entropy

Article

A Fast Algorithm for Estimating Two-Dimensional Sample Entropy Based on an Upper Confidence Bound and Monte Carlo Sampling

Zeheng Zhou, Ying Jiang *, Weifeng Liu, Ruifan Wu, Zerong Li and Wenchao Guan

School of Computer Science and Engineering, Guangdong Province Key Laboratory of Computational Science, Sun Yat-sen University, Guangzhou 510275, China
* Correspondence: jiangy32@mail.sysu.edu.cn

Abstract: The two-dimensional sample entropy marks a significant advance in evaluating the regularity and predictability of images in the information domain. Unlike the direct computation of sample entropy, which incurs a time complexity of $O(N^2)$ for the series with N length, the Monte Carlo-based algorithm for computing one-dimensional sample entropy (MCSampEn) markedly reduces computational costs by minimizing the dependence on N. This paper extends MCSampEn to two dimensions, referred to as MCSampEn2D. This new approach substantially accelerates the estimation of two-dimensional sample entropy, outperforming the direct method by more than a thousand fold. Despite these advancements, MCSampEn2D encounters challenges with significant errors and slow convergence rates. To counter these issues, we have incorporated an upper confidence bound (UCB) strategy in MCSampEn2D. This strategy involves assigning varied upper confidence bounds in each Monte Carlo experiment iteration to enhance the algorithm's speed and accuracy. Our evaluation of this enhanced approach, dubbed UCBMCSampEn2D, involved the use of medical and natural image data sets. The experiments demonstrate that UCBMCSampEn2D achieves a 40% reduction in computational time compared to MCSampEn2D. Furthermore, the errors with UCBMCSampEn2D are only 30% of those observed in MCSampEn2D, highlighting its improved accuracy and efficiency.

Keywords: sample entropy; Monte Carlo algorithm; upper confidence bound strategy

check for updates

Citation: Zhou, Z.; Jiang, Y.; Liu, W.; Wu, R.; Li, Z.; Guan, W. A Fast Algorithm for Estimating Two-Dimensional Sample Entropy Based on an Upper Confidence Bound and Monte Carlo Sampling. *Entropy* **2024**, *26*, 155. https://doi.org/10.3390/e26020155

Academic Editor: George Manis

Received: 9 January 2024
Revised: 7 February 2024
Accepted: 7 February 2024
Published: 10 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Information theory serves as a foundational framework for developing tools to represent and manipulate information [1], particularly in the realms of signal and image processing. Within this paradigm, entropy stands out as a key concept, functioning as a metric for quantifying uncertainty or irregularity within a system or dataset [2]. Stemming from Shannon's pioneering work on entropy [1], subsequent researchers have advanced the field by introducing diverse methods. Notable examples include one-dimensional approximate entropy (ApEn) [3,4], dispersion entropy [5], sample entropy (SampEn) [2], and other innovative approaches. Multiscale entropy, hierarchical entropy, and their variants have been applied to various fields, such as fault identification [6,7] and feature extraction [8], beyond physiological time series analysis.

In 1991, the concept of ApEn was introduced as a method for quantifying the irregularity of time series [3]. ApEn relies on the conditional probability of the negative average natural logarithm, specifically examining the likelihood that two sequences, initially similar at m points, will remain similar at the subsequent point. Addressing the computational challenges associated with self-similar patterns in ApEn, SampEn was subsequently developed, obtaining sampling points using global random sampling to represent the signal, leading to more robust estimations [1]. Notably, in the domain of biomedical signal processing, SampEn has been successfully employed, demonstrating its effectiveness and applicability [9].

Computing SampEn involves enumerating the number of similar templates within a time series, essentially requiring the count of matching template pairs for the given series. The direct computation of SampEn inherently has a computational complexity $O(N^2)$, where N represents the length of the time series under analysis. To expedite this process, kd-tree based algorithms have been proposed for sample entropy computation. These algorithms effectively reduce the time complexity to $O(N^{2-\frac{1}{m+1}})$, where m is denoting the template (or pattern) length [10,11]. Additionally, various approaches like box-assisted [12,13], bucket-assisted [14], lightweight [15], and assisted sliding box (SBOX) [16] algorithms have been developed. Nonetheless, the computational complexity for all these algorithms remains at $O(N^2)$.To tackle the challenge of computational complexity, a rapid algorithm for estimating Sample Entropy using the Monte Carlo algorithm (MCSampEn) has been introduced in [17]. This algorithm features computational costs that are independent of N, and its estimations converge to the exact sample entropy as the number of repeated experiments increases. Experimental results reported in [17] demonstrate that MCSampEn achieves a speedup of 100 to 1000 times compared to kd-tree and assisted sliding box algorithms, while still delivering satisfactory approximation accuracy.

However, the MCSampEn algorithm utilizes a random sampling pattern where the importance of each sampling point varies. The application of averaging methods in this context leads to significant fluctuations in errors, resulting in a large standard deviation and slow convergence of the entire process. To address this, we introduce the upper confidence bound (UCB) strategy to set different confidence bounds for various sampling points, assigning varying levels of importance to these points. This approach mitigates errors caused by averaging methods, reduces the standard deviation of the MCSampEn algorithm, accelerates the convergence speed, and significantly improves computational speed.

