For the directionality, the 5p (3p) strand retains the original 5' (3') end of the pre-miRNA.

it by considering the loop and bulge lengths [9]. [10] proposed an ensemble learning approach, using a gradient boosting machine for better accuracy. [11] developed a deep learning model, namely DiCleave. This model used an autoencoder to learn the secondary structure embeddings of pre-miRNAs from all the species in the miRBase database and leveraged this information. All these methods begin with curated pre-miRNA sequences from the miRBase database. Their secondary structures are predicted. Patterns are extracted from the sequence and the secondary structure. They create the positive cleavage patterns by setting the cleavage sites at the middle of the patterns. The follow-up work of [11], which created the cleavage pattern by allowing cleavage sites to appear at any position within the pattern, instead of the middle only [12]. It created a much larger dataset. This increased dataset facilitates the learning of the deep learning method at the cost of increased running time. We utilized the original dataset setting [8–11]. DiCleave is the current state-of-the-art (SOTA) for this problem with the original dataset setting.

These models suffer several limitations. They rely heavily on complicated feature engineering or opaque deep learning models [10–12]. It results in a lack of generalizability and a long running time. There is a need to design a simpler model so that it can be easily extended to other prediction tasks on RNA data. One way to analyze sequence data is to transform it into time series data. In response to this, we proposed a multivariate time series classification-based method. Our contributions are shown as follows.

1. To the best of our knowledge, we are the first to frame the prediction of the cleavage sites as a multivariate time series classification problem.
2. We introduced several encoding methods to convert RNA data to time series.
3. We proposed utilizing the base-pair probabilities in the predicted secondary structure for the prediction. To our surprise, this information has been ignored in the existing works.
4. We conducted perturbation-based experiments. It shows that regions close to the cleavage sites are important for this problem. It agrees with the existing study [10].

2 Methods

The overall pipeline of this study is summarized in Fig. 2.

2.1 Data Preparation

We used miRBase database [13][5]. The database comprises miRNA data from various organisms [14]. The database contains 38589 miRNA records. Each record refers to an miRNA sequence, along with other properties such as name, accession, organism, and information on its derivative miRNA products. We are interested in pri-miRNA in humans. The derivative miRNA products are the mature miRNAs. The database also annotates the location of the mature miRNA within the original sequence and indicates whether its existence has experimental evidence.

[5]The website is www.mirbase.org, and the newest version of the database is Release 22.1 (Accessed on 2025-06-22).

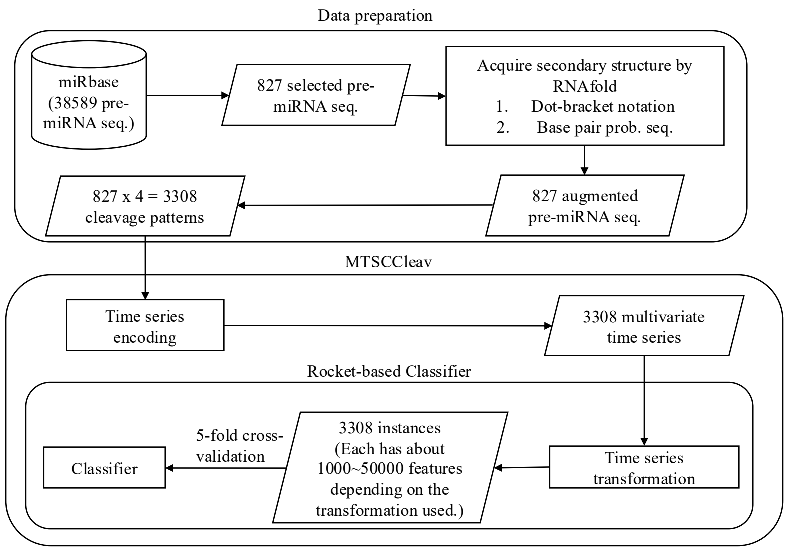

Data preparation
miRbase (38589 pre-miRNA seq.)
827 selected pre-miRNA seq.
Acquire secondary structure by RNAfold
1. Dot-bracket notation
2. Base pair prob. seq.
827 augmented pre-miRNA seq.
827 x 4 = 3308 cleavage patterns
MTSCCleav
Time series encoding
3308 multivariate time series
Rocket-based Classifier
Time series transformation
3308 instances (Each has about 1000~50000 features depending on the transformation used.)
5-fold cross-validation
Classifier

Fig. 2: The overall pipeline of this study. Symbol notations: Cylinder - Dataset, Rectangle - Process, Parallelogram - Input / Output, Rounded Rectangle - Component.

Accession	Name	Organism	Sequence	Mature miRNA 1	Mature miRNA 2
MI0000001	cel-let-7	Caenorhabditis elegans	$UACAC \cdots UUCGA$	cel-let-7-5p 17:38 experimental	cel-let-7-3p 60:81 experimental
MI0000060	hsa-let-7a-1	Homo sapiens	$UGGGA \cdots UCCUA$	hsa-let-7a-5p 6:27 experimental	hsa-let-7a-3p 57:77 experimental
MI0000114	hsa-mir-107	Homo sapiens	$CUCUC \cdots ACAGA$	hsa-miR-107 50:72 experimental	NA
MI0000238	hsa-mir-196a-1	Homo sapiens	$GUGAA \cdots UUCAC$	hsa-miR-196a-5p 7:28 experimental	hsa-miR-196a-1-3p 45:65 not experimental

Table 1: Selected representative records from miRBase. For the last two columns, the first line shows the name, the second line shows its location in the original sequence, and the third line indicates whether its existence has experimental evidence. The selected one is highlighted in **bold**.

Table 1 shows its four representative records. We first selected the records from humans (Homo sapiens). It resulted in 1917 records. To identify the actual locations of the two cleavage sites in the pri-miRNA sequence supported by experimental evidence, we selected records that have two mature miRNAs resulting from cleavage at the 5p arm and the 3p arm, both of which have experimental support. Hence, only "MI0000060" ("hsa-let-7a-1") would be selected in the table. It would serve as our running example. The whole sequence of it is listed in Table 2. After the selection process, we selected 827 experimental validated pre-miRNA sequences, each with its two mature miRNA products. This formed our dataset.

Sequence	Secondary Structure (In Dot-bracket notation)
1 UGGGA**UGAGGUAGUAGGUUGUAUAGUU** 27 28 UUAGGGUCACACCCACCACUGGGAGAU 54 55 AA**CUAUACAAUCUACUGUCUUUC**CUA 80	1 (((((**.(((((((((((((((((((((** 27 28 UUAGGGUCACACCCACCACUGGGAGAU 54 55))**)))))))))))))))))))))**))) 80
Base-pair probabilities sequence (the first 10 bases)	
1 (0.549, 0.946, 0.987, 0.987, 0.904) 5 6 (**0.000**, 0.841, 0.974, 0.981, 0.890) 10	

Table 2: The whole sequence of "hsa-let-7a-1" and its predicted secondary structure by RNAfold. The corresponding positions of the two mature miRNAs and the probability of the unpaired "U" are highlighted in **bold**.

2.1.1 Argument the dataset with Secondary Structure information

We leveraged the predicted secondary structure of these sequences to enhance the accuracy of the classification. Recall that a specific three-dimensional (3D) structure is required for DNA, RNA, and protein to perform functions [15]. However, finding these 3D structures using experimental methods such as X-ray crystallography or nuclear magnetic resonance (NMR) is costly and time-consuming. Hence, prediction methods for such 3D structures are necessary and helpful for downstream analysis. However, predicting the 3D structures is challenging. One of the reasons is that there are some "nonconventional" base-pair interactions (e.g., noncanonical and rare A-G) that allow an RNA sequence to fold into a 3D structure, in addition to the (G, U) wobble pair, which is common and functionally important in RNA secondary structures. It makes the search space for prediction much larger than, in the 2D case, the secondary structure. The local structures of the 3D structures, the secondary structures, only focus on the conventional base-pair interactions [2]. Hence, predicting secondary structures is easier and faster. We employed RNAfold from the ViennaRNA Package[6] to predict the secondary structure for a given pri-miRNA S [16]. RNAfold returns the secondary structure in the dot-bracket notation and a matrix of base-pair probabilities. The matrix is a square matrix with the side length $|S|$, where each entry m_{ij} is the probability of base s_i paired up with base s_j. Dot-bracket notation is a way of representing the secondary structure of S. Open parentheses "(" (Close parentheses ")") indicates that the base is paired with a complementary base further (earlier) along in S. Dot "." indicates that the base is unpaired. Equipped with the matrix, we can construct the base-pair probability sequence of S. The predicted secondary structure and the base-pair probability sequence of our running example are shown in Table 2.

