

Time Series Searching, Forecasting, and Classification 1
with Applications in Bioinformatics 2
時系列の探索・予測・分類とその生命情報学へ 3
の応用 4

Coleman Yu 5
余 浩旻

Abstract

7

Time series data are ubiquitous across many different fields. Much of the data are inherently time series data. Additionally, some data, such as strings, images, and object shapes, that are not originally time series data, can be transformed into time series. Many data mining tasks, such as classification, clustering, and motif finding, have been defined for time series data. Hence, by developing appropriate transformation methods, we can apply a plethora of well-established time series methods to our problems.

This thesis contributes two aspects in time series data mining and bioinformatics. They are a novel application of time series classification to solve complex biological problems by transforming biological data into time series and developing a more expressive distance measure framework that removes certain underlying assumptions.

In the first part, we demonstrate the utility of time series analysis in bioinformatics by studying the problem of predicting Human Dicer Cleavage Sites. Recall that bioinformatics operates at the intersection of **biology**, **biotechnology**, and **informatics**. In this work, we formulate a specific **biology** problem, which is predicting Human Dicer Cleavage sites in microRNA biogenesis, into a machine learning framework. In particular, a multivariate time series classification problem, which is the **informatics** part. Due to current limitations in **biotechnology**, we are constrained to using 1-D RNA sequence inputs rather than 2-D or 3-D data, because these are more expensive to obtain. We propose MTSCCleav, a method that encodes RNA sequences and the probabilities of base pairs in predicted secondary structures into time series data. To the best of our knowledge, we are the first to make use of the probabilities of base pairs in this kind of classification task on RNA data. By doing this, we frame the problem of predicting Human Dicer Cleavage sites into a Multivariate Time Series Classification (MTSC) problem. Existing approaches rely on opaque deep neural networks or complex feature engineering. They are slow, and the feature engineering is over-designed. In contrast, our approach is simple, intuitive, and computationally efficient. The proposed transformation methods allow us to use any well-established time series tools to analyze this biological problem. Experiments demonstrate that MTSCCleav achieves comparable and even better accuracy to state-of-the-art methods while delivering a 3.7X to 28.8X speedup. Furthermore, our perturbation experiments reveal that regions near the center of pre-miRNAs are essential for cleavage-site prediction, consistent with the existing literature.

39 In the second part, we address the limitations of existing similarity measures. Simi-
40 larity search is a core subroutine in time series data mining tasks. For example, recent
41 studies show that a simple 1-NN classifier with an appropriate distance measure can out-
42 perform many advanced, complicated methods. While Dynamic Time Warping (DTW)
43 and Uniform Scaling (US) are prevailing measures for handling local distortions and
44 global scaling, respectively, and some studies have demonstrated that combining both
45 DTW and US is necessary to obtain meaningful results. Current approaches apply a
46 single scaling factor to the entire sequence. We argue that since distinct phases of a pro-
47 cess often evolve at different speeds, a single scaling factor is insufficient. We introduce
48 the first distance measure framework, namely PSD, that achieves invariance to multiple
49 scaling factors. We also provide speed-up techniques to enable efficient computation of
50 the PSD. Experiments show that PSD better reflects the similarity between time series
51 with multiple phases, and that the identified phases (segmentation) provide a clearer
52 understanding of the data.

53 Collectively, this thesis advances the fields of time series data mining and bioinfor-
54 matics by demonstrating the use of time series analysis to address fundamental biological
55 questions and proposing a new, more expressive distance measure framework.

Acknowledgements

56

To begin, I want to express my deepest gratitude to my PhD supervisor at Kyoto University, Prof. Tatsuya Akutsu. I thank him for his kindness and patience. He is a brilliant researcher who can explain complex concepts simply. In a seminar, he showed that simply using clear notation makes presentations much easier to follow. More importantly, he helps students when they are in “valleys” in their lives. When a car is off track, it needs a push—so do people. I am equally indebted to my MPhil supervisor at The Hong Kong University of Science and Technology (HKUST), Prof. Raymond Chi-Wing Wong. He has continued to help and advise me even after I graduated from HKUST.

Akutsu-sensei taught me the importance of reading good materials. In the weekly seminars, he organized reading seminars, assigning some of his favorite books for us to read and discuss. His favorite book is “Probability and Computing” by Mitzenmacher and Upfal. Additionally, through the journal club he organized, students were asked to present the core ideas from their ten selected papers published in leading venues, such as Nature, Science, Cell, and PNAS. Reading beyond my fields helped me identify research gaps others missed. I think some great ideas are born when we merge the ideas from different fields. He also emphasized the importance of mathematics and proof. It is what computer scientists can stand out in the field of bioinformatics. I think it becomes especially true now, since advances in AI make it possible for almost everyone to code (i.e., Vibe coding). He advised me to use the existing tools, libraries, and studies to leverage my own research. He also taught me that a good researcher must master both oral and written presentation; otherwise, others may misinterpret your talent and the effort you put in. When we are reading a paper, he urges us to identify the novelty and analyze the pros and cons of the paper.

Raymond taught me the power of focus and “First Principles”. I was always surprised that he did not use reference management software like Zotero, preferring to annotate hard copies and even type .bib files manually. He told me that when he starts to do research, he will print out the papers and get focused on the stack of papers (hard copies) in front of him. I am not saying it is beneficial not to use tools, but I want to emphasize the power of focus that underlies his work routine. He taught me that every good research starts with a set of good papers. He also emphasized that there is no right or wrong in research, only what you choose to do about it. His research receipt works as follows. When you

88 are tackling a problem, you first review the relevant existing studies to understand the
89 state of the art (SOTA) for it. If the existing work addresses your particular problem,
90 you can apply and adapt it to your problem setting. But most of the time, since your
91 problem must be a particular version of a general problem, the existing general solution
92 should not work well for it. It means that you have some room to improve it. And this
93 is the research gap!. It reminds me of when we deal with the NP-complete problem.
94 As Kleinberg and Tardos's Algorithm Design (Chapter 10) suggests, an NP-complete
95 problem (assuming $P \neq NP$) does not allow us to have an algorithm that possesses
96 all three of the following desired properties simultaneously: Efficiency, Correctness, and
97 Generalization. Hence, it is sometimes preferable to address a specific instance of the
98 problem rather than the general one. He also emphasized the theory. He consistently
99 noted that you need to add theory to the paper to strengthen it. He always mentioned
100 that you need three motivations (why you are doing it) and three contributions (what you
101 have done). Sometimes, I ask why he can write so fast, and he replies, "If you know what
102 you are doing, then you write fast.". He also taught me that when you don't know some
103 of the ideas, you just go back to the original idea. Doing a "deep first search" can help
104 you reach the first principle, a concept that Elon Musk, one of the greatest entrepreneurs
105 of our time, always emphasized. Without these two supervisors, I would not have made
106 it this far. I could not have finished this program.

107 I would like to take this opportunity to share some of my thoughts on my PhD journey.
108 I hope the readers may learn something from what I have gone through. My PhD journey
109 was, to quote Dickens, "the best of times and the worst of times", and to quote Churchill,
110 "This was their finest hour". For the best parts, it allows me to explore both in my daily
111 life and in my research. I tried many things, walked many roads, drank many colas and
112 alcohol, and met many people. This helped me see problems from new perspectives.
113 Regarding the worst parts, I am reminded of a quote from one of my favorite movies,
114 "Les Choristes": "Fond de l'étang". It literally means "Bottom of the Pond". At times, I
115 felt like a frog at the well's bottom, trying hard to get out. Research is fun but also hard.
116 Research is about exploring something new. It is about publishing (so others can learn
117 from it). When I am stuck, the best solution is to aim for a reachable, well-defined goal.
118 The goal should be clear, with obvious rewards and requirements. Also, make sure the
119 effort of your actions can be accumulated. Like the frog, do not jump randomly, but aim
120 to move to stable platforms towards escape. Then, each jump matters for your progress.
121 Two of the materials helped me; they are "Eat the Frog!" and "THE PH.D. GRIND".

122 I would like to thank my thesis committee members. I would like to acknowledge
123 Tamura-sensei and Mori-sensei (a former member) of the Akutsu laboratory. Tamura-
124 sensei taught me to focus on the input-output relationship when encountering new algo-
125 rithms/methods, and Mori-sensei introduced me to the fascinating application of time-
126 series analysis to gene expression data, especially in trajectory inferences (i.e., pseudo-

time). I would like to thank the many communities that made my life in Japan colorful. 127
Thank you to the people of the dormitories where I lived at the Uji and Yoshida cam- 128
puses. I am grateful to my Japanese language teachers. I also cherish the friends I met 129
through my extracurricular activities, including Kendo, Table Tennis, Naginata, Karate, 130
and Aikido. I also met many interesting people from various international student events. 131

I am honored to receive the Asian Future Leaders Scholarship Program (AFLSP) 132
scholarship to support my studies in Japan. I have made a great choice by enrolling in 133
Kyoto University Design School, which has provided me with many valuable interdisci- 134
plinary experiences and opportunities to meet people outside my field. In this program, 135
I have conducted fieldwork in Okinawa, Hong Kong, and Bali, a rare opportunity for 136
researchers in computer science. I would also like to thank the Institute for Chemical 137
Research (ICR) for the Research Assistant position in the Akutsu laboratory. 138

Finally, I would like to express my deepest gratitude to my family and my friends who 139
have stayed by my side with their countless acts of support; I have nothing to return to 140
them but my love and time. 141

A special acknowledgement goes to my niece. As you grow up, I hope you explore 142
the world and fulfill your eagerness for knowledge. Find materials that interest you. One 143
day, you might find this thesis online and, I hope, find it worth reading and inspiring. 144

List of Publications

145

This thesis is based on the following papers.

146

- (Chapter 3) **Coleman Yu**, Raymond Chi-Wing Wong, and Tatsuya Akutsu, “MTSC-¹⁴⁷
Cleav: a Multivariate Time Series Classification (MTSC)-based Method for Predict-¹⁴⁸
ing Human Dicer Cleavage Sites”, submitted to *IEEE Access*, under review¹⁴⁹
 - (Chapter 4) **Coleman Yu**, Tatsuya Akutsu, and Raymond Chi-Wing Wong, “Scal-¹⁵⁰
ing with Multiple Scaling Factors and Dynamic Time Warping in Time Series¹⁵¹
Searching”, submitted to *IEEE Access*, under review¹⁵²
 - (Chapter 5) **Coleman Yu**, Raymond Chi-Wing Wong, and Tatsuya Akutsu, “Lever-¹⁵³
aging Nearest Neighbors for Time Series Forecasting with Matrix Profile”, in prepara-¹⁵⁴
tion¹⁵⁵
- Other publications¹⁵⁶
- **Coleman Yu** and Raymond Chi-Wing Wong, “A Melody Composer for both Tonal¹⁵⁷
and Non-Tonal Languages”, the 43rd International Computer Music Conference¹⁵⁸
2017, Shanghai, China on 16-20 Oct, 2017¹⁵⁹
 - Yi Zheng, Bogdan Enescu, Jiancang Zhuang, and **Coleman Yu**, “Data replenish-¹⁶⁰
ment of five moderate earthquake sequences in Japan, with semi-automatic cluster¹⁶¹
selection”, *Earthquake Science*, 34:310-322, 2021¹⁶²

Contents

163

Abstract	i	164
Acknowledgements	iii	165
List of Publications	vii	166
List of Figures	xiii	167
List of Tables	xv	168
1 Introduction	1	169
1.1 Background	1	170
1.2 Contributions	2	171
1.3 Organization	2	172
2 Preliminaries	3	173
2.1 Distance Measures	3	174
2.1.1 Euclidean Distance (ED)	4	175
2.1.2 Dynamic Time Warping (DTW)	4	176
2.1.3 Derivative Dynamic Time Warping (DDTW)	5	177
2.1.4 Weighted Dynamic Time Warping (WDTW)	5	178
2.1.5 Weighted Derivative Dynamic Time Warping (WDDTW)	6	179
2.1.6 Shape Dynamic Time Warping (shapeDTW)	6	180
2.1.7 Amercing Dynamic Time Warping (ADTW)	6	181
2.2 Classification	7	182
2.2.1 Ridge Classifier	7	183
3 MTSCCleav: a Multivariate Time Series Classification (MTSC)-based Method for Predicting Human Dicer Cleavage Sites	9	184
3.1 Background	10	186
3.2 Methods	12	187
3.2.1 Data Preparation	13	188

189	3.2.2	Time Series Encoding	16
190	3.2.3	Time Series Classification	19
191	3.2.4	Evaluation Metrics	23
192	3.3	Results	24
193	3.3.1	Channel Ablation Study	25
194	3.3.2	Predictive Performance	26
195	3.3.3	Running Time Analysis	28
196	3.3.4	Subsequence Importance	29
197	3.3.5	Summary	30
198	3.4	Discussion	30
199	3.5	Concluding Remarks	31
200	4	Scaling with Multiple Scaling Factors in Time Series Searching	33
201	4.1	Background	34
202	4.2	Related Work	37
203	4.3	Preliminaries	37
204	4.3.1	Euclidean Distance (ED)	38
205	4.3.2	Dynamic Time Warping (DTW)	39
206	4.3.3	Uniform Scaling (US)	41
207	4.3.4	Uniform Scaling & Dynamic Time Warping (USDTW)	42
208	4.3.5	Lower Bounds for DTW and USDTW	43
209	4.4	Piecewise Scaling Distance	46
210	4.4.1	Piecewise Scaling & Dynamic Time Warping (PSDTW)	47
211	4.4.2	Speedup Techniques	49
212	4.4.3	Guided Distance	54
213	4.5	Experiments	54
214	4.5.1	Experimental Setup	54
215	4.5.2	Experimental Results	57
216	4.6	Concluding Remarks	60
217	5	Leveraging Nearest Neighbors for Time Series Forecasting with Matrix Profile	63
218	5.1	Introduction	63
219	5.2	Related Work	66
220	5.3	Method	67
221	5.3.1	Problem Formulation	68
222	5.3.2	Evaluation Method	68
223	5.3.3	Gradient Boosting Regression Tree (GBRT)	70
224	5.3.4	Matrix Profile	70

5.4	Experiments	71	226
5.5	Conclusion and Future work	72	227
5.5.1	Conclusion	72	228
5.5.2	Future Work	72	229
6	Conclusion and Future Directions	75	230

List of Figures

231

3.1	Predicted secondary structure of the sequence S of pri-miRNA “hsa-let-7a-1” ¹ . 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 ² .	232 233 234 235 236 237 238 239 240 11 241
3.2	The overall pipeline of this study. Symbol notations: Cylinder - Dataset, Rectangle - Process, Parallelogram - Input / Output, Rounded Rectangle - Component.	242 243 13 244
3.3	Relationship of the proposed encoding methods.	16 245
3.4	Features generation in the transformation	20 246
3.5	Convolutions of HYDRA for each input time series with a set of random kernels w , organized into g groups with k kernels each.	21 247 248
3.6	CD diagrams of channel ablation study.	26 249
3.7	CD diagrams to compare different classifiers.	27 250
3.8	CD diagrams to compare different encoding methods.	28 251
3.9	Results of the perturbation experiment.	30 252
4.1	Applying nearest neighbor interpolation on Q , which result in Scaled Q , that can better reflect its similarity with C .	253 35 254
4.2	Q and C are in different rates. A stretching version of Q is similar to a prefix of C , but not the whole C .	255 35 256
4.3	Intuition of piecewise scaling (PS).	36 257
4.4	Z-normalization. Resulting time series have mean = 0 and std = 1.	38 258
4.5	Visualization of D with local constraints. The black cells form the warping path.	40 259 260
4.6	Visualization of LB_{Keogh} and LB_{Shen}	44 261

262	4.7	Relationship of L_Q and L_C	51
263	4.8	Visualization of the GunPoint dataset. Left (Right): A time series of the “Gun” (“Point”) scenario. Critical periods, such as “Aiming”, are annotated.	55
264	4.9	The red solid time series shows an instant in the target set. The blue dashed time series shows an instant in the query set.	55
265	4.10	Runtime and number of distance calls comparison of GunPoint dataset. (a)-(d) $P = 1$, (e)-(h) $P = 2$, (i)-(l) $P = 3$	59
266	5.1	Flattening a 2D window of size $(1 + M) \times w$ to a 1D vector of length $w + M$. The resulting vector serves as the <i>predictor</i> for the regressor. Its expected <i>response</i> is the vector of length h . The regressor learns from this predictor-response pair.	64
267	5.2	The <i>left</i> (<i>right</i>) matrix profile and the <i>matrix profile</i> of a time series. The <i>matrix profile</i> shows the distances between each m -subsequence and its nearest neighbor, where m is a user-given value. The <i>left</i> (<i>right</i>) matrix profile shows the same information but is limited to its left (right) nearest neighbor. For a particular m -subsequence shown by the right gray box, its nearest neighbor is the left gray box, as indicated by the dashed line in the <i>left matrix profile</i> and the <i>matrix profile</i> . Similarly, the nearest neighbor of the left gray box is the right gray box, as indicated by the dashed line in the <i>right matrix profile</i> and the <i>matrix profile</i> . The first (last) point in each box is denoted by a <i>red</i> circle (triangle). h denotes the length of the immediate subsequence of the nearest neighbor. Intuitively, this subsequence should be similar to the immediate subsequence (i.e., future) of the right gray box. To note, the <i>left</i> (<i>right</i>) matrix profile starts (ends) at a later (earlier) index because the corresponding nearest neighbor with length m does not exist.	65
268			
269			
270			
271			
272			
273			
274			
275			
276			
277			
278			
279			
280			
281			
282			
283			
284			
285			
286			
287			

List of Tables

288

3.1	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	289 290 291 13 292
3.2	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	293 294 14 295
3.3	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”.	296 297 15 298
3.4	Time series encoding. P is the corresponding base-pair probability sequence of S . $p_i = 1$ if we encode S without incorporating base-pair probability sequence.	299 300 18 301
3.5	Comparison of rocket-based classifiers [1]. $\mathcal{N}(0, 1)$: a standard normal distribution, $U(0, 1)$: a uniform distribution between 0 and 1, 1 st order difference: $\Delta T = t_2 - t_1, t_3 - t_2, \dots, t_n - t_{n-1}$	302 303 23 304
3.6	Channel ablation study. The best results are highlighted in bold	25 305
3.7	Performance on the 45 combinations between encoding methods and the ROCKET-based classifiers. The best results are highlighted in bold	306 27 307
3.8	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. The best results are highlighted in bold	308 309 310 311 28 312
4.1	Details of the ten datasets for the experiments	56 313
4.2	The accuracy comparison for eight distance measures of the GunPoint dataset	314 57 315
4.3	The accuracy comparison for six PSED-guided distance measures of the GunPoint dataset	316 58 317
4.4	The accuracy comparison for eight distance measures of the ten datasets	60 318

319	4.5	The accuracy comparison for six PSED-guided distance measures of the		
320		ten datasets	60	
321	4.6	The number of distance calls and runtime on PSED_vanilla and PSED_parallel_bsf		
322		of the ten datasets	61	
323	5.1	Dataset Statistics. N is the number of time series in the dataset, while		
324		$ T = n$ is the length of each time series. “rate” refers to the measuring		
325		rate. w is the size of the look-back window. h is the size of the forecasting		
326		window, also known as the forecasting horizon. T_{train} is the training subse-		
327		quence of T . T_{test} is the test subsequence of T . To note, $ T_{\text{train}} + T_{\text{test}} = n$.		
328		71	

Chapter 1

329

Introduction

330

1.1 Background

331

Human generates a ton of data nowadays. We are producing more new data in one single day today than in the first twenty-one centuries of AD combined. We are drowning in information but thirsty for knowledge. It is natural for us to develop computational methods to accelerate the process of “harvesting” knowledge from information.

332

333

334

335

Computational tasks focus on the relationship between input and output. We would like to find the hidden function behind. To note, there are two ways to solve a problem. One is the algorithmic approach, and the other is the machine-learning approach. In Chapter 3, it demonstrates a machine learning approach to analyze a biology problem. In Chapter 4, we solve an algorithmic problem on distance measure.

336

337

338

339

340

There are two large categories of machine learning. They are supervised learning and unsupervised learning. Classification may be the most intuitive form of supervised learning. The input is data points with labels. We learn a model from the relationship between data points and the labels. The model predicts the labels for the new data points. They have many applications. For example, in medical applications, it involves classifying patients as healthy or diseased, or tumors as benign or malignant. The term “supervised” means the model has access to labeled data. In other words, it requires labeled training examples that provide ground truth. So, the model can learn the boundary between the categories. Our first study focuses on a classification problem in biology.

341

342

343

344

345

346

347

348

349

A representative of unsupervised learning is undoubtedly clustering. We aim to group data into distinct clusters. The key difference is that the data lack predefined labels. By grouping them, we aim to identify natural patterns hidden in the data. One example is clustering cells based on their gene-expression profiles. These clusters might reveal distinct cell types. Note that we do not have the ground truth for the cluster set. In bioinformatics, we typically use enrichment analysis to determine whether specific gene functions are enriched in these clusters. Cell types often show enrichment for genes

350

351

352

353

354

355

357 responsible for specific functions. This set of genes defines their biological role. In
358 clustering, we first need to define a measure of similarity between two objects. And
359 what should be ignored. This is called invariance. For example, in image classification,
360 it should be invariant with respect to the zooming effect and the rotation effect. In
361 Chapter 4, we define a new distance measure framework that achieves invariance when
362 two time series are in different scaling factors.

363 1.2 Contributions

364 In this study, our contributions mainly include two parts. First, we propose the usage
365 of the base pair probability sequence from the predicted secondary structure of RNA se-
366 quence as a new information for the classification task. We apply Rocket-based classifiers
367 to identify the human dicer cleavage sites. Because of the simplicity of the transformation
368 method and the classifiers, our proposed method achieves 3.7X to 28.8X speedup while
369 achieving better or comparable results than the current state-of-the-art method. Second,
370 we propose a new distance measure framework, namely PSD, that can incorporate any
371 existing distance measures to achieve invariance for two time series with multiple rates.
372 Experiments show that our methods outperform ED, DTW and the other five DTW-
373 based methods. Besides, we propose to use the segmentation result returned by PSD to
374 improve the accuracy of other distance measures.

375 1.3 Organization

376 In Chapter 2, we review some of the basic knowledge in biology and time series data
377 mining, in particular, we focus on distance measures and Rocket-based classifiers. In
378 Chapter 3, we introduce our study of the problem of predicting human dicer cleavage
379 sites. We proposed a novel approach to frame this task as a multivariate time series
380 classification problem by introducing nine encoding methods and making use of Rocket-
381 based classifiers. In Chapter 4, we introduce a new distance measure framework, namely
382 PSD. It releases the assumption that there is only one scaling factor existing throughout
383 the whole time series. In Chapter 6, we give a conclusion to these two studies and provide
384 future work on them.

Chapter 2

385

Preliminaries

386

In this chapter, we provide background on time series, with a focus on distance measures 387
and classification, particularly the ridge classifier, which is used in the ROCKET-based 388
classifiers for time series classification. The remaining preliminary knowledge will be 389
provided in the corresponding chapters. Section 2.1 gives an overview of the existing 390
distance measures used in the evaluation in Chapter 4. Section 2.2 reviews the additional 391
knowledge about the classifiers used in Chapter 3. We start with the definition of a time 392
series. 393

2.1 Distance Measures

394

A general form of a time series T is an ordered pair of n real-valued variables, $T = 395$
 $(b_1, c_1), (b_2, c_2), \dots, (b_n, c_n)$, where b_i is the behavioral attribute and c_i is the contextual 396
attribute, where $1 \leq i \leq n$. c_i refers to the time stamp at which the measurement b_i 397
is taken. Since the measurements are always taken in a uniform manner, t_i is simply 398
incrementing from 1 to n uniformly. Hence, we can represent a time series more concisely 399
as $T = t_1, t_2, \dots, t_n$, where $t_i = b_i$. 400

We may be interested not only in the entire time series but also in a segment of it, 401
called a subsequence. A subsequence $T(i : j)$ of a time series T is a shorter time series, 402
which is a contiguous subset of time points in T , that starts from position i and ends at 403
position j . Formally, $T(i : j) = t_i, t_{i+1}, \dots, t_j$, where $1 \leq i \leq j \leq n$. We call $T(1 : m)$ as 404
the prefix of length m of T , m -prefix in short. 405

To quantify the similarity between two time series, we need to define a distance 406
measure, also known as a similarity measure, between them. Many distance measures 407
have been proposed in the literature. Among them, the most established measures are 408
undoubtedly Euclidean Distance (ED) and Dynamic Time Warping (DTW). They are 409
representatives of the two board classes of distance measures, namely “lock-step” and 410
“elastic”. 411

412 2.1.1 Euclidean Distance (ED)

413 ED is a lock-step distance measure. Given two time series Q and C with the same length
414 n , it compares the time point q_i of Q with the time point c_i of C at the same time (index).
415 Note that, traditionally, lockstep distance measures require the two time series to have
416 the same length because of the one-to-one alignment. However, in the setting of query by
417 content, where it is always the case that $|Q| < |C|$, we can still apply a lock-step distance
418 measure by either comparing Q with $C(1 : |Q|)$ or padding Q using its last element to
419 lengthen it to the same length of C . Minkowski distance is a generalization of Euclidean
420 distance. Minkowski distance is the L_p -norm of the difference between the two time series
421 X and Y , defined as:

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (2.1)$$

422 When $p = 2$, it corresponds to the Euclidean distance. When $p = 1$, it corresponds to
423 the Manhattan distance. When $p = \infty$, it corresponds to the Chebyshev distance. In our
424 studies, we focus on the Euclidean distance. Other lock-step distance measures include
425 Pearson correlation distance. It accounts for the linear association between the two time
426 series using the Pearson correlation coefficient. To note, there is another measure called
427 Edit Distance for strings. And Edit Distance is also sometimes abbreviated as ED in
428 string processing or bioinformatics. In this study, we focus on time series analysis and
429 use ED to denote Euclidean distance rather than edit distance.

430 2.1.2 Dynamic Time Warping (DTW)

431 Dynamic Time Warping is an elastic measure [2]. In contrast to lock-step distance mea-
432 sures, elastic distance measures allow one-to-many point matching. The one-to-many
433 point matching allows the elastic distance measures to warp in the time axis (i.e., tem-
434 porally) such that it can handle the local temporal distortions. While it will be detailed
435 in Chapter 4, it is briefly explained here. In short, it minimizes the cumulative distance
436 between two time series, subject to constraints, by finding an optimal warping path W^*
437 in a cost matrix, where W is the set of all possible paths and W^* is the optimal one. The
438 constraints typically are (1) Boundary conditions, (2) Continuity, and (3) Monotonicity.
439 Note that we can reduce pathological warping and accelerate computation by introduc-
440 ing a warping window. DTW with a warping window constraint is called constrained
441 DTW (cDTW). Two famous windows are the Sakoe-Chiba band [2] and the Itakura
442 parallelogram [3]. In the study, we focus on the Sakoe-Chiba band.

2.1.3 Derivative Dynamic Time Warping (DDTW)

The Derivative Dynamic Time Warping (DDTW) is a variant of DTW [4]. Instead of comparing original raw values, it compares two time series using their first-order derivatives, but with an approximation. In DTW, a point on a rising trend may be mapped to a point on a falling trend. It goes against our intuition. It can be solved by comparing their first-order derivatives, which encodes the slope information. The derivative T' of a time series T is computed approximately as follows.

$$t'_i = \frac{(t_i - t_{i-1}) + \frac{t_{i+1} - t_{i-1}}{2}}{2} \quad (2.2)$$

This estimate is simply the average of “the slope of the line through t_i and t_{i-1} (i.e., its left neighbor)” and “the slope of the line through t_{i-1} (i.e., its left neighbor) and t_{i+1} (i.e., its right neighbor)”. The $1/2$ term in the second item of the numerator comes from the fact that the separation in time of the t_{i-1} and t_{i+1} is 2. Note that the estimate is not defined for the first and last elements of the time series in the above equation. In these boundary cases, we use the estimates of the second and penultimate (i.e., the second-to-last thing) as the estimates for the first and last elements, respectively.