In this paper, we extend MCSampEn to compute two-dimensional sample entropy, referred to as MCSampEn2D. To mitigate the challenges of convergence brought about by MCSampEn2D, we integrate MCSampEn2D with the upper confidence bound (UCB) strategy to reduce variance [18]. We call this refined algorithm as the UCBMCSampEn2D. By establishing a confidence upper bound for each experimental round, we differentiate the importance of each experiment. The higher the importance of an experiment, the greater its assigned confidence upper bound. Conversely, for experiments with lower confidence upper bounds, our objective is to minimize the errors they introduce to the greatest extent possible.

The UCBMCSampEn2D algorithm is a notable enhancement, enabling swift convergence and minimal errors, even in scenarios with a limited number of sampling points. The UCBMCSampEn2D algorithm eliminates the need for explicit knowledge of the data distribution. By optimistically adjusting the weights for each round of experiments and estimating the upper bound of the expected value, the UCBMASampEn2D algorithm operates without introducing additional computational burden. This algorithm continuously optimizes weights through online learning [19], providing a solution without the requirement for explicit knowledge of the data distribution. In this study, we systematically assess the performance of the UCBMCSampEn2D algorithm across medical image and natural image datasets. Our investigation reveals two primary advantages of the proposed UCBMCSampEn2D algorithm: (1) The UCBMCSampEn2D algorithm places greater emphasis on the importance of different rounds of experiments by assigning importance levels, leading to a reduction in overall errors and faster convergence speed. (2) Leveraging a reinforcement learning approach, the UCBMCSampEn2D algorithm utilizes local optima to represent true entropy. Through the application of upper confidence bounds, it effectively addresses the challenge of determining how to set importance levels.

Further detailed analysis and results will be provided in subsequent sections to expound upon these advantages and demonstrate the effectiveness of the UCBMCSampEn2D algorithm in comparison to conventional methods.

2. Fast Algorithms for Estimating Two-Dimensional Sample Entropy

In this section, we introduce the MCSampEn2D and UCBMCSampEn2D algorithms. Before delving into the details of these approaches, we will first establish the key mathematical symbols and fundamental concepts integral to understanding MCSampEn2D. This groundwork provides a clear and comprehensive exposition of both the MCSampEn2D and UCBMCSampEn2D algorithms.

2.1. Groundwork of Two-Dimensional Sample Entropy

Let $\mathbf{U} = \{u_{i,j} \in \mathbb{R}\}_{i=1,2,\ldots,h}^{j=1,2,\ldots,w}$ be an image of size $h \times w$. For all $k \in \{1,2,\ldots,h-(m_h-1)\}$ and $l \in \{1,2,\ldots,w-(m_w-1)\}$, define two-dimensional matrices $\mathbf{X}^{\mathbf{m}}_{k,l}$ with size $m_h \times m_w$, named template matrices, by

$$\mathbf{X^m}_{k,l} = \begin{bmatrix} u_{k,l} & u_{k,l+1} & \cdots & u_{k,l+(m_w-1)} \\ u_{k+1,l} & u_{k+1,l+1} & \cdots & u_{k+1,l+(m_w-1)} \\ \vdots & \vdots & \vdots & \vdots \\ u_{k+(m_h-1),l} & u_{k+(m_h-1),l+1} & \cdots & u_{k+(m_h-1),l+(m_w-1)} \end{bmatrix}, \quad (1)$$

where $\mathbf{m} = [m_h, m_w]$ is the embedding dimension vector [20]. We also define $\mathbf{X^m} := \{\mathbf{X}^{\mathbf{m}}_{a,b} : a \in \{1,2,\ldots,h-(m_h-1)\}\text{and } b \in \{1,2,\ldots,w-(m_w-1)\}\}$. For all $k,a \in \{1,2,\ldots,h-(m_h-1)\}$ and $l,b \in \{1,2,\ldots,w-(m_w-1)\}$, let $d(\mathbf{X}^{\mathbf{m}}_{k,l}, \mathbf{X}^{\mathbf{m}}_{a,b})$ be the greatest element of the absolute differences between $\mathbf{X}^{\mathbf{m}}_{k,l}$ and $\mathbf{X}^{\mathbf{m}}_{a,b}$. We denote by $\#E$ the cardinality of a set E. Then, for fixed k and l, we count $\#\{\mathbf{X}^{\mathbf{m}}_{a,b} \in \mathbf{X^m} : d(\mathbf{X}^{\mathbf{m}}_{k,l}, \mathbf{X}^{\mathbf{m}}_{a,b}) \le r \text{ and } (k-a)^2 + (l-b)^2 \neq 0\}$, and compute

$$\phi^{\mathbf{m}}_{k,l}(r) = \frac{\#\{\mathbf{X}^{\mathbf{m}}_{a,b} \in \mathbf{X^m} : d(\mathbf{X}^{\mathbf{m}}_{k,l}, \mathbf{X}^{\mathbf{m}}_{a,b}) \le r \text{ and } (k-a)^2 + (l-b)^2 \neq 0\}}{(h-m_h)(w-m_w)-1}, \quad (2)$$

where r is the predefined threshold (tolerance factor). We also define $\phi^{\mathbf{m}}(r)$ as

$$\phi^{\mathbf{m}}(r) = \frac{1}{(h-m_h)(w-m_w)} \sum_{k=1}^{k=h-m_h} \sum_{l=1}^{l=w-m_w} \phi^{\mathbf{m}}_{k,l}(r). \quad (3)$$