[6]The latest stable release is Version 2.7.0 (Accessed on 2025-06-22).

2.1.2 Extract cleavage patterns

The locations of the two mature miRNAs on the whole sequence indicate the probable locations of the two cleavage sites. The 5p cleavage site must be beyond and near the ending location of the 5p mature miRNA. We deemed the immediate bond next to the 5p mature miRNA's ending position the 5p cleavage site, with the knowledge that the actual cleavage site may not be this immediate bond but rather the nearby bonds after it. The same applies to the 3p cleavage site. It is located at the immediate bond before the starting position of the 3p mature miRNA.

For each arm of each whole sequence, we extracted a 14-string[7] with the cleavage site located at the center of the string. The first 7 nt (nucleotide) before the center are highlighted in **bold**. In our running example, it would be "**UAUAGUU**UUAGGU" for the 5p cleavage site and "**GAGAUAA**CUAUACA" for the 3p cleavage site. We refer to these 14-strings as cleavage patterns. We also generate non-cleavage patterns by selecting a 14-string with the center 6 nt away from the corresponding cleavage sites towards the corresponding mature miRNA [9, 10] for each arm of each whole sequence. So, in our running example, the 5p non-cleavage pattern would be "**AGGUUGU**AUAGUUU". The 3p non-cleavage pattern would be "**ACUAUAC**AAUCUAC".

In conclusion, for a given pri-miRNA sequence, we can generate two cleavage patterns and two non-cleavage patterns. We call these four patterns simply the "four strings" of a given pri-miRNA. We also call each string a strand. The "four strings" of our running example are listed in Table 3.

	5p cleav	5p non-cleav	3p cleav	3p non-cleav
Input strand	UAUAGUUUUAGGGU	AGGUUGUAUAGUUU	GAGAUAACUAUACA	ACUAUACAAUCUAC
Complementary strand	AUAUCAA_____UA	UCUAACAUAUCAA_	C_CUGUUGAUAUGU	UGAUAUGUUGGAUG

Table 3: The first row shows the "four strings" of "hsa-let-7a-1". Their complementary strands are shown in the second row. As a whole, they are referred to as the "eight strings".

We can construct the complementary strand of each of the strands in the "four strings" by finding the corresponding paired base for each of the bases in the input strand by considering the secondary structure information. We use "_" to denote the unpaired base in the complementary strand. For example, in Fig. 1, "UUAGG" in the 5p cleavage pattern is unpaired, while other bases pair with some bases, the resulting complementary strand is "AUAUCAA_____UA". There is a loop/ budge there. We refer to the "four strings" and the four complementary strands together as the "eight strings" of the input pre-miRNA. It is also shown in Table 3.

[7]String with length = 14.

2.2 Time Series Encoding

A time series $T = t_1, t_2, ..., t_n$ is a sequence of real-valued numbers[8]. A short contiguous region of T is called a subsequence. A subsequence $T(i : j) = t_i, t_{i+1}, ..., t_j$ of a time series T is a shorter time series that starts from position i and ends at position j, where $i < j$.

Strings and time series are temporal sequences. The difference between strings and time series lies in their behavioral attributes [17]. For strings, an entry is a letter from a predefined set called the alphabet. For example, the alphabet is $\{A, C, G, T\}$ in the DNA string, while $\{A, C, G, U\}$ in the RNA string. For time series, an entry is a real number. Unlike real numbers, there is no ordering in the alphabet unless some external domain knowledge is introduced.

The study of applying signal processing techniques to genomic data is called "Genomic Signal Processing" (GSP) [18, 19]. In the field of GSP, the time series representations of DNA strings are referred to as DNA numeric representations (DNR). Many DNRs have been proposed. We noted that DNA strings and RNA strings are equivalent from a computational standpoint. Many transformation methods designed for DNA can be applied to RNA by simply substituting T with U. We present nine encoding methods. The relationship among them is shown in Fig. 3.

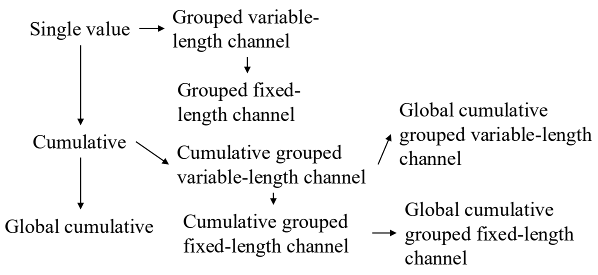

Single value
Grouped variable-length channel
Grouped fixed-length channel
Cumulative
Cumulative grouped variable-length channel
Global cumulative grouped variable-length channel
Global cumulative
Cumulative grouped fixed-length channel
Global cumulative grouped fixed-length channel

Fig. 3: Relationship of the proposed encoding methods.

2.2.1 Single value versus Cumulative

One of the simple, if not the simplest, encoding is to map the letters into numbers. Domain knowledge can be utilized. This approach is called the "Single value mapping" [18, 20–23]. One single value is assigned to each of the letters. [24] employed the atomic number of each nucleotide as the transformed values, where $\{G = 78, A = 70, C = 58, T = 66\}$. [25] used electron-ion interaction potential representation (EIIP) as such value, where $\{G = 0.0806, A = 0.1260, C = 0.1340, T = 0.1335\}$. Our goal is to transform the input strand and its complementary strand into time series, aiming to capture the information contained in these sequences and the secondary structure implied by them. We employed the following reasoning to assign the value:

[8]Unless otherwise specified, we denote entries of a time series (e.g., T) using the corresponding lowercase letter (e.g., t).

1. We employ the complementary property [22, 26] during encoding. Recall that in the base-pairing rules, G pairs with C to form three hydrogen bonds while A pairs with U[9] to form two hydrogen bonds. G-C pairs are more stable than A-U pairs. G (U) can be regarded as the "inverse" of C (A). We can preserve these base-pairing rules in the encoding by assigning G (A) and C (U) opposite values.
2. G and A have a two-ring structure. They are purines. C and U have a single-ring structure. They are pyrimidines. Hence, we put G and A (C and U) on the same side of the number line with zero in the middle.
3. The lower stability of A-U pairs promotes strand separation, thereby facilitating the unwinding of the miRNA duplex during RISC loading. Regions rich in A and U are thus more likely to undergo strand selection and cleavage events. We assigned A (U) with a larger absolute value than G (C) to reflect this functional relevance. It aims to highlight sequence regions with higher cleavage potential.

It results in our baseline transformation method, namely "Single value mapping" , shown in row 1 of Table 4. S is the input strand. When we encode S without incorporating the corresponding base-pair probability sequence P, we set $p_i = 1$ for all the entries of P. We use the first ten nucleotides of the complementary strand of the 3p cleav of "hsa-let-7a-1", as shown in Table 3 as S in the examples in Table 4.

With the assigned value to each nucleotide defined in single-value mapping, we can compute a cumulative sum of those values over time. It captures the aggregated signal by accumulating past events, allowing us to focus on the trend [27, 28]. We named this method as "Cumulative mapping", shown in row 4 of Table 4.

2.2.2 Grouped variable-length channel versus Grouped local-length channel

We can transform the input strand into a multivariate time series with two channels using grouped binary encoding, where nucleotides are grouped into (A, U) and (G, C). It releases our third assumption that A (U) has a larger absolute value than G (C). We proposed two variations. The first one allows the output to be variable-length sequences per channel, depending on group-specific occurrences. The second one always returns two resulting sequences of fixed length. Two variations extended from single value mapping are shown in rows 2 and 3, while those extended from cumulative mapping are shown in rows 5 and 6 in Table 4.

2.2.3 Global cumulative versus Local Cumulative

In cumulative mapping and its variations, we can choose where to start the accumulation. For a given subsequence S' of the whole sequence S, accumulation can start from the beginning of S even if only S' is used downstream. Or it can begin just at the start of the S'. The first one preserves the global context. It can be useful when previous nucleotides (those before S') influence later interpretation. The second one focuses solely on local history in S', ignoring global history. It is helpful if the previous nucleotides do not affect the chemical property of S'.

[9]In DNA, A pairs with T.