2.1.4 Weighted Dynamic Time Warping (WDTW)

The Weighted Dynamic Time Warping (WDTW) is a variant of DTW [5]. It is a penalty-based DTW designed to prevent pathological paths. Recall that a warping window (e.g., Sakoe-Chiba band) is enforced on the cost matrix of DTW, such that some paths are excluded. Only the paths that reside entirely in the warping window are feasible. This constraint may be too strict. WDTW uses a softer way for the same purpose. Instead of using a window to forbid the alignment of x_i and y_j that are far away in time. WDTW weights the cost of such alignment by multiplying it by a modified logistic weight function (MLWF) $\omega(k)$, defined as follows.

$$\omega(k) = \frac{\omega_{\max}}{1 + \exp(-g \cdot (k - m_c))} \quad (2.3)$$

Where:

- $k = |i - j|$. It is the phase difference (i.e., distance on the time axis from the diagonal. The diagonal refers to the line where $i = j$.)
- ω_{\max} is the desired upper bound for the weight parameter, which is suggested to be set to 1.
- m_c is the midpoint of a sequence. $m_c = m/2$.

- 472 • g is a constant that controls the level of penalization. It controls the curvature
473 (slope) of the function.

474 Intuitively, if x_i and y_j are far apart temporally, it will have a larger weight to discourage
475 their alignment and vice versa.

476 2.1.5 Weighted Derivative Dynamic Time Warping (WDDTW)

477 [5] also proposed the weighted version of DDTW. In brief, a weight is applied to the local
478 cost function when computing DTW on the first derivative.

479 2.1.6 Shape Dynamic Time Warping (shapeDTW)

480 The Shape Dynamic Time Warping (shapeDTW) is a variant of DTW [6]. The main mod-
481 ification to the original DTW is the way the local distance between points is computed.
482 Recall that DTW compares single scalar points. shapeDTW compares local descriptors.
483 The local descriptors are constructed using a sliding window on the original series, such
484 that for each point x , a L -subsequence with x as the center is extracted to compute the
485 higher-level feature of x . L is the user-given length of the subsequence to consider. By
486 default, it is set to 15. There are several ways to construct such a descriptor. For exam-
487 ple, a raw subsequence (i.e., a set of neighbor points surrounding the point of interest),
488 Piecewise aggregate approximation (PAA) [7, 8], slope, derivative, HOG-1D [9].

489 Then, the distance between descriptors is calculated rather than between raw values.
490 When a raw subsequence is chosen to construct the local descriptors, a common metric
491 used for comparing two local descriptors is the Euclidean distance. In the evaluation, a
492 raw subsequence is chosen to construct the local descriptors.

493 2.1.7 Amercing Dynamic Time Warping (ADTW)

494 The Amercing Dynamic Time Warping (WDTW) is a variant of DTW [10]. It is also
495 designed to constrain the amount of warping, as in cDTW and WDTW. While cDTW
496 imposes a hard window and WDTW uses multiplicative weights (i.e., MLWF), ADTW
497 introduces an additive penalty for non-diagonal alignment. The word “Amercing” means
498 “fining”. The non-diagonal alignments are required to pay the fines. Unlike WDTW,
499 which uses a multiplicative weight based on the position of the alignment, ADTW applies
500 an additive penalty ω based on the action of warping. The non-diagonal alignments are
501 penalized. Formally, the recursive relation for ADTW is defined as:

$$D(i, j) = d(q_i, c_j)$$

$$+ \min \begin{cases} D(i - 1, j - 1), \\ D(i - 1, j) + \omega, \\ D(i, j - 1) + \omega \end{cases} \quad (2.4)$$

ADTW penalizes the last two alignment actions. ω is a user-given hyperparameter. 502
 It should be a non-negative scalar constant. In practice, it is defined through cross- 503
 validation, which determines the optimal ω by training on a subset of data or heuristic 504
 search, which searches values in a user-given range. 505

To note, ADTW generalizes ED and DTW. If $\omega = 0$, no need to pay the fine for the 506
 non-diagonal alignment, which reduces to DTW. If $\omega \rightarrow \infty$, the non-diagonal alignment 507
 becomes prohibitive, and it reduces to ED. 508

2.2 Classification

In Chapter 3, we use ROCKET (RandOm Convolutional KErnel Transform) and its 510
 variants, including MiniRocket, MultiRocket, and Hydra, as the time series classifiers on 511
 the time series resulting from our encoding methods. Technically, they are not classifiers 512
 in their own right but rather feature extractors. These features are also called summary 513
 statistics. They are high-dimensional feature vectors that capture the characteristics of 514
 the original time series. The summary statistics are then fed to the classifiers to output 515
 the final classification results. 516

2.2.1 Ridge Classifier

The classifier that is usually chosen to work with ROCKET and its variants is a ridge 518
 classifier. The main advantage of it is speed. ROCKET and its variants generate a large 519
 number of features. 520

A ridge classifier is a wrapper that uses a ridge regression model as a routine to perform 521
 classification. It first maps the categorical labels of targets into continuous numbers, does 522
 the regression, and finally thresholds the numerical results from the regressor to obtain 523
 the classification result. 524

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^n$ with n instances, where $x_i \in \mathbb{R}^P$ is the 525
 feature vector with P dimensions and $y_i \in \{+1, -1\}$ is its label, it minimize the following 526
 optimization function. 527

$$\min_w \left(\sum_{i=1}^n (x_i^T w - y_i)^2 + \lambda \|w\|_2^2 \right) \quad (2.5)$$

528 Where $\|w\|_2^2 = \sum_{j=1}^p w_j^2$ is the L_2 norm of the weight vector and $\lambda > 0$ control the
 529 penalty. We explain the equation in brief. There are two terms inside the bracket. The
 530 first term is simply the sum of the residual, same as the one in the least squares method.
 531 The second term is called the L_2 penalty and is used to introduce bias in the fit to avoid
 532 overfitting. Hence, λ serves as a regularization hyperparameter between the trade-off
 533 between bias and variance.

534 Since the above optimization function in a ridge classifier has a closed-form solution,
 535 it can be solved using linear algebra rather than iterative optimization, as in logistic
 536 regression. Besides, the generated features by the random kernels in ROCKET and its
 537 variants are highly correlated. Ridge regularization, also known as the L_2 norm, can
 538 handle this case. Ridge regression shrinks regression coefficients toward zero by adding
 539 an L2 penalty. It reduces model complexity and helps with multicollinearity.

Chapter 3

540

MTSCCleav: a Multivariate Time Series Classification (MTSC)-based Method for Predicting Human Dicer Cleavage Sites

541

542

543

544

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.

545

546

547

548

549

550

551

552

We proposed a novel approach to frame the task as a multivariate time series classification problem. Nine 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 in terms of using a simpler, more intuitive model and less computational time. It achieves 3.7X to 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.

553

554

555

556

557

558

559

560

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.

561

562

563

Code is available at: <https://github.com/colemanyu/time-series-classification-cleavage>.

564

565

566 3.1 Background

567 One of the most important theories in molecular biology is the central dogma. It depicts
568 the flow of genetic information [11, 12]. Proteins are the functional units. The information
569 stored in DNA is used to create them. Genes (segments) in DNA are used as templates for
570 messenger RNAs (mRNAs) synthesis. An mRNA acts as a set of instructions to assemble
571 a chain of amino acids, which form a linear polypeptide. To become biologically active,
572 this chain is folded into a specific 3D structure, a proper configuration that enables it to
573 perform its desired functions. This folded polypeptide is called a functional protein, or
574 simply a protein. This entire process closely resembles how a computer program runs on
575 a machine. The source code does not function by itself. First, it is translated into an
576 assembly code (a lower-level, less human-readable form) and then into an executable file
577 that can actually perform the intended tasks [13].

578 These mRNAs are called “coding RNAs” because they code for proteins. There are
579 other genes in which the final product is the RNA molecule itself. They are called
580 non-coding RNAs (ncRNAs). Two types of small ncRNAs are particularly important.
581 They are microRNAs (miRNAs) and small interfering RNAs (siRNAs). Their discovery
582 was recognized with the 2006 Nobel Prize in Physiology or Medicine¹, awarded for work
583 completed only eight years prior [11].

584 In this study, we focus on miRNAs. An miRNA can regulate the expression of several
585 proteins. Hence, understanding the biogenesis of miRNAs is of great value. It involves
586 the processing of primary miRNAs (pri-miRNAs). RNAs are 3D molecules. However, it
587 is hard to measure the 3D structure (tertiary structure) from the experiment and predict
588 it from 1D sequence. We can understand their properties by analyzing their 1D sequence
589 or 2D structure, known as secondary structure. RNA sequence is easily obtained through
590 sequencing. The sequence and its predicted secondary structure of a pri-miRNA “hsa-
591 let-7a-1” is shown in Figure 3.1.

592 Recall that a pri-miRNA contains a hairpin loop, also called a stem loop. A mi-
593 croprocessor complex comprising Drosa and DCGR8 cleaves the pri-miRNA to form a
594 precursor miRNA (pre-miRNA) inside the nucleus. The stem-loop is still preserved, but
595 the two arms become shorter. After that, the pri-miRNA is transported by Exportin 5
596 from the nucleus to the cytoplasm. It is further cleaved by an enzyme called dicer [14].
597 Dicer cleaves the stem-loop from the two arms at the two cleavage sites, shown as the two
598 scissors in Figure 3.1. The stem-loop is removed. It results in a short double-stranded
599 miRNA molecule, known as an miRNA duplex, which consists of the 5p strand and the

¹The Nobel Prize in Physiology or Medicine 2006 - NobelPrize.org:

<https://www.nobelprize.org/prizes/medicine/2006/summary/> (Accessed on: 2025-06-13).

²Its miRBase entry: <https://mirbase.org/hairpin/MI0000060>. (Accessed on: 2025-06-12).

³RNAfold web server: <http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RFold.cgi>. (Accessed on: 2025-06-12). The figure is viewed in “forna”. This view option can be chosen on the website.

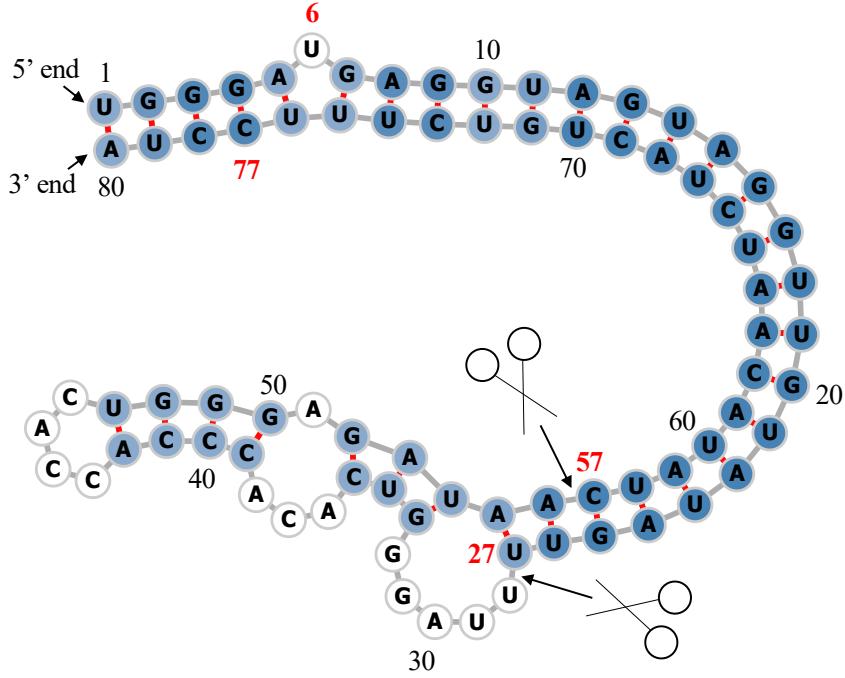


Figure 3.1: Predicted secondary structure of the sequence S of pri-miRNA “hsa-let-7a-1”². Experimental evidence suggests that the two deviated mature miRNAs are $UGA \cdots GUU$ and $CUA \cdots UUC$. They are $S(6 : 27)$ and $S(57 : 77)$ (Both ends are inclusive.). The ends are highlighted in **bold**. 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 server³.

3p strand⁴. 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 [15, 16, 17] 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 for the classification [18]. LBSizeCleav improved upon it by considering the loop

⁴The 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.

and bulge lengths [19]. [20] proposed an ensemble learning approach, using a gradient boosting machine for better accuracy. [21] 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 [21], which created the cleavage pattern by allowing cleavage sites to appear at any position within the pattern, instead of the middle only [22]. 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 [18, 19, 20, 21]. 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 [20, 21, 22]. 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 studies.
4. For computational efficiency, our method achieves a 3.7X to 28.8X speedup compared to the state-of-the-art (SOTA).
5. We conducted perturbation-based experiments. It shows that regions close to the cleavage sites are important for this problem. It is consistent with the existing study [20].

3.2 Methods

The overall pipeline of this study is summarized in Figure 3.2.

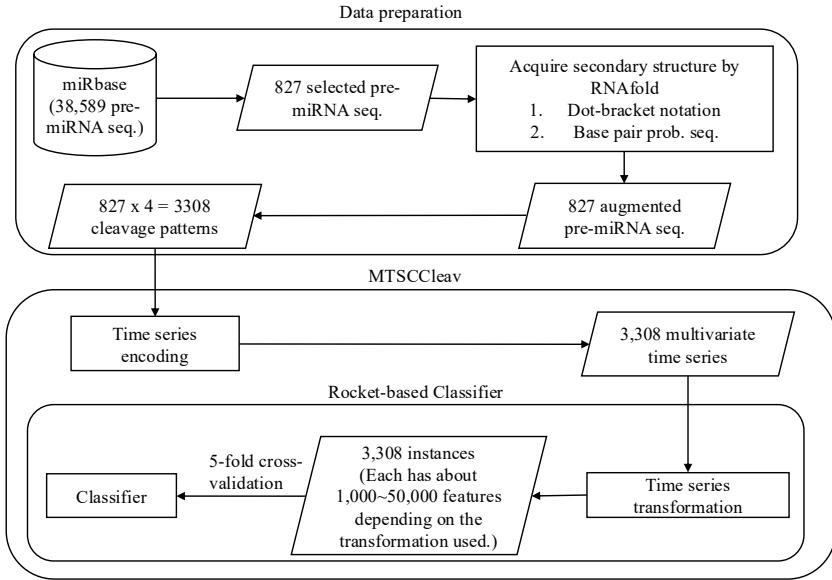


Figure 3.2: The overall pipeline of this study. Symbol notations: Cylinder - Dataset, Rectangle - Process, Parallelogram - Input / Output, Rounded Rectangle - Component.

3.2.1 Data Preparation

647

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

Table 3.1: 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**.

We used miRBase database [23]⁵. The database comprises miRNA data from various organisms [24]. The database contains 38,589 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.

648

649

650

651

652

653

654

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

655 Table 3.1 shows its four representative records. We first selected the records from
 656 humans (*Homo sapiens*). It resulted in 1,917 records. To identify the actual locations of
 657 the two cleavage sites in the pri-miRNA sequence supported by experimental evidence, we
 658 selected records that have two mature miRNAs resulting from cleavage at the 5p arm and
 659 the 3p arm, both of which have experimental support. Hence, only “MI0000060” (“hsa-
 660 let-7a-1”) would be selected in the table. It would serve as our running example. Its whole
 661 sequence is listed in Table 3.2. After the selection process, we selected 827 experimental
 662 validated pre-miRNA sequences, each with its two mature miRNA products. This formed
 663 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 3.2: 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**.

664 Augment the Dataset with Secondary Structure Information

665 We leveraged the predicted secondary structure of these sequences to enhance the ac-
 666 curacy of the classification. Recall that a specific three-dimensional (3D) structure is
 667 required for DNA, RNA, and protein to perform functions [25]. However, finding these
 668 3D structures using experimental methods such as X-ray crystallography or nuclear mag-
 669 netic resonance (NMR) is costly and time-consuming. Hence, prediction methods for such
 670 3D structures are necessary and helpful for downstream analysis. However, predicting the
 671 3D structures is challenging. One of the reasons is that there are some “nonconventional”
 672 base-pair interactions (e.g., noncanonical and rare A-G) that allow an RNA sequence to
 673 fold into a 3D structure, in addition to the (G, U) wobble pair, which is common and
 674 functionally important in RNA secondary structures. It makes the search space for pre-
 675 diction much larger than, in the 2D case, the secondary structure. The local structures
 676 of the 3D structures, the secondary structures, only focus on the conventional base-pair
 677 interactions [12]. Hence, predicting secondary structures is easier and faster. We em-
 678 ployed RNAfold from the ViennaRNA Package⁶ to predict the secondary structure for
 679 a given pri-miRNA S [26]. RNAfold returns the secondary structure in the dot-bracket
 680 notation and a matrix of base-pair probabilities. The matrix is a square matrix with the
 681 side length $|S|$, where each entry m_{ij} is the probability of base s_i paired up with base s_j .

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

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 3.2. 682
683
684
685
686
687

Extract Cleavage Patterns 688

The locations of the two mature miRNAs on the whole sequence indicate the probable 689 locations of the two cleavage sites. The 5p cleavage site must be beyond and near the 690 ending location of the 5p mature miRNA. We deemed the immediate bond next to the 691 5p mature miRNA’s ending position the 5p cleavage site, with the knowledge that the 692 actual cleavage site may not be this immediate bond but rather the nearby bonds after 693 it. The same applies to the 3p cleavage site. It is located at the immediate bond before 694 the starting position of the 3p mature miRNA. 695

For each arm of each whole sequence, we extracted a 14-string⁷ with the cleavage 696 site located at the center of the string. The first 7 nt (nucleotide) before the center are 697 highlighted in **bold**. In our running example, it would be “**UAUAGUU**UUAGGU” for 698 the 5p cleavage site and “**GAGAUAA**CUAUACA” for the 3p cleavage site. We refer to 699 these 14-strings as cleavage patterns. We also generate non-cleavage patterns by selecting 700 a 14-string with the center 6 nt away from the corresponding cleavage sites towards the 701 corresponding mature miRNA [19, 20] for each arm of each whole sequence. So, in our 702 running example, the 5p non-cleavage pattern would be “**AGGUUGU**AUAGUUU”. 703 The 3p non-cleavage pattern would be “**ACUAUAC**AAUCUAC”. 704

In conclusion, for a given pri-miRNA sequence, we can generate two cleavage patterns 705 and two non-cleavage patterns. We call these four patterns simply the “four strings” of 706 a given pri-miRNA. We also call each string a strand. The “four strings” of our running 707 example are listed in Table 3.3. 708

	5p cleav	5p non-cleav	3p cleav	3p non-cleav
Input strand	UAUAGUU UUAGGU	AGGUUGU AUAGUUU	GAGAUAA CUAUACA	ACUAUAC AAUCUAC
Complementary strand	AUAUCAA____UA	UCUAACAUAAUCAA_	C_CUGUUGAU AUGU	UGAUAUGUUGGAUG

Table 3.3: 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” 709 by finding the corresponding paired base for each of the bases in the input strand by 710 considering the secondary structure information. We use “_” to denote the unpaired base 711

⁷String with length = 14.

in the complementary strand. For example, in Figure 3.1, “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 3.3.

3.2.2 Time Series Encoding

A *time series* $T = t_1, t_2, \dots, t_n$ is a sequence of real-valued numbers⁸. A short contiguous region of T is called a subsequence. A *subsequence* $T(i : j) = t_i, t_{i+1}, \dots, t_j$ 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 [27]. 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) [28, 29]. 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 Figure 3.3.

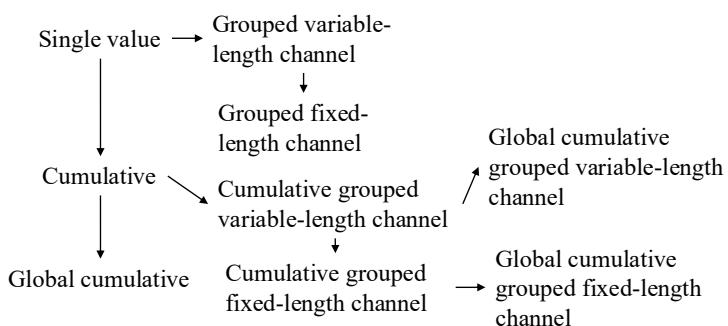


Figure 3.3: Relationship of the proposed encoding methods.

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 map-

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

ping” [30, 31, 32, 33, 28]. One single value is assigned to each of the letters. [34] employed the atomic number of each nucleotide as the transformed values, where $\{G = 78, A = 70, C = 58, T = 66\}$. [35] 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:

1. We employ the complementary property [36, 32] 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” as shown in row 1 of Table 3.4. 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 3.3 as S in the examples in Table 3.4.

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 [37, 38]. We named this method as “Cumulative mapping”, shown in row 4 of Table 3.4.

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

⁹In DNA, A pairs with T .

	Encoding	Algorithm	Example
1	Single value mapping [30, 31, 32, 33, 28]	$\text{for } i = 1 \text{ to } S :$ $t_i = \begin{cases} 2 \cdot p_i & \text{if } s_i = A \\ 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \\ -2 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ return T	$S = C, -, C, U, G, U, U, G, A, U$ $P = 0.843, 0.000, 0.807, 0.807, 0.793,$ $0.914, 0.982, 1.000, 0.999, 0.999$ 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	$j = 1, k = 1$ $\text{for } i = 1 \text{ to } S :$ $t_j^1 = \begin{cases} 1 \cdot p_i & \text{if } s_i = A \\ -1 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ $t_k^2 = \begin{cases} 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \\ \text{if } (s_i = G) \text{ or } (s_i = C): \\ \quad \text{increment } k \text{ by 1} \\ \text{else:} \\ \quad \text{increment } j \text{ by 1} \end{cases}$ 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	$\text{for } i = 1 \text{ to } S :$ $t_i^1 = \begin{cases} 1 \cdot p_i & \text{if } s_i = A \\ -1 \cdot p_i & \text{if } s_i = U \\ 0 & \text{otherwise} \end{cases}$ $t_i^2 = \begin{cases} 1 \cdot p_i & \text{if } s_i = G \\ -1 \cdot p_i & \text{if } s_i = C \\ 0 & \text{otherwise} \end{cases}$ return T^1, T^2	Without base-pair probability sequence: $T^1 = 0, 0, -1, 0, -1, -1, 0, 1, -1$ $T^2 = -1, 0, -1, 0, 1, 0, 0, 1, 0, 0$ 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 [37, 38]	$t_1 = 0$ $\text{for } i = 1 \text{ to } S :$ $t_{i+1} = \begin{cases} t_i + 2 \cdot p_i & \text{if } s_i = A \\ t_i + 1 \cdot p_i & \text{if } s_i = G \\ t_i - 1 \cdot p_i & \text{if } s_i = C \\ t_i - 2 \cdot p_i & \text{if } s_i = U \\ t_i & \text{otherwise} \end{cases}$ 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	$t_1^1 = 0, t_1^2 = 0$ $j = 1, k = 1$ $\text{for } i = 1 \text{ to } S :$ $t_{j+1}^1 = \begin{cases} t_j^1 + 1 \cdot p_i & \text{if } s_i = A \\ t_j^1 - 1 \cdot p_i & \text{if } s_i = U \\ t_j^1 & \text{if } s_i = - \end{cases}$ $t_{k+1}^2 = \begin{cases} t_k^2 + 1 \cdot p_i & \text{if } s_i = G \\ t_k^2 - 1 \cdot p_i & \text{if } s_i = C \\ t_k^2 & \text{otherwise} \end{cases}$ $\text{if } (s_i = G) \text{ or } (s_i = C):$ $\quad \text{increment } k \text{ by 1}$ else: $\quad \text{increment } j \text{ by 1}$ 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	$t_1^1 = 0, t_1^2 = 0$ $\text{for } i = 1 \text{ to } S :$ $t_{i+1}^1 = \begin{cases} t_i^1 + 1 \cdot p_i & \text{if } s_i = A \\ t_i^1 - 1 \cdot p_i & \text{if } s_i = U \\ t_i^1 & \text{otherwise} \end{cases}$ $t_{i+1}^2 = \begin{cases} t_i^2 + 1 \cdot p_i & \text{if } s_i = G \\ t_i^2 - 1 \cdot p_i & \text{if } s_i = C \\ t_i^2 & \text{otherwise} \end{cases}$ return $T^1, T^2 // T^1 = T^2 = S + 1$	Without base-pair probability sequence: $T^1 = 0, 0, 0, -1, -1, -2, -3, -2, -3$ $T^2 = 0, -1, -1, -2, -2, -1, -1, 0, 0, 0$ 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 3.4: Time series encoding. P is the corresponding base-pair probability sequence of S . $p_i = 1$ if we encode S without incorporating base-pair probability sequence.

per channel, depending on group-specific occurrences. The second one always returns two resulting sequences of a 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 3.4. 771
772
773
774

Global Cumulative versus Local Cumulative 775

In cumulative mapping and its variations, we can choose where to start the accumulation. 776
For a given subsequence S' of the whole sequence S , accumulation can start from the 777
beginning of S even if only S' is used downstream. It can also begin just at the start 778
of the S' . The first one preserves the global context. It can be useful when previous 779
nucleotides (those before S') influence later interpretation. The second one focuses solely 780
on local history in S' , ignoring global history. It is helpful if the previous nucleotides do 781
not affect the chemical property of S' . 782

Consider $T = 0, -1, \dots, -6$ of the input string S in “Cumulative mapping” in Table 3.4, 783
which accumulates from 0. S is the suffix with length = 10 of the constructed complemen- 784
tary strand of $S(1 : 63)$ in Figure 3.1. If we start the accumulation from the first entry 785
of the constructed complementary strand instead, it will yield a different result. Suppose 786
that the last entry of the time series encoded in the cumulative mapping of the con- 787
structed complementary strand is -8, the time series encoded in the “Global cumulative 788
mapping” for S would accumulate from -8 instead of 0. The result is $T = -8, -9, \dots, -14$. 789
Note that it has the same trend as the original T . This “Global cumulative” concept can 790
be applied to every cumulative-based method, as shown in Figure 3.3. 791

Incorporating Base-Pair Probabilities 792

We can incorporate the base-pair probabilities P in the encoding by thinking of it as the 793
weight or confidence p_i in the value assignment of each nucleotide s_i . It is implemented 794
by multiplying the base-pair probability p_i of the nucleotide s_i with the assigned value of 795
the kind of nucleotide of s_i during encoding, as shown in Table 3.4. 796

Transforming the Secondary Structure into a Time Series 797

We can transform the secondary structure in the dot-bracket notation into a time series 798
by “Single value mapping”, where “(” maps to 1, “.” maps to 0, and “)” maps to -1. 799

3.2.3 Time Series Classification 800

In univariate time series classification, an instance in the dataset consists of a time series 801
 $x = x_1, x_2, \dots, x_m$ with m observations and a discrete class label y , which takes c possible 802

values [39, 40]. 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 [1, 40, 39]. We employed convolution-based classifiers due to their simplicity and accuracy.

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:

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

To note, $|T(i : i + l - 1)| = |\omega| = l$. Entries M_i 's are calculated by sliding ω across T and computing a dot product. Additionally, although the original paper [41] 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 .