Finally, SampEn2D is defined as follows [20]:

$$\mathrm{SampEn2D}(\mathbf{U}, \mathbf{m}, r) = -\log \frac{\phi^{\mathbf{m}+1}(r)}{\phi^{\mathbf{m}}(r)}, \quad (4)$$

where $\mathbf{m}+1 = [m_h+1, m_w+1]$. The parameter $\mathbf{m}$ indicates the size of the matrices, which are analyzed or compared along images. In this study, [20,21], $\mathbf{m}$ is chosen to obtain squared template matrices, and let $m_h = m_w \in \mathbb{Z}$. For all $w,h \in \mathbb{Z}$, we denote $Z_{w,h} := \{(i,j) \in \mathbb{Z}^2 : 1 \le i \le w, 1 \le j \le h\}$. The process for computing $\mathrm{SampEn2D}(\mathbf{U}, \mathbf{m}, r)$ is summarized in Algorithm 1.

The parameter r is selected to strike a balance between the quality of the logarithmic likelihood estimates and the potential loss of signals or image information. If r is chosen to be too small (less than 0.1 of the standard deviation of an image), it leads to poor conditional probability estimates. Additionally, to mitigate the influence of noise on the data, it is advisable to opt for a larger r. Conversely, when r exceeds 0.4 of the standard deviation, excessive loss of detailed data information occurs. Therefore, a trade-off between large and small r values is essential. For a more in-depth discussion on the impact of these parameters in SampEn2D, please refer to [20].

Algorithm 1 Two-dimensional sample entropy

```
Require: Sequence $\mathbf{U} := \{u_{i,j} : 1 \leq i \leq w, 1 \leq j \leq h\}$, $\mathbf{s} \subset \mathbb{Z}_{w,h}$ template length $m$ and
    threshold $r$.
 1: procedure SampEn2D($\mathbf{U}, \mathbf{s}, m, r$)
 2:     Set $count = 0$,
 3:     Set $N_0 = \#\mathbf{s}$,
 4:     for $i = 1$ to $N_0$ do
 5:         for $j = i$ to $N_0$ do
 6:             $(k, l) = \mathbf{s}_i$; $(a, b) = \mathbf{s}_j$,
 7:             $\mathbf{X^m}_{i,j} = \mathbf{U}[i : i + m - 1][j : j + m - 1]$,
 8:             $\mathbf{X^m}_{a,b} = \mathbf{U}[a : a + m - 1][b : b + m - 1]$,
 9:             if $d(\mathbf{X^m}_{i,j}, \mathbf{X^m}_{a,b}) \leq r$ then
10:                 $count = count + 1$,
11:     return $count$
```

2.2. A Monte Carlo-Based Algorithm for Estimating Two-Dimensional Sample Entropy

The SampEn is fundamentally defined as $-\log(B/A)$, where B represents the number of matching template pairs of length m, and A represents the number of matching template pairs of length $m+1$. The most computationally intensive step in calculating SampEn involves determining the ratio B/A for templates of lengths m and $m+1$. Notably, $\frac{A}{N(N-1)}$ (resp. $\frac{B}{N(N-1)}$) denotes the probability of template matches of length m (resp. $m+1$), and the ratio B/A can be interpreted as a conditional probability. The statement indicates that the computation time of the MCSampEn method becomes independent of the data size and, instead, depends on the number of sampling points N_0 and the number of repetitions N_1. This complexity is denoted as $O(N_1(N_0 + N_0^2))$ [17].

The objective of the MCSampEn algorithm [17] is to approximate this conditional probability for the original dataset by considering the conditional probability of a randomly subsampled dataset. Specifically, the MCSampEn randomly selects N_0 templates of length m and N_0 templates of length $m+1$ from the original time series. It subsequently computes the number of matching pairs in the selected templates of length m (resp. $m+1$), denoted as $\tilde{A}$ (resp. $\tilde{B}$). This selection process is repeated N_1 times, and the average value of $\{\tilde{A}_k : k = 1, 2, \ldots, N_1\}$ (resp. $\{\tilde{B}_k : k = 1, 2, \ldots, N_1\}$), represented as $\bar{A}_{N_1}$ (resp. $\bar{B}_{N_1}$), is then calculated. Finally, $-\log(\bar{B}_{N_1}/\bar{A}_{N_1})$ is employed to approximate the complexity measurement $-\log(B/A)$ for the time series. The entire process can be expressed succinctly using the following formula:

$$\bar{A}_{N_1} := \frac{1}{N_1}\sum_{k=1}^{N_1}\tilde{A}_k, \quad \text{and} \quad \bar{B}_{N_1} := \frac{1}{N_1}\sum_{k=1}^{N_1}\tilde{B}_k,$$

$$\mathrm{MCSampEn} := -\log\frac{\bar{B}_{N_1}}{\bar{A}_{N_1}}, \tag{5}$$

where $\tilde{A}_k$ (resp. $\tilde{B}_k$) means the number of matching pairs in the selected templates of length m (resp. $m+1$) in in the k-th experiment.