	Encoding	Algorithm	Example $S = C, \_, C, U, G, U, U, G, A, U$ $P = 0.843, 0.000, 0.807, 0.807, 0.793, 0.914, 0.982, 1.000, 0.999, 0.999$
1	Single value mapping [18, 20–23]	for $i = 1$ to $\|S\|$: $t_i = \begin{cases} 2 \cdot p_i & \text{if } s_i = A \\ 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \\ -2 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ return T	Without base-pair probability sequence: $T = -1, 0, -1, -2, 1, -2, -2, 1, 2, -2$ With base-pair probability sequence: $T = -0.843, 0.000, -0.807, -1.614, 0.793, -1.829, -1.963, 1.000, 1.999, -1.998$
2	Grouped variable-length channel mapping	$j = 1, k = 1$ for $i = 1$ to $\|S\|$: $t^1_j = \begin{cases} 1 \cdot p_i & \text{if } s_i = A \\ -1 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ $t^2_k = \begin{cases} 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \end{cases}$ if $(s_i = G)$ or $(s_i = C)$: increment k by 1 else: increment j by 1 return T^1, T^2	Without base-pair probability sequence: $T^1 = 0, -1, -1, -1, 1, -1$ $T^2 = -1, -1, 1, 1$ With base-pair probability sequence: $T^1 = 0.000, -0.807, -0.914, -0.982, 0.999, -0.999$ $T^2 = -0.843, -0.807, 0.793, 1.000$
3	Grouped fixed-length channel mapping	for $i = 1$ to $\|S\|$: $t^1_i = \begin{cases} 1 \cdot p_i & \text{if } s_i = A \\ -1 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ $t^2_i = \begin{cases} 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \\ 0 & \text{otherwise} \end{cases}$ return T^1, T^2	Without base-pair probability sequence: $T^1 = 0, 0, 0, -1, 0, -1, -1, 0, 1, -1$ $T^2 = -1, 0, -1, 0, 1, 0, 0, 1, 0, 0$ With base-pair probability sequence: $T^1 = 0.000, 0.000, 0.000, -0.807, 0.000, -0.914, -0.982, 0.000, 0.999, -0.9999$ $T^2 = -0.843, 0.000, -0.807, 0.000, 0.793, 0.000, 0.000, 1.000, 0.000, 0.000$
4	Cumulative mapping [27, 28]	$t_1 = 0$ for $i = 1$ to $\|S\|$: $t_{i+1} = \begin{cases} t_i + 2 \cdot p_i & \text{if } s_i = A \\ t_i + 1 \cdot p_i & \text{if } s_i = G \\ t_i - 1 \cdot p_i & \text{if } s_i = C \\ t_i - 2 \cdot p_i & \text{if } s_i = U \\ t_i & \text{otherwise} \end{cases}$ return T // $\|T\| = \|S\| + 1$	Without base-pair probability sequence: $T = 0, -1, -1, -2, -4, -3, -5, -7, -6, -4, -6$ With base-pair probability sequence: $T = 0.000, -0.843, -0.843, -1.650, -3.265, -2.471, -4.300, -6.263, -5.264, -3.265, -5.263$
5	Cumulative grouped variable-length channel mapping	$t^1_1 = 0, t^2_1 = 0$ $j = 1, k = 1$ for $i = 1$ to $\|S\|$: $t^1_{j+1} = \begin{cases} t^1_j + 1 \cdot p_i & \text{if } s_i = A \\ t^1_j - 1 \cdot p_i & \text{if } s_i = U \\ t^1_j & \text{if } s_i = \_ \end{cases}$ $t^2_{k+1} = \begin{cases} t^2_k + 1 \cdot p_i & \text{if } s_i = G \\ t^2_k - 1 \cdot p_i & \text{if } s_i = C \end{cases}$ if $(s_i = G)$ or $(s_i = C)$: increment k by 1 else: increment j by 1 return T^1, T^2	Without base-pair probability sequence: $T^1 = 0, -1, -2, -3, -2, -3$ $T^2 = 0, -1, -2, -1, 0$ With base-pair probability sequence: $T^1 = 0.000, -0.807, -1.722, -2.703, -1.704, -2.703$ $T^2 = 0.000, -0.843, -1.650, -0.857, 0.143$
6	Cumulative grouped fixed-length channel mapping	$t^1_1 = 0, t^2_1 = 0$ for $i = 1$ to $\|S\|$: $t^1_{i+1} = \begin{cases} t^1_i + 1 \cdot p_i & \text{if } s_i = A \\ t^1_i - 1 \cdot p_i & \text{if } s_i = U \\ t^1_i & \text{otherwise} \end{cases}$ $t^2_{i+1} = \begin{cases} t^2_i + 1 \cdot p_i & \text{if } s_i = G \\ t^2_i - 1 \cdot p_i & \text{if } s_i = C \\ t^2_i & \text{otherwise} \end{cases}$ return T^1, T^2 // $\|T^1\| = \|T^2\| = \|S\| + 1$	Without base-pair probability sequence: $T^1 = 0, 0, 0, 0, -1, -1, -2, -3, -3, -2, -3$ $T^2 = 0, -1, -1, -2, -2, -1, -1, -1, 0, 0, 0$ With base-pair probability sequence: $T^1 = 0.000, 0.000, 0.000, 0.000, -0.807, -0.807, -1.722, -2.703, -2.703, -1.704, -2.703$ $T^2 = 0.000, -0.843, -0.843, -1.650, -1.650, -0.857, -0.857, -0.857, 0.143, 0.143, 0.143$

Table 4: Time series encoding. P is the corresponding base-pair probability sequence of S. $p_i = 1$ if we encode S without base-pair probability sequence.

Consider $T = 0, -1, ..., -6$ of the input string S in "Cumulative mapping" in Table 4, which accumulates from 0. S is the suffix with length = 10 of the constructed complementary strand of $S(1:63)$ in Fig. 1. If we start the accumulation from the first entry of the constructed complementary strand instead, it will yield a different result. Suppose the last entry of the time series encoded in the cumulative mapping of the constructed complementary strand is 2, the time series encoded in the "Global cumulative mapping" for S would accumulate from 2 instead of 0. The result is $T = 2, 1, ..., -4$. Note that it has the same trend as the original T. This "Global cumulative" concept can be applied to every cumulative-based method, as shown in Fig. 3.

2.2.4 Incorporating base-pair probabilities

We can incorporate the base-pair probabilities P in the encoding by thinking of it as the weight or confidence p_i in the value assignment of each nucleotide s_i. It is implemented by multiplying the base-pair probability p_i of the nucleotide s_i with the assigned value of the kind of nucleotide of s_i during encoding, as shown in Table 4.

2.2.5 Transforming the secondary structure into time series

We can transform the secondary structure in the dot-bracket notation into a time series by "Single value mapping", where "(" maps to 1, "." maps to 0, and ")" maps to -1.

2.3 Time series classification

In univariate time series classification, an instance in the dataset consists of a time series $x = x_1, x_2, ..., x_m$ with m observations and a discrete class label y, which takes c possible values [29, 30]. If $c = 2$, we refer to binary classification. If $c > 2$, we refer to multi-class classification. In multivariate time series classification, the time series is not a single sequence but a list of sequences. Each sequence is called a channel. There are many classifiers defined for time series data, including distance-based, feature-based, interval-based, shapelet-based, dictionary-based, convolution-based, and deep learning-based classifiers. Additionally, two or more of the above approaches can be combined, resulting in hybrid approaches [29–31]. We employed convolution-based classifiers due to their simplicity and accuracy.

2.3.1 Convolution-based classifiers

Convolution-based classifiers first use randomly parameterized kernels to perform convolutions on the original time series T. A kernel is referred to as parameterized because its behavior is governed by a set of parameters, which will be discussed in detail later. Convolution is an operation to transform T to another time series M, where M is called the activation map. Its entry M_i is calculated by applying a kernel ω with length l to T at position i, defined as follows:

$$M_i = A(i : i + l - 1) * \omega = \sum_{j=0}^{l-1} A_{i+j} \cdot \omega_{1+j}$$

To note, $|A(i : i + l - 1)| = |\omega| = l$. Entries M_i's are calculated by sliding ω across T and computing a dot product. Additionally, although the original paper used the term "convolution" to refer to the above operation, "cross-correlation" may be a more suitable term for this operation. Recall T with length m has $(m-l+1)$ sliding windows of length l, given that the increment is 1[10], which defines the length of M.

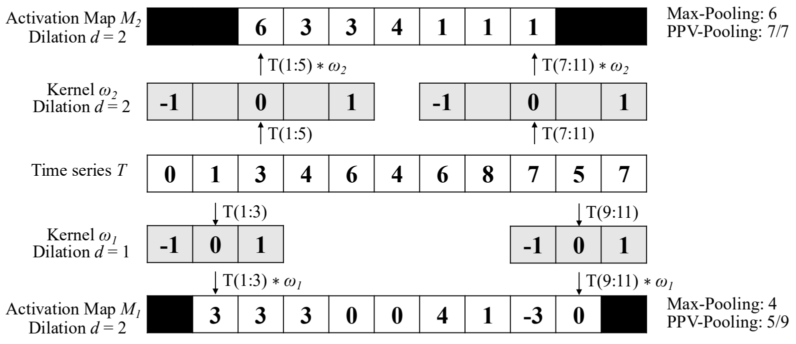


Fig. 4: Features generation in the transformation

Fig. 4 shows two kernels ω_1 and ω_2 with lengths 3 and 5, respectively. Each of which performs a convolution with T and returns two activation maps, M^1 and M^2, respectively. For example, $M^1_1 = T(1 : 3) * \omega_1 = 3$. By sliding ω_1 one time stamp at a time, an activation map M^1 with length $= (m - l + 1) = 11 - 3 + 1 = 9$ is obtained. Then, pooling operations, such as the maximum (MAX) and proportion of positive values (PPV), are applied on M^1 to derive the summary features. In Fig. 4, MAX and PPV are applied on M^1 and M^2. The summary features of M^1 are 4 and 5/9, which correspond to MAX and PPV, respectively. Dilation refers to a method that enables a kernel to cover a larger portion by creating empty spaces between entries in the kernel. The dilation d of ω_2 is 2. It introduces a gap of 1 in every two values of ω_2.