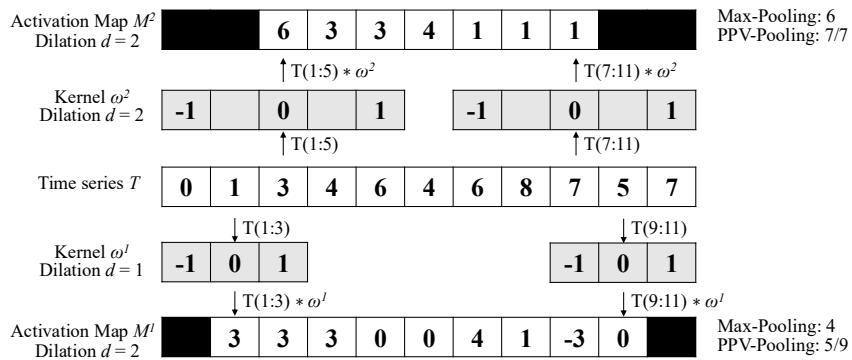


Figure 3.4: Features generation in the transformation

Figure 3.4 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 ,

¹⁰One step to the right per time.

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 Figure 3.4, 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) [41]. 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 Figure 3.4 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 10,000.

There are two extensions of ROCKET. They are MiniROCKET [42] and MultiROCKET [43]. 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 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 3.5 and add three new pooling operations [43]. The three added operations are mean of positive values (MPV), mean of indices of positive values (MIPV) and longest stretch of positive values (LSPV).

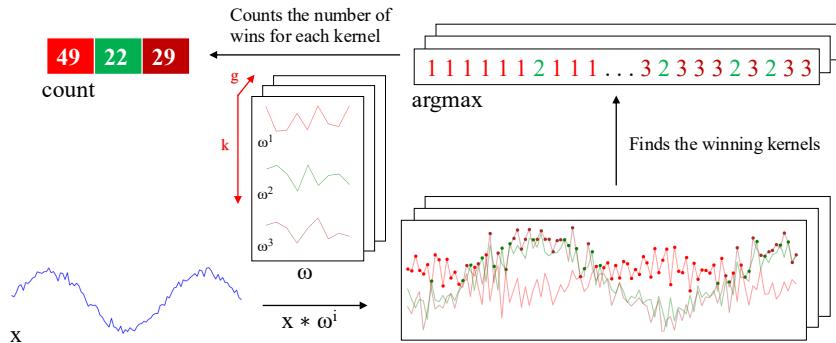


Figure 3.5: Convolutions of HYDRA for each input time series with a set of random kernels w , organized into g groups with k kernels each.

852 The HYbrid Dictionary-ROCKET Architecture (Hydra) combines dictionary-based
853 and convolution-based models [44]. Similar to ROCKET-based classifiers, it uses random
854 kernels to extract features from the input time series. But it groups the kernels into g
855 groups of k kernels each, as shown in Figure 3.5. Each time series is passed through
856 all the groups. For each group of kernels, we slide them across T and compute the dot
857 product at each timestamp. Recall that the dot product of two input vectors (x and
858 w_i) has the maximum value when the two vectors align in the same direction and the
859 minimum value when they are oriented in opposite directions. We record the kernel
860 that best matches the subsequence of T at each timestamp in each group (i.e., argmax).
861 We refer to these kernels as the winning kernels. This results in a k -dimensional count
862 vector for each of the g groups, where $k = 3$ in Figure 3.5. This results in a total of
863 $g \times k$ features, with default values of $g = 64$ and $k = 8$. It uses a total of $k \times g = 512$
864 kernels per dilation. In addition to recording the kernel with the maximum response,
865 we can also record the kernel with the minimum response, knowing that this kernel will
866 be the best match with the “inverted” subsequence of T . Hydra is applied to both the
867 original time series and its first-order differences. Hydra generated approximately 1000
868 features for each instance in our dataset. [44] found that it can improve the accuracy
869 by concatenating features generated from Hydra with those from MultiRocket. This
870 classifier is called MultiROCKET-Hydra.

871 These five classifiers share the same simple design pattern. It involves the overpro-
872 duction of features followed by a selection strategy. A large number of features (1,000 ~
873 50,000) are generated for each instance. The features are then fed into a simple linear
874 classifier. It determines which features are most useful and returns the final classification
875 result. A ridge classifier is used in this study. It is a linear classifier that extends ridge
876 regression to classification tasks by applying a threshold to the predicted values. It uses
877 L2 regularization to prevent overfitting. The regularization strength is selected by in-
878 ternal cross-validation. A Ridge classifier is suggested for small datasets, as in our case,
879 while a logistic regression classifier is suggested for large datasets [1].

880 While these five classifiers are often referred to as classifiers [1], they are technically
881 time series transformation methods for generating features that are then fed to a down-
882 stream classifier. The comparison of them is shown in Table 3.5. For MiniROCKET
883 and MultiROCKET, the bias is determined from the convolution output, and the di-
884 lation depends on the length of the input time series [42, 43]. The main differences
885 among ROCKET-based classifiers lie in how the summary features are generated. The
886 generation of the summary features depends on:

- 887 1. Kernels, which are defined based on the parameters, which consist of kernel length,
888 kernel weights, bias, and dilation.

2. The way that padding applies to T , which leads to activation maps with different lengths. 889
890
3. The pooling operations, which are used in extracting features on the activation map. 891
892

	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
1 st order difference	no	no	yes	yes
feature vector size	20k	10k	50k	relative to input

Table 3.5: Comparison of rocket-based classifiers [1]. $\mathcal{N}(0, 1)$: a standard normal distribution, $\mathcal{U}(0, 1)$: a uniform distribution between 0 and 1, 1st order difference: $\Delta T = t_2 - t_1, t_3 - t_2, \dots, t_n - t_{n-1}$.

3.2.4 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) [45]. 893
894
895
896
897

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

$$Sp = \frac{TN}{TN + FP}$$

$$Sn = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

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

where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives, respectively. 897
898

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 of doing 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 899
900
901
902
903
904

905 can be defined in terms of a confusion matrix C for K classes, where $C_{i,j}$ is the number
906 of observations that are actually in class i and predicted to be in class j [46].

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

907 where $t_k = \sum_i^K C_{i,k}$ (denoting the number of times class k actually occurred), $p_k =$
908 $\sum_i^K C_{k,i}$ (denoting the number of times class k was predicted), $c = \sum_k^K C_{k,k}$ (denoting
909 the total number of samples correctly predicted) and $s = \sum_i^K \sum_j^K C_{i,j}$ (denoting the total
910 number of samples).

911 3.3 Results

912 The code implementing our method is available at <https://github.com/colemanyu/time-series-classification-cleavage>. The dataset of this study is available at
913 <https://www.mirbase.org>.

914 In all experiments, the models were trained and tested using 5-fold cross-validation.
915 We retrieved 827 empirically validated sequences of pre-miRNAs. There are 5p arm
916 and 3p arm in each sequence. For each arm, we defined a cleavage pattern and a non-
917 cleavage pattern. Three datasets, namely “5p arm”, “3p arm”, and “multi-class” were
918 constructed by these patterns. We refer to the cleavage patterns as positive instances
919 and the non-cleavage patterns as negative instances. The 5p arm dataset comprises 827
920 positive instances and an equal number of negative instances. The 5p arm and 3p arm
921 datasets are binary-class datasets. The multi-class dataset comprises all patterns from
922 both the 5p arm and the 3p arm. There are 827 “5p” instances¹¹, 827 “3p” instances,
923 and 1,654 negative instances.

924 For every fold in 5-fold cross-validation, the dataset was divided into a training set
925 and a test set with sizes of 80% and 20% of the whole dataset, respectively. We kept the
926 class distribution approximately the same in each fold, since it is in the original dataset.
927 In each fold derived from the 5p arm and 3p arm datasets, the training set has a size of
928 1,323, and the test set has a size of 331. In each fold derived from the multi-class dataset,
929 the training set has a size of 2,262, and the test set has a size of 662. We reported the
930 average of the five classification metrics.

931 The ROCKET-based classifiers require all channels in the multivariate time series to
932 have equal length. We applied padding to the shorter channels using the constant value
933 100, which does not appear in the original time series. It ensures the padding does not
934 introduce ambiguity or interfere with the semantic meaning of the encoded nucleotide
935 signals.

¹¹Cleavage patterns from the 5p arm.

3.3.1 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 Table 3.4. 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.

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 Structure (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 3.6: Channel ablation study. The best results are highlighted in **bold**.

960 We used single value mapping as the encoding method. Table 3.6 shows the result.
 961 From the table, we can see that the addition of secondary structure, base-pair probability
 962 as a standalone channel, and base-pair probability incorporated in the encoding of the
 963 complementary strand can improve the performance. We plotted the critical difference
 964 (CD) diagram as shown in Figure 3.6 to visualize Table 3.6 to make the performances
 965 of different combinations more obvious. In CD diagrams, lower-ranked methods (toward
 966 the right) are better. A horizontal bar connecting combinations indicates no statistically
 967 significant difference.

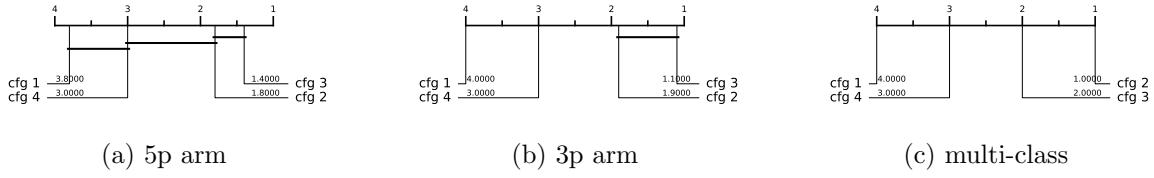


Figure 3.6: CD diagrams of channel ablation study.

968 From Figure 3.6, we can see that including time series derived from secondary struc-
 969 ture information and base-pair probability as a separate channel can significantly improve
 970 the performance of the classifiers. Incorporating the base-pair probability sequence in the
 971 time series encoding of the complementary strand can also improve the classifier, but to
 972 a minor degree compared to serving as a standalone channel. In our downstream analy-
 973 sis, we adopted the combination of RNA sequence time series, secondary structure time
 974 series, and base-pair probability time series as our multivariate time series input, with 4
 975 to 6 channels, depending on the encoding used.

976 3.3.2 Predictive Performance

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

980 The result is shown in Table 3.7. The best combination of encoding method and
 981 classifier is shown in Table 3.8. For the 5p arm dataset, the best combination is “Global
 982 Cumulative grouped fixed-length channel mapping + ROCKET”. For all five classifica-
 983 tion metrics, it outperforms the state-of-the-art (SOTA) method, DiCleave. For the 3p
 984 arm dataset, the best combination is “Global Cumulative grouped fixed-length channel
 985 mapping + ROCKET”. Out of the five classification metrics, it outperforms DiCleave,
 986 except in specificity. For the multi-class dataset, the best combination is “Global Cu-
 987 mulative grouped fixed-length channel mapping + ROCKET”. For all five classification
 988 metrics, it outperforms DiCleave. Note that for the 3p arm and the multi-class datasets,

	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 3.7: Performance on the 45 combinations between encoding methods and the ROCKET-based classifiers. The best results are highlighted in **bold**.

the combination of “Cumulative grouped fixed-length channel mapping + ROKCET” 989 also attains the best result. 990

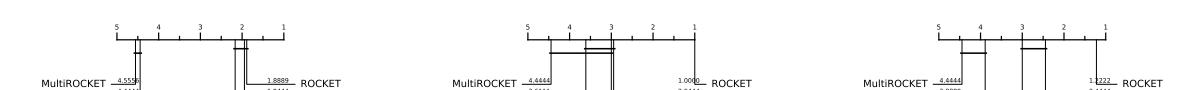


Figure 3.7: CD diagrams to compare different classifiers.

To summarize Table 3.7, we plot the CD diagrams for finding the best classifier, as shown in Figure 3.7, and the best encoding method, as shown in Figure 3.8. 991

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 3.8: 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. The best results are highlighted in **bold**.

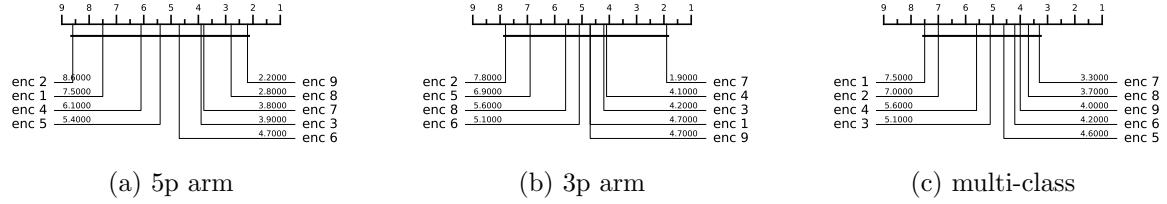


Figure 3.8: CD diagrams to compare different encoding methods.

3.3.3 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 website¹², 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 3.8. 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

¹²

<https://github.com/MGuard0303/DiCleave> (Accessed on: 2025-07-13).

case of the 3p arm dataset, the performance of MiniROCKET is only slightly worse than 1009
DiCleave. In the case of the multi-class dataset, even the performance of MiniROCKET 1010
is better than DiCleave. DiCleave is a deep learning-based method that requires sub- 1011
stantial time for model inference, while MTSCleav leverages efficient ROCKET-based 1012
classifiers. This significant reduction in runtime makes MTSCcleav more suitable for 1013
large-scale data and real-time applications. 1014

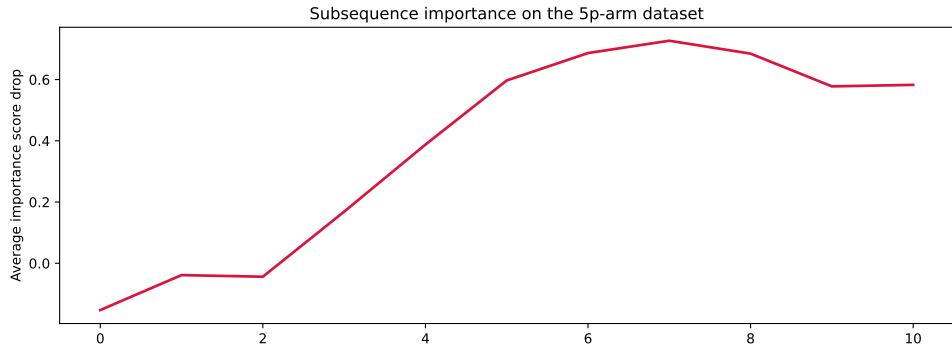
3.3.4 Subsequence Importance

To evaluate the sensitivity of MTSCcleav to subsequences of the input, we conducted a 1016
perturbation experiment to evaluate the importance of subsequences based on masking 1017
windows. The goal of this experiment is to identify which subsequences of the entire 1018
time series are critical for classification. We examine how various modifications to the 1019
original input impact model performance. It suggests which features are essential for 1020
classification. 1021

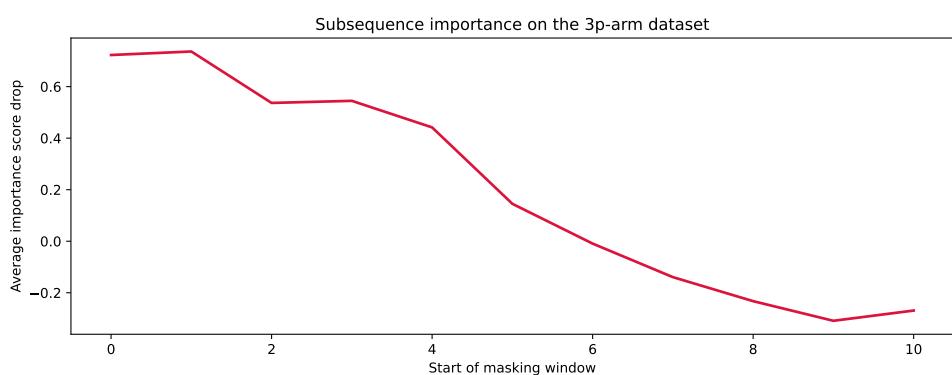
The model was trained on the original training dataset. For each instance in the test 1022
dataset, we measure its original score and the masked score. We slid a masking window 1023
 w with a fixed length over the input time series T . $|w|$ was set to 4. For each window 1024
position $i \in \{1, 2, \dots, |T| - |w| + 1\}$, we masked all entries across all the channels of T 1025
within the window. Hence, we removed or hid that portion of information from the model 1026
during inference. The changes in classification performance in terms of accuracy relative 1027
to the unmasked original score of each i are recorded. Intuitively, if the information of a 1028
subsequence is critical for the classification, the masking of this subsequence would lead 1029
to a great drop in classification performance. We aggregated the importance score across 1030
the test dataset. 1031

The result is shown in Figure 3.9. For the encoding methods, we cannot use the 1032
methods derived from the cumulative mapping because the accumulation would leak in- 1033
formation from the masked region. We adopted “Grouped fixed-length channel mapping” 1034
as the encoding method and ROCKET as the classifier. “Grouped fixed-length channel 1035
mapping” is the best encoding, other than the methods derived from the cumulative map- 1036
ping, in all datasets, as shown in Figure 3.8. ROCKET is the best classifier, as shown in 1037
Figure 3.7. 1038

In the 5p arm dataset, we found that masking subsequences at the tailing part caused 1039
a significant drop in the importance score, as shown in Figure 3.9 (a). In the 3p arm 1040
dataset, we found that masking subsequences at the leading part caused a significant 1041
drop in the importance score, as shown in Figure 3.9 (b). 1042



(a) 5p arm



(b) 3p arm

Figure 3.9: Results of the perturbation experiment.

3.3.5 Summary

Our method achieves better or comparable predictive results and a 3.7X to 28.8X speedup compared to the state-of-the-art (SOTA).

3.4 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. Furthermore, 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 fea-

tures 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) [43]. In contrast, the pooling operator that is only present in ROCKET but not in MiniROCKET, enhances the classification performance. It is maximum (MAX). 1058
1059
1060
1061
1062

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. 1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074

One limitation of DiCleave is overfitting during training because of the relatively small size of the dataset [21]. 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. 1075
1076
1077
1078
1079
1080
1081
1082
1083

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 is consistent with the previous study [20]. 1084
1085
1086
1087
1088
1089

3.5 Concluding Remarks 1090

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 1091
1092
1093
1094

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 classification 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.

Chapter 4

1111

Scaling with Multiple Scaling

1112

Factors in Time Series Searching

1113

Time series data are ubiquitous across many different fields. Many data mining tasks, 1114 such as classification, clustering, and motif finding, have been defined for time series data. 1115 They utilize similarity search as a core subroutine, making it crucial to design similarity 1116 measures that align with our intuitions. To facilitate efficient computation, speedup 1117 techniques are essential. Dynamic Time Warping (DTW) is arguably the most prevailing 1118 distance measure for time series data. However, studies have shown that for certain data, 1119 another distance measure, namely Uniform Scaling (US), is equally crucial as DTW. DTW 1120 handles the local distortion, while US handles the global scaling. In addition, studies have 1121 demonstrated that combining DTW and US is necessary to obtain meaningful results in 1122 some cases. Surprisingly, all existing studies employ only a single scaling factor for the 1123 entire time series. A time series could consist of phases. Since each phase of a time series 1124 expresses at its own rate, using a single scaling factor is insufficient when comparing two 1125 time series that share similar phases but differ in their expression rates. We introduce 1126 the first framework that accounts for multiple scaling factors, Piecewise Scaling Distance 1127 (PSD). PSD employs other existing distance measures as subroutines. Because the naive 1128 implementation of PSD is slow, we propose a constrained version of PSD that enforces 1129 constraints based on the allowed segment lengths derived from the given scaling factor 1130 bound. It also prevents pathological results. In addition, two other speedup techniques 1131 have been proposed, which achieve 10.10X to 191.46X speedup. We also demonstrate 1132 the usage of a lower bound when DTW is used as the subroutine of PSD. Moreover, we 1133 show that the segmentation results returned by PSD can improve the accuracy of other 1134 distance measures. 1135

1136 4.1 Background

1137 To study the mechanism of a process, we take measurements. Measurements are usually
1138 taken continuously by the sensors. Measurements of processes always yield continuous
1139 values at discrete timestamps. They are time series data. For example, smartphones
1140 collect users' GPS data. ECG monitors measure patients' heart rate. The continuous
1141 measurements compose a time series. It is not hard to see why time series data are
1142 ubiquitous across many different fields. In GPS data, each time series data point consists
1143 of the user's latitude and longitude information. They are multivariate time series. In
1144 ECG data, each data point represents the amplitude of the patient's cardiac electrical
1145 activity. They are univariate time series. In this study, we focus on univariate time series.

1146 Many data mining tasks can be defined on time series data. For example, given a
1147 time series database, we can perform clustering based on the pairwise similarity of the
1148 time series instances. A classifier can be trained when categorical labels are available.
1149 Alternatively, given a long time series, for motif finding, we identify recurring patterns.
1150 In contrast, for anomaly detection, we identify abnormal subsequences. Almost all time
1151 series data mining tasks can be reduced to arguing the similarity between two time series.
1152 A good distance measure, also known as a similarity measure, can determine the success
1153 or failure of the algorithms built on it. The choice of an appropriate distance measure is
1154 particularly evident in classification. Studies show that simple nearest-neighbor classifi-
1155 cation (1-NN) is difficult to beat and can compete with more complex methods [39].

1156 A time series is treated as a whole rather than as a collection of individual values.
1157 The relationships between values are important. They constitute trends and shapes.
1158 Hence, similarity search in time series data is approximate-based rather than exact match-
1159 based [47]. Besides, different invariances should be allowed during the comparison.

1160 Dynamic Time Warping (DTW) is one of, if not the most common, similarity mea-
1161 sures. DTW provides invariance to time distortion by aligning and measuring the sim-
1162 ilarity between two series that may be misaligned in time. However, it assumes that
1163 the time series are expressed on a similar global expression rate. This assumption limits
1164 its performance when comparing two time series expressed at different global expression
1165 rates. We often see this behavior in domains such as speech recognition, motion analysis,
1166 patient biomedical signals, and sensor data in the manufacturing industry.

1167 Uniform Scaling (US) can achieve global scaling invariance by scaling the two time
1168 series to the same length via interpolation, such as nearest-neighbor interpolation, before
1169 comparison, as shown in Figures 4.1. It is reported that in some domains, such as
1170 gestures [48, 49] and music performance [50], the scaling is about 10-15% (i.e., scaling
1171 factors: 1.1 to 1.15). The scaling factors are relatively small, since the nature of the
1172 music and the gait will change with significant scaling factors. However, in some other
1173 domains, we may encounter larger scaling factors. In bioinformatics, gene expression time

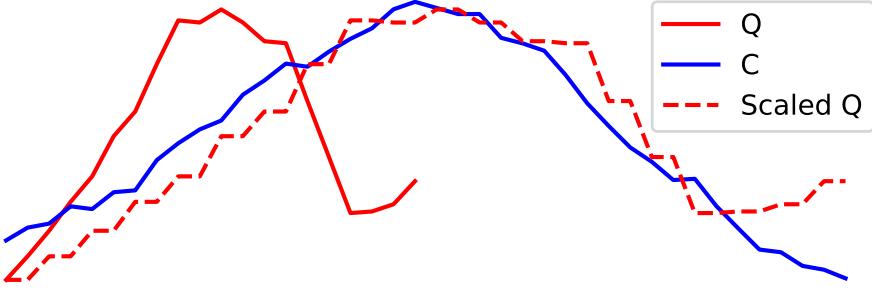


Figure 4.1: Applying nearest neighbor interpolation on Q , which result in Scaled Q , that can better reflect its similarity with C .

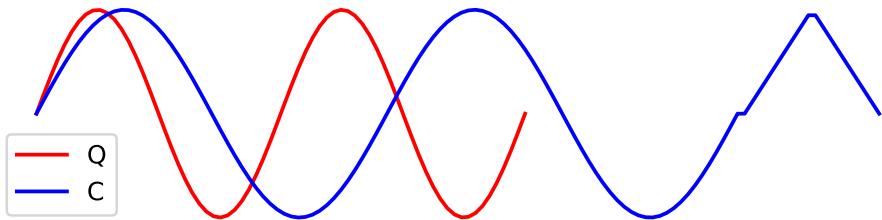


Figure 4.2: Q and C are in different rates. A stretching version of Q is similar to a prefix of C , but not the whole C .

series data could differ by a factor of 1.41 [51, 52]. In Figure 4.2, Q and a prefix of C 1174
are similar, but at different rates. In searching, we typically have a query Q and a longer 1175
candidate C . We seek a prefix of C that is close to Q . For better comparison, we need 1176
to eliminate the scaling effect. These observations demonstrate the necessity of uniform 1177
scaling. 1178

DTW and US are used to achieve different kinds of invariance. DTW handles local 1179
distortion, while US handles global scaling. Furthermore, some studies show that the 1180
combination of US and DTW, namely USDTW, better reflects similarity [53, 54, 55]. US 1181
is first applied to transform the two time series into the same length to eliminate the effect 1182
resulting from the different rates. Then, DTW, rather than ED, is applied to address 1183
local misalignment. USDTW is computationally more expensive than DTW because it 1184
involves the calculation of the DTW between Q and different lengths of each prefix of C . 1185
The different lengths of the prefixes correspond to different scaling factors. 1186

It is not uncommon for the data sampling strategy to change over time [56]. There are 1187
different phases, each with its own rate. To achieve invariance for this kind of scaling effect 1188
resulting from multiple rates, rather than using a single scaling factor, it is beneficial to 1189
identify these different phases and use the appropriate scaling factors for these segments, 1190
also known as pieces. We refer to this as piecewise scaling (PS). Figure 4.3 shows the 1191
intuition of PS. The prefix of C (i.e., $C(1 : k)$) and Q share the same set of segments, 1192
but each has a different scaling. Multiple scaling factors must be used. It motivates us 1193

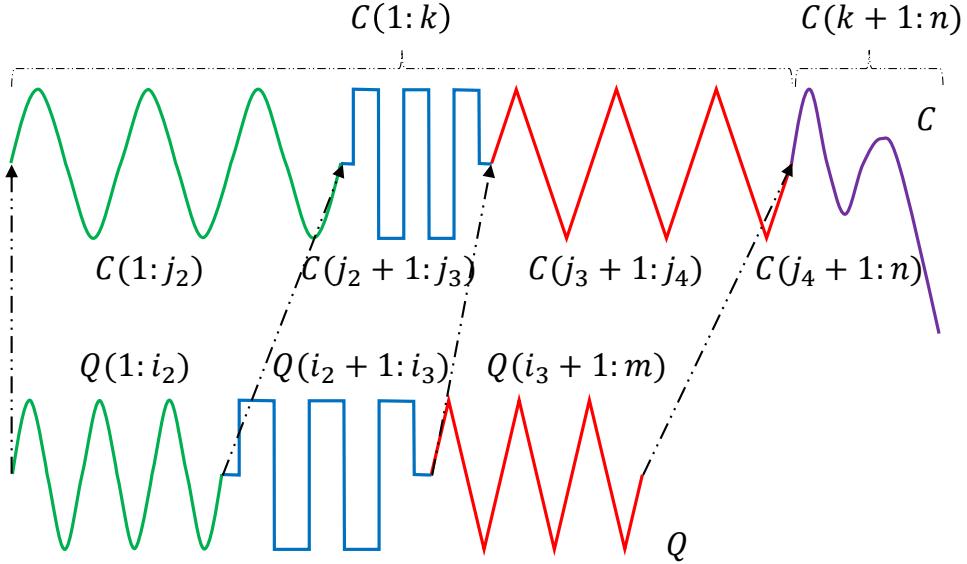


Figure 4.3: Intuition of piecewise scaling (PS).

1194 to design a new distance measure or framework that considers applying a scaling factor
 1195 on each of the phases as defined by dashed lines in Figure 4.3, during the comparison of
 1196 two time series.