When extending the MCSampEn algorithm to process two-dimensional data, a random sampling step is essential at the outset to acquire N data points. The technique employs a specific sampling strategy to sample N_0 positive integer sets, denoted as $V = \{v_i : i = 1, 2, \ldots, N_0\}$ and $v_i \leq (h - m_h) \times (w - m_w)$. Subsequently, for each v_i, compute $k = v_i/h$ and $l = v_i\%h$, which indicates a two-dimensional matrix $\mathbf{X}^{\mathbf{m}}_{k,l} \in \mathbf{X}^{\mathbf{m}}$. Then, we randomly select N_0 templates of size $\mathbf{m}$ and N_0 templates of length $\mathbf{m}+1$ from the original two-dimensional data. It subsequently computes the number of matching pairs, with tolerant factor r, in the selected templates of length $\mathbf{m}$ (resp. $\mathbf{m}+1$), denoted as $\tilde{\phi}^{\mathbf{m}}$ (resp. $\tilde{\phi}^{\mathbf{m}+1}$). This selection process is repeated N_1 times, and the average value, represented as $\bar{\phi}^{\mathbf{m}}_{N_1}$ (resp.

$\bar{\phi}^{\mathbf{m}+1}_{N_1}$), is then calculated. Replacing $\bar{A}_{N_1}$ and $\bar{B}_{N_1}$ by $\bar{\phi}^{\mathbf{m}}_{N_1}$ and $\bar{\phi}^{\mathbf{m}+1}_{N_1}$ in (5), respectively, we obtain an approximation of SampEn2D. This process is summarized in Algorithm 2, which calls Algorithm 1.

Algorithm 2 Two-dimensional Monte Carlo sample entropy (MCSampEn2D)

```
Require: Sequence $\mathbf{U} := \{u_{i,j} : 1 \le i \le w, 1 \le j \le h\}$, template length $m$, threshold $r$,
    Sample numbers $N_0$ and experimental rounds $N_1$.
 1: procedure MCSampEn2D($\mathbf{U}, m, r, N_0, N_1$)
 2:     Set $\bar{\phi}^{\mathbf{m}}_{N_1} = 0$ and $\bar{\phi}^{\mathbf{m}+1}_{N_1} = 0$,
 3:     for $k = 1$ to $N_1$ do
 4:         Set Cor $= \{(h_s, w_s) : 1 \le s \le N_0\}$ where $h_s$ and $w_s$ are selected on $\mathbf{U}$ pixel
    coordinates with uniform distribution,
 5:         Compute $\tilde{\phi}^{\mathbf{m}}_k$ by calling SampEn2D($\mathbf{U}$, Cor, $m, r$),
 6:         Compute $\tilde{\phi}^{\mathbf{m}+1}_k$ by calling SampEn2D($\mathbf{U}$, Cor, $m+1, r$),
 7:         $\bar{\phi}^{\mathbf{m}}_{N_1} = \bar{\phi}^{\mathbf{m}}_{N_1} + \frac{1}{N_1}\sum_{k=1}^{N_1}\tilde{\phi}^{\mathbf{m}}_k$,
 8:         $\bar{\phi}^{\mathbf{m}+1}_{N_1} = \bar{\phi}^{\mathbf{m}+1}_{N_1} + \frac{1}{N_1}\sum_{k=1}^{N_1}\tilde{\phi}^{\mathbf{m}+1}_k$
 9:     $entropy = -\log \frac{\bar{\phi}^{\mathbf{m}+1}_{N_1}}{\bar{\phi}^{\mathbf{m}}_{N_1}}$,
10:     return $entropy$
```

It is easy to check the computational cost of MCSampEn2D is $O(N_1 N_0^2)$ when $\mathbf{m}$ is fixed. Through a proof process similar to that of Theorem 5 in [17], we can see that the output of MCSampEn2D, $-\log \frac{\bar{\phi}^{\mathbf{m}+1}_{N_1}}{\bar{\phi}^{\mathbf{m}}_{N_1}}$, is approximating the output of SampEn2D with the rate $O(N_1^{-1}\log N_1)$ in the sense of almost sure convergence when N_0 is fixed. Furthermore, in our examination of MCSampEn2D, we observed variability in the significance of the randomly subsampled dataset. This is manifested as large fluctuations in the errors between the values of $\tilde{\phi}^{\mathbf{m}}$ (or $\tilde{\phi}^{\mathbf{m}+1}$) obtained in each of the N_1 rounds of experiments and their respective average values, $\bar{\phi}^{\mathbf{m}}_{N_1}$ (or $\bar{\phi}^{\mathbf{m}+1}_{N_1}$). Such fluctuations amplify the errors in the output of MCSampEn2D. This phenomenon made us realize that if we could capture the importance of different experimental rounds and use this importance to calculate a weighted average of $\tilde{\phi}^{\mathbf{m}}$ (or $\tilde{\phi}^{\mathbf{m}+1}$) obtained in the N_1 rounds of experiments, we could achieve a faster algorithm than MCSampEn2D.