The most popular convolution-based approach is the Random Convolutional Kernel Transform (ROCKET) [32]. It generates a large number of randomly parameterized kernels, ranging from thousands to tens of thousands. The kernel's parameters include length, weights (the entries inside the kernel), bias (the value added to the result of the convolution operation), and dilation. Additionally, padding can be applied to T at the start and end, ensuring M has the same length as the input. To note, T, M_1, and M_2 in Fig. 4 have different lengths. The summary statistics of the activation map are obtained through two pooling operations: MAX and PPV. Hence, for k kernels, the transformed data has $2k$ features. The default value of k is 10000. The feature vector size is then 20000.

There are two extensions of ROCKET. They are MiniROCKET [33] and MultiROCKET [34]. MiniROCKET removes unnecessary operations and many of the random components in the definition of kernels used by ROCKET. It speeds up Rocket by over an order of magnitude with no significant difference in accuracy, making the

[10]One step to the right per time

classifier almost deterministic. For example, the kernel length is fixed, and only two weight values are used. Only PPV is used for the summary statistics. MultiROCKET is extended from MiniROCKET. The main improvement of it is to extract features from first-order differences as defined in Table 5 and add three new pooling operations [34]. The three added operations are mean of positive values (MPV), mean of indices of positive values (MIPV) and longest stretch of positive values (LSPV).

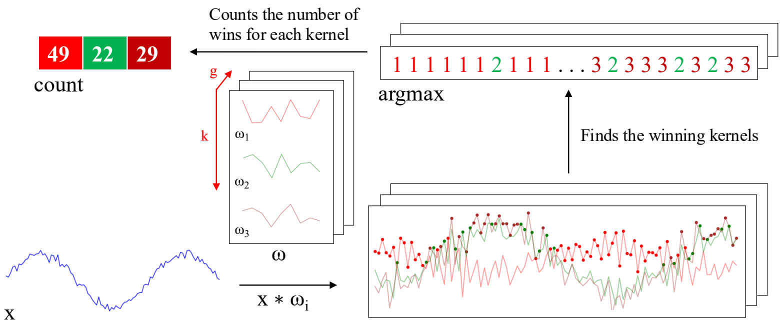


Fig. 5: Convolutions of HYDRA for each input time series with a set of random kernels w, organized into g groups with k kernels each.

The HYbrid Dictionary-ROCKET Architecture (Hydra) combines dictionary-based and convolution-based models [35]. Similar to ROCKET-based classifiers, it uses random kernels to extract features from the input time series. But it groups the kernels into g groups of k kernels each, as shown in Fig. 5. Each time series is passed through all the groups. For each group of kernels, we slide them across T and compute the dot product at each timestamp. Recall that the dot product of two input vectors (x and w_i) has the maximum value when the two vectors align in the same direction and the minimum value when they are oriented in opposite directions. We record the kernel that best matches the subsequence of T at each timestamp in each group (i.e., argmax). We refer to these kernels as the winning kernels. This results in a k-dimensional count vector for each of the g groups, where $k = 3$ in Fig. 5. This results in a total of $g \times k$ features, with default values of $g = 64$ and $k = 8$ It uses a total of $k \times g = 512$ kernels per dilation. In addition to recording the kernel with the maximum response, we can also record the kernel with the minimum response, knowing that this kernel will be the best match with the "inverted" subsequence of T. Hydra is applied to both the original time series and its first-order differences. Hydra generated approximately 1000 features for each instance in our dataset. [35] found that it can improve the accuracy by concatenating features generated from Hydra with those from MultiRocket. This classifier is called MultiROCKET-Hydra.

These five classifiers share the same simple design pattern. It involves the overproduction of features followed by a selection strategy. A large number of features (1000 $\sim$ 50000) are generated for each instance. The features are then fed into a simple linear classifier. It determines which features are most useful and returns the final classification result. A ridge classifier is used in this study. It is a linear classifier that extends ridge regression to classification tasks by applying a threshold to the predicted values. It uses L2 regularization to prevent overfitting. The regularization strength is selected by internal cross-validation. A Ridge classifier is suggested for small datasets, as in our case, while a logistic regression classifier is suggested for large datasets [31].

While these five classifiers are often referred to as classifiers, they are technically time series transformation methods for generating features that are then fed to a downstream classifier. The comparison of them is shown in Table 5. For MiniROCKET and MultiROCKET, the bias is determined from the convolution output, and the dilation depends on the length of the input time series [33]. The main differences among ROCKET-based classifiers lie in how the summary features are generated. The generation of the summary features depends on:

1. Kernels, which are defined based on the parameters, which consist of kernel length, kernel weights, bias, and dilation.
2. The way that padding applies to T, which leads to activation maps with different lengths.
3. The pooling operations, which are used in extracting features on the activation map.

	ROCKET	MiniROCKET	MultiROCKET	Hydra
kernel length	{7, 9, 11}	9	9	9
kernel weights	$\mathcal{N}(0,1)$	{-1, 2}	{-1, 2}	$\mathcal{N}(0,1)$
bias	$\mathcal{U}(0,1)$	from output	from output	none
dilation	random	fixed (input-relative)	fixed (input-relative)	random
padding	random	fixed	fixed	always
pooling operations	MAX, PPV	PPV	PPV, MPV, MIPV, LSPV	Response per Kernel/Group
1st order difference	no	no	yes	yes
feature vector size	20k	10k	50k	relative to input

Table 5: Comparison of rocket-based classifiers [31]. $\mathcal{N}(0,1)$: a standard normal distribution, $U(0,1)$: a uniform distribution between 0 and 1, 1st order difference: $\Delta T = t_2 - t_1, t_3 - t_2, ..., t_n - t_{n-1}$.

2.4 Evaluation metrics

To evaluate the performance of our time series-based classification (MTSC) model, we adopted five standard classification metrics. They are Accuracy (Acc), Specificity (Sp), Sensitivity (Sn), F1 score (F1), and Matthews Correlation Coefficient (MCC) [36].

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Sp = \frac{TN}{TN + FP}$$
$$Sn = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$
$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives, respectively.

To extend a binary metric to multi-class problems, we can treat the data as a collection of binary problems, one for each class. One class is treated as positive while the other classes are treated as negative. Then, the multi-class metrics can be obtained by averaging binary metric calculations across the set of classes. There are different ways to do the averaging. Here, we adopted a macro-averaging approach. It treats each class equally and calculates the mean of the binary metrics. To use MCC in the multiclass case, it can be defined in terms of a confusion matrix C for K classes, where $C_{i,j}$ is the number of observations that are actually in class i and predicted to be in class j [37].

$$MCC_{multi} = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) \times (s^2 - \sum_k^K t_k^2)}}$$

Where $t_k = \sum_i^K C_{i,k}$ (The number of times class k actually occurred), $p_k = \sum_i^K C_{k,i}$ (The number of times class k was predicted), $c = \sum_k^K C_{k,k}$ (The total number of samples correctly predicted) and $s = \sum_i^K \sum_j^K C_{i,j}$ (The total number of samples).

3 Results

In all experiments, the models were trained and tested using 5-fold cross-validation. We retrieved 827 empirically validated sequences of pre-miRNAs. There are 5p arm and 3p arm in each sequence. For each arm, we defined a cleavage pattern and a non-cleavage pattern. Three datasets, namely “5p arm”, “3p arm”, and “multi-class” were constructed by these patterns. We refer to the cleavage patterns as positive instances and the non-cleavage patterns as negative instances. The 5p arm dataset comprises 827 positive instances and an equal number of negative instances. The 5p arm and 3p arm datasets are binary-class datasets. The multi-class dataset comprises all patterns from both the 5p arm and the 3p arm. There are 827 “5p” instances[11], 827 “3p” instances, and 1654 negative instances.