1197 Our contributions are as follows:

- 1198 • We propose the first framework to achieve piecewise scaling (PS) invariance. In
 1199 particular, we focus on two instantiations of PSD, namely PSED (i.e., ED with PS
 1200 invariance) and PSDTW (i.e., DTW with PS invariance).
- 1201 • We design a dynamic programming method to compute PSD.
- 1202 • We propose a constrained version of PSD (cPSD) based on the allowed segment
 1203 lengths. Besides, two other speedup techniques have been proposed. For a partic-
 1204 ular instantiation of PSD, PDTW, we demonstrate the usage of a lower bound to
 1205 further speed it up.
- 1206 • We demonstrate that the segmentation results returned by PSD can improve the
 1207 accuracy of other distance measures.

1208 The rest of this paper is structured as follows. We present related work in Section 5.2
 1209 and preliminaries in Section 4.3. Section 5.3 introduces our new distance measure frame-
 1210 work, its constrained version, and speedup techniques. It is experimentally demonstrated
 1211 in Section 4.5 for the problem of querying. In Section 5.5, we conclude this study with
 1212 some future work.

4.2 Related Work

1213

This study focuses on distance measures of time series. For the overall review of time 1214 series, we direct the readers to [47, 57] for a more comprehensive understanding of this 1215 field. 1216

For many tasks, having appropriate distance measures that align with our intuition 1217 for the domains we work with is essential. One well-known distance measure is Dynamic 1218 Time Warping (DTW). It is initially designed for speech analysis [2]. However, DTW 1219 is computationally expensive. Lower bounds are used to speed up time series similarity 1220 search by admissibly pruning the unpromising candidates. One of the popular exact 1221 lower bounds of DTW is LB_{Keogh} . [58] improves the scalability of DTW by introducing a 1222 subsequence search suite of their four novel ideas, namely the UCR suite. For an overall 1223 review of lower bounds, we refer readers to [59, 60]. There is an approximate algorithm 1224 that approximates DTW with high accuracy while drastically cutting down the time and 1225 space requirements [61]. 1226

While ED is sensitive to distortions in the time axis, uniform scaling (US) has been 1227 shown to be a critical invariance in domains such as motion capture. [62] demonstrated 1228 that DTW is insufficient for handling global scaling effects, and that identifying DTW is 1229 not the solution to achieve this kind of invariance. There is a need for US. [63] extends 1230 the importance of uniform scaling to motif discovery. The authors show that meaningful 1231 motifs often suffer from a global scaling effect, causing standard motif finding algorithms 1232 to miss them completely. 1233

To the best of our knowledge, three studies analyze the combination of US and DTW, 1234 namely USDTW. It was first proposed by [53]. It extended LB_{Keogh} to bound the US- 1235 DTW. However, the extended LB_{Keogh} is still too loose with invariance to large amounts 1236 of uniform scaling. [54] and its follow-up study [55] proposed a new lower bound, namely 1237 LB_{Shen} ¹, which has been shown to be tighter than LB_{Keogh} on USDTW. 1238

To our surprise, despite a fruitful discussion of DTW, US, and USDTW, no study has 1239 proposed a distance measure capable of handling scaling effects across multiple scaling 1240 factors. This is precisely what we will address in this study. 1241

4.3 Preliminaries

1242

We refer to time as the contextual attribute because it provides the context for the mea- 1243 surements to be made. We refer to the measurements as the behavioral attributes. Time 1244 series are multivariate when more than one behavioral attribute is present. Otherwise, it 1245 is called univariate. We focus on the univariate case. 1246

¹It is denoted as LB_{New} in the original study. We rename it to prevent any confusion.

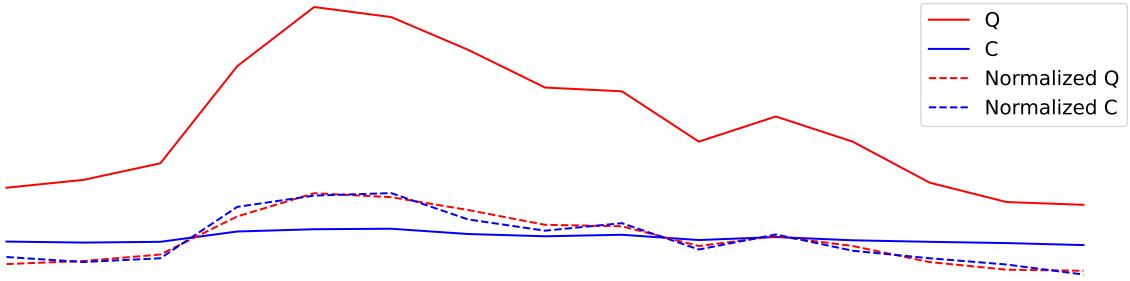


Figure 4.4: Z-normalization. Resulting time series have mean = 0 and std = 1.

1247 **Definition 1** (Time Series). A time series $T = t_1, t_2, \dots, t_n$ is a sequence of real-valued
1248 numbers with length = n .

1249 When two time series are involved in the discussion, we denote them as Q (Query)
1250 and C (Candidate), with lengths m and n , respectively. Since Q is the query sequence,
1251 it is not longer than C (i.e., $m \leq n$). The requirement of “ $m \leq n$ ” is a natural setting.
1252 In “Query by Content”, a user is going to search for a candidate in the database from a
1253 user-input query in which the query may only contain partial information of the target
1254 candidate. For example, a user often wants to find a song or tune that is lingering in
1255 their head by humming a part but not whole of the tune [64]. We are also interested in
1256 a segment or subsequence of a time series.

1257 **Definition 2** (Subsequence). A subsequence $T(i : j)$ of a time series T is a shorter time
1258 series that starts from position i and ends at position j with length = $j - i + 1$. Both
1259 ends are inclusive. Formally, $T(i : j) = t_i, t_{i+1}, \dots, t_j$, $1 \leq i \leq j \leq n$.

1260 We call $T(1 : j)$ the prefix of T of length j .

1261 Before comparison, we need to standardize or normalize them. A common way is
1262 Z-Normalization, which is $T = (T - \text{mean}(T)) / \text{std}(T)$, as shown in Figure 4.4. The most
1263 fundamental distance measure is the Euclidean Distance (ED).

1264 4.3.1 Euclidean Distance (ED)

1265 Given two points on a plane, it is intuitive to define the distance between them as the
1266 length of the line segment between them. This idea extends to the case of n -dimensions
1267 in time series with length n .

1268 **Definition 3** (Euclidean Distance (ED)). Given two series Q and C both with length n ,
1269 the Euclidean Distance between them is defined as:

$$\text{ED}(Q, C) = \sqrt{\sum_{i=1}^n (q_i - c_i)^2} \quad (4.1)$$

A square root is usually involved in the computation of distance measures. It is a ₁₂₇₀ monotonic function. Since it does not change the relative ranking of the results, we can ₁₂₇₁ omit the square root operation for simplicity and optimization. ED aligns the entries ₁₂₇₂ between two series in a one-to-one manner. Two similar series will have a large distance ₁₂₇₃ under ED if they are not aligned well in the time dimension. ED cannot handle local ₁₂₇₄ distortion along the time axis because warping in the alignment is not allowed. A stan- ₁₂₇₅ dard distance measure that provides warping invariance is called Dynamic Time Warping ₁₂₇₆ (DTW). ₁₂₇₇

4.3.2 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is an algorithm that measures similarity between two ₁₂₇₉ time series while accounting for local distortions. DTW aligns the two series by nonlin- ₁₂₈₀ early warping the time axis to minimize the final cumulative distance. ₁₂₈₁

Given two time series, Q and C , we first construct an m by n distance matrix M . ₁₂₈₂ The origin $(1, 1)$ is set at the bottom-left element of M . The (i, j) element of M ₁₂₈₃ contains the distance $d(q_i, c_j)$ between points q_i and c_j . This local distance $d(\cdot, \cdot)$ is ₁₂₈₄ usually calculated by $(q_i - c_j)^2$. Each (i, j) element refers to an alignment or mapping ₁₂₈₅ of the two points. A contiguous set of such elements forms a warping path W . W ₁₂₈₆ represents the non-linear alignment of Q and C . $W = w_1, w_2, \dots, w_K$ in which $w_k = (i, j)_k$ ₁₂₈₇ represents the mapping between q_i in Q and c_j in C , where $\max(m, n) \leq K \leq m + n - 1$. ₁₂₈₈ “ $\max(m, n) \leq K$ ” because the alignment of two series must include every point in both Q ₁₂₈₉ and C . “ $K \leq m + n - 1$ ” because the longest warping path is either the “concatenation of ₁₂₉₀ the bottom row and the rightmost column” (with the bottom-right cell being overlapped) ₁₂₉₁ or the “concatenation of the leftmost column and the top row” (with the top-left cell being ₁₂₉₂ overlapped). ₁₂₉₃

The warping path W is typically subject to the following three constraints. ₁₂₉₄

- Boundary conditions: $w_1 = (1, 1)$ and $w_K = (m, n)$. The first (last) point of Q ₁₂₉₅ must map to that of C . ₁₂₉₆
- Continuity: Given $w_k = (i, j)$ and $w_{k-1} = (i', j')$, $i - i' \leq 1$ and $j - j' \leq 1$. An ₁₂₉₇ entry in the warping path W is adjacent to its one-step previous entry. ₁₂₉₈
- Monotonicity: Given $w_k = (i, j)$ and $w_{k-1} = (i', j')$, $i - i' \geq 0$ and $j - j' \geq 0$. The ₁₂₉₉ warping path W does not go back. ₁₃₀₀

We denote W^* as a set of all allowed possible paths. ₁₃₀₁

Definition 4 (Dynamic Time Warping (DTW) [2]). DTW returns the minimum warping ₁₃₀₂ cost: ₁₃₀₃

$$\text{DTW}(Q, C) = \min_{W \in W^*} \sum_{k=1}^{|W|} w_k \quad (4.2)$$

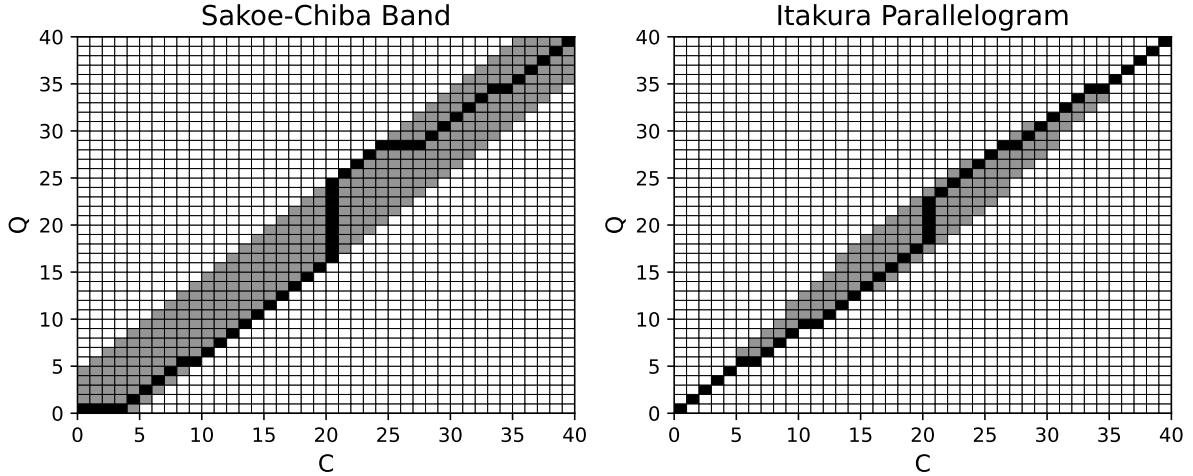


Figure 4.5: Visualization of D with local constraints. The black cells form the warping path.

To find the minimum warping cost and its corresponding warping path, we can use dynamic programming (DP) to evaluate the following recurrence.

$$D(i, j) = d(q_i, c_j) + \min \begin{cases} D(i - 1, j - 1), \\ D(i - 1, j), \\ D(i, j - 1) \end{cases} \quad (4.3)$$

It can be solved by building the accumulated cost matrix D , where the y-axis refers to Q , and the x-axis refers to C . The base cases, which are the first row and the first column, are defined as $D(1, j) = \sum_{k=1}^j d(q_1, c_k), j \in [1, n]$ and $D(i, 1) = \sum_{k=1}^i d(q_k, c_1), i \in [1, m]$. After we have initialized the base cases, we can fill up D starting from the bottom up. There are $m \times n$ entries in D . It takes $\mathcal{O}(mn)$ to fill it up. Once D is built, we can find the path corresponding to this minimum cost by simple backtracking from the end cell $D(m, n)$ to the origin cell $D(1, 1)$.

Some constraints have been proposed to prevent pathological warping paths with the aim of accuracy and efficiency. Two of the most popular are Sakoe-Chiba Band [2] and Itakura Parallelogram [3]. They only allow the warping paths to pass through the allowed region, as shown in Figure 4.5, by setting the cells outside this region in D to have ∞ accumulated distance cost. [65] suggested that these constraints can be viewed as constraints on the warping path entry $w_k = (i, j)_k$. It represents these constraints as inequalities applying to the indices i and j locally, without depending on the main diagonal in D . In the Sakoe-Chiba Band, the constraints can be represented as $j - r \leq$

$i \leq j + r$, where r is an integer. It is sometimes specified as a fraction (or percentage) 1321 of the longer time series length to ensure length invariance. For clarity, we assume r is 1322 an integer unless specified otherwise. This means that q_i can only align with c_j if their 1323 indices differ by at most r . Since r defines the maximum allowed difference between the 1324 mapping indices, $|n - m| \leq r$, to ensure that the end points of Q and C can map. In 1325 the Itakura Parallelogram, r is a function of i rather than a constant value. ED can 1326 be seen as a special case of DTW where the warping path is fixed to be diagonal. q_i 1327 aligns to c_i for every i (i.e., $r = 0$). DTW minimizes over all possible warping paths, 1328 and the warping path of ED is one of them. Because of the band, we only need to fill 1329 up the cells within the band. The complexity is $\mathcal{O}(\#\text{of_cells_inside})$. In the case of the 1330 Sakoe-Chiba Band, the band has a constant width $w = 2r + 1$. The complexity becomes 1331 $\mathcal{O}(w \times \text{length_of_diagonal}) = \mathcal{O}(rn)$. The constrained DTW is denoted as DTW_r . 1332

4.3.3 Uniform Scaling (US)

In US, we compare the whole sequence of Q to a prefix of C , as shown in Figure 4.2. 1334 The two compared sequences are scaled to the same length via interpolation before ED 1335 is applied. A common interpolation method is nearest neighbor interpolation. 1336

Definition 5 (Nearest Neighbor Interpolation). Given a time series T of length n and 1337 an integer L , Nearest Neighbor Interpolation scales T into T^L as follows: 1338

$$T_j^L = T_{\lceil n(j/L) \rceil} \quad \text{where } 1 \leq j \leq L \quad (4.4)$$

We can scale up or down a given series using Equation 4.4, as shown in Example 1. 1339

Example 1. Given a series $T = 1, 2, \dots, 6$ with length $n = 6$. Let $T(1 : 4)$ be the prefix 1340 of T of length $k = 4$ (i.e., $T(1 : 4) = 1, 2, 3, 4$). Given an integer $L = 8$, we compute 1341 $T(1 : 4)^8$ as follows. 1342

$$\begin{aligned} T(1 : 4)^8 &= T_{\lceil 4(1/8) \rceil}, T_{\lceil 4(2/8) \rceil}, T_{\lceil 4(3/8) \rceil}, \dots, T_{\lceil 4(8/8) \rceil} \\ &= T_1, T_1, T_2, \dots, T_4 \\ &= 1, 1, 2, \dots, 4. \end{aligned}$$

When $L > k$, T is said to be stretched. When $L < k$, T is said to be shrunk. 1343

Definition 6 (Uniform Scaling (US) [62]). Given two series Q and C , of length m and 1344 n respectively, and a scaling factor bound l , where $l \geq 1$. Let $C(1 : k)$ be the prefix of 1345 C , where $\lceil m/l \rceil \leq k \leq \min(\lfloor lm \rfloor, n)$, and $C(1 : k)^L$ be a rescaled version of $C(1 : k)$ with 1346 length L , where $L = \min(\lfloor lm \rfloor, n)$. L is called the alignment factor. $\min(\lfloor lm \rfloor, n)$ is the 1347

1348 largest alignment factor.

$$\text{US}(Q, C, l, L) = \min_{k=\lceil m/l \rceil}^{\min(\lfloor lm \rfloor, n)} \text{ED}(Q^L, C(1:k)^L) \quad (4.5)$$

1349 L is set as the largest alignment factor [55] to ensure all the points in Q and $C(1:k)$ are
1350 preserved during interpolation because of up-sampling, and the scaled version of all the
1351 prefixes is going to have the same length (i.e., L), for fair comparison, since comparison
1352 between two longer time series generally results in a larger distance measure. Through
1353 Equation 4.5, we find the minimum value and the corresponding argument (i.e., k) by
1354 checking the minimum value of the ED function from $\lceil m/l \rceil$ to $\min(\lfloor lm \rfloor, n)$. The scaling
1355 factor is defined by the argument minimum value of k . The smaller k is, the more we
1356 need to “stretch” C for Q to compare with $C(1:k)$, which is m/k times.

1357 Consider a time series database D comprising a set of candidate instances, and a
1358 single query series Q . The search task aims to retrieve the most similar instance (or
1359 top- k instances) from D to Q . We maintain Q as a fixed reference and extract only the
1360 prefixes from each instance in D for comparison. Furthermore, to simplify notation, we
1361 apply scaling exclusively to the instances in D , leaving Q unscaled.

1362 4.3.4 Uniform Scaling & Dynamic Time Warping (USDTW)

1363 Uniform Scaling & Dynamic Time Warping (USDTW) measures the similarity by applying
1364 scaling with an appropriate scaling factor on C and then applying DTW. The tail
1365 part of C can be ignored, as shown in Figure 4.2.

1366 **Definition 7** (Uniform Scaling & Dynamic Time Warping (USDTW) [55]). With the
1367 same notations defined in Definition 6,

$$\text{USDTW}_r(Q, C, l, L) = \min_{k=\lceil m/l \rceil}^{\min(\lfloor lm \rfloor, n)} \text{DTW}_r(Q^L, C(1:k)^L) \quad (4.6)$$

1368 where r is the DTW constraint parameter.

1369 We replace the ED function in Equation 4.5 by DTW function to form Equation 4.6.
1370 As mentioned in Section 5.1, there may exist more than one scaling factor. Hence,
1371 both the existing distance measures, US and USDTW, which are designed to handle
1372 only one scaling factor, are insufficient. This motivates us to design a new framework of
1373 distance measures.

4.3.5 Lower Bounds for DTW and USDTW

1374

We first introduce the concept of lower bounds and explain how they can benefit search. 1375
 DTW is computationally more expensive than ED. They compute an accumulated cost 1376
 matrix of size $m \times n$, where m and n denote the lengths of the two time series. It 1377
 results in quadratic complexity. This complexity poses a challenge for similarity search. 1378
 The most common approach is to compute a lower bound of the real value, which is 1379
 computationally cheap and tight. We can use this lower bound to filter out unpromising 1380
 candidates and perform the expensive distance computation only on the small set of 1381
 promising candidates. In searching, we would keep track of the best_so_far distance bsf 1382
 between the query Q and the testing candidates. When testing a new candidate C , we 1383
 first compute the $\text{LB}(Q, C)$. We only compute the actual DTW when $\text{LB}(Q, C) \leq \text{bsf}$. 1384

We then introduce some common lower bounds. 1385

Kim Lower Bound [66]: LB_{Kim} is a simple and fast lower bound of DTW. The 1386
 complexity is $O(1)$. It uses the four features in Q and C . We denote t_{-1} as the last point 1387
 and t_{\max} (t_{\min}) as the maximum (minimum) point in time series T . 1388

$$\text{LB}_{\text{Kim}} = \max \begin{cases} d(q_1, c_1) \\ d(q_{-1}, c_{-1}) \\ d(q_{\max}, c_{\max}) \\ d(q_{\min}, c_{\min}) \end{cases} \quad (4.7)$$

$d(\cdot, \cdot)$ refers to the local distance used in the point alignment. The first two lines come 1389
 from the boundary condition in DTW. The alignment between the first pair of points and 1390
 the last pair of points must contribute to the accumulated sum of DTW. Each point in 1391
 Q must align with some point in C , and vice versa. Each alignment contributes a local 1392
 distance to the final sum. The minimum possible local distance for the alignment of q_{\max} 1393
 would be the one aligned with c_{\max} . The same applies to the last line in the equation. 1394

There is a simplification of LB_{Kim} . Only the first and last pair are used in the lower 1395
 bound computation. In the normalized time series, the third and fourth rows in Equa- 1396
 tion 4.7 should have small values [58]. Ignoring them can improve the complexity from 1397
 $\mathcal{O}(n)$ to $\mathcal{O}(1)$. The simplified lower bound is $\text{LB}_{\text{KimFL}} = d(q_1, c_1) + d(q_{-1}, c_{-1})$. 1398

Keogh Lower Bound [65]: LB_{Keogh} builds the upper U and lower envelopes L of 1399
 one of the time series out of the two compared sequences. Usually, the envelopes are 1400
 constructed for Q instead of C as we will compare one Q against many C 's. Otherwise, 1401
 we need to build the envelopes for each candidate instead [58]. 1402

To the best of our knowledge, they are the first to interpret the constraint as a 1403
 restriction on the indices of the warping path $w_k = (i, j)_k$ such that $j - r \leq i \leq j + r$, 1404
 where r defines the allowable deviation of alignment between q_i and c_j . For ease of 1405

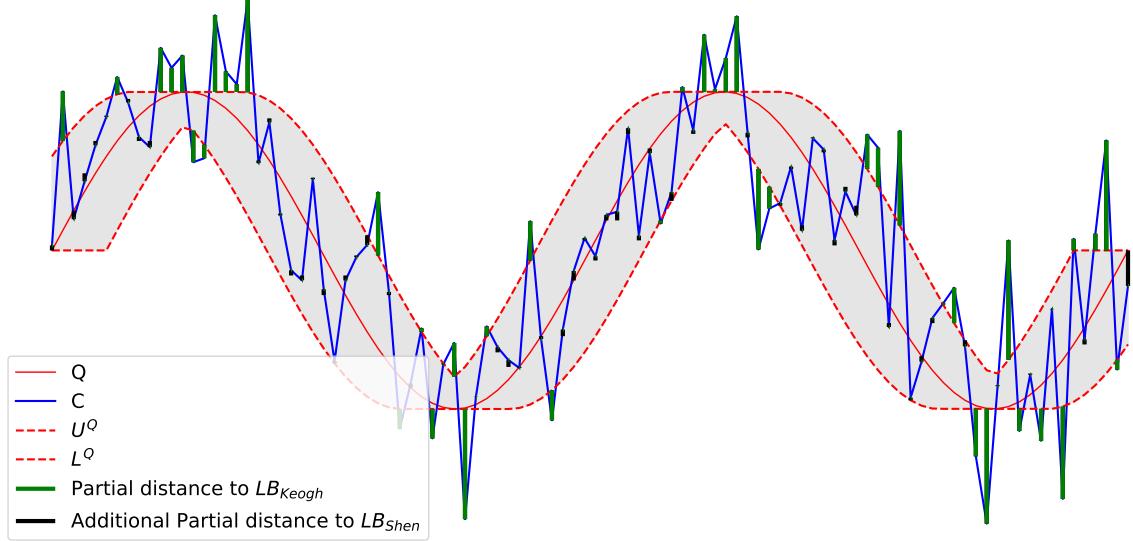


Figure 4.6: Visualization of LB_{Keogh} and LB_{Shen}

exposition, we focus on the most used constraint in the literature, which is the Sakoe-Chiba Band [67]. Two sequences are constructed for Q , namely the upper U^Q and lower envelopes L^Q of Q as shown in Figure 4.6. For each q_i , we would assign a window of q_i based on its index i as follows.

$$\begin{aligned} U_i^Q &= \max(q_{\max(1,i-r)} : q_{\min(i+r,m)}) \\ L_i^Q &= \min(q_{\max(1,i-r)} : q_{\min(i+r,m)}) \end{aligned} \quad (4.8)$$

$\max(1, \cdot)$ and $\min(\cdot, m)$ are used for handling the boundary cases. U^Q and L^Q together form a bounding envelope that encloses the original Q , it is the grey region in the figure. For each c_j , either (c_j, U_j^Q) or (c_j, L_j^Q) corresponds to the possible alignment that contributes the minimum distances if c_j falls outside the envelope. Herein, the lower bound is the sum of these distances, as shown in the following equation.

$$\text{LB}_{\text{Keogh}}(Q, C) = \sum_{j=1}^n \begin{cases} d(c_j, U_j^Q) & \text{if } c_j > U_j^Q \\ d(c_j, L_j^Q) & \text{if } c_j < L_j^Q \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

Visually, these distances are the distances between c_j outside the envelope and the vertically corresponding points on the envelope. The distances are the green bars in the figure. Equation 4.9 returns the sum of the green bars.

Shen Lower Bound [54, 55]: It leverages the boundary and continuity conditions to create a lower bound of DTW. It can be used to lower-bound the USDTW with slight modification. The intuition is to find the minimum possible alignment of each c_j with

points in Q , subject to the local constraint r from DTW and the global constraint l from US. 1421
1422

We will first introduce LB_{Shen} for DTW. Given the candidate sequence C with length n , for each c_j , we create its possible reach \mathbf{q}_j in Q under DTW as in Equation 4.10. The elements \mathbf{q}_j form an indexed collection $\mathbb{Q} = (\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{\min(\lfloor lm \rfloor, n)})$. 1423
1424
1425

$$\mathbf{q}_j = (q_{\max(1, j-r)}, \dots, q_{\min(j+r, m)}) \quad (4.10)$$

We define $\delta(x, Y) = \min_{y \in Y} d(x, y)$. The possible minimum cost contributed by the alignment of c_j and some point in Q to the accumulated sum would be $\delta(c_j, \mathbf{q}_j)$. The lower bound LB_{Shen} is defined as: 1426
1427
1428

$$\text{LB}_{\text{Shen}}(Q, C) = d(c_1, q_1) + \sum_{j=2}^{n-1} \delta(c_j, \mathbf{q}_j) + d(c_n, q_m) \quad (4.11)$$

We direct the reader to [54] for the formal proof, while an intuition of the proof is presented here. There are three items on the right-hand side of Equation 4.11. The continuity requirement ensures that each c_i is involved in at least one alignment. The first and the last items are from the boundary condition. The middle item returns the possible minimum distances contributed by each c_j 's, where $2 \leq j \leq n - 1$. It is obvious that LB_{Shen} in Equation 4.11 is tighter when we use the first pair of points and the last pair of points instead of doing the middle summation all from $j = 1$ to n . It is because the distance contributed by the first pair (last pair) must be greater than $\delta(c_1, \mathbf{q}_1)$ ($\delta(c_n, \mathbf{q}_n)$). 1429
1430
1431
1432
1433
1434
1435
1436

It is proven that it is tighter than LB_{Keogh} [54]. It is shown in Figure 4.6 visually. The black bars refer to the additional lower bound distance sum on top of LB_{Keogh} . We will give an intuitive proof here. Both LB_{Keogh} and LB_{Shen} compute the lower bounds by summing the local distance resulting from the alignment of each c_j with some points in Q , which is guaranteed to be not greater than the partial distance contributed by the real alignment, which we only know until we compute the exact distance measure. In LB_{Keogh} , if this c_j falls outside the envelope, this local distance is the vertical distance between c_j and the envelope. If c_j falls inside, this local distance would be 0. For those points outside the envelope, the local distances for c_j of LB_{Keogh} and LB_{Shen} are the same. But LB_{Shen} aims to return the minimum possible partial distance for each c_j , even within the envelope. Hence, $\text{LB}_{\text{Keogh}} \leq \text{LB}_{\text{Shen}}$. 1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447

In order to compute Equation 4.11 efficiently, we first sort sequences \mathbf{q}_j and the resulting sorted sequences are denoted as $\tilde{\mathbf{q}}_j$. The sorted sequences allow us to do a binary search when we are calculating $\delta(c_j, \tilde{\mathbf{q}}_j)$ in contrast to $\delta(c_j, \mathbf{q}_j)$. The sorting only needs to be done once because we are testing the same Q with different candidates. 1448
1449
1450
1451

It can be extended to the USDTW case [54, 55]. The possible reach now is not only defined by r , but also by the scaling factor bound l . 1452
1453

$$\mathbf{q}_j = (q_{\max(1, \lceil j/l \rceil - r)}, \dots, q_{\min(\lfloor j/l \rfloor + r, m)}) \quad (4.12)$$

[54] proves the following theorem to allow us to consider the lower bound between each prefix of C and Q without the scaling up of each prefix of C (i.e., $C(1 : k)^L$) and the scaling up of Q (i.e., Q^L).