2.3. Monte Carlo Sample Entropy Based on the UCB Strategy

The UCB strategy is a refinement in the field of optimization, particularly tailored for the intricate problem of the multi-armed bandit. This problem, often conceptualized as a machine or 'gambler' with multiple levers (or 'arms'), each offering random rewards on each interaction, demands finding the optimal lever that maximizes reward yield with minimal experimentation. The UCB strategy's core principle is to meticulously balance the pursuit of exploration and exploitation. Exploration, in this context, signifies the willingness to experiment with untested options, while exploitation underscores the preference to capitalize on the rewards offered by gamblers with a proven track record of high performance. This balancing act, inherent in the UCB strategy, aids in optimizing the overall reward yield by efficiently determining the optimal balance between exploring new options and exploiting known high-reward choices, thereby minimizing the number of trials required to reach the optimal solution [22,23].

In the preceding section, we presented the MCSampEn2D for estimating two-dimensional sample entropy. The precision of MCSampEn2D is contingent upon factors such as the number of experimental rounds, denoted as N_1, the quantity of sampled points, N_0, and the representativeness of these sampled points. Given that distinct experimental rounds involve the selection of different sampled points, their representativeness inherently varies. In Equation (5), the weight assigned to each round is statically set to $1/N_1$. To enhance accuracy, we can dynamically adjust the weights based on the representativeness of the sampled points in each round.

The adjustment of weights based on the representativeness of the sampled points serves to refine the estimation of sample entropy in each round, thereby enhancing the overall precision of the algorithm. This approach aims to ensure that the sampling process more accurately reflects the inherent characteristics of the dataset.

In the realm of decision-making under limited resources, the multi-armed bandit problem represents a classic challenge in reinforcement learning, involving a series of choices with the goal of maximizing cumulative rewards. The UCB strategy emerges as a crucial approach for tackling the multi-armed bandit problem. Its central concept revolves around dynamically assessing the potential value of each choice [24,25], seeking a balance between the exploration of unknown options and the exploitation of known ones.

At the core of the UCB strategy is the principle of making selections based on the upper confidence bounds. In this strategy, each arm represents a potential choice, and the true reward value of each arm remains unknown. In Algorithm 2, we refer to the process executed from step 5 to step 6 as one epoch. Under the UCB strategy, we conceptualize each epoch as an arm, which means that N_1 epochs represent N_1 arms, and dynamically update its upper confidence bound based on a designated reward function for each epoch. The UCB strategy involves the following three steps at each time step.

Firstly, the average reward for each round of epochs is calculated by

$$\hat{q}(i) = \frac{X(i)}{K(i)},$$

where $\hat{q}(i)$ signifies the average reward for the i-th round. In our design, each epoch is utilized only once per round, thus we set $K(i) = 1$ for all $i \in \{1, 2, \ldots, N_1\}$. Here, $X(i)$ represents the reward for the current epoch, estimated from historical data.

To design $X(i)$, we define the following notations. For all $1 \leq i \leq N_1$, let $\bar{r}_i$ be the average of the sample entropy of the preceding $i-1$ rounds, r_i, be the entropy computed for the ongoing round, and $e_i := \bar{r}_{i-1} - r_i$, where $\bar{r}_0 := 0$. Considering that different images inherently possess distinct entropies, we introduce the error ratio $er_i = \frac{e_i}{\bar{r}_{i-1}}$ to characterize the error situation for the i-th round. This ratio is then used as an input to the reward function $X(i)$. In formulating the reward function, our objective is to assign higher rewards to rounds where errors are closer to 0. Multiple choices exist for defining the reward function R, provided that it aligns with the design specifications of the particular scenario. Various mathematical functions, such as the cosine function and normal distribution function, among others, are viable options for constructing the reward function. The selection of a specific function is contingent upon its ability to meet the desired criteria and effectively capture the intended behavior in the context of the given problem. Then, we set the average reward $\hat{q}(i)$ for the i-th round of epochs formula as

$$\hat{q}(i) = X(i) := a \times R(b \times er_i), \tag{6}$$

where a is a scaling factor for the reward and b controls the scale of er_i.

Secondly, we calculate the upper confidence limit boundary for each round of epochs by

$$\mathrm{ucb}_i = \hat{q}(i) + c\sqrt{2\ln(i)}, \tag{7}$$

where ucb_i represents the upper confidence bound for the i-th round and set $K(i)$ as a constant equal to 1, we use a parameter c to control the degree of exploration. Denote $\tilde{U} := \{\mathrm{ucb}_i : i \in \{1, 2, \ldots, N_1\}\}$, which is the set of UCB.