For every fold in 5-fold cross-validation, the dataset was divided into a training set and a test set with sizes of 80% and 20% of the whole dataset, respectively. We kept the class distribution approximately the same in each fold, as it is in the original dataset. In each fold derived from the 5p arm and 3p arm datasets, the training set has a size of 1323, and the test set has a size of 331. In each fold derived from the multi-class dataset, the training set has a size of 2262, and the test set has a size of 662. We reported the average of the five classification metrics.

[11]cleavage patterns from the 5p arm

The ROCKET-based classifiers require all channels in the multivariate time series to have equal length. We applied padding to the shorter channels using the constant value 100, which does not appear in the original time series. It ensures the padding does not introduce ambiguity or interfere with the semantic meaning of the encoded nucleotide signals.

3.1 Channel ablation study

We utilized three types of data as the input features for each instance. They are (1) the RNA sequence, which consists of the primary strand and its complementary strand, (2) the secondary structure information, and (3) the base-pair probability sequence. To input the data into our time series-based classifiers, we converted them into multivariate time series. The primary strand and its complementary strand are each encoded into one or two channels, using the encoding methods in Tables 4. For example, single value mapping encodes a strand in one channel, while grouped variable-length channel mapping encodes in two channels. The secondary structure information is converted into a univariate time series. The base-pair probability sequence is already in numerical form and does not require further transformation. It can be used either as a standalone channel or incorporated into the encoding of the complementary strand. We performed a channel ablation study to determine the most informative combination of the above channels.

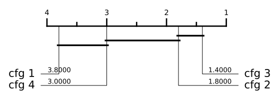
We referred to the multivariate time series that consists of the channels from the RNA sequence only as the baseline setting. We added the other channels to this baseline. It leads to the following configurations (cfgs):

1. (cfg 1) Baseline: Time series derived only from the RNA sequence.
2. (cfg 2) Baseline + Secondary structure: Baseline + time series representation of the secondary structure.
3. (cfg 3) Baseline + Base-pair probability (Standalone): Baseline + the base-pair probability sequence as a standalone channel.
4. (cfg 4) Baseline + Base-Pair probability (Incorporated): Baseline with the base-pair probability sequence incorporated into the encoding of the complementary strand.

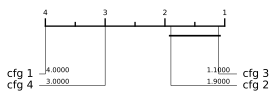
We used single value mapping as the encoding method. Table 6 shows the result. From the table, we can see that the addition of secondary structure, base-pair probability as a standalone channel, and base-pair probability incorporated in the encoding of the complementary strand can improve the performance. We plotted the critical difference (CD) diagram to visualize Table 6 to make the performances of different combinations more obvious. In CD diagrams, lower-ranked methods (toward the right) are better. A horizontal bar connecting combinations indicates no statistically significant difference. From Figure 6, we can see that including time series derived from secondary structure information and base-pair probability as a separate channel can significantly improve the performance of the classifiers. Incorporating the base-pair probability sequence in the time series encoding of the complementary strand can also improve the classifier, but to a minor degree compared to serving as a standalone channel. In our downstream analysis, we adopted the combination of RNA sequence time

	Classifier	5p arm					3p arm					multi-class				
		Acc	Sp	Sn	F1	MCC	Acc	Sp	Sn	F1	MCC	Acc	Sp	Sn	F1	MCC
Baseline (cfg 1)	ROCKET	0.781	0.743	0.819	0.789	0.563	0.790	0.773	0.807	0.793	0.580	0.717	0.838	0.685	0.700	0.538
	MiniROCKET	0.755	0.728	0.782	0.762	0.512	0.788	0.781	0.794	0.789	0.576	0.685	0.823	0.653	0.662	0.486
	MultiROCKET	0.784	0.767	0.801	0.787	0.569	0.803	0.792	0.814	0.805	0.606	0.691	0.830	0.667	0.672	0.501
	Hydra	0.830	0.800	0.860	0.835	0.663	0.808	0.797	0.820	0.810	0.617	0.731	0.844	0.696	0.712	0.560
	MultiROCKET-Hydra	0.796	0.778	0.815	0.800	0.594	0.807	0.767	0.816	0.808	0.614	0.701	0.836	0.681	0.686	0.520
Baseline + Secondary Structre (cfg 2)	ROCKET	**0.847**	**0.832**	0.862	**0.849**	**0.695**	**0.855**	0.842	**0.868**	**0.857**	**0.711**	**0.836**	**0.907**	**0.828**	**0.833**	**0.736**
	MiniROCKET	0.825	0.807	0.843	0.827	0.652	0.822	0.802	0.843	0.826	0.646	0.823	0.900	0.812	0.818	0.715
	MultiROCKET	0.812	0.803	0.822	0.814	0.626	0.824	0.809	0.839	0.826	0.649	0.796	0.888	0.791	0.792	0.673
	Hydra	0.845	0.816	**0.873**	**0.849**	0.691	0.846	0.817	0.874	0.850	0.693	0.830	0.901	0.814	0.826	0.724
	MultiROCKET-Hydra	0.817	0.809	0.826	0.819	0.635	0.825	0.816	0.834	0.826	0.652	0.803	0.891	0.798	0.800	0.684
Baseline + Base-pair probability (Standalone) (cfg 3)	ROCKET	0.842	0.828	0.855	0.844	0.684	**0.855**	**0.856**	0.854	0.855	0.710	0.795	0.885	0.783	0.789	0.670
	MiniROCKET	0.817	0.820	0.814	0.816	0.634	0.836	0.834	0.838	0.836	0.673	0.772	0.872	0.757	0.764	0.632
	MultiROCKET	0.822	0.813	0.832	0.824	0.645	0.825	0.831	0.820	0.824	0.651	0.758	0.866	0.747	0.750	0.612
	Hydra	0.846	0.827	0.865	**0.849**	0.693	0.851	0.840	0.861	0.852	0.702	0.789	0.879	0.769	0.780	0.658
	MultiROCKET-Hydra	0.822	0.809	0.834	0.824	0.644	0.835	0.840	0.830	0.834	0.670	0.759	0.866	0.746	0.750	0.611
Baseline + Base-pair probability (Incorporated) (cfg 4)	ROCKET	0.799	0.771	0.827	0.805	0.600	0.809	0.786	0.832	0.813	0.619	0.737	0.850	0.712	0.724	0.573
	MiniROCKET	0.776	0.756	0.797	0.781	0.554	0.801	0.808	0.794	0.799	0.603	0.705	0.835	0.675	0.684	0.521
	MultiROCKET	0.814	0.801	0.828	0.817	0.630	0.816	0.812	0.820	0.816	0.634	0.726	0.848	0.706	0.712	0.556
	Hydra	0.822	0.787	0.857	0.828	0.647	0.834	0.828	0.840	0.835	0.669	0.759	0.862	0.734	0.746	0.608
	MultiROCKET-Hydra	0.814	0.802	0.820	0.817	0.629	0.820	0.825	0.816	0.819	0.642	0.736	0.853	0.717	0.723	0.874

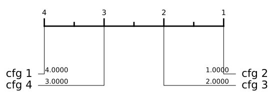
Table 6: Channel ablation study.



(a) 5p arm



(b) 3p arm



(c) multi-class

Fig. 6: CD diagrams of channel ablation study.

series, secondary structure time series, and base-pair probability time series as our multivariate time series input, with 4 to 6 channels, depending on the encoding used.

3.2 Predictive performance

The experiment was conducted on three datasets: the 5p arm, the 3p arm, and the multi-class datasets. Recall that we have nine encoding methods and five ROCKET-based classifiers. It results in 45 combinations of encoding methods and classifiers.

The result is shown in Table 7. The best combination of encoding method and classifier is shown in Table 8. For the 5p arm dataset, the best combination is “Global Cumulative grouped fixed-length channel mapping + ROCKET”. For all five classification metrics, it outperforms the state-of-the-art (SOTA) method, DiCleave. For the 3p arm dataset, the best combination is “Global Cumulative grouped fixed-length channel mapping + ROCKET”. Out of the five classification metrics, it outperforms DiCleave, except in specificity. For the multi-class dataset, the best combination is “Global Cumulative grouped fixed-length channel mapping + ROCKET”. For all five classification metrics, it outperforms DiCleave. Note that for the 3p arm and the multi-class datasets, the combination of “Cumulative grouped fixed-length channel mapping + ROKCET” also attains the best result.

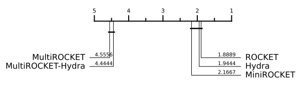
To summarize Table 7, we plot the CD diagrams for finding the best classifier, as shown in Figure 7, and the best encoding method, as shown in Figure 8.