Theorem 8. *For any $\lceil m/l \rceil \leq k \leq \min(\lfloor lm \rfloor, n)$, $\text{DTW}_r(Q^{\min(\lfloor lm \rfloor, n)}, C(1 : k)^{\min(\lfloor lm \rfloor, n)})$ is always lower bounded by $\sum_{j=1}^k \delta(c_j, \mathbf{q}_j)$.*

Recall that USDTW calculates DTW distances between each rescaled prefix of C to the rescaled Q , and outputs the minimum DTW distance, as in Equation 4.6. The incremental nature of the lower bound frees us to calculate the lower bound of each DTW distance from scratch. This theorem allows us to first calculate the lower bound of the shortest prefix $C(1 : \lceil m/l \rceil)$ of C and Q , and then incrementally calculate the lower bound of the longer prefix with length from $\lceil m/l \rceil + 1$ to $\min(\lfloor lm \rfloor, n)$ by adding on each $\delta(c_j, \mathbf{q}_j)$. To note, it also means that if $\text{LB}_{\text{Shen}}(Q, C(1 : k))$ is larger than a value, namely bsf, $\text{LB}_{\text{Shen}}(Q, C(1 : k'))$, where $k' > k$, would also be larger than bsf. To note, we can tighten LB_{Shen} by using $d(c_1, q_1)$ instead of $\delta(c_1, \mathbf{q}_1)$.

The above analysis can also apply to the case of US distance. We only need to define the corresponding reaches as follows:

$$\mathbf{q}_j = (q_{\max(1, \lceil j/l \rceil)}, \dots, q_{\min(\lfloor j/l \rfloor, m)}) \quad (4.13)$$

4.4 Piecewise Scaling Distance

The motivation stems from the limitations of US and USDTW. They assume that the relationship between Q and C is governed by only a single, global scaling factor. This assumption fails when applied to multi-rate data, where different phases of the time series express at different rates.

Note that existing studies [62, 58] can find all scaling factors, defined by the chosen prefix of C , ranging from $\lceil m/l \rceil$ to $\min(\lfloor lm \rfloor, n)$. However, they can use only one of them in the scaling. Consider the following illustrative example in ASCII text [58], where character repetition represents the duration of spoken phonemes, and space indicates a pause:

- “time series 20 25” and “time series 20 25”. Here, the Hamming distance (the discrete analogue of ED) fails due to misalignment in the underlined locations, but the string edit distance (the discrete analogue of DTW) can resolve it.
- “time sseerriiss 222000222555”. This sequence exhibits three distinct phases with scaling factors of 1, 2, and 3, respectively. The corresponding invariance cannot be

achieved by DTW or US (which is restricted to a single global scalar). US would 1485
enforce a single compromised global scale instead. 1486

In Query-by-Content scenarios, such as query-by-humming or gesture retrieval, the 1487
query is generated by a human. Humans do not maintain a consistent rate for each 1488
phase. For example, humans rush through a familiar sequence but slow down for a 1489
new or complex sequence. It is commonly observed in a piece of music performed by a 1490
beginner. The piece's tempo is not uniform. 1491

We introduce a novel distance measure framework, termed Piecewise Scaling Distance 1492
(PSD). It addresses the local scaling effect within each phase by employing a scaling factor 1493
to each phase, instead of using a single scaling factor for the whole time series. It releases 1494
the basic constraint or assumption made in US and USDTW. PSD employs an existing 1495
distance metric, such as ED or DTW, to quantify the similarity of aligned segment pairs. 1496
While the PSD framework is agnostic to the underlying metric, this study focuses on its 1497
two fundamental instantiations: 1498

- PSED, which employs ED to compute the similarity of aligned segment pairs. 1499
- PSDTW, which employs DTW to compute the similarity of aligned segment pairs. 1500

In the formulation that follows, we utilize PSDTW as the running example. PSDTW 1501
generalizes PSED. The PSED formulation can be trivially derived from it. 1502

4.4.1 Piecewise Scaling & Dynamic Time Warping (PSDTW) 1503

Problem formulation: To simplify the discussion, we focus on the comparison between 1504
 Q and the entire sequence C , rather than a prefix of C . Note that this formulation can 1505
be generalized to prefix matching as in Definition 6 and Definition 7. 1506

Given two sequences Q and C , where Q is not longer than C , $|Q| = m \leq |C| = n$, and 1507
the number of segments or pieces P allowed, our goal is to segmentalize both Q and C 1508
into P contiguous segments automatically in a way that minimize the total sum of DTW 1509
distance of aligned segment pairs with interpolation. 1510

Definition 9 (Piecewise Scaling & Dynamic Time Warping (PSDTW)). With the same 1511
notations defined in Definition 7, 1512

$$\begin{aligned} \text{PSDTW}_r(Q, C, l, L, P) = \\ \min_{\substack{i_1 < i_2 < \dots < i_{P+1} \\ j_1 < j_2 < \dots < j_{P+1}}} \sum_{p=1}^P \text{DTW}_r(Q(i_p + 1 : i_{p+1})^L, \\ C(j_p + 1 : j_{p+1})^L) \end{aligned} \quad (4.14)$$

, where $i_1 = 0$, $i_{P+1} = m$, $j_1 = 0$, $j_{P+1} = n$ and the setting of L will be discussed later. 1513

Algorithm 1 Naive PSDTW

Input: Query series Q , Candidate series C , DTW constraint parameter r (in fraction), Number of pieces P , Scaling parameter L

Output: Distance Matrix D of size $(m + 1) \times (n + 1) \times (P + 1)$

```
1: Initialize  $D$  with  $\infty$ 
2:  $D[0, 0, 0] \leftarrow 0$ 
3: for  $p \leftarrow 1$  to  $P$  do
4:   for  $i \leftarrow 1$  to  $m$  do
5:     for  $j \leftarrow 1$  to  $n$  do
6:        $D[i, j, p] \leftarrow \min_{i' < i, j' < j} \{D[i', j', p - 1]$ 
          $+ \text{DTW}_r(Q(i' + 1 : i)^L, C(j' + 1 : j)^L)\}$ 
7: return  $D[m, n, P]$ 
```

Algorithm 2 Line 6 in Algorithm 1

```
1: for  $i' \leftarrow 0$  to  $i - 1$  do
2:   for  $j' \leftarrow 0$  to  $j - 1$  do
3:      $dist_{\text{prev}} \leftarrow D[i', j', p - 1]$ 
4:      $dist_{\text{seg}} \leftarrow \text{DTW}_r(Q(i' + 1 : i)^L, C(j' + 1 : j)^L)$ 
5:      $D[i, j, p] \leftarrow \min(D[i, j, p], dist_{\text{prev}} + dist_{\text{seg}})$ 
```

1514 Essentially, L needs to be at least the length of all segments to preserve all the points
1515 by up-sampling.

1516 We refer to Figure 4.3 to clarify Equation 4.14. Given two sequences Q and C , with the
1517 aid of different colors and the dashed lines, we observe that Q consists of three segments
1518 while C consists of four segments, with the first three segments of C similar to those
1519 of Q . They form three segment pairs. The scaling factor used in each segment pair is
1520 determined from the length of the two subsequences involved, i.e., $(i_{p+1} - i_p)/(j_{p+1} - j_p)$.
1521 These three parts have different scaling factors. For example, the first part in C is the
1522 stretched version of that in Q . The second part in C is the compressed version of that in
1523 Q .

1524 Equation 4.14 can be formulated in a recurrence relation as follows. Let $D[i, j, p]$ be
1525 the minimum cost to align the first i points in Q (i.e., $Q(1 : i)$) with the first j points in
1526 C (i.e., $C(1 : j)$) using exactly p segments:

$$D[i, j, p] = \min_{\substack{i' < i \\ j' < j}} \left\{ D[i', j', p - 1] + \text{DTW}_r(Q(i' + 1 : i)^L, C(j' + 1 : j)^L) \right\} \quad (4.15)$$

1527 **Naive PSDTW:**

Algorithm 3 Initialization of PSDTW

- 1: $L_{\text{gmin}}^Q \leftarrow \lceil (m/P)/\sqrt{l} \rceil, \quad L_{\text{gmax}}^Q \leftarrow \lfloor (m/P)\sqrt{l} \rfloor$ ▷ “g” refers to “global”.
 - 2: $L_{\text{gmin}}^C \leftarrow \lceil (n/P)/\sqrt{l} \rceil, \quad L_{\text{gmax}}^C \leftarrow \lfloor (n/P)\sqrt{l} \rfloor$
 - 3: $L = \max(L_{\text{gmax}}^Q, L_{\text{gmin}}^C)$
 - 4: Initialize D of size $(m+1) \times (n+1) \times (P+1)$ with ∞
 - 5: $D[0, 0, 0] \leftarrow 0$
-

Our goal is $D[m, n, P]$. Equation 4.15 can be solved exactly by dynamic programming 1528 (DP). The base case is $D[0, 0, 0] = 0$. It refers to the zero cost to align the first 0 point 1529 (i.e., the empty prefix) of Q with that of C . Other cells in D are first initialized with ∞ . 1530 They are calculated using a bottom-up approach via Equation 4.15. A straightforward 1531 implementation in DP is shown in Algorithm 1. Line 6 is achieved by looping all the 1532 previous indices of the current i and j as in Algorithm 2. 1533

We explain the lines in Algorithm 2. Line 3 retrieves the accumulated distance cost 1534 from the beginning up to the endpoints (i', j') , and saves it as $dist_{\text{prev}}$. Line 4 considers 1535 the current aligned segment pair, which consists of $Q(i'+1 : i)$ and $C(j'+1, j)$, and they 1536 are interpolated to the length L . The DTW distance of this pair is calculated and saved 1537 as $dist_{\text{seg}}$. 1538

We now analyze the time complexity of Algorithm 1. There are Pmn entries in D . 1539 The \min operator in line 6 takes $\mathcal{O}(mn)$. Hence, the time complexity of Algorithm 1 1540 is $\mathcal{O}(Pm^2n^2) = \mathcal{O}(Pn^4)$, multiplied by the running time of the DTW. It is slow, which 1541 prevents us from using it in practice. To note, we use r in a fraction instead of a fixed 1542 integer here. This allows the deviation tolerance to scale adaptively with pieces of varying 1543 lengths. 1544

4.4.2 Speedup Techniques

1545

Length constraints of the segment: A way to reduce complexity and to prevent 1546 pathological segment pairs is to limit the possible segment lengths that are considered by 1547 constraining the minimum and maximum lengths of segments. It is similar to constrained 1548 DTW, in which we limit the search space of the warping path as in Figure 4.5. The 1549 version of PSDTW that considers the segment constraint is termed constrained PSDTW 1550 (cPSDTW). It is shown in Algorithm 4. The uncolored part shows the main logic, while 1551 the colored part shows the speedup techniques, which will be explained later. 1552

We initialize in Algorithm 3. For a given number of segments P , the expected length 1553 of each segment in Q and C would be m/P and n/P , respectively. For Q , we set the 1554 minimum possible segment length L_{gmin}^Q to be $\lceil (m/P)/\sqrt{l} \rceil$ and the maximum possible 1555 length L_{gmax}^Q to be $\lfloor (m/P)\sqrt{l} \rfloor$ in line 1 such that the scaling ratio of any two segments 1556 in Q would be bounded by l . It allows some deviation in the length of the segments from 1557 their expected length. Similarly, we compute the segment constraints for C in line 2. We 1558

Algorithm 4 Constrained PSDTW (cPSDTW) with early abandoning and lower bounding

Input: Query series Q , Candidate series C , DTW constraint parameter r (in fraction), Number of pieces P , best_so_far bsf

Output: The final distance $D[m, n, P]$ if $D[m, n, P] \leq \text{bsf}$, otherwise ∞

- 1: Execute Algorithm 3 for initialization
- 2: **for** $p \leftarrow 1$ **to** P **do**
- 3: **for** $i \leftarrow (p \cdot L_{\text{gmin}}^Q)$ **to** $\min(p \cdot L_{\text{gmax}}^Q, m)$ **do** ▷ The iterations can be parallelized.
- 4: **for** $L^Q \leftarrow L_{\text{gmin}}^Q$ **to** L_{gmax}^Q **do**
- 5: $i' \leftarrow i - L^Q$ ▷ i' : End point of previous segment on Q .
- 6: $Q' \leftarrow \text{rev}(Q(i' + 1 : i))$ ▷ $\text{rev}(T)$: Reverse the input series T .
- 7: $L_{\text{min}}^C \leftarrow \max(L_{\text{gmin}}^C, \lceil L^Q/l \rceil)$
- 8: $L_{\text{max}}^C \leftarrow \min(\lfloor L^Q/l \rfloor, L_{\text{gmax}}^C)$
- 9: $r' \leftarrow r \times \max(L^Q, L_{\text{max}}^C)$ ▷ Compute the integer value of r .
- 10: **for** $k \leftarrow 1$ **to** L_{max}^C **do**
- 11: $\mathbf{q}_k \leftarrow (q'_{\max(1, \lceil k/l \rceil - r')}, \dots, q'_{\min(\lfloor k/l \rfloor + r', L^Q)})$ ▷ Construct indexed collection \mathbb{Q} .
- 12: $\tilde{\mathbf{q}}_k \leftarrow \text{sort}(\mathbf{q}_k)$
- 13: **for** $j \leftarrow (p \cdot L_{\text{gmin}}^C)$ **to** $\min(p \cdot L_{\text{gmax}}^C, n)$ **do**
- 14: **for** $L^C \leftarrow L_{\text{min}}^C$ **to** L_{max}^C **do**
- 15: $j' \leftarrow j - L^C$
- 16: $C' \leftarrow \text{rev}(C(j' + 1 : j))$
- 17: $dist_{\text{prev}} \leftarrow D[i', j', p - 1]$
- 18: **if** $dist_{\text{prev}} = \infty$ **then**
- 19: **continue**
- 20: **if** $dist_{\text{prev}} > \text{bsf}$ **then**
- 21: **continue**
- 22: **if** $dist_{\text{prev}} > D[i, j, p]$ **then** ▷ $D[i, j, p]$ stores the best so far.
- 23: **continue**
- 24: **if** $L^C = L_{\text{min}}^C$ **then** ▷ Compute the lower bound from sketch.
- 25: $lb = (c'_1 - q'_1)^2$ ▷ Partial distance contributed by the first alignment.
- 26: **for** $k \leftarrow 2$ **to** L^C **do**
- 27: $lb = lb + \delta(c'_k, \tilde{\mathbf{q}}_k)$
- 28: **else**
- 29: $lb = lb + \delta(c'_{L^C}, \tilde{\mathbf{q}}_{L^C})$ ▷ Compute the lower bound incrementally.
- 30: $lb_{\text{check}} = lb - \delta(c'_{L^C}, \tilde{\mathbf{q}}_{L^C}) + (c'_{-1} - q'_{-1})^2$ ▷ Tighten the lower bound by using the last alignment.
- 31: **if** $dist_{\text{prev}} + lb_{\text{check}} > D[i, j, p]$ **then**
- 32: **continue**
- 33: $dist_{\text{seg}} \leftarrow \text{DTW}_r(Q'^L, C'^L)$
- 34: **if** $dist_{\text{prev}} + dist_{\text{seg}} < D[i, j, p]$ **then**
- 35: $D[i, j, p] \leftarrow dist_{\text{prev}} + dist_{\text{seg}}$ ▷ Also save pointer (i', j') for the cut.
- 36: **return** $D[m, n, P]$

1559 set the maximum segment length be the alignment factor L in line 3. We set the base
1560 case in line 5.

1561 We fill the table D in Algorithm 4. In line 3, given p pieces in Q , the ending index
1562 of the last piece (i.e., the p^{th} piece) will range from $(p \cdot L_{\text{gmin}}^Q)$, given all the p pieces are
1563 in minimum length L_{gmin}^Q , to $\min(p \cdot L_{\text{gmax}}^Q, m)$, given all the p pieces are in maximum
1564 length L_{gmax}^Q .

1565 In line 4, we enumerate for all the allowed lengths L^Q . For the segment with length
1566 L^Q in Q , the length L^C of the corresponding aligned segment in C will have a range from

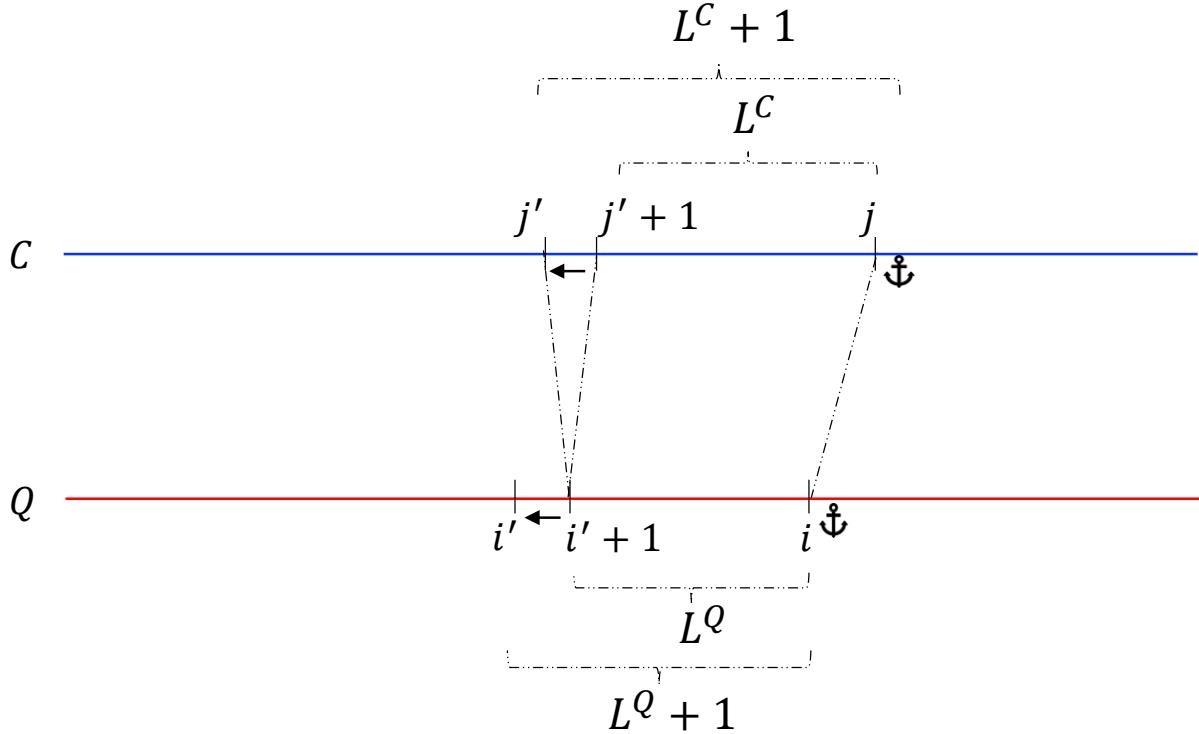


Figure 4.7: Relationship of L_Q and L_C .

L_{\min}^C to L_{\max}^C , as defined in lines 7–8, such that the ratio of L^Q and L^C would be bounded by l . Because L^Q and L^C increase in line 4 and line 14, and the i and j are fixed by the outer loop, in line 3 and line 13, respectively, the i' and j' decrease correspondingly. It is visualized in Figure 4.7. A segment in Q with ending index i and starting index $i' + 1$ would be compared to a set of segments in C , all of which have a fixed end at j and a decreasing starting index, starting at $j' + 1$. For example, the starting index of the first segment with being compared is $j' + 1$, which has length L^C , and that of the second segment is j' , which has length $L^C + 1$, as in Figure 4.7.

In line 18, we terminate the current iteration if there are no previously valid segment pairs (i.e., $dist_{\text{prev}} = \infty$).

In line 22, if the accumulated cost $dist_{\text{prev}}$ exceeds the current best so far, we stop the current iteration as the resulting ($dist_{\text{prev}} + dist_{\text{seg}}$) is guaranteed to be greater than the current best so far, which is stored in $D[i, j, p]$.

To be consistent with the result of using the lower bound speedup technique, which will be introduced later, we compute the DTW in the reverse manner in line 33. Due to the nearest neighbor interpolation, $\text{DTW}_r(Q'^L, C'^L)$ may not equal to $\text{DTW}_r(\text{rev}(Q')^L, \text{rev}(C')^L)$.

In line 35, we also save the pairs (i', j') that serve as cutting points between segments to obtain the segmentation result.

Parallel computing: We observe that the recurrence relation for the state $D[i, j, p]$ at stage p depends exclusively on the states computed at stage $p - 1$, as shown in Equa-

1587 tion 4.15. There are no stage dependencies over indices i and j . It allows us to parallelize
 1588 the loops over indices i and j on lines 3 and 13. In the following experimental section, we
 1589 distribute the i-loop iterations (i.e., line 3 in Algorithm 4, which is colored in blue) across
 1590 available threads only because there are already sufficient iterations from the i-loop to
 1591 fill the available threads. There are $\left(\min(p \cdot L_{\text{gmax}}^Q, m) - (p \cdot L_{\text{gmin}}^Q) + 1\right)$ iterations from
 1592 the i-loop. The parallel execution of the i -loop is implemented by `prange` in Numba in
 1593 Python.

1594 **Early Abandoning in nearest neighbor search:** To accelerate the nearest neighbor
 1595 search (or top- k search) for a query Q on a candidate set, we employ an early abandoning
 1596 strategy. It prunes the search branch within the PSD computation of a specific candidate
 1597 C as soon as the result corresponding to this search branch is determined to be sub-
 1598 optimal. We maintain a variable, bsf (best-so-far), which represents the minimum final
 1599 distance among the candidates processed with Q so far. bsf serves as an upper-bound
 1600 threshold. During the evaluation of a new candidate C , we monitor the accumulated
 1601 partial distance, $\text{dist}_{\text{prev}}$. If $\text{dist}_{\text{prev}}$ exceeds bsf , the corresponding final distance is guar-
 1602 anteed to exceed bsf . In such cases, the current search branch is immediately terminated.
 1603 The implementation of this pruning mechanism is detailed in lines 20–21 in Algorithm 4,
 1604 which are colored in orange. In the case of top- k search, we must maintain the top- k
 1605 final distances and use the k -th distance as the threshold.

1606 Lower bound:

1607 When we use DTW as a routine in PSD, the computation of the DTW of the interpo-
 1608 lated segment can be sped up by computing LB_{Shen} for the lower bound. From Figure 4.7,
 1609 we observe that a segment of Q , with length L^Q , is compared to a set of growing seg-
 1610 ments of C , with a fixed end at j . The length of these segments is from L_{min}^C to L_{max}^C ,
 1611 as indicated in line 14 in Algorithm 4. They share the same suffix with length L_{min}^C . It
 1612 encourages us to view both Q and C reversely. The reversed segments are denoted as Q'
 1613 and C' , as in lines 6 and 16 in Algorithm 4. In this reversed view, they share the same
 1614 prefix with length L_{min}^C . Hence, we construct an indexed collection $\tilde{\mathbb{Q}}$ for Q' with the
 1615 maximum length of the segments being compared in C , which is L_{max}^C , as in lines 10–12.

1616 In lines 24–27, we first compute the lower bound between Q' and the shortest segment
 1617 of C . We add the minimum possible contribution of each c' to the distance contributed
 1618 by the first alignment, which is $lb = (c'_1 - q'_1)$. For the lower bound of the subsequent
 1619 segments, we compute them incrementally in line 29. Because the last point of segment
 1620 C' must map to the last point of Q' , we further tighten the lower bound by using the last
 1621 alignment in line 30 instead.

1622 Furthermore, we can reduce the computational overhead of constructing the sorted
 1623 reaches $\tilde{\mathbb{q}}_k$ (lines 9–12). As illustrated in Figure 4.7, the segment of Q involved in the
 1624 comparison grows incrementally. Since the reversed versions of these segments share a
 1625 common prefix, the sorted reaches $\tilde{\mathbb{q}}_k$ computed for a segment Q' of length L^Q can be

Algorithm 5 Replace lines 3 to 9 in Algorithm 5 to reuse the computed sorted reaches \tilde{q}_k .

```

1:  $r'_{\text{cache}} \leftarrow -1$ 
2: for  $i \leftarrow (p \cdot L_{\text{gmin}}^Q)$  to  $\min(p \cdot L_{\text{gmax}}^Q, m)$  do
3:   for  $L^Q \leftarrow L_{\text{gmin}}^Q$  to  $L_{\text{gmax}}^Q$  do
4:      $i' \leftarrow i - L^Q$ 
5:      $Q' \leftarrow \text{rev}(Q(i' + 1 : i))$ 
6:      $L_{\text{min}}^C \leftarrow \max(L_{\text{gmin}}^C, \lceil L^Q/l \rceil)$ 
7:      $L_{\text{max}}^C \leftarrow \min(\lfloor L^Q/l \rfloor, L_{\text{gmax}}^C)$ 
8:      $r' \leftarrow r \times \max(L^Q, L_{\text{max}}^C)$ 
9:     if  $r'_{\text{cache}} = r'$  then
10:      for  $k \leftarrow 1$  to  $L_{\text{max}}^C$  do
11:        if  $k \leq |Q'|$  then
12:           $e_{\text{prev}} \leftarrow \min(\lfloor kl \rfloor + r', L^Q - 1)$ 
13:           $e_{\text{new}} \leftarrow \min(\lfloor kl \rfloor + r', L^Q)$ 
14:          if  $e_{\text{new}} > e_{\text{prev}}$  then
15:             $\tilde{q}_k \leftarrow \tilde{q}_k \cup q'_{e_{\text{new}}}$ 
16:        else
17:           $q_k \leftarrow (q'_{\max(1, \lceil k/l \rceil - r')}, \dots,$ 
18:           $q'_{\min(\lfloor kl \rfloor + r', L^Q)})$  ▷  $L^Q$  equals to  $|Q'|$ .
19:        else
20:          for  $k \leftarrow 1$  to  $L_{\text{max}}^C$  do
21:             $q_k \leftarrow (q'_{\max(1, \lceil k/l \rceil - r')}, \dots,$ 
22:             $q'_{\min(\lfloor kl \rfloor + r', L^Q)})$ 
23:             $\tilde{q}_k \leftarrow \text{sort}(\tilde{q}_k)$ 
 $r_{\text{cache}} \leftarrow r'$ 

```

reused to compute those for the subsequent segments. This optimization is detailed in 1626
Algorithm 5, with the new components highlighted in blue. It is important to note that 1627
because r' is a function of L^Q , and the construction of \tilde{q}_k depends on r , reuse is limited 1628
to cases where r' remains constant. We construct reaches \tilde{q}_k from the sketch in lines 1629
20–22 and keep track of the r' used for construction in line 23. If r' remains constant, 1630
we can use previously computed reaches \tilde{q}_k to compute the new set of reaches \tilde{q}_k . Since 1631
the ending index of reaches depends on $\min(\lfloor kl \rfloor + r', L^Q)$, we need to check whether we 1632
have a new ending index when L^Q increases. If the ending index has been changed, we 1633
need to add the new data points $q'_{e_{\text{new}}}$ to the sorted sequence \tilde{q}_k , as in line 15. 1634

L_{max}^C depends of L^Q . If L_{max}^C increase because L^Q increase, we construct the new reach 1635
 \tilde{q}_k and sort it in lines 16–18. 1636

1637 4.4.3 Guided Distance

1638 For faster computation, one would want to use a distance measure with linear complexity,
1639 such as ED, as the base measure for PSD. While PSED is effective for identifying phase-
1640 scaling changes, certain applications require capturing complex properties within those
1641 segments that ED cannot handle. There are two ways to address it. One approach is
1642 to use an alternative distance measure that captures these complex properties as the
1643 base distance for computing the PSD, such as DTW. But PSDTW is slower than PSED.
1644 The other approach is to use the segmentation result returned by PSED. To address
1645 this, we propose a two-stage framework in which PSED-derived segmentation guides the
1646 application of advanced distance metrics M . Let $\mathcal{P}^* = \{(i_1, j_1), (i_2, j_2), \dots, (i_{P+1}, j_{P+1})\}$
1647 be the set of optimal cut points on Q and C obtained by minimizing the PSED. We
1648 utilize \mathcal{P}^* to partition both series into P aligned pairs of segments. The final distance,
1649 denoted as M^{PSED} , is calculated by summing the distances of these pairs using a target
1650 metric M :

$$M^{\text{PSED}}(Q, C) = \sum_{p=1}^P M(Q_p^L, C_p^L) \quad (4.16)$$

1651 , where $Q_p = Q(i_p + 1 : i_{p+1})$ and $C_p = C(j_p + 1 : j_{p+1})$.