Thirdly, for the set $\tilde{U}$, we employ the softmax function to determine the proportional weight each epoch round should have, thereby replacing the average proportion used in MCSampEn2D. We then calculate $\bar{A}_{\mathrm{Ucb}}$ and $\bar{B}_{\mathrm{Ucb}}$ for the UCBMCSampEn2D by

$$\bar{A}_{\tilde{U}} = \sum_{k=1}^{N_1} S_k \tilde{A}_k, \quad \bar{B}_{\tilde{U}} = \sum_{k=1}^{N_1} S_k \tilde{B}_k,$$

where $S = \mathrm{softmax}(\tilde{U})$ and the UCBMCSampEn can be calculated by

$$\mathrm{UCBMCSampEn} = -\log \frac{\bar{B}_{\tilde{U}}}{\bar{A}_{\tilde{U}}}.$$

The pseudocode for the entire UCBMCSampEn is outlined in Algorithm 3.

Algorithm 3 Monte Carlo sample entropy based on UCB strategy

```
Require: Sequence $\mathbf{U} := \{u_{i,j} : 1 \le i \le w, 1 \le j \le h\}$, template length $m$, threshold $r$,
    Sample numbers $N_0$ and epoch numbers $N_1$.
 1: procedure UCBMCSampEn($\mathbf{U}, m, r, N_0, N_1$)
 2:     Set $\bar{A}_{\tilde{U}} = 0$ and $\bar{B}_{\tilde{U}} = 0$,
 3:     for $k = 1$ to $N_1$ do
 4:         Set Cor $= \{(h_s, w_s) : 1 \le s \le N_0\}$ where $h_s$ and $w_s$ are selected on $\mathbf{U}$ pixel
    coordinates with uniform distribution,
 5:         Compute $\tilde{A}_k$ by calling SampEn2D($\mathbf{U}$, Cor, $m, r$),
 6:         Compute $\tilde{B}_k$ by calling SampEn2D($\mathbf{U}$, Cor, $m+1, r$),
 7:         Compute $e_k = -\log \frac{\tilde{B}_k}{\tilde{A}_k}$, $e_m = \frac{1}{k}\sum_1^k e_k$,
 8:         Compute $er_k = \frac{e_m - e_k}{e_m}$,
 9:         Compute $\mathrm{ucb}_k = a \times R(b \times er_k) + c\sqrt{2\ln(i)}$,
10:     Set $\tilde{U} = \{\mathrm{ucb}_k : k \in (1, 2, \ldots, N_0)\}$,
11:     Set $S = \mathrm{softmax}(\tilde{U})$,
12:     for $k = 1$ to $N_1$ do
13:         $\bar{A}_{\tilde{U}} = \bar{A}_{\tilde{U}} + \tilde{A}_k \times S_k$,
14:         $\bar{B}_{\tilde{U}} = \bar{B}_{\tilde{U}} + \tilde{B}_k \times S_k$,
        $entropy = -\log \frac{\bar{B}_{\tilde{U}}}{\bar{A}_{\tilde{U}}}$
15:     return $entropy$
```

Because the averaging strategy in the MCSampEn method does not consider the varying importance of sampling points across different epochs, it can result in significant errors. Although the UCB strategy introduces bias [26], it can mitigate the errors introduced by the averaging strategy, transforming the uniform strategy in the original method into an importance-based strategy. This adjustment aligns the sampling more closely with the actual characteristics of the data.

3. Experiments

This section is dedicated to thoroughly evaluating the effectiveness of the UCBMC-SampEn algorithm by implementing it across various domains. All of our experiments were carried out on the Linux platform. The platform utilizes an Intel(R) Xeon(R) Gold 6248R processor with a clock frequency of 3.00 GHz.

3.1. Datasets

To facilitate a thorough investigation, our experiments incorporate a range of sequences characterized by distinct features. These sequences are categorized primarily into datasets encompassing medical image data and natural image data. The medical image dataset, named Warwick QU dataset, is derived from the Colon Histology Images Challenge Contest for Gland Segmentation (GlaS), organized by MICCAI 2015, where participants developed algorithms for segmenting benign and diseased tissues [27]. It contains 165 samples extracted from H&E-stained colon histology slides. The slides were derived from 16 different patients, from which malignant and benign visual fields were extracted. The dataset example is illustrated in Figure 1.