	Classifier	5p arm					3p arm					multi-class				
		Acc	Sp	Sn	F1	MCC	Acc	Sp	Sn	F1	MCC	Acc	Sp	Sn	F1	MCC
Single value mapping (enc 1)	ROCKET	0.849	0.842	0.857	0.851	0.699	0.863	0.854	0.873	0.865	0.727	0.853	0.917	0.847	0.851	0.764
	MiniROCKET	0.823	0.809	0.837	0.825	0.647	0.823	0.828	0.817	0.822	0.647	0.835	0.906	0.828	0.833	0.735
	MultiROCKET	0.821	0.802	0.840	0.824	0.643	0.839	0.826	0.852	0.841	0.679	0.811	0.894	0.806	0.809	0.697
	Hydra	0.843	0.820	0.867	0.847	0.688	0.838	0.819	0.857	0.841	0.677	0.831	0.901	0.815	0.827	0.727
	MultiROCKET-Hydra	0.820	0.803	0.837	0.823	0.640	0.840	0.830	0.850	0.841	0.680	0.816	0.896	0.810	0.814	0.704
Grouped variable-length channel mapping (enc 2)	ROCKET	0.835	0.826	0.844	0.836	0.670	0.855	0.849	0.861	0.856	0.710	0.846	0.913	0.839	0.844	0.752
	MiniROCKET	0.843	0.833	0.853	0.844	0.686	0.831	0.821	0.842	0.833	0.663	0.837	0.907	0.828	0.834	0.737
	MultiROCKET	0.819	0.809	0.828	0.820	0.638	0.817	0.814	0.820	0.818	0.634	**0.890**	0.894	0.806	0.808	0.695
	Hydra	0.825	0.780	0.869	0.832	0.653	0.811	0.769	0.854	0.819	0.626	0.818	0.892	0.765	0.812	0.705
	MultiROCKET-Hydra	0.818	0.814	0.822	0.819	0.636	0.831	0.825	0.837	0.832	0.662	0.820	0.900	0.815	0.818	0.710
Grouped fixed-length channel mapping (enc 3)	ROCKET	0.851	0.843	0.859	0.852	0.702	0.863	0.850	0.875	0.864	0.726	0.849	0.915	0.843	0.847	0.757
	MiniROCKET	0.844	0.836	0.853	0.845	0.689	0.840	0.826	0.855	0.843	0.682	0.851	0.915	0.844	0.849	0.760
	MultiROCKET	0.831	0.815	0.848	0.834	0.663	0.824	0.813	0.836	0.826	0.649	0.811	0.896	0.808	0.808	0.698
	Hydra	0.848	0.816	0.880	0.853	0.699	0.862	0.839	0.884	0.864	0.724	0.843	0.908	0.837	0.839	0.746
	MultiROCKET-Hydra	0.836	0.813	0.859	0.839	0.672	0.833	0.820	0.845	0.835	0.665	0.828	0.905	0.824	0.826	0.725
Cumulative mapping (enc 4)	ROCKET	0.850	0.834	0.866	0.852	0.701	0.863	0.855	0.871	0.864	0.726	0.852	0.915	0.842	0.850	0.762
	MiniROCKET	0.840	0.821	0.860	0.843	0.682	0.840	0.837	0.844	0.841	0.682	0.843	0.911	0.835	0.840	0.747
	MultiROCKET	0.822	0.809	0.834	0.824	0.644	0.832	0.830	0.834	0.832	0.665	0.820	0.898	0.810	0.816	0.709
	Hydra	0.848	0.819	0.878	0.853	0.698	0.853	0.856	0.869	0.855	0.705	0.845	0.910	0.830	0.841	0.749
	MultiROCKET-Hydra	0.824	0.811	0.856	0.825	0.647	0.838	0.833	0.843	0.839	0.677	0.821	0.898	0.810	0.817	0.711
Cumulative grouped variable-length channel mapping (enc 5)	ROCKET	0.843	0.821	0.866	0.847	0.688	0.856	0.840	0.871	0.857	0.712	0.855	0.916	0.843	0.851	0.766
	MiniROCKET	0.845	0.826	0.865	0.848	0.691	0.836	0.833	0.838	0.836	0.672	0.840	0.909	0.833	0.838	0.742
	MultiROCKET	0.826	0.814	0.838	0.828	0.653	0.815	0.820	0.810	0.814	0.631	0.826	0.902	0.820	0.824	0.721
	Hydra	0.850	0.819	**0.880**	0.854	0.701	0.834	0.807	0.861	0.838	0.669	0.833	0.903	0.818	0.829	0.731
	MultiROCKET-Hydra	0.824	0.810	0.838	0.826	0.649	0.833	0.833	0.833	0.833	0.666	0.830	0.903	0.821	0.827	0.726
Cumulative grouped fixed-length channel mapping (enc 6)	ROCKET	0.856	0.836	0.876	0.858	0.712	**0.870**	**0.861**	**0.879**	**0.871**	**0.741**	0.863	**0.921**	**0.852**	**0.860**	**0.780**
	MiniROCKET	0.856	0.837	0.874	0.858	0.712	0.842	0.839	0.845	0.843	0.685	0.845	0.912	0.837	0.843	0.751
	MultiROCKET	0.820	0.802	0.839	0.824	0.642	0.798	0.798	0.798	0.798	0.597	0.809	0.894	0.806	0.807	0.694
	Hydra	0.850	0.814	0.885	0.855	0.701	0.855	0.840	0.869	0.857	0.711	0.847	0.910	0.831	0.843	0.752
	MultiROCKET-Hydra	0.820	0.801	0.839	0.823	0.641	0.807	0.813	0.802	0.806	0.615	0.821	0.900	0.817	0.819	0.713
Global Cumulative mapping (enc 7)	ROCKET	0.850	0.834	0.866	0.852	0.701	0.863	0.855	0.871	0.864	0.726	0.852	0.915	0.842	0.850	0.762
	MiniROCKET	0.847	0.832	0.862	0.849	0.695	0.848	0.839	0.857	0.850	0.697	0.845	0.911	0.836	0.843	0.750
	MultiROCKET	0.827	0.819	0.834	0.828	0.653	0.847	0.842	0.853	0.848	0.695	0.825	0.901	0.817	0.822	0.718
	Hydra	0.851	0.821	0.880	0.855	0.703	0.861	0.848	0.874	0.863	0.722	0.847	0.911	0.834	0.844	0.753
	MultiROCKET-Hydra	0.829	0.823	0.834	0.830	0.658	0.843	0.838	0.849	0.844	0.688	0.832	0.905	0.823	0.829	0.730
Global Cumulative grouped variable-length channel mapping (enc 8)	ROCKET	0.840	0.814	0.867	0.844	0.682	0.853	0.838	0.867	0.854	0.706	0.856	0.917	0.845	0.853	0.768
	MiniROCKET	0.848	0.834	0.862	0.850	0.697	0.841	0.824	0.859	0.844	0.683	0.844	0.911	0.856	0.842	0.748
	MultiROCKET	0.834	0.828	0.839	0.834	0.668	0.831	0.821	0.842	0.833	0.663	0.828	0.904	0.823	0.826	0.724
	Hydra	0.857	0.821	0.894	0.862	**0.717**	0.822	0.786	0.857	0.828	0.645	0.826	0.898	0.806	0.820	0.717
	MultiROCKET-Hydra	0.837	0.834	0.839	0.837	0.674	0.834	0.827	0.840	0.835	0.668	0.835	0.907	0.828	0.832	0.734
Global Cumulative grouped fixed-length channel mapping (enc 9)	ROCKET	0.856	0.836	0.876	0.858	0.712	**0.870**	**0.861**	**0.879**	**0.871**	**0.741**	0.863	**0.921**	**0.852**	**0.860**	**0.780**
	MiniROCKET	**0.857**	**0.845**	0.870	**0.859**	0.715	0.840	0.821	0.859	0.843	0.681	0.844	0.911	0.837	0.842	0.749
	MultiROCKET	0.829	0.825	0.833	0.830	0.658	0.820	0.816	0.823	0.820	0.640	0.819	0.900	0.816	0.817	0.710
	Hydra	0.856	0.817	0.894	0.861	0.713	0.859	0.838	0.880	0.862	0.719	0.846	0.911	0.832	0.843	0.752
	MultiROCKET-Hydra	0.829	0.824	0.834	0.830	0.658	0.822	0.825	0.819	0.821	0.644	0.827	0.904	0.823	0.824	0.722

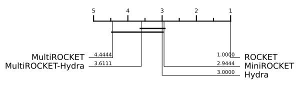
Table 7: Performance on the 45 combinations between encoding methods and the ROCKET-based classifiers.