1652 4.5 Experiments

1653 We evaluate the performance of our proposed distance measure framework, PSD, via its
1654 two instantiations: PSED and PSDTW. Specifically, we focus on a query retrieval task
1655 in which the query exhibits piecewise-scaled distortion, that is, distinct phases of the
1656 query exhibit different expression rates relative to the target in the candidate dataset.
1657 Our objective is to verify whether PSD achieves invariance to these multiple scaling
1658 distortions, thereby allowing it to correctly retrieve the most similar candidate. We detail
1659 the experimental setup in Section 4.5.1 and present the results in Section 4.5.2. The code
1660 and data are available at <https://github.com/colemanyu/k-scaling-factor-dtw>.

1661 4.5.1 Experimental Setup

1662 We utilize the GunPoint dataset, originally released in 2003 [68], as the running example.
1663 Widely regarded as the “iris” dataset of the time series community [69], it has appeared
1664 in over one thousand publications. Beyond its ubiquity, the dataset addresses a critical
1665 distinction: misinterpreting the act of aiming a gun as merely pointing a finger could
1666 have life-threatening consequences.

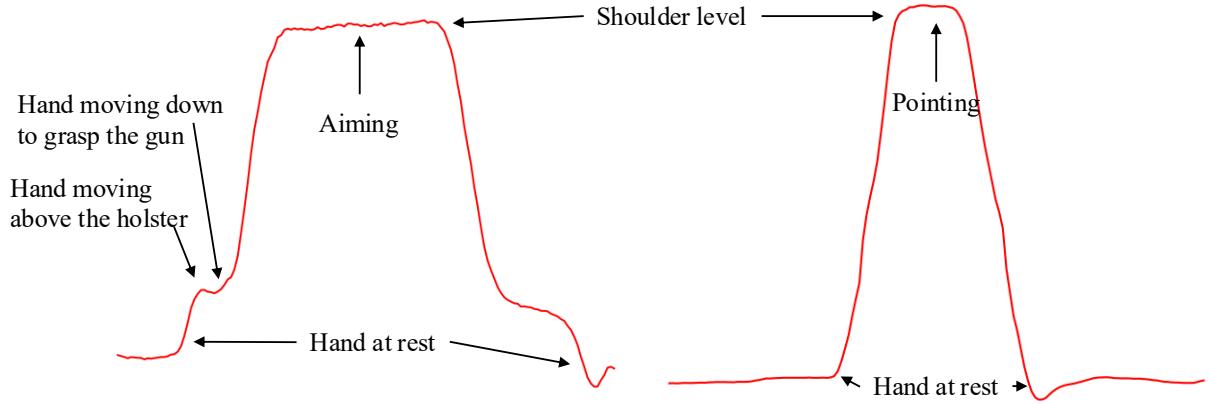


Figure 4.8: Visualization of the GunPoint dataset. Left (Right): A time series of the “Gun” (“Point”) scenario. Critical periods, such as “Aiming”, are annotated.

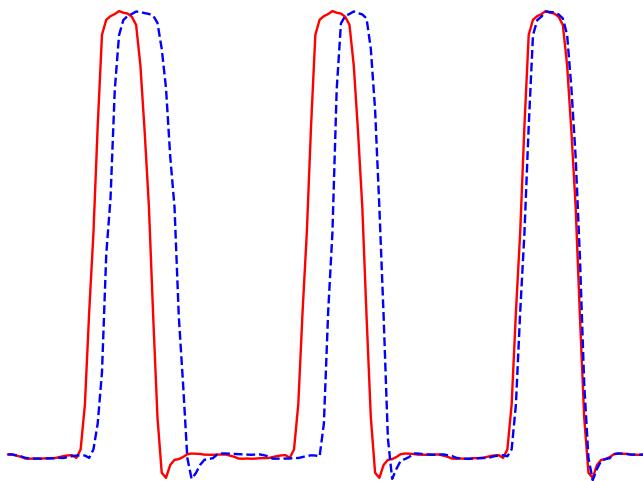


Figure 4.9: The red solid time series shows an instant in the target set. The blue dashed time series shows an instant in the query set.

The dataset contains two classes: “Gun” and Point”. Actors aim at an eye-level target 1667 using either a replica gun or their index finger, as illustrated in Figure 4.8. The resulting 1668 time series represent the X-axis centroid of the actor’s right hand. Each action lasts 1669 for 5 seconds, with the pointing/aiming phase occurring for approximately one second. 1670 Recorded at 30 fps, each sample consists of 150 data points. The dataset comprises 50 1671 training and 150 test series, all of length 150. 1672

A key limitation of the original dataset is the assumption that every action lasts 1673 exactly 5 seconds. In reality, actors perform actions at varying speeds. If an actor is 1674 asked to perform the action three times continuously in a row, we are likely to observe a 1675 time series containing three phases, each with a unique rate. The first phase should take 1676 longer than subsequent phases because it is the first time the action is performed. This 1677 phenomenon is depicted in Figure 4.9, where the red curve represents an ideal case that 1678 consists of three identical phases (i.e., identical rate), while the blue curve represents a 1679

1680 realistic scenario with varying rates. The first action is slower than the second action.
 1681 Consequently, retrieving such patterns requires assigning different scaling factors to each
 1682 phase to accommodate the phase-specific rates.

1683 We now describe the procedure for generating the target and query sets for the re-
 1684 trieval task. To construct the target set, we concatenate P repetitions of each time series
 1685 instance from the source dataset. To ensure a fair comparison, we constrain the resulting
 1686 time series to match the original length n . This is achieved by rescaling each phase to a
 1687 length of n/P prior to concatenation. Note that we must handle remainders to ensure
 1688 that the final time series length is exactly n . An example of such a target (where $P = 3$)
 1689 is depicted by the red solid line in Figure 4.9.

1690 To construct the query set, we first determine the specific lengths for the P segments.
 1691 Starting with an expected mean length of n/P , we define the minimum segment length
 1692 as $L_{\min} = (n/P)\sqrt{l}$ and the maximum as $L_{\max} = (n/P)\sqrt{l}$. This formulation ensures
 1693 that the ratio of the lengths of any two segments is bounded by l . We then generate P
 1694 random integers within $[L_{\min}, L_{\max}]$ subject to the constraint that their sum is exactly n .
 1695 Finally, we construct the query by rescaling P copies of the source time series to these
 1696 generated random lengths and concatenating them. The resulting time series maintains
 1697 the original length n . An example of such a query (where $P = 3$) is depicted by the blue
 1698 dashed line in Figure 4.9.

1699 Both the query and target sets are derived from the same source dataset. Conse-
 1700 quently, the ground truth target for a given query is defined as the instance in the target
 1701 set that is generated from the same underlying source time series.

Table 4.1: Details of the ten datasets for the experiments

Name	Type	Train/Test?	Size	Class	Length
SonyAIBORobotSurface1	Sensor	Test	601	2	70
ECG200	ECG	Train	100	2	96
MedicalImages	Image	Train	381	10	99
CBF	Simulated	Train	30	3	128
SwedishLeaf	HAR	Train	500	15	128
Plane	Sensor	Train	105	7	144
PowerCons	Device	Train	180	2	144
GunPoint	HAR	Train	50	2	150
Adiac	Image	Train	390	37	176
Epilepsy	HAR	Train	137	4	207

1702 Table 4.1 details the additional datasets used in this study. The column labeled
 1703 “Train/Test?” specifies which split was employed as the source dataset.

1704 In our experiments, we set the warping window parameter $r = 0.1$, as suggested in
 1705 the literature. The algorithms were implemented in Python. We utilized the `aeon` [70]

library to obtain the baseline distance measures. All experiments were conducted on a 1706
workstation equipped with an Intel Xeon Gold 6326 CPU and 256GB of RAM. 1707

4.5.2 Experimental Results

1708

We employ Top- k Accuracy (often referred to as Precision@ k [71]) as the primary evaluation metric. For a single query Q , this metric indicates whether the correct match is successfully retrieved within the top k candidates: 1709
1710
1711

$$P@k(Q) = \begin{cases} 1 & \text{if the ground truth is in the top-}k \text{ results} \\ 0 & \text{otherwise} \end{cases} \quad (4.17)$$

To determine the top- k results, we compute the distance between Q and every time series in the target set, generating a distance profile of length equal to the dataset size. 1712
1713
1714 The top- k results correspond to the k instances with the smallest distances in this profile. Finally, we evaluate the overall performance by computing the mean Top- k Accuracy across the entire query set \mathcal{D} : 1715
1716

$$\overline{P@k} = \frac{\sum_{Q \in \mathcal{D}} P@k(Q)}{|\mathcal{D}|} \quad (4.18)$$

We choose $k \in \{1, 3\}$. $k = 1$ refers to the exact retrieval. $k = 3$ give some tolerance for the retrieval. 1717
1718

Table 4.2: The accuracy comparison for eight distance measures of the GunPoint dataset

P	l	ED		DTW [2]		ADTW [10]		DDTW [4]		shapeDTW [6]		WDDTW [5]		WDTW [5]		PSED		PSDTW	
		$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$	$P@1$	$P@3$
2	1.25	0.30	0.56	0.82	1.00	0.54	0.84	0.84	0.92	<u>0.96</u>	1.00	0.88	0.96	0.82	0.98	0.98	1.00	0.86	1.00
	1.50	0.14	0.36	0.88	1.00	0.38	0.64	0.78	0.94	1.00	1.00	0.80	0.92	0.88	0.96	1.00	1.00	<u>0.96</u>	1.00
	1.75	0.16	0.34	0.82	1.00	0.38	0.66	0.64	0.80	0.90	1.00	0.66	0.80	0.82	0.96	1.00	1.00	<u>0.94</u>	1.00
	2.00	0.12	0.24	0.84	0.98	0.34	0.66	0.72	0.88	<u>0.96</u>	1.00	0.68	0.86	0.82	0.98	1.00	1.00	<u>0.96</u>	1.00
	1.25	0.30	0.50	0.84	0.92	0.70	0.88	0.80	0.92	0.96	0.98	0.84	0.94	0.86	0.98	<u>0.94</u>	1.00	0.88	0.96
3	1.50	0.10	0.28	0.84	0.96	0.60	0.78	0.66	0.86	<u>0.90</u>	1.00	0.72	0.86	0.84	0.98	0.96	1.00	0.86	0.96
	1.75	0.10	0.28	<u>0.88</u>	1.00	0.42	0.78	0.72	0.88	0.76	0.94	0.74	0.88	<u>0.88</u>	1.00	1.00	1.00	<u>0.88</u>	0.98
	2.00	0.02	0.12	<u>0.86</u>	0.94	0.38	0.52	0.60	0.78	0.70	0.94	0.60	0.80	0.82	0.92	0.98	1.00	<u>0.86</u>	0.98
	1.25	0.34	0.50	0.70	0.86	0.68	0.88	0.60	0.78	0.70	0.90	0.64	0.80	0.70	0.86	0.86	0.94	<u>0.72</u>	0.88
4	1.50	0.22	0.38	0.64	0.80	0.60	0.92	0.52	0.64	0.68	0.90	0.52	0.66	0.64	0.82	0.86	0.98	<u>0.72</u>	0.82
	1.75	0.16	0.30	0.72	0.92	0.56	0.80	0.60	0.78	0.56	0.80	0.56	0.80	0.70	0.88	0.92	1.00	<u>0.80</u>	0.94
	2.00	0.06	0.12	0.58	0.78	0.50	0.72	0.46	0.64	0.50	0.82	0.50	0.64	0.56	0.82	0.88	0.98	<u>0.64</u>	0.88

Results of GunPoint dataset: We utilize the GunPoint dataset to evaluate how the parameters P and l affect the ability of PSD to achieve invariance under piecewise scaling. 1719
1720 The results are presented in Table 4.2, where the best performance of $\overline{P@1}$ is highlighted in bold, and the second-best is underlined. We benchmark our method against several state-of-the-art distance measures, including ADTW [10], DDTW [4], shapeDTW [6], WDDTW [5], and WDTW [5]. PSED achieves the highest accuracy, followed closely by PSDTW. We attribute PSED’s superior performance over PSDTW to the specific nature 1721
1722
1723
1724
1725

of the distortions in the query set, that is, the “pure” piecewise scaling distortions. Since
 the query set exhibits only piecewise scaling distortions, the corresponding segments of the
 query and target time series differ solely in length (scale). Consequently, the additional
 flexibility provided by DTW in PSDTW is unnecessary and may inadvertently increase
 the similarity of incorrect matches, thereby reducing discriminative power relative to the
 stricter PSED measure. However, PSDTW remains theoretically essential for handling
 local distortions within the phase. PSDTW applies DTW within the scaled segment,
 enabling robust alignment across local nonlinearities.

Table 4.3: The accuracy comparison for six PSED-guided distance measures of the Gun-Point dataset

P	l	DTW ^{PSED}		ADTW ^{PSED}		DDTW ^{PSED}		shapeDTW ^{PSED}		WDDTW ^{PSED}		WDTW ^{PSED}	
		$\overline{P@1}$	$\overline{P@3}$	$\overline{P@1}$	$\overline{P@3}$	$\overline{P@1}$	$\overline{P@3}$	$\overline{P@1}$	$\overline{P@3}$	$\overline{P@1}$	$\overline{P@3}$	$\overline{P@1}$	$\overline{P@3}$
2	1.25	0.86	1.00	0.98	1.00	0.72	0.92	0.98	1.00	0.74	0.96	0.86	1.00
	1.50	0.88	1.00	1.00	1.00	0.60	0.90	0.98	1.00	0.64	0.90	0.92	1.00
	1.75	0.94	1.00	1.00	1.00	0.84	0.94	1.00	1.00	0.86	0.96	0.94	1.00
	2.00	0.98	1.00	1.00	1.00	0.60	0.78	0.98	1.00	0.66	0.80	0.98	1.00
3	1.25	0.82	0.96	0.94	1.00	0.74	0.90	0.92	0.98	0.76	0.90	0.82	0.96
	1.50	0.86	1.00	0.96	1.00	0.72	0.90	0.94	1.00	0.80	0.90	0.88	1.00
	1.75	0.88	1.00	1.00	1.00	0.68	0.90	1.00	1.00	0.70	0.94	0.90	1.00
	2.00	0.86	0.96	0.98	1.00	0.54	0.86	0.96	1.00	0.62	0.86	0.88	0.96
4	1.25	0.74	0.88	0.86	0.94	0.78	0.80	0.80	0.92	0.78	0.82	0.74	0.88
	1.50	0.72	0.84	0.86	0.98	0.68	0.84	0.84	0.96	0.66	0.84	0.72	0.86
	1.75	0.74	0.96	0.92	1.00	0.54	0.90	0.90	1.00	0.56	0.88	0.74	0.96
	2.00	0.66	0.90	0.88	0.98	0.58	0.82	0.88	0.98	0.62	0.80	0.66	0.90

We further investigate whether the segmentation results returned by PSED can serve
 as a guide to enhance other distance measures. As shown in Table 4.3, this approach
 generally improves accuracy. The exceptions are highlighted in bold, indicating cases
 where the segmentation led to worse performance. Notably, in most cases, only DDTW
 and WDDTW failed to benefit from PSED-guided segmentation. We argue that the
 performance degradation of DDTW and WDDTW stems from their derivative-based
 nature. They rely on matching slope or shape features. Consequently, they are highly
 sensitive to segmentation boundaries. A non-perfect cut that falls within a shape feature
 segmentizes it, and these features are then destroyed. When the algorithm subsequently
 attempts to map these features, it fails to find correct correspondences, resulting in high
 distances.

Figure 4.10 illustrates the runtime performance across varying parameters P and l .
 We evaluate two variants of PSD, PSED and PSDTW, each implemented with three
 levels of

- 1748 1. `vanilla` (i.e., Basic implementation)
- 1749 2. `parallel_bsf` (i.e., With parallelization with Best-So-Far early abandoning)

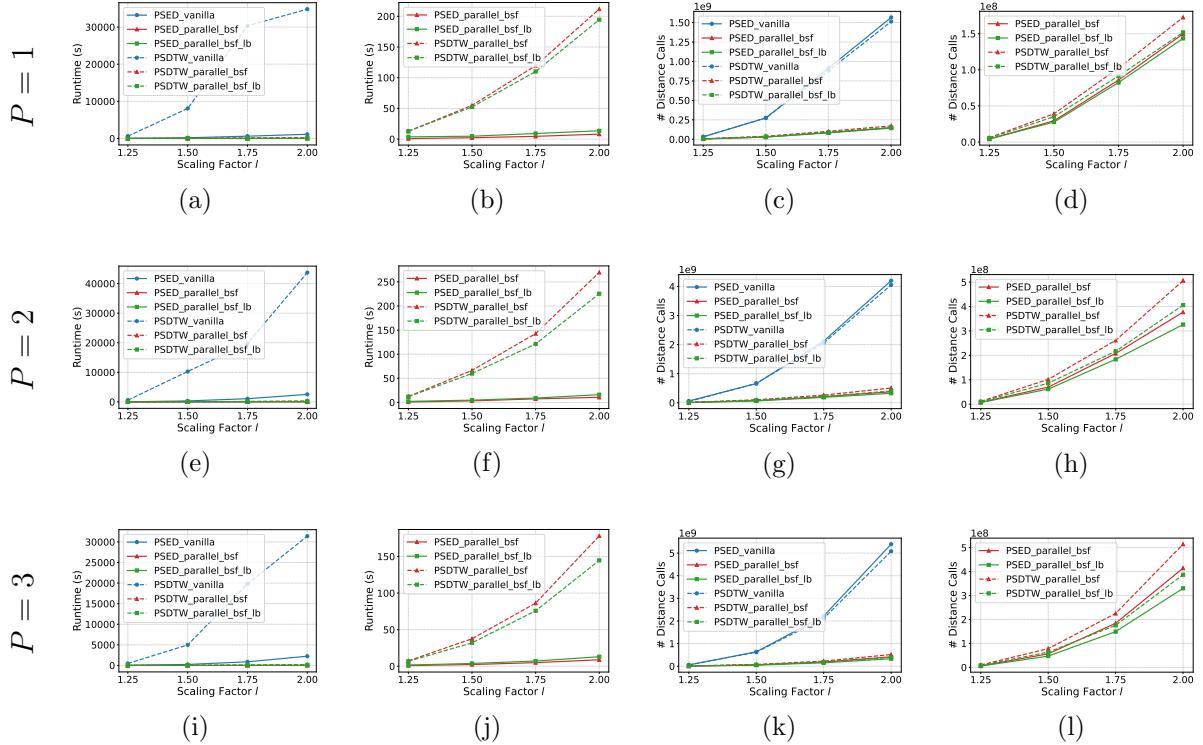


Figure 4.10: Runtime and number of distance calls comparison of GunPoint dataset. (a)-(d) $P = 1$, (e)-(h) $P = 2$, (i)-(l) $P = 3$.

3. parallel_bsf_lb (i.e., Incorporating lower bound pruning)

1750

This yields a total of six methods. The figure is organized into four columns plotted against the scaling factor l . The rows refer to the number of pieces P . The first and third columns display the running time and the number of distance calculations, respectively, for all six methods. To better visualize the performance differences among the efficient implementations, the second and fourth columns omit the `vanilla` baselines and focus exclusively on the four optimized variants. 1751
1752
1753
1754
1755
1756

The results indicate that `vanilla_PSDTW` is orders of magnitude slower than the other approaches, whereas the optimized methods operate within a similar performance range. 1757
1758 As anticipated, the computation time for all methods increases with the scaling factor 1759 l . The fourth column confirms that the lower bounding strategy effectively reduces the 1760 number of distance calculations (pruning power). However, the second column reveals 1761 a critical trade-off in actual runtime. While the lower bound successfully accelerates 1762 PSDTW, it actually slows down PSED. This suggests that for the computationally lighter 1763 ED, the overhead of calculating the lower bound outweighs the time saved by pruning. 1764 Overall, the results demonstrate that PSED is significantly faster than PSDTW. 1765

Results of the ten datasets:

For the remaining nine datasets, we fix the parameters at $P = 3$ and $l = 1.5$. We 1767 have the following findings from the previous experiment: 1768

Table 4.4: The accuracy comparison for eight distance measures of the ten datasets

Dataset	ED		DTW		ADTW		DDTW		shapeDTW		WDDTW		WDTW		PSED	
	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3
SonyAIBORobotSurface1	0.04	0.08	0.51	0.63	<u>0.60</u>	0.74	0.45	0.57	0.20	0.29	0.45	0.58	0.51	0.64	0.67	0.77
ECG200	0.12	0.20	0.70	0.81	<u>0.78</u>	0.83	0.60	0.74	0.59	0.76	0.59	0.73	0.70	0.80	0.81	0.88
MedicalImages	0.08	0.19	0.63	0.78	0.69	0.85	0.52	0.68	0.57	0.75	0.51	0.68	0.62	0.78	0.76	0.88
CBF	0.53	0.67	0.83	1.00	<u>0.90</u>	1.00	0.73	0.87	0.73	0.90	0.77	0.90	<u>0.90</u>	1.00	1.00	1.00
SwedishLeaf	0.07	0.12	0.82	0.92	<u>0.84</u>	0.93	0.70	0.83	0.78	0.91	0.69	0.83	0.83	0.92	0.97	0.99
Plane	0.09	0.14	0.50	0.74	<u>0.59</u>	0.78	0.38	0.64	0.53	0.79	0.37	0.61	0.49	0.75	0.75	0.93
PowerCons	0.26	0.43	0.77	0.98	0.80	1.00	0.77	0.98	0.77	0.99	0.77	0.99	0.77	0.99	0.80	1.00
GunPoint	0.10	0.28	<u>0.84</u>	0.96	0.60	0.78	0.66	0.86	0.90	1.00	0.72	0.86	<u>0.84</u>	0.98	0.96	1.00
Adiac	0.01	0.02	<u>0.37</u>	0.50	0.12	0.21	0.32	0.42	0.27	0.41	0.31	0.41	0.36	0.50	0.57	0.71
Epilepsy	0.18	0.26	0.74	0.88	<u>0.80</u>	0.91	0.65	0.82	0.38	0.47	0.58	0.79	0.70	0.83	0.83	0.91

Table 4.5: The accuracy comparison for six PSED-guided distance measures of the ten datasets

Dataset	DTW ^{PSED}		ADTW ^{PSED}		DDTW ^{PSED}		shapeDTW ^{PSED}		WDDTW ^{PSED}		WDTW ^{PSED}	
	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3
SonyAIBORobotSurface1	0.53	0.64	0.64	0.76	0.37	0.53	0.65	0.76	0.38	0.53	0.53	0.65
ECG200	0.73	0.84	0.80	0.88	0.54	0.68	0.81	0.88	0.53	0.69	0.73	0.84
MedicalImages	0.66	0.78	0.76	0.88	0.51	0.69	0.74	0.88	0.51	0.70	0.66	0.79
CBF	0.90	1.00	0.93	1.00	0.73	0.93	0.93	0.97	0.70	0.93	0.90	1.00
SwedishLeaf	0.82	0.92	0.97	0.99	0.72	0.87	0.97	0.99	0.73	0.88	0.82	0.92
Plane	0.60	0.80	0.75	0.94	0.50	0.67	0.74	0.90	0.50	0.69	0.59	0.80
GunPoint	0.86	1.00	0.96	1.00	0.72	0.90	0.94	1.00	0.80	0.90	0.88	1.00
PowerCons	0.78	0.99	0.79	1.00	0.79	1.00	0.79	1.00	0.79	1.00	0.78	1.00
Adiac	0.33	0.45	0.57	0.71	0.21	0.34	0.53	0.70	0.20	0.33	0.34	0.45
Epilepsy	0.77	0.88	0.78	0.88	0.66	0.79	0.74	0.89	0.66	0.82	0.80	0.89

1769 1. From Table 4.3, PSED outperformed PSDTW in handling piecewise scaling distortions.

1770

1771 2. From Figure 4.10, the lower bound offered no efficiency gain for PSED.

1772 Hence, we select PSED_parallel_bsfs as the representative method for this evaluation.

1773 The accuracy results are presented in Table 4.4, while the results for the PSED-guided
1774 distance measures are detailed in Table 4.5. Finally, the runtime efficiency for all datasets
1775 is summarized in Table 4.6. It shows a significant speedup, ranging from 10.10X to
1776 191.46X.

1777 4.6 Concluding Remarks

1778 In this paper, we proposed a novel distance measure framework, Piecewise Scaling Dis-
1779 tance (PSD), which relaxes the strict assumption of a single uniform scaling factor across
1780 the entire time series. We presented an exact dynamic programming (DP) algorithm to
1781 solve this problem. To enhance efficiency and prevent pathological segment alignments,
1782 we introduced a constraint version, which limits the search space of segment lengths
1783 based on given scaling factors. To enhance computational efficiency, we integrated two

Table 4.6: The number of distance calls and runtime on PSED_vanilla and PSED_parallel_bsf of the ten datasets

Name	Size	Length	PSED_vanilla	PSED_parallel_bsf	% distance calls pruned	Speed Up
			Time (s)	Time (s)		
SonyAIBORobotSurface1	601	70	697	69	90.40%	10.10X
ECG200	100	96	91	3	91.08%	30.33X
MedicalImages	381	99	2082	40	96.03%	52.05X
CBF	30	128	43	2	75.31%	21.50X
SwedishLeaf	500	128	16601	107	96.24%	155.15X
Plane	105	144	621	6	96.05%	103.50X
PowerCons	180	144	3629	49	82.32%	74.06X
GunPoint	50	150	159	3	89.42%	53.00X
Adiac	390	175	23550	123	96.71%	191.46X
Epilepsy	137	206	13131	336	39.88%	39.08X

optimization techniques for the general PSD framework. In addition, we propose incorporating a lower-bounding strategy to accelerate PSDTW. Our experimental results demonstrate the necessity and effectiveness of PSD when identifying matches between a query Q and a candidate C under piecewise scaling distortions.

We outline several directions for future research. First, we aim to develop a lower bound specifically optimized for PSED that can improve actual runtime. Second, while the current PSDTW algorithm requires the number of segments P to be specified as a hyperparameter, it is preferable for the algorithm to determine this value adaptively. A simple heuristic is to test a range of P values and select the configuration that yields the minimum distance. Finally, we plan to investigate efficiency improvements for PSDTW. Currently, PSDTW computes dynamic time warping on two growing subsequences after scaling. While techniques for Incremental DTW [72] allow for the reuse of the accumulating cost matrix D to avoid redundant calculations, applying this to PSDTW is non-trivial. The interpolation performed before DTW fundamentally alters the sequence structure, preventing the straightforward extension of D (e.g., by appending rows or columns) to reuse the computed D that is possible in standard DTW.