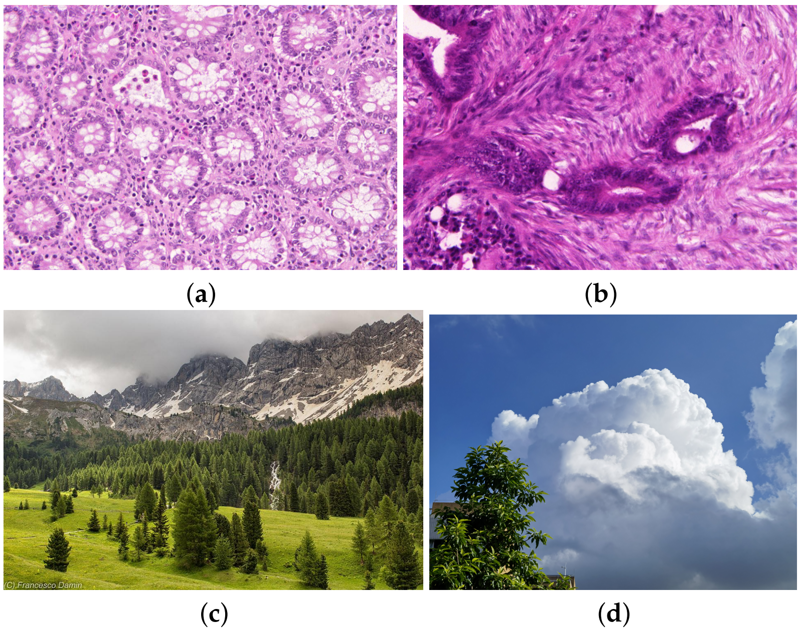

Figure 1. Examples of reference images include: images (**a**,**b**) in the Warwick QU Dataset, which represent benign and malignant cases, respectively, each with a size of 580 × 440; image (**c**) is a natural image with a size of 775 × 522; and image (**d**), called the wallpaper, with a size of 3000 × 3000, is used to verify the method's performance on large-scale data.

3.2. Main Results

In this section, we validate the effectiveness of the UCBMCSampEn2D, comparing its computational time and computational error with the MCSampEn2D algorithm. Figure 2 illustrates the variation in entropy mean error with the number of epochs N_1 using sampling points N_0 on the Warwick QU dataset and natural image dataset. The formula for calculating the mean error is as follows:

$$\text{MeanError} = \frac{\sum_{i=1}^{N} |d_i - e_i|}{W},$$

where W represents the number of images in the dataset, d_i is the entropy of the i-th image calculated using the direct algorithm, and e_i is the entropy of the i-th image calculated using the MCSampEn2D (or UCBMCSampEn2D) algorithm. The results demonstrate that the UCBMCSampEn2D algorithm converges more quickly, significantly reducing the error in comparison to the MCSampEn2D algorithm.

In Equation (6), we discussed that the reward function R offers flexibility, providing multiple choices to meet the design requirements of the scenario. In Figure 3, we conducted experiments using two different reward functions, and the results indicate that, with reasonable parameter settings, different reward functions R exhibit similar trends in average error changes. This suggests that the UCB strategy has a certain degree of generality and is not confined to specific forms. The remaining experiments were all conducted using the cosine function.

Figure 2 provides a detailed view of the situation with sampling points set at $N_0 = 128$. Based on the experimental results, it is evident that the UCBMCSampEn2D algorithm demonstrates more rapid convergence with an increase in the number of experiments (N_1)

compared to the MCSampEn2D. In Figure 2, with $N_0 = 128$, the UCBMCSampEn2D algorithm initially exhibits a larger error during the first 150 rounds of experiments. However, as the number of experiments increases, the UCBMCSampEn2D algorithm quickly reduces the error and achieves a lower convergence error than the MCSampEn2D. This substantial improvement in accuracy is consistently observed across various values of N_0.

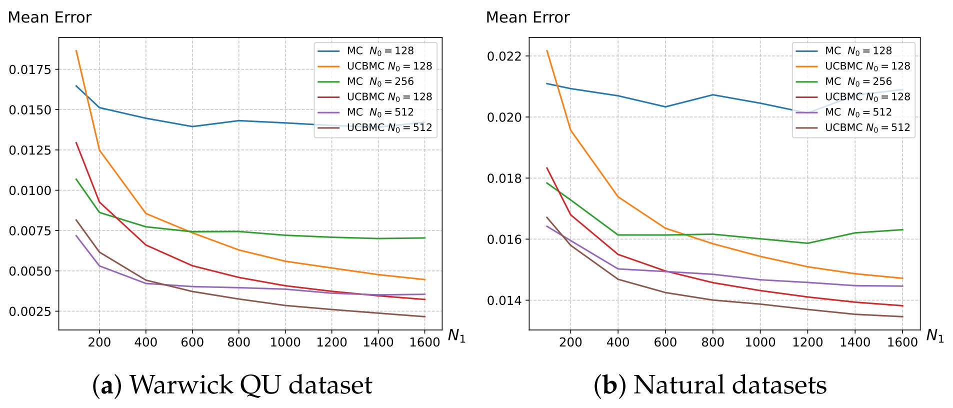

Figure 2. (**a**) depicts the average error variation in MCSampEn2D and UCBMCSampEn2D experiments on the Warwick QU dataset with changing N_1, where parameters are set to $m = 2$ and $r = 0.3$; (**b**) depicts the average error variation in MCSampEn2D and UCBMCSampEn2D experiments on natural datasets with changing N_1, where parameters are set to $m = 2$, $r = 0.3$, $a = 5$ and $b = 1$. The reward function is set as the cosine function.