Dataset	Methods	Acc	Sp	Sn	F1	MCC	Time (s)
5p arm	enc 9 + MiniROCKET	**0.857**	**0.845**	**0.870**	**0.859**	**0.715**	**0.787**
	DiCleave	0.818	0.790	0.846	0.822	0.653	21.249
3p arm	enc 9 + ROCKET	**0.870**	0.861	**0.879**	**0.871**	**0.741**	4.311
	enc 7 + MiniROCKET	0.848	0.839	0.857	0.850	0.697	**0.989**
	DiCleave	0.854	**0.891**	0.817	0.847	0.715	15.919
multi-class	enc 9 + ROCKET	**0.863**	**0.921**	**0.852**	**0.860**	**0.780**	12.208
	enc 3 + MiniROCKET	0.851	0.915	0.844	0.849	0.760	**4.550**
	DiCleave	0.820	0.895	0.804	0.815	0.710	131.151

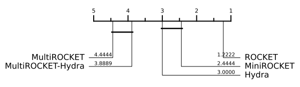
Table 8: Comparative analysis between MTSCCleav with the best combination of the encoding method and classifier, with the SOTA, DiCleave, on the three datasets. The best results of using MiniROCKET have also been shown to compare the computational efficiency.



(a) 5p arm

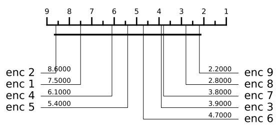


(b) 3p arm

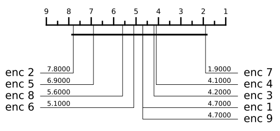


(c) multi-class

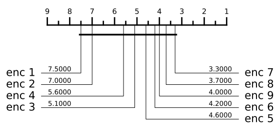
Fig. 7: CD diagrams to compare different classifiers.



(a) 5p arm



(b) 3p arm



(c) multi-class

Fig. 8: CD diagrams to compare different encoding methods.

3.3 Running time analysis

To compare the computational efficiency of MTSCCleav and DiCleave, we conducted a comparative analysis of their running times. For DiCleave, we employed the code from its supporting website[12], without any modifications. All experiments were conducted on the same machine (a personal laptop equipped with an Apple M1 Pro chip and 16 GB of memory) and using the same splits of the training and test datasets under 5-fold cross-validation to ensure fairness. The reported running times are the averages of the five runs. The timing results were measured from the training phase to the return of the five classification metrics. The result is shown in Table 8. MiniROCKET is the most computationally efficient of the five rocket-based classifiers. We also included its best result, along with the corresponding encoding method, even though this combination may not be the best overall.

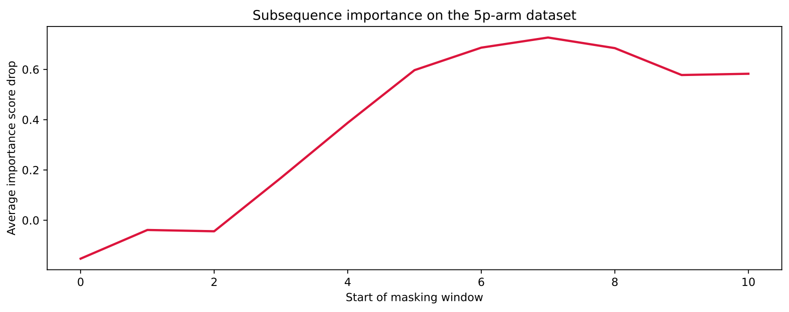
MTSCCleav demonstrated a significant advantage in computational efficiency, achieving an average 27.0X, 3.7X, and 10.7X speedup over DiCleave, for the 5p arm, 3p arm, and multi-class datasets, respectively. If we consider using the MiniROCKET in the case of 3p arm and multi-class datasets, it achieves 16.1X and 28.8X speedup. To note, in the case of the 3p arm dataset, the performance of MiniROCKET is only slightly lower than DiCleave. In the case of the multi-class dataset, even the performance of MiniROCKET is better than DiCleave. DiCleave is a deep learning-based method that requires substantial time for model inference, while MTSCleav leverages efficient ROCKET-based classifiers. This significant reduction in runtime makes MTSCCleav more suitable for large-scale data and real-time applications.

[12] https://github.com/MGuard0303/DiCleave (Accessed on: 2025-07-13).

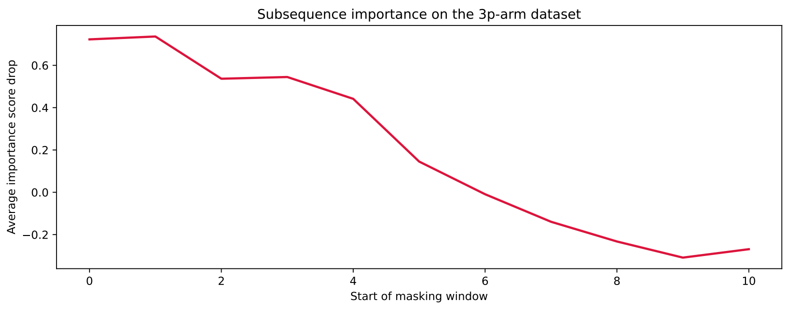
3.4 Subsequence importance

To evaluate the sensitivity of MTSCCleav to subsequences of the input, we conducted a perturbation experiment to evaluate the importance of subsequences based on masking windows. The goal of this experiment is to identify which subsequences of the entire time series are critical for classification. We examine how various modifications to the original input impact model performance. It suggests which features are essential for classification.

The model was trained on the original training dataset. For each instance in the test dataset, we measure its original score and the masked score. We slid a masking window w with a fixed length over the input time series T. $|w|$ was set to 4. For each window position $i \in \{1, 2, ..., |T|-|w|+1\}$, we masked all entries across all the channels of T within the window. Hence, we removed or hid that portion of information from the model during inference. The changes in classification performance in terms of accuracy relative to the unmasked original score of each i are recorded. Intuitively, if the information of a subsequence is critical for the classification, the masking of this subsequence would lead to a great drop in classification performance. We aggregated the importance score across the test dataset. The result is shown in Figure 9. For the



(a) 5p arm



(b) 3p arm

Fig. 9: Results of the perturbation experiment.

encoding methods, we cannot use the methods derived from the cumulative mapping because the accumulation would leak information from the masked region. We adopted "Grouped fixed-length channel mapping" as the encoding method and ROCKET as the classifier. "Grouped fixed-length channel mapping" is the best encoding, other than the methods derived from the cumulative mapping, in all datasets, as shown in Figure 8. ROCKET is the best classifier, as shown in Figure 7..

In the 5p arm dataset, we found that masking subsequences at the tailing part caused a significant drop in the importance score, as shown in Figure 9 (a). In the 3p arm dataset, we found that masking subsequences at the leading part caused a significant drop in the importance score, as shown in Figure 9 (b).

4 Discussion

The channel ablation study reveals that the involvement of the time series derived from the secondary structure can improve accuracy. It suggests the importance of RNA folding in dicer processing. Additionally, we found that the base-pair probability sequence of the secondary structure can also enhance accuracy. To the best of our knowledge, it is a novel application of the base-pair probability sequence. Experiments show that using the probability sequence as an additional channel can enhance accuracy more than incorporating it in the encoding. It is likely because keeping it as an additional channel can preserve more information, of both the probability sequence itself and the complementary strand.

Out of the three datasets, the best classifier is ROCKET. The ranking of the five classifiers by performance, starting from the best, is as follows: ROCKET, Hydra, MiniROCKET, MultiROCKET-Hydra, and MultiROCKET. It indicates that the features created from the pooling operations that are only in MultiROCKET but not in MiniROCKET, confuse the final classifier. They are mean of positive values (MPV), mean of indices of positive values (MIPV) and longest stretch of positive values (LSPV) [34]. In contrast, the pooling operator that is only present in ROCKET but not in MiniROCKET, enhances the classification performance. It is maximum (MAX).

For the encoding methods, we have the following observations. Fixed-length grouped channel mappings outperform variable-length counterparts with one exception in the multi-class dataset, likely because fixed-length schemes better preserve the original positional information of nucleotides within the sequence. Global cumulative methods consistently yield better performance than local cumulative methods. It suggests that the upstream information of the cleavage pattern plays a critical role in identifying cleavage sites. Cumulative-based encodings perform better than single-value mappings, with one exception in the 3p dataset, suggesting that the accumulated nucleotide signal is more informative for cleavage site prediction than the local or isolated presence of nucleotides. In the 5p arm dataset, encoding RNA sequence in two channels appears to worsen the result. This suggests that the 5p arm dataset and the 3p arm dataset need different nucleotide grouping methods for the encoding.

One limitation of DiCleave is overfitting during training because of the relatively small size of the dataset [11]. DiCleave is a deep learning-based method. Deep learning models typically require a large amount of training data to generalize effectively. They are data-hungry. In contrast, MTSCCleav leverages ROCKET-based methods for the classification. They rely on random convolutional feature extraction followed by a simple linear classifier. The Ridge classifier used in this study is less data-hungry compared to deep learning methods due to its use of L2 regularization and the simplicity of its linear model nature. It allows ROCKET-based classifiers, and hence MTSCCleav, to maintain strong predictive performance even in settings with a relatively small dataset size.