Chapter 5

1800

Leveraging Nearest Neighbors for Time Series Forecasting with Matrix Profile

1801

1802

1803

5.1 Introduction

1804

Humans have made predictions since ancient times. In ancient societies, accurate predictions were important to the success of subsistence activities such as hunting, planting, and harvesting. There was a need to predict weather dynamics, such as rainfall and temperature. For example, it is crucial to plant during the period with sufficient rainfall and appropriate temperatures. Our ancestors used divination tools such as turtle shells, wooden blocks (moon blocks), or bones to make predictions. Without doubt, the accuracy was not guaranteed. Even in modern societies, prediction is still essential. Predicting traffic-jam patterns only a few hours ahead can save time by enabling the selection of an alternative route [73]. A wealth could be created by forecasting stock market trends. Predicting the future, also known as time series forecasting, is crucial.

1814

With advances in hardware technologies, we collect enormous amounts of data from diverse sources, such as smart sensors and social media platforms, continuously in the form of time series data. A time series is an ordered sequence of measures, represented in real-valued numbers, at discrete equal-interval timestamps [74]. The vast data collections have created the era of “Big Data”, which provides a wealth of datasets for developing and deploying reliable, robust, data-driven forecasting techniques to discover patterns and extract valuable information [75]. Applications can be found in the financial sector, such as predicting business cycles and stock market movements [76, 73, 77, 78] and the medical field, such as the status of critical patients according to their vital signs [79, 80] and the propagation of diseases such as influenza [78, 81] and COVID-19 [79, 82].

1824

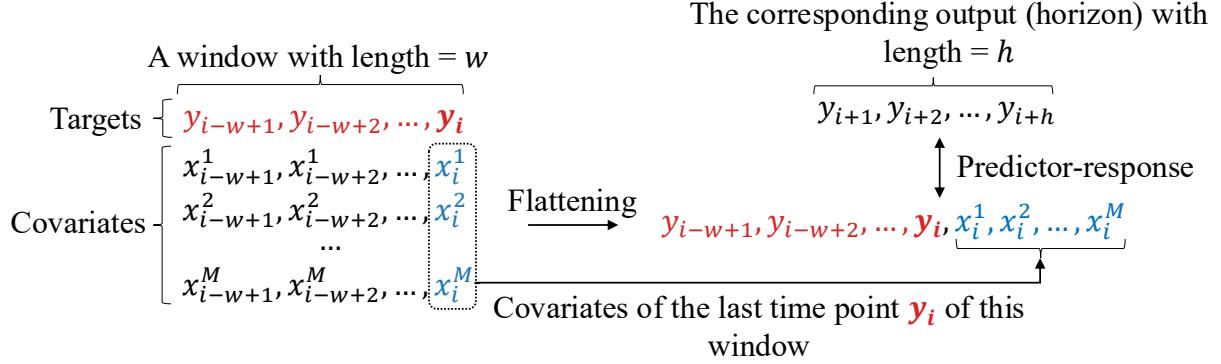


Figure 5.1: Flattening a 2D window of size $(1+M) \times w$ to a 1D vector of length $w+M$. The resulting vector serves as the *predictor* for the regressor. Its expected *response* is the vector of length h . The regressor learns from this predictor-response pair.

1825 A recent study [1] shows that, in time series classification, a non-parametric, instance-
 1826 based method, namely nearest-neighbor classifiers (1-NN) and its generalized form k -NN,
 1827 with appropriate distance measures, such as Dynamic Time Warping (DTW), despite
 1828 their simplicity, perform well and are therefore commonly used as benchmarks. In de-
 1829 tail, when a new instance is to be classified, k -NN finds its k nearest neighbors in the
 1830 training set and returns their majority label among them. k -NN is considered a lazy
 1831 learner because the training steps involve only memorizing all the instances verbatim; no
 1832 higher-level concepts have been learned. In addition, in time series forecasting, a recent
 1833 study demonstrates that a well-known machine learning baseline, Gradient Boosting Re-
 1834 gression Tree (GBRT), such as XGBoost, equipped with an appropriate data engineering
 1835 of the data, can achieve competitive or even superior performance than the deep learning
 1836 method. In detail, they transform the time series forecasting task into a window-based
 1837 regression problem, as shown in Figure 5.1. For each training window of length w with
 1838 the last time point \mathbf{y}_i , and its lagged values $\mathbf{y}_{i-1}, \mathbf{y}_{i-2}, \dots, \mathbf{y}_{i-w+1}$ are concatenated with
 1839 covariates $x_i^1, x_i^2, \dots, x_i^M$ to form a *predictor* for a multi-output GBRT. This transfor-
 1840 mation is called flattening. The corresponding response is the following h points of \mathbf{y}_i . It
 1841 provides a simple, more efficient yet accurate method for time series forecasting.

1842 Moreover, k -NN has also shown to be a promising method for time series forecast-
 1843 ing [83]. The k -NN uses the lagged values of the last time point to form a query Q .
 1844 It identifies the k previous similar subsequences to Q and uses their immediate subse-
 1845 quences to predict the immediate subsequence, which is the forecasting window, of Q .
 1846 The intuition is that history repeats itself. The previous (historical) subsequences that
 1847 are similar to Q can provide a hint about the future of Q . They are similar, and so are
 1848 their immediate subsequences. Figure 5.2 depicts this idea. Observe that the immediate
 1849 subsequence of the right gray box is similar to that of the left gray box. The left gray
 1850 box is the nearest neighbor of the right gray in the “past”.

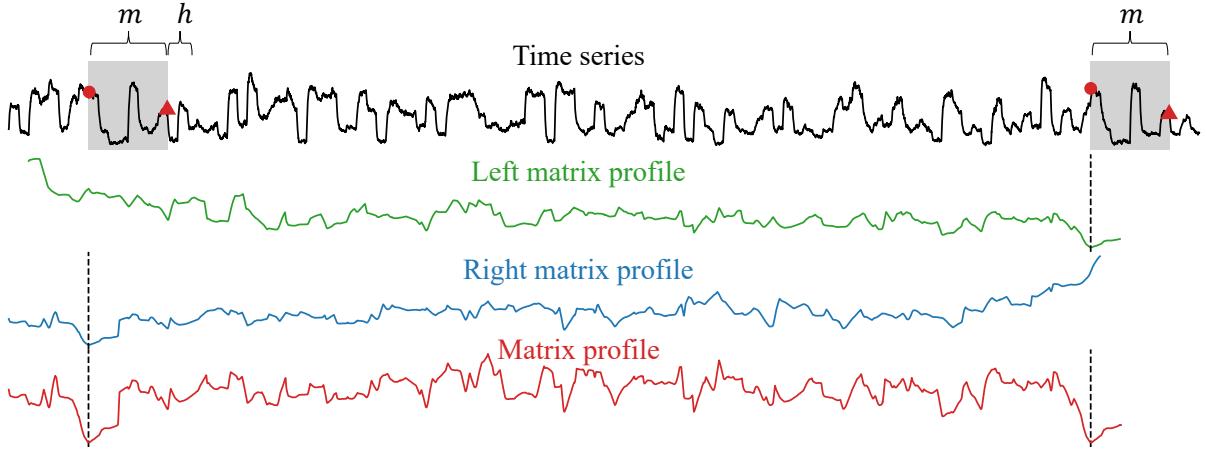


Figure 5.2: The **left** (**right**) matrix profile and the **matrix profile** of a time series. The **matrix profile** shows the distances between each m -subsequence and its nearest neighbor, where m is a user-given value. The **left** (**right**) matrix profile shows the same information but is limited to its left (right) nearest neighbor. For a particular m -subsequence shown by the right gray box, its nearest neighbor is the left gray box, as indicated by the dashed line in the **left matrix profile** and the **matrix profile**. Similarly, the nearest neighbor of the left gray box is the right gray box, as indicated by the dashed line in the **right matrix profile** and the **matrix profile**. The first (last) point in each box is denoted by a red circle (triangle). h denotes the length of the immediate subsequence of the nearest neighbor. Intuitively, this subsequence should be similar to the immediate subsequence (i.e., future) of the right gray box. To note, the **left** (**right**) matrix profile starts (ends) at a later (earlier) index because the corresponding nearest neighbor with length m does not exist.

Algorithm 6 The brute force approach to compute Matrix Profile

```

1: for  $i \leftarrow 1$  to  $n - m + 1$  do
2:   for  $j \leftarrow 1$  to  $n - m + 1$  do
3:     Compute the z-normalized Euclidean distance between  $t_{i,m}$  and  $t_{j,m}$ .

```

Based on these findings, this study proposes a method to improve the performance of 1851 existing forecasters by leveraging information from the nearest neighbors of each subsequence in the target variable. For each time point y_i of the target variable Y , a window 1852 of length w is constructed with y_i as the last point, then we retrieve the window's historical 1853 nearest neighbors and use their information to create new covariates for the window. 1854 The information includes the similarities between the window and its nearest neighbors, 1855 as well as the immediate subsequences of them. The similarity can be interpreted as a 1856 measure of confidence or weight in using the information from the corresponding nearest 1857 neighbor. The intuition is that, if the similarity of the window and a neighbor is high, 1858 then the future (i.e., immediate subsequence) of the neighbor should also be similar to 1859 the future of the window. The fundamental difference between this study and previous 1860 approaches [84, 83, 85] is that they directly use the subsequent points for prediction, 1861 whereas we use the nearest neighbor information for each subsequence as covariates, 1862 1863

1864 which are used as primitives for other forecasters. We explain this subtle difference by
1865 Figure 5.2. The previous approaches simply use the information of the nearest neighbors
1866 of the last look-back window, which consisted of the last time point and its lagged values
1867 (i.e., the right gray box), for prediction. In contrast, we use the information of the nearest
1868 neighbors of **all** of the windows.

1869 We use the Matrix Profile [86, 87] to annotate the nearest neighbor for each m -
1870 subsequence of a time series T of length n . The distance is the z-normalized Euclidean
1871 distance. It may seem computationally expensive to perform this annotation at first
1872 glance. Algorithm 6 shows the brute force approach to compute the matrix profile.
1873 The two for-loops and the computation of z-normalized Euclidean distance, which takes
1874 $\mathcal{O}(m)$, indicate that the computational complexity is $\mathcal{O}(n^2m)$. The space complexity is
1875 $\mathcal{O}(n^2)$ because of the pairwise distance of each subsequence with the other subsequence.
1876 However, the matrix profile can be computed in $\mathcal{O}(n^2)$ using an exact method, namely
1877 STOMP [86] or its community-open-sourced version, STUMP [88]. Besides, the running
1878 time can be further sped up by parallelization for a single machine with multiple compu-
1879 tation units, such as CPUs or GPUs. The tool also allows us to compute the left matrix
1880 profile to find the left nearest neighbor of each window. To note, the matrix profile an-
1881notates a time series with information about the nearest neighbor of each subsequence,
1882 including the similarity with its nearest neighbor and its location, as shown in Figure 5.2.
1883

In this study, we make the following contributions:

- 1884 • We are the first to propose leveraging the matrix profile to create meaningful co-
1885 variates that improve forecaster performance.

1886 The rest of this paper is organized as follows. Section 5.2 presents the related-work.
1887 In Section 5.3, we introduce the necessary background knowledge, then introduce our
1888 method. Section 5.4 contains an empirical evaluation. Finally, we conclude this paper
1889 and provide future work in Section 5.5.

1890 5.2 Related Work

1891 Many methods have been developed for time series forecasting. Traditional methods in-
1892 clude rolling averages (RA), vector auto-regression (VAR) [73, 89], and auto-regressive
1893 integrated moving averages (ARIMA) [90, 91, 89]. Because of their rigorous statisti-
1894 cal properties, they have long been the standard. The shortcomings of ARIMA and
1895 its variants include their high computational cost [73]. In contrast, VAR is arguably
1896 the most widely used method, particularly in multivariate time series analysis, owing to
1897 its simplicity. However, most of these traditional approaches have certain limitations.
1898 They perform well when the data meet specific statistical assumptions, such as station-
1899 arity [92], which means that the mean and variance of the time series remain constant

over time. It motivates the community to develop machine learning methods, particularly deep learning methods for time series forecasting. Many deep learning models have been proposed, including RNN-based models, CNN-based models, GNN-based models, Transformer-based models, and compound models that incorporate different base models mentioned before [78]. The compound models are promising. For example, RNNs are well suited to capturing long-term dependencies, whereas CNNs are well suited to capturing short-term dependencies. A good way to improve performance is to compound them. For example, LSTnet [73] integrates CNN, RNN, and autoregressive [93] techniques to extract both short-term and long-term patterns. Using the occupancy rate of a freeway as an example [73] to explain these two patterns, the “short-term” patterns refer to the morning peaks against evening peaks, while the “long-term” patterns refer to the workday patterns against weekend patterns. Clearly, a good forecaster needs to capture and distinguish both kinds of patterns. Despite the superior performance deep learning methods have achieved, they tend to be overly complex, opaque, and incur high computational costs compared to traditional techniques.

5.3 Method

In this section, we first formulate the time series forecasting problem, followed by the evaluation method. We then explain how to use a Gradient Boosting Regression Tree (GBRT) for forecasting. Subsequently, we discuss how to leverage the nearest neighbors’ information of each subsequence to improve GBRT’s performance. Finally, we discuss how to compute those nearest neighbors using the Matrix Profile.

To begin, we define the data type of interest: time series.

Definition 10 (Time series). A *time series* $T = t_1, t_2, \dots, t_n$ is a sequence of real-valued numbers with length $= n$.

In Definition 10, T is a univariate time series where each entry is a scalar number. If each entry is a vector consisting of scalar numbers with size > 1 , T is a multivariate time series. A multivariate time series can be regarded as a sequence of vectors. It can also be represented as a vector of univariate time series, where each univariate time series is referred to as a channel. In a dataset with more than one time series, we use T_i to denote a time series in a dataset with N time series, where $1 \leq i \leq N$.

The local properties of T can be analyzed through its subsequences.

Definition 11 (Subsequence). A *subsequence* $T_{i,m} = t_i, t_{i+1}, \dots, t_{i+m-1} = t_{i:i+m-1}$ of a T is a sequence that consists of a continuous subset of the entries from T of length m starting from i .

1934 5.3.1 Problem Formulation

1935 Time series forecasting is the task of predicting h -future values $y_{t+1}, y_{t+2}, \dots, y_{t+h}$ of a
1936 target Y at the current time point t . In this study, there is only one target variable Y . h is
1937 the number of steps we want to predict in the future. The simplest case is one-step-ahead
1938 forecasting, where $h = 1$. The predicted value is denoted as \hat{y}_{t+1} where the actual value
1939 is y_{t+1} . It is preferable to predict multiple points in the future. It is called multi-horizon
1940 (multi-step) forecasting, where $h > 1$. The task of forecasting is encoded in Equation 5.1.

$$\hat{y}_{t+\tau} = f(y_{t-w+1:t}, x_{t-w+1:t}, u_{t-w+1:t+\tau}, \tau) \quad (5.1)$$

1941 where

- 1942 • $\hat{y}_{t+\tau}$ is a prediction of the target value at $t + \tau$, where $\tau \in \{1, 2, \dots, h\}$.
- 1943 • $y_{t-w+1:t}$ are the actual values consisting of the current value y_t and the lag values
1944 before it. y_{t-i} is called the lag of i or i -lag.
- 1945 • x_t are inputs that can only be known historically at time t . x_{t+1} is not known at t .
- 1946 • u_t are known inputs for all time. For example, the date information such as the
1947 day of the week or month [92]. Even at t , u_{t+i} where $1 \leq i \leq \infty$ are known.

1948 x_t and u_t are called covariates of y_t . The input of Equation 5.1 is a look-back window w .

1949 We explain the task of forecasting in terms of Equation 5.1. The forecasting process
1950 estimates the value of $y_{t+\tau}$, denoted by $\hat{y}_{t+\tau}$ with the aim to minimize the error function,
1951 typically represented as a function of $y_{t+\tau} - \hat{y}_{t+\tau}$ for each τ . It is obvious that date
1952 information is useful when the target variable depends on when the measurement is
1953 taken. For example, if the target variable is the electricity consumption rate, there is a
1954 clear pattern by month: consumption is higher during the winter and summer months,
1955 when air conditioners and heaters are used. x_t provides additional information about the
1956 state of y_t . For example, if the target variable is the body temperature, and x_t tells us
1957 the severity of the sore throat, we may guess the body temperature will raise tomorrow.

1958 5.3.2 Evaluation Method

1959 Given a dataset D of N time series T_i , where $1 \leq i \leq N$, we explain how to evaluate the
1960 performance of a forecaster on T_i . The error made by the forecaster on D is simply the
1961 summation of errors made by the forecaster on each T_i . We now focus on a single time
1962 series; hence, we drop the index i . T is divided into two subsequences, namely training
1963 subsequence T_{train} and test subsequence T_{test} .

1964 In our study, the forecaster is only allowed to train on T_{train} . Recall that we are
1965 predicting the h future values from the w values just before them. Hence, we use a

sliding window of length $w + h$ to generate the w -predictor- h -response pairs in T_{train} , enabling the forecaster to train on them. During the test (evaluation) phase, we do rolling forecasts. We predict based on the ground truth, not results generated from the model. It is used to prevent the accumulation of errors.

This concept is called “Teacher forcing” [94] because the teacher’s values are “force fed” into the forecaster when we “roll” the forecaster on the T_{test} [95]. The intuition is that, suppose each question (except the first one) in an exam depends on the answers to the previous questions, rather than simply grading every answer in the end, a teacher would grade (evaluate) the answer once it is given by the student, and provide the correct answer to the student so he can answer the next question based on the correct answer.

To evaluate the forecaster, we used the following three evaluation metrics defined as:

- Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (5.2)$$

- Weighted Absolute Percentage Error (WAPE)

$$\text{WAPE} = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{\sum_{i=1}^n |y_i|} \quad (5.3)$$

- Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (5.4)$$

where n is the length of the time series, y_i , \hat{y}_i is ground true value and predicted value, respectively. RMSE and MAE are widely used metrics. MAE can better reflect the actual error situation than RMSE [76]. WAPE was introduced by [96]. By rewriting Equation 5.3 to Equation 5.5, it is more obvious that it is a weighted absolute percentage error.

$$\text{WAPE} = \sum_{i=1}^n w_i \frac{|y_i - \hat{y}_i|}{|y_i|} \quad (5.5)$$

where the weights are given by

$$w_i = \frac{|y_i|}{\sum_{i=1}^n |y_i|} \quad (5.6)$$

For all of them, a lower value is better.

1987 **5.3.3 Gradient Boosting Regression Tree (GBRT)**

1988 Gradient boosting [97] is a boosting algorithm that ensembles a group of weak learners
1989 (usually decision trees) to make predictions. It sequentially adds learners to an ensemble,
1990 with each learner connecting its predecessor. It constructs weak learners in a way that
1991 each learner strategically corrects the predecessor’s mistakes by fitting the new learner
1992 to the residual errors made by the predecessor [98, 99]. It can be used in classification
1993 and regression. In this study, we focus on its use in regression. For the usage in regres-
1994 sion, the model is called “Gradient Boosting Regression Tree (GBRT)”. Some popular
1995 optimized implementations of gradient boosting are XGBoost [100], CatBoost [101], and
1996 LightGBM [102]. In this study, we use XGBoost.

1997 In order to apply GBRT into time series forecasting problem, we need to cast the
1998 input into an appropriate format to input into GBRT. The casting approach is similar to
1999 successful time-series forecasting models, which reconfigure the time series into windowed
2000 inputs [90]. Figure 5.1 presents the reconfiguration. For each entry of the target variable
2001 y_i , we retrieve its u_i , such as the day information from the calendar. Hence, for each y_i , it
2002 is associated with x_i and u_i . To simplify the notation, we absorb u_i into x_i , and it is called
2003 the covariates of y_i . The 2D window, as shown on the left-hand side in Figure 5.1, with
2004 size $w \times M$, where M is the total number of covariates, is flattened into a 1D array on the
2005 right-hand side with length $w + M$. To note, as suggested in the literature [90], only the
2006 covariates of the last time-point i are kept and appended to the final vector. By recon-
2007 figuration, we obtain the predictor-response pairs for training. In detail, the predictor is
2008 $y_{i-w+1}, y_{i-w+2}, \dots, \textcolor{red}{y_i}, \textcolor{blue}{x_i^1}, \textcolor{blue}{x_i^2}, \dots, \textcolor{blue}{x_i^M}$ where the red part refers to the current target value
2009 $\textcolor{red}{y_i}$ and its lag values, and the blue part refers to the covariates of $\textcolor{red}{y_i}$. The corresponding
2010 output is $y_{i+1}, y_{i+2}, \dots, y_{i+h}$, with length = h . We predict the h -horizon from the w -look
2011 back window (i.e. $w + M$ -flattened predictor). With this predictor-response formulation,
2012 the forecasting problem becomes a multi-output regression problem. Standard XGBoost
2013 cannot return a sequence of predicted values; it only returns a single number [90]. To
2014 note, a multi-output regression problem is simply a group of single-output regression
2015 problems. In other words, XGBoost internally simply treats the prediction of h -steps as
2016 h individual problems. Hence, the final output is produced by the h regressors rather
2017 than by a single model. One may argue that the h regressors operate individually and
2018 hence the temporal relationship in the output sequence is lost. However, as the individual
2019 regressors are trained on the same flattened input, the prediction would still preserve the
2020 temporal relationship [90].

2021 **5.3.4 Matrix Profile**

2022 We would like to find

Definition 12 (Distance profile). A *distance profile* $D_i = d_{i,1}, d_{i,2}, \dots, d_{i,n-m+1}$ of a T is a vector of the Euclidean distances between a given subsequence $T_{i,m}$ and each subsequences in T , where $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$, $1 \leq i, j \leq n - m + 1$. 2023
2024
2025

The distances are measured between z-normalized time series. 2026

Definition 13 (Matrix profile). A *matrix profile* $P = \min(D_1), \min(D_2), \dots, \min(D_{n-m+1})$ of T is a vector of Euclidean distances between every subsequence $T_{i,m}$ of T and its nearest neighbor in T . 2027
2028
2029

Definition 14 (Matrix profile index). A *matrix profile index* $I = I_1, I_2, \dots, I_{n-m+1}$ of T is a vector of integers, where $I_i = j$ if $d_{i,j} = \min D_i$. 2030
2031

Definition 15 (Left distance profile). A *left distance profile* $D_i^L = d_{i,1}, d_{i,2}, \dots, d_{i,i-\lceil m/4 \rceil - 1}$ of T is a vector of Euclidean distances between a given subsequence $T_{i,m}$ and each subsequence that appears before $T_{i,m}$. To note, $i - \lceil m/4 \rceil - 1$ is the index location of the last eligible subsequence before $T_{i,m}$ because of the exclusion zone. 2032
2033
2034
2035

Definition 16 (Left matrix profile). A *left matrix profile* $P^L = \min(D_1^L), \min(D_2^L), \dots, \min(D_{n-m+1}^L)$ of T is a vector of Euclidean distances between every subsequence $T_{i,m}$ of T and its nearest neighbor in T before it. 2036
2037
2038

Definition 17 (Left matrix profile index). A *left matrix profile index* $I^L = I_1^L, I_2^L, \dots, I_{n-m+1}^L$ of T is a vector of integers, where $I_i^L = j$ if $d_{i,j} = \min D_i^L$. 2039
2040

5.4 Experiments

2041

Table 5.1: Dataset Statistics. N is the number of time series in the dataset, while $|T| = n$ is the length of each time series. “rate” refers to the measuring rate. w is the size of the look-back window. h is the size of the forecasting window, also known as the forecasting horizon. T_{train} is the training subsequence of T . T_{test} is the test subsequence of T . To note, $|T_{\text{train}}| + |T_{\text{test}}| = n$.

Dataset	Data			Forecasting Task		
	N	n	rate	w, h	$ T_{\text{train}} $	$ T_{\text{test}} $
Electricity [103]	70	26,136	hourly	24	25,968	168
Traffic [103]	90	10,560	hourly	24	10,392	168
PeMSD7(M) [103]	228	12,672	/5 mins	9	11,232	1,440
Exchange-Rate [73]	8	7,536	daily	24	6,048	1,488

2042 **5.5 Conclusion and Future work**

2043 **5.5.1 Conclusion**

2044 **5.5.2 Future Work**

2045 **Leveraging nearest neighbors' location information:** It would be beneficial to re-
2046 trieve the nearest neighbors for each subsequence in a specific range with respect to it.
2047 Real-world applications often require the separation of information of short-term and
2048 long-term repeating patterns for making accurate predictions [73]. Notably, the matrix
2049 profile also provides the locations of the nearest neighbors from the matrix profile in-
2050 dex. Using this location (index) information, we can retrieve the nearest neighbors of
2051 each subsequence in a specified range with respect to it to find those "short-term" and
2052 "long-term" patterns.

2053 **Extend to multidimensional case:** This study focuses on univariate time series fore-
2054 casting, where there is a single target and no exogenous inputs. It only requires us
2055 to find one-dimensional nearest neighbors. When there are multiple targets or a sin-
2056 gle target with multiple exogenous inputs, we need to identify multidimensional nearest
2057 neighbors [104] and leverage their information for forecasting.

2058 **Top- k motifs:** We can use motifs instead of neighbors to receive more stable "future"
2059 information for each window. Recall that a time series motif is a repeated pattern that
2060 consists of at least two occurrences. A motif can be considered as a family of nearest
2061 neighbors A nearest neighbor is a historical occurrence that instantiates this motif. The
2062 motif captures the ideal behavior. By finding the occurrences of a motif and considering
2063 their immediate subsequences, we can make a more confident guess about this motif. We
2064 outline the approach for finding members of a motif¹ Given a subsequence A in a time
2065 series T , we denote the left-hand side of A in T as T_L . We find A 's nearest neighbor in
2066 T_L , denoted as B . They are the two members of a motif M . We want to identify other
2067 subsequences in T_L that belong to M . We define a threshold $\theta = r \times \text{ED}(A, B)$, where
2068 $r > 1$. The center M_C of M is defined as the average of A and B . Then, we compute
2069 the distance profile between M_C and T_L . Any part of the distance profile that is smarter
2070 than θ points to a member of M in T_L . These members can then be added to M . After
2071 identifying all the members of M and excluding these members in the next consideration,
2072 we can find the next left nearest neighbor of A in T_L , and repeat the same process for
2073 finding the next motif. Given a set of immediate subsequences of members (occurrences)
2074 of M , we can compute a more stable immediate subsequence (future) associated with M
2075 by excluding the outliers among them or using the ensemble value, such as the mean or
2076 median of them, to cancel the noise.

¹The idea has been mentioned in <https://www.cs.ucr.edu/~eamonn/TimeSeriesMotifs/>.

Identify outliers of immediate subsequences: When we have a set of immediate 2077 subsequences, we can determine whether an immediate subsequence is an outlier or not 2078 by comparing it with others. We provide a heuristic to identify an outlier as follows. 2079 Recall that the length of a nearest neighbor and its immediate subsequence is m and h , 2080 respectively. We concatenate all nearest neighbors into a single long time series T' . To 2081 establish a clear boundary, we append a NaN value after each of them. It ensures that all 2082 matrix profile computations do not consider subsequences that span multiple neighbors. 2083 Then, we compute a distance profile of T' to find the nearest neighbor distance d_i of each 2084 neighbor i . Let S_i be the sequence consisting of neighbor i of length m and its immediate 2085 subsequence of length h . The expected nearest neighbor distance of S_i (found within the 2086 set of all extended sequences) should be proportional to the increase in length: $d_i \times \frac{m+h}{m}$. 2087 If the actual nearest neighbor distance of S_i is greater than $r' \times (d_i \times (m + h)/m)$ among 2088 the others, where r' is a user-given value, the immediate subsequence in S_i is considered 2089 as an outlier. 2090

Chapter 6

2091

Conclusion and Future Directions

2092

In this thesis, we contribute to time series analysis by addressing two aspects, with applications in bioinformatics. The first is to frame the prediction of biological sequence problems as time series classification tasks. The second addresses a fundamental limitation in existing time series distance measures.