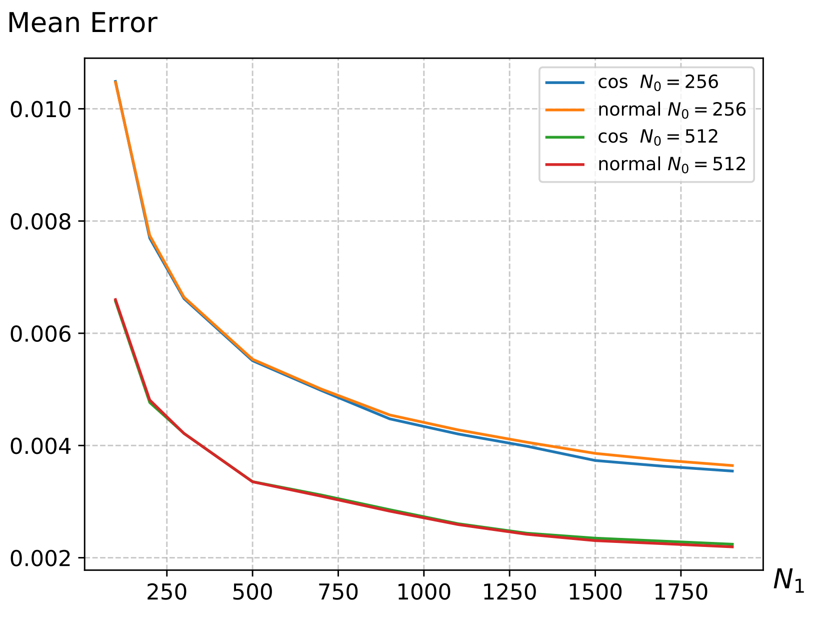

Figure 3. The average error variation in different reward function R with changing N_1 on the Warwick QU dataset, where parameters are set to $m = 2$ and $r = 0.3$. The parameters for the cosine function in the reward function are set as $a = 8$ and $b = 0.5$, while for the normal distribution function, the parameters are set as $a = 8$ and $b = 2$.

This phenomenon is elucidated in Section 2.3 of our algorithm, where the first i rounds of epochs are utilized to calculate the average entropy, simulating the true entropy. When i is small, there is not enough historical data to support it, the average entropy at this point introduces a relatively large error. However, since the entropy calculated from the previous i rounds is relatively close, the reward obtained for these initial rounds in (6) tends to be too high, leading to a larger error. As i increases, the average entropy more closely approximates the true entropy, and the weights assigned subsequently become more accurate in reflecting the real situation, thereby enabling the algorithm to converge more effectively.

Table 1 details the specifics of the UCBMCSampEn2D algorithm. Throughout the experiment, we maintained a consistent template length of $m = 2$ and a fixed similarity threshold of $r = 0.3$. Adjustments were made only to the number of sampling points, N_0, and the number of epochs, N_1.

Table 1. The mean error comparison among different algorithms under the same amount of time. The UCB parameters were set at $a = 8$ and $b = 1$.

Method	N_0/N_1	Mean Error (Proportion)	Mean Time (s)
SampEn2D	/	/	402.98
MCSampEn2D	128/1300	13.21×10^{-3} (0.906×10^{-3})	4.91
	256/1100	6.78×10^{-3} (0.465×10^{-3})	4.29
	512/900	3.69×10^{-3} (0.253×10^{-3})	4.87
UCBMCSampEn2D	128/1300	4.80×10^{-3} (0.329×10^{-3})	4.91
	256/1100	3.96×10^{-3} (0.271×10^{-3})	4.29
	512/900	2.78×10^{-3} (0.191×10^{-3})	4.87

In Table 2, under identical time constraints and with the same values of N_0 and N_1, the UCBMCSampEn2D algorithm demonstrates an error that is only 30% of the error observed with the MCSampEn2D algorithm when N_0 is small. Additionally, when N_0 is large, the UCBMCSampEn algorithm consistently outperforms the MCSampEn2D algorithm in terms of error reduction.

We can see that the MCSampEn2D algorithm has a significantly improved computation speed compared to the traditional SampEn2D, with an acceleration ratio exceeding a thousand-fold. Additionally, we conducted a time comparison between the UCBMCSampEn2D algorithm and the MCSampEn2D algorithm, setting the error below 5×10^{-3}. The UCBMCSampEn2D algorithm also demonstrated advantages, as shown in Table 2. In comparison to the MCSampEn2D algorithm, the UCBMCSampEn2D algorithm reduced the computation time by nearly 40%. Moreover, for larger-sized sequences, the UCBMCSampEn2D algorithm exhibited a significant advantage over the MCSampEn2D algorithm in terms of computation time and error.

Table 2. The comparison of time and error for image (d) in Figure 1 under different methods.

	SampEn2D	MCSampEn2D	UCBMCSampEn2D
time(s)	161,322	1.5816	0.9057
error	/	1.499×10^{-3}	0.442×10^{-3}

Simultaneously, we conducted numerical