The subsequence importance reveals some connections between RNA secondary structure and human dicer cleavage site prediction. The perturbation experiment shows that the leading part of 5p arm and the tailing part of 3p arm are important for the classification. These parts are close to the center of the RNA secondary structure of pre-miRNA. It indicates that the center region is more crucial for human dicer cleavage site prediction. It agrees with the previous study [10].

5 Conclusions

We proposed an accurate, fast, and simple multivariate time series classification (MTSC)-based method, termed MTSCCleav, for predicting human dicer cleavage sites. Base-pair probability sequences of the secondary structures have also been leveraged in the classification. MTSCCleav consists of three parts: time series encoding, time series transformation, and classification. ROCKET-based methods were used for time series transformation. Ridge Classifier was used for classification. For the computational experiments, we evaluated nine time series encoding methods in conjunction with five time series transformation methods. MTSCCleav outperformed the SOTA method in all five evaluation metrics for the 5p-arm and multi-class datasets, and four of the metrics for the 3p-arm dataset. In terms of computational efficiency, MTSCCleav with the optimal setting achieved an average 3.7X to 27.0X speedup over the SOTA method on the three datasets. With the use of a less accurate but faster time series transformation method, MTSCCleav achieved an average speedup of 16.1X to 28.8X, respectively. We analyzed the subsequence importance of the input multivariate time series. The results show that subsequences near the center of the pre-miRNA sequences are more important. This aligns with the findings from previous work. This study demonstrates that time series analysis provides a powerful alternative to conventional modeling in the context of RNA processing. This framework may be extended to other RNA-processing tasks. Notably, the encoding of RNA sequence into time series enables us to utilize any well-established tools from the time series community.

References

[1] Urry, L.A., Cain, M.L., Wasserman, S.A., Minorsky, P.V., Orr, R.B., Campbell, N.A.: Campbell Biology, Twelfth edition edn., New York, NY (2020)

[2] Alberts, B.: Molecular Biology of the Cell, Seventh edition edn., New York (2022)

[3] Cohen, W.W.: A Computer Scientist's Guide to Cell Biology: A Travelogue from a Stranger in a Strange Land, New York, NY (2007)

[4] Lee, Y., Jeon, K., Lee, J.-T., Kim, S., Kim, V.N.: Microrna maturation: Stepwise processing and subcellular localization. The EMBO Journal **21**(17), 4663–4670 (2002)

[5] Gu, S., Jin, L., Zhang, Y., Huang, Y., Zhang, F., Valdmanis, P.N., Kay, M.A.: The loop position of shrnas and pre-mirnas is critical for the accuracy of dicer processing in vivo. Cell **151**(4), 900–911 (2012)

[6] Feng, Y., Zhang, X., Graves, P., Zeng, Y.: A comprehensive analysis of precursor microrna cleavage by human dicer. RNA **18**(11), 2083–2092 (2012)

[7] MacRae, I.J., Zhou, K., Doudna, J.A.: Structural determinants of rna recognition and cleavage by dicer. Nature Structural & Molecular Biology **14**(10), 934–940 (2007)

[8] Ahmed, F., Kaundal, R., Raghava, G.P.: Phdcleav: A svm based method for predicting human dicer cleavage sites using sequence and secondary structure of mirna precursors. BMC Bioinformatics **14**(14), 9 (2013)

[9] Bao, Y., Hayashida, M., Akutsu, T.: Lbsizecleav: Improved support vector machine (svm)-based prediction of dicer cleavage sites using loop/bulge length. BMC Bioinformatics **17**(1), 487 (2016)

[10] Liu, P., Song, J., Lin, C.-Y., Akutsu, T.: Recgbm: A gradient boosting-based method for predicting human dicer cleavage sites. BMC Bioinformatics **22**(1), 63 (2021)

[11] Mu, L., Song, J., Akutsu, T., Mori, T.: Dicleave: A deep learning model for predicting human dicer cleavage sites. BMC Bioinformatics **25**(1), 13 (2024)

[12] Mu, L., Akutsu, T.: DiCleavePlus: A Transformer-Based Model to Detect Dicer Cleavage Sites within Cleavage Patterns. Submitted, under review (2025). https://github.com/MGuard0303/DiCleavePlus

[13] Griffiths-Jones, S., Saini, H.K., van Dongen, S.: mirbase: Tools for microrna genomics. Nucleic Acids Research **36**(suppl_1), 154–158 (2008)

[14] Xu, T., Su, N., Liu, L., Zhang, J., Wang, H., Zhang, W., Gui, J., Yu, K., Li, J., Le, T.D.: mirbaseconverter: An r/bioconductor package for converting and retrieving mirna name, accession, sequence and family information in different versions of mirbase. BMC Bioinformatics **19**(19), 514 (2018)

[15] Zvelebil, M.J., Baum, J.O., Zvelebil, M.: Understanding Bioinformatics, New York (2008)

[16] Lorenz, R., Bernhart, S.H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P.F., Hofacker, I.L.: Viennarna package 2.0. Algorithms for Molecular

Biology **6**(1), 26 (2011)

[17] Aggarwal, C.C.: Data Mining: The Textbook, Cham (2015)

[18] Mendizabal-Ruiz, G., Román-Godínez, I., Torres-Ramos, S., Salido-Ruiz, R.A., Morales, J.A.: On dna numerical representations for genomic similarity computation. PLOS ONE **12**(3), 0173288 (2017)

[19] Anastassiou, D.: Genomic signal processing. IEEE Signal Processing Magazine **18**(4), 8–20 (2001)

[20] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12, pp. 262–270, New York, NY, USA (2012)

[21] Cristea, P.D.: Conversion of nucleotides sequences into genomic signals. Journal of Cellular and Molecular Medicine **6**(2), 279–303 (2002)

[22] Chakravarthy, N., Spanias, A., Iasemidis, L.D., Tsakalis, K.: Autoregressive modeling and feature analysis of dna sequences. EURASIP Journal on Advances in Signal Processing **2004**(1), 1–16 (2004)

[23] Zhao, J., Yang, X.W., Li, J.P., Tang, Y.Y.: Dna sequences classification based on wavelet packet analysis. In: Wavelet Analysis and Its Applications, pp. 424–429 (2001)

[24] Holden, T., Subramaniam, R., Sullivan, R., Cheung, E., Schneider, C., Jr, G.T., Flamholz, A., Lieberman, D.H., Cheung, T.D.: Atcg nucleotide fluctuation of deinococcus radiodurans radiation genes. In: Instruments, Methods, and Missions for Astrobiology X, vol. 6694, pp. 402–411 (2007)

[25] Nair, A.S., Sreenadhan, S.P.: A coding measure scheme employing electron-ion interaction pseudopotential (eiip). Bioinformation **1**(6), 197–202 (2006)

[26] Akhtar, M., Epps, J., Ambikairajah, E.: On dna numerical representations for period-3 based exon prediction. In: 2007 IEEE International Workshop on Genomic Signal Processing and Statistics, pp. 1–4 (2007)

[27] Zhang, R., Zhang, C.-T.: Z curves, an intutive tool for visualizing and analyzing the dna sequences. Journal of Biomolecular Structure and Dynamics **11**(4), 767–782 (1994)

[28] Berger, J.A., Mitra, S.K., Carli, M., Neri, A.: Visualization and analysis of dna sequences using dna walks. Journal of the Franklin Institute **341**(1), 37–53 (2004)

[29] Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series

classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)

[30] Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **35**(2), 401–449 (2021)

[31] Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: A review and experimental evaluation of recent time series classification algorithms. Data Mining and Knowledge Discovery **38**(4), 1958–2031 (2024)

[32] Dempster, A., Petitjean, F., Webb, G.I.: Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery **34**(5), 1454–1495 (2020)

[33] Dempster, A., Schmidt, D.F., Webb, G.I.: Minirocket: A very fast (almost) deterministic transform for time series classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. KDD '21, pp. 248–257, New York, NY, USA (2021)

[34] Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: Multirocket: Multiple pooling operators and transformations for fast and effective time series classification. Data Mining and Knowledge Discovery **36**(5), 1623–1646 (2022)

[35] Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: Competing convolutional kernels for fast and accurate time series classification. Data Mining and Knowledge Discovery **37**(5), 1779–1805 (2023)

[36] Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure **405**(2), 442–451 (1975)

[37] Gorodkin, J.: Comparing two k-category assignments by a k-category correlation coefficient. Computational Biology and Chemistry **28**(5), 367–374 (2004)