2093
2094
2095
2096

In the first study, we investigate the prediction of human Dicer cleavage sites. This task is important for the biogenesis of microRNAs (miRNAs). Accurate prediction of Dicer cleavage sites is crucial for elucidating mechanisms of post-transcriptional gene regulation. Computationally, this task is a classification problem. First, we curate the dataset based on existing studies. The resulting datasets are 14-strings. Then, they are transformed into time series. We employ ROCKET-based classifiers for the classification. The main contributions are summarized as follows. We are the first to frame this problem as a multivariate time series classification problem. We introduced nine encoding methods for the transformation. In the transformation, to our surprise, we are the first to use the base-pair probabilities derived from the predicted secondary structure. We employ state-of-the-art time-series classifiers, namely ROCKET-based classifiers. They use random convolutional kernels to generate the summary statistics and then use a simple ridge classifier to generate the final results. Because of the simplicity of the transformation method and the classifiers we adopted, our framework, namely MTSCCleave, is fast. It achieved predictive performance comparable to or even better than deep learning-based state-of-the-art methods. Furthermore, MTSCCleave demonstrated substantial computational efficiency, with speedups ranging from 3.7X to 28.8X relative to existing methods. We carried out perturbation-based experiments to identify the subsequence that are important for the classification. We found that regions near the center of the pre-miRNA secondary structure are most critical for Dicer cleavage site determination. It aligns with the existing study. Future work for this study is as follows. We make use of the predicted secondary structure information to construct the complementary strand and the base pair probability sequence for the input strand. However, there is more than one predicted secondary structure for the given RNA sequence. One future work is to make

use of all potential secondary structures, each with its own pair probability sequence, and encode this data into a multivariate time series with more channels. Another area for future work is to use interpretable time series classifiers, such as those based on time series shapelets. By doing this, we can study which subsequence is critical for the definition of the classes, namely “5p cleav”, “5p non-cleav”, “3p leav”, and “3p non-cleav” because shapelets serve as the subsequence that has the most discriminating power between classes.

The second study develops a new distance measure framework, namely PSD. It aims to release a fundamental assumption that the prior studies have overlooked. There is only one scaling rate throughout the entire time series. However, there are much data that exhibits multiple rates. For example, human motion or music performance. They consist of phases. Each phase has its own expression rate. Existing distance measures cannot account for such variations. For example, DTW is designed for handling local distortions. US assumes that there is only one scaling factor in the whole series. To address this, we introduced the Piecewise Scaling Distance (PSD) framework, the first of its kind to account for multiple scaling factors. Recall that PSD is agnostic to the base measure we used. We can use any existing distance measure as the base measure. We studied the two instantiations of it. They are PSED (using Euclidean Distance as the base measure) and PSDTW (using DTW as the base measure). We provided an exact dynamic programming solution to compute PSD and three general ways to speed it up. In particular, we proposed a constrained version of it that limits the search space based on allowed segment lengths derived from scaling factor bounds. Besides, we use parallel computing and early abandoning to further accelerate it. For PSDTW, due to its quadratic complexity, we can further speed it up using lower-bounding techniques. Experiments show that PSD, and in particular PSED, perform best when the query contains multi-rate distortions, compared with ED, DTW and the other five DTW-based methods.

Future works on it are as follows. Currently, the number of segments P is given by users. It is preferable to develop a heuristic or algorithmic approach to automatically determine P . A simple heuristic is to test a range of possible P . Besides, while we have successfully applied a lower bound on PSDTW and accelerated it, the computational overhead of calculating a lower bound on PSED outweighs the pruning benefit and makes the computation slower eventually. Developing a specialized lower bound for PSED could further improve its running time. In addition, some existing works on speeding up DTW such as “incremental DTW” can reuse the accumulated cost matrix D . However, it is challenging to apply a similar method to USDTW and PSDTW due to time series interpolation.

Bibliography

2158

- [1] M. Middlehurst, P. Schäfer, and A. Bagnall, “Bake off redux: A review and experimental evaluation of recent time series classification algorithms,” *Data Mining and Knowledge Discovery*, vol. 38, no. 4, pp. 1958–2031, Jul. 2024. 2159
2160
2161
- [2] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978. 2162
2163
2164
- [3] F. Itakura, “Minimum prediction residual principle applied to speech recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, Feb. 1975. 2165
2166
2167
- [4] E. J. Keogh and M. J. Pazzani, “Derivative Dynamic Time Warping,” in *Proceedings of the 2001 SIAM International Conference on Data Mining (SDM)*, ser. 2168
2169
Proceedings. Society for Industrial and Applied Mathematics, Apr. 2001, pp. 1–11. 2170
- [5] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, “Weighted dynamic time warping 2171
for time series classification,” *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2172
Sep. 2011. 2173
- [6] J. Zhao and L. Itti, “shapeDTW: Shape Dynamic Time Warping,” *Pattern Recognition*, vol. 74, pp. 171–184, Feb. 2018. 2174
2175
- [7] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, “Dimensionality Reduction 2176
for Fast Similarity Search in Large Time Series Databases,” *Knowledge and 2177
Information Systems*, vol. 3, no. 3, pp. 263–286, Aug. 2001. 2178
- [8] B.-K. Yi and C. Faloutsos, “Fast Time Sequence Indexing for Arbitrary L_p Norms,” 2179
in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. 2180
VLDB ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Sep. 2181
2000, pp. 385–394. 2182
- [9] J. Zhao and L. Itti, “Decomposing time series with application to temporal seg- 2183
mentation,” in *2016 IEEE Winter Conference on Applications of Computer Vision 2184
(WACV)*, Mar. 2016, pp. 1–9. 2185

- 2186 [10] M. Herrmann and G. I. Webb, "Amercing: An intuitive and effective constraint for
2187 dynamic time warping," *Pattern Recognition*, vol. 137, p. 109333, May 2023.
- 2188 [11] L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, R. B. Orr, and N. A.
2189 Campbell, *Campbell Biology*, twelfth ed. New York, NY: Pearson, 2020.
- 2190 [12] B. Alberts, *Molecular Biology of the Cell*, 7th ed. New York: W. W. Norton &
2191 Company, 2022.
- 2192 [13] W. W. Cohen, *A Computer Scientist's Guide to Cell Biology: A Travelogue from a*
2193 *Stranger in a Strange Land*. New York, NY: Springer-Verl, 2007.
- 2194 [14] Y. Lee, K. Jeon, J.-T. Lee, S. Kim, and V. N. Kim, "MicroRNA maturation:
2195 Stepwise processing and subcellular localization," *The EMBO Journal*, vol. 21,
2196 no. 17, pp. 4663–4670, Sep. 2002.
- 2197 [15] S. Gu, L. Jin, Y. Zhang, Y. Huang, F. Zhang, P. N. Valdmanis, and M. A. Kay,
2198 "The Loop Position of shRNAs and Pre-miRNAs Is Critical for the Accuracy of
2199 Dicer Processing In Vivo," *Cell*, vol. 151, no. 4, pp. 900–911, Nov. 2012.
- 2200 [16] Y. Feng, X. Zhang, P. Graves, and Y. Zeng, "A comprehensive analysis of precursor
2201 microRNA cleavage by human Dicer," *RNA*, vol. 18, no. 11, pp. 2083–2092, Nov.
2202 2012.
- 2203 [17] I. J. MacRae, K. Zhou, and J. A. Doudna, "Structural determinants of RNA recogni-
2204 tion and cleavage by Dicer," *Nature Structural & Molecular Biology*, vol. 14,
2205 no. 10, pp. 934–940, Oct. 2007.
- 2206 [18] F. Ahmed, R. Kaundal, and G. P. Raghava, "PHDcleav: A SVM based method for
2207 predicting human Dicer cleavage sites using sequence and secondary structure of
2208 miRNA precursors," *BMC Bioinformatics*, vol. 14, no. 14, p. S9, Oct. 2013.
- 2209 [19] Y. Bao, M. Hayashida, and T. Akutsu, "LBSizeCleav: Improved support vector
2210 machine (SVM)-based prediction of Dicer cleavage sites using loop/bulge length,"
2211 *BMC Bioinformatics*, vol. 17, no. 1, p. 487, Nov. 2016.
- 2212 [20] P. Liu, J. Song, C.-Y. Lin, and T. Akutsu, "ReCGBM: A gradient boosting-based
2213 method for predicting human dicer cleavage sites," *BMC Bioinformatics*, vol. 22,
2214 no. 1, p. 63, Feb. 2021.
- 2215 [21] L. Mu, J. Song, T. Akutsu, and T. Mori, "DiCleave: A deep learning model for
2216 predicting human Dicer cleavage sites," *BMC Bioinformatics*, vol. 25, no. 1, p. 13,
2217 Jan. 2024.

- [22] L. Mu and T. Akutsu, “DiCleavePlus: A Transformer-Based Model to Detect Human Dicer Cleavage Sites Within Cleavage Patterns,” *Genes to Cells*, vol. 31, no. 1, p. e70074, 2026. 2218
2219
2220
- [23] S. Griffiths-Jones, H. K. Saini, and S. van Dongen, “miRBase: Tools for microRNA 2221
genomics,” *Nucleic Acids Research*, vol. 36, no. suppl_1, pp. D154–D158, Jan. 2008. 2222
- [24] T. Xu, N. Su, L. Liu, J. Zhang, H. Wang, W. Zhang, J. Gui, K. Yu, J. Li, and T. D. 2223
Le, “miRBaseConverter: An R/Bioconductor package for converting and retrieving 2224
miRNA name, accession, sequence and family information in different versions of 2225
miRBase,” *BMC Bioinformatics*, vol. 19, no. 19, p. 514, Dec. 2018. 2226
- [25] M. J. Zvelebil, J. O. Baum, and M. Zvelebil, *Understanding Bioinformatics*. New 2227
York: Garland Science, 2008. 2228
- [26] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. 2229
Stadler, and I. L. Hofacker, “ViennaRNA Package 2.0,” *Algorithms for Molecular 2230
Biology*, vol. 6, no. 1, p. 26, Nov. 2011. 2231
- [27] C. C. Aggarwal, *Data Mining: The Textbook*. Cham: Springer International Pub- 2232
lishing, 2015. 2233
- [28] G. Mendizabal-Ruiz, I. Román-Godínez, S. Torres-Ramos, R. A. Salido-Ruiz, and 2234
J. A. Morales, “On DNA numerical representations for genomic similarity compu- 2235
tation,” *PLOS ONE*, vol. 12, no. 3, p. e0173288, Mar. 2017. 2236
- [29] D. Anastassiou, “Genomic signal processing,” *IEEE Signal Processing Magazine*, 2237
vol. 18, no. 4, pp. 8–20, Jul. 2001. 2238
- [30] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Za- 2239
karia, and E. Keogh, “Searching and mining trillions of time series subsequences 2240
under dynamic time warping,” in *Proceedings of the 18th ACM SIGKDD Interna- 2241
tional Conference on Knowledge Discovery and Data Mining*, ser. KDD ’12. New 2242
York, NY, USA: Association for Computing Machinery, Aug. 2012, pp. 262–270. 2243
- [31] P. D. Cristea, “Conversion of nucleotides sequences into genomic signals,” *Journal 2244
of Cellular and Molecular Medicine*, vol. 6, no. 2, pp. 279–303, 2002. 2245
- [32] N. Chakravarthy, A. Spanias, L. D. Iasemidis, and K. Tsakalis, “Autoregressive 2246
Modeling and Feature Analysis of DNA Sequences,” *EURASIP Journal on Ad- 2247
vances in Signal Processing*, vol. 2004, no. 1, pp. 1–16, Dec. 2004. 2248
- [33] J. Zhao, X. W. Yang, J. P. Li, and Y. Y. Tang, “DNA Sequences Classification 2249
Based on Wavelet Packet Analysis,” in *Wavelet Analysis and Its Applications*. 2250
Springer, Berlin, Heidelberg, 2001, pp. 424–429. 2251

- 2252 [34] T. Holden, R. Subramaniam, R. Sullivan, E. Cheung, C. Schneider, G. T. Jr,
2253 A. Flamholz, D. H. Lieberman, and T. D. Cheung, “ATCG nucleotide fluctuation
2254 of *Deinococcus radiodurans* radiation genes,” in *Instruments, Methods, and*
2255 *Missions for Astrobiology X*, vol. 6694. SPIE, Oct. 2007, pp. 402–411.
- 2256 [35] A. S. Nair and S. P. Sreenadhan, “A coding measure scheme employing electron-
2257 ion interaction pseudopotential (EIIP),” *Bioinformation*, vol. 1, no. 6, pp. 197–202,
2258 Oct. 2006.
- 2259 [36] M. Akhtar, J. Epps, and E. Ambikairajah, “On DNA Numerical Representations
2260 for Period-3 Based Exon Prediction,” in *2007 IEEE International Workshop on*
2261 *Genomic Signal Processing and Statistics*, Jun. 2007, pp. 1–4.
- 2262 [37] R. Zhang and C.-T. Zhang, “Z Curves, An Intutive Tool for Visualizing and An-
2263 alyzing the DNA Sequences,” *Journal of Biomolecular Structure and Dynamics*,
2264 vol. 11, no. 4, pp. 767–782, Feb. 1994.
- 2265 [38] J. A. Berger, S. K. Mitra, M. Carli, and A. Neri, “Visualization and analysis of DNA
2266 sequences using DNA walks,” *Journal of the Franklin Institute*, vol. 341, no. 1, pp.
2267 37–53, Jan. 2004.
- 2268 [39] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series
2269 classification bake off: A review and experimental evaluation of recent algorithmic
2270 advances,” *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, May
2271 2017.
- 2272 [40] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, “The great multi-
2273 variate time series classification bake off: A review and experimental evaluation
2274 of recent algorithmic advances,” *Data Mining and Knowledge Discovery*, vol. 35,
2275 no. 2, pp. 401–449, Mar. 2021.
- 2276 [41] A. Dempster, F. Petitjean, and G. I. Webb, “ROCKET: Exceptionally fast and
2277 accurate time series classification using random convolutional kernels,” *Data Mining*
2278 and *Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020.
- 2279 [42] A. Dempster, D. F. Schmidt, and G. I. Webb, “MiniRocket: A Very Fast (Almost)
2280 Deterministic Transform for Time Series Classification,” in *Proceedings of the 27th*
2281 *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD
2282 ’21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp.
2283 248–257.
- 2284 [43] C. W. Tan, A. Dempster, C. Bergmeir, and G. I. Webb, “MultiRocket: Multiple
2285 pooling operators and transformations for fast and effective time series classifica-

- tion,” *Data Mining and Knowledge Discovery*, vol. 36, no. 5, pp. 1623–1646, Sep. 2022. 2286
2287
- [44] A. Dempster, D. F. Schmidt, and G. I. Webb, “Hydra: Competing convolutional kernels for fast and accurate time series classification,” *Data Mining and Knowledge Discovery*, vol. 37, no. 5, pp. 1779–1805, Sep. 2023. 2288
2289
2290
- [45] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, Oct. 1975. 2291
2292
2293
- [46] J. Gorodkin, “Comparing two K-category assignments by a K-category correlation coefficient,” *Computational Biology and Chemistry*, vol. 28, no. 5, pp. 367–374, Dec. 2004. 2294
2295
2296
- [47] T.-c. Fu, “A review on time series data mining,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, Feb. 2011. 2297
2298
- [48] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen, “Efficient generation of motion transitions using spacetime constraints,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, Aug. 1996, pp. 147–154. 2299
2300
2301
2302
2303
- [49] Y. Li, T. Wang, and H.-Y. Shum, “Motion texture: A two-level statistical model for character motion synthesis,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 465–472, Jul. 2002. 2304
2305
2306
- [50] R. B. Dannenberg, W. P. Birmingham, G. P. Tzanetakis, C. P. Meek, N. P. Hu, and B. P. Pardo, “The MUSART Testbed for Query-by-Humming Evaluation,” *Comput. Music J.*, vol. 28, no. 2, pp. 34–48, Jun. 2004. 2307
2308
2309
- [51] K.-C. Li, M. Yan, and S. Yuan, “A Simple Statistical Model for Depicting the Cdc15-Synchronized Yeast Cell-Cycle Regulated Gene Expression Data,” *Statistica Sinica*, vol. 12, no. 1, pp. 141–158, 2002. 2310
2311
2312
- [52] G. E. A. P. A. Batista, E. J. Keogh, O. M. Tataw, and V. M. A. de Souza, “CID: An efficient complexity-invariant distance for time series,” *Data Mining and Knowledge Discovery*, vol. 28, no. 3, pp. 634–669, May 2014. 2313
2314
2315
- [53] A. W.-C. Fu, E. Keogh, L. Y. H. Lau, C. A. Ratanamahatana, and R. C.-W. Wong, “Scaling and time warping in time series querying,” *The VLDB Journal*, vol. 17, no. 4, pp. 899–921, Jul. 2008. 2316
2317
2318

- 2319 [54] Y. Shen, Y. Chen, E. Keogh, and H. Jin, “Searching Time Series with Invariance to
2320 Large Amounts of Uniform Scaling,” in *2017 IEEE 33rd International Conference*
2321 *on Data Engineering (ICDE)*, Apr. 2017, pp. 111–114.
- 2322 [55] ——, “Accelerating Time Series Searching with Large Uniform Scaling,” in *Pro-
2323 ceedings of the 2018 SIAM International Conference on Data Mining (SDM)*, ser.
2324 Proceedings. Society for Industrial and Applied Mathematics, May 2018, pp.
2325 234–242.
- 2326 [56] C. C. Aggarwal and C. K. Reddy, Eds., *Data Clustering: Algorithms and Applica-
2327 tions*, 1st ed. Boca Raton: Chapman and Hall/CRC, Aug. 2013.
- 2328 [57] P. Esling and C. Agon, “Time-series data mining,” *ACM Computing Surveys*,
2329 vol. 45, no. 1, pp. 1–34, Nov. 2012.
- 2330 [58] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu,
2331 J. Zakaria, and E. Keogh, “Addressing Big Data Time Series: Mining Trillions of
2332 Time Series Subsequences Under Dynamic Time Warping,” *ACM Trans. Knowl.
2333 Discov. Data*, vol. 7, no. 3, pp. 10:1–10:31, Sep. 2013.
- 2334 [59] C. W. Tan, F. Petitjean, and G. I. Webb, “Elastic bands across the path: A new
2335 framework and method to lower bound DTW,” in *Proceedings of the 2019 SIAM*
2336 *International Conference on Data Mining (SDM)*, ser. Proceedings. Society for
2337 Industrial and Applied Mathematics, May 2019, pp. 522–530.
- 2338 [60] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, “Elastic similarity and distance
2339 measures for multivariate time series,” *Knowledge and Information Systems*, vol. 65,
2340 no. 6, pp. 2665–2698, Jun. 2023.
- 2341 [61] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time
2342 and space,” *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, Oct. 2007.
- 2343 [62] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopoulos, and M. Cardle, “Indexing large
2344 human-motion databases,” in *Proceedings of the Thirtieth International Conference*
2345 *on Very Large Data Bases - Volume 30*, ser. VLDB ’04. Toronto, Canada: VLDB
2346 Endowment, Aug. 2004, pp. 780–791.
- 2347 [63] D. Yankov, E. Keogh, J. Medina, B. Chiu, and V. Zordan, “Detecting time series
2348 motifs under uniform scaling,” in *Proceedings of the 13th ACM SIGKDD Interna-
2349 tional Conference on Knowledge Discovery and Data Mining*, ser. KDD ’07. New
2350 York, NY, USA: Association for Computing Machinery, Aug. 2007, pp. 844–853.
- 2351 [64] Y. Zhu and D. Shasha, “Warping indexes with envelope transforms for query by
2352 humming,” in *Proceedings of the 2003 ACM SIGMOD International Conference on*

- Management of Data*, ser. SIGMOD '03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 181–192. 2353
- [65] E. Keogh and C. A. Ratanamahatana, “Exact indexing of dynamic time warping,” 2355
Knowledge and Information Systems, vol. 7, no. 3, pp. 358–386, Mar. 2005. 2356
- [66] S.-W. Kim, S. Park, and W. Chu, “An index-based approach for similarity search 2357
supporting time warping in large sequence databases,” in *Proceedings 17th Intern- 2358
ational Conference on Data Engineering*, Apr. 2001, pp. 607–614. 2359
- [67] C. W. Tan, M. Herrmann, G. Forestier, G. I. Webb, and F. Petitjean, “Efficient 2360
search of the best warping window for Dynamic Time Warping,” in *Proceedings of 2361
the 2018 SIAM International Conference on Data Mining (SDM)*, ser. Proceedings. 2362
Society for Industrial and Applied Mathematics, May 2018, pp. 225–233. 2363
- [68] C. A. Ratanamahatana and E. Keogh, “Everything you know about Dynamic Time 2364
Warping is Wrong,” in *3 Rd International Workshop on Mining Temporal and 2365
Sequential Data (TDM-04)*. Citeseer, 2004. 2366
- [69] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. 2367
Ratanamahatana, and E. Keogh, “The UCR time series archive,” *IEEE/CAA Jour- 2368
nal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019. 2369
- [70] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bu- 2370
latova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, and A. Bagnall, “Aeon: 2371
A Python Toolkit for Learning from Time Series,” *Journal of Machine Learning 2372
Research*, vol. 25, no. 289, pp. 1–10, 2024. 2373
- [71] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, “Precision and Recall 2374
for Time Series,” in *Advances in Neural Information Processing Systems*, vol. 31. 2375
Curran Associates, Inc., 2018. 2376
- [72] M. Leodolter, C. Plant, and N. Brändle, “IncDTW: An R Package for Incremental 2377
Calculation of Dynamic Time Warping,” *Journal of Statistical Software*, vol. 99, 2378
pp. 1–23, Sep. 2021. 2379
- [73] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, “Modeling Long- and Short-Term 2380
Temporal Patterns with Deep Neural Networks,” in *The 41st International ACM 2381
SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR 2382
’18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 2383
95–104. 2384
- [74] C. Chatfield, *The Analysis of Time Series: An Introduction*, Sixth ed. 2385
New York: Chapman and Hall/CRC, Jul. 2003. 2386

- 2387 [75] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha,
2388 “Efficient Machine Learning for Big Data: A Review,” *Big Data Research*, vol. 2,
2389 no. 3, pp. 87–93, Sep. 2015.
- 2390 [76] W. Li and K. L. E. Law, “Deep Learning Models for Time Series Forecasting: A
2391 Review,” *IEEE Access*, vol. 12, pp. 92 306–92 327, 2024.
- 2392 [77] N. I. Sapankevych and R. Sankar, “Time Series Prediction Using Support Vector
2393 Machines: A Survey,” *IEEE Computational Intelligence Magazine*, vol. 4, no. 2,
2394 pp. 24–38, May 2009.
- 2395 [78] Z. Chen, M. Ma, T. Li, H. Wang, and C. Li, “Long sequence time-series forecasting
2396 with deep learning: A survey,” *Information Fusion*, vol. 97, p. 101819, Sep. 2023.
- 2397 [79] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, “Deep
2398 Learning for Time Series Forecasting: A Survey,” *Big Data*, vol. 9, no. 1, pp. 3–21,
2399 Feb. 2021.
- 2400 [80] D. B. da Silva, D. Schmidt, C. A. da Costa, R. da Rosa Righi, and B. Eskofier,
2401 “DeepSigns: A predictive model based on Deep Learning for the early detection of
2402 patient health deterioration,” *Expert Systems with Applications*, vol. 165, p. 113905,
2403 Mar. 2021.
- 2404 [81] N. Wu, B. Green, X. Ben, and S. O’Banion, “Deep Transformer Models for Time
2405 Series Forecasting: The Influenza Prevalence Case,” Jan. 2020.
- 2406 [82] F. Martínez-Álvarez, G. Asencio-Cortés, J. F. Torres, D. Gutiérrez-Avilés,
2407 L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. C. Riquelme, and
2408 A. Troncoso, “Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic
2409 Based on the COVID-19 Propagation Model,” *Big Data*, vol. 8, no. 4, pp. 308–322,
2410 Aug. 2020.
- 2411 [83] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, “A methodology for ap-
2412 plying k-nearest neighbor to time series forecasting,” *Artificial Intelligence Review*,
2413 vol. 52, no. 3, pp. 2019–2037, Oct. 2019.
- 2414 [84] S. Tajmouati, B. E. L. Wahbi, A. Bedoui, A. Abarda, and M. Dakkon, “Applying
2415 k-nearest neighbors to time series forecasting: Two new approaches,” *Journal of
2416 Forecasting*, vol. 43, no. 5, pp. 1559–1574, 2024.
- 2417 [85] D. T. Thi, N. T. Phong, and N. D. Bui, “Forecast Timeseries Based On Matrix
2418 Profile,” *Journal of Informatics and Innovative Technologies (JIIT)*, 2021.

- [86] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, “Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets,” in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Barcelona, Spain: IEEE, Dec. 2016, pp. 1317–1322.
- [87] Y. Zhu, M. Imamura, D. Nikovski, and E. Keogh, “Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining,” in *2017 IEEE International Conference on Data Mining (ICDM)*, Nov. 2017, pp. 695–704.
- [88] S. M. Law, “STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining,” *Journal of Open Source Software*, vol. 4, no. 39, p. 1504, Jul. 2019.
- [89] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, May 2015.
- [90] S. Elsayed, D. Thyssens, A. Rashed, H. S. Jomaa, and L. Schmidt-Thieme, “Do We Really Need Deep Learning Models for Time Series Forecasting?” Oct. 2021.
- [91] G. E. P. Box and D. A. Pierce, “Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models,” *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.
- [92] B. Lim and S. Zohren, “Time-series forecasting with deep learning: A survey,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20200209, Feb. 2021.
- [93] G. U. Yule, “On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer’s Sunspot Numbers,” *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 226, pp. 267–298, 1927.
- [94] R. J. Williams and D. Zipser, “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks,” *Neural Computation*, vol. 1, no. 2, pp. 270–280, Jun. 1989.
- [95] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts London, England: The MIT Press, 2022.
- [96] S. Kolassa and W. Schütz, “Advantages of the MAD/Mean Ratio over the MAPE,” *Foresight: The International Journal of Applied Forecasting*, vol. 6, pp. 40–43, 2007.

- 2451 [97] J. H. Friedman, “Greedy function approximation: A gradient boosting machine.”
2452 *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.
- 2453 [98] L. G. Serrano and S. Thrun, *Grokking Machine Learning*. Shelter Island: Manning,
2454 2021.
- 2455 [99] A. Géron, *Hands-On Machine Learning with Scikit-Learn and PyTorch: Concepts,*
2456 *Tools, and Techniques to Build Intelligent Systems*. US: O'Reilly Media, 2025.
- 2457 [100] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794.
- 2461 [101] A. V. Dorogush, V. Ershov, and A. Gulin, “CatBoost: Gradient boosting with
2462 categorical features support,” in *Workshop on ML Systems at NIPS 2017*, 2017.
- 2463 [102] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu,
2464 “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in*
2465 *Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.
- 2466 [103] R. Sen, H.-F. Yu, and I. S. Dhillon, “Think Globally, Act Locally: A Deep Neural
2467 Network Approach to High-Dimensional Time Series Forecasting,” in *Advances in*
2468 *Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
- 2469 [104] C.-C. M. Yeh, N. Kavantzas, and E. Keogh, “Matrix Profile VI: Meaningful Multi-
2470 dimensional Motif Discovery,” in *2017 IEEE International Conference on Data*
2471 *Mining (ICDM)*, Nov. 2017, pp. 565–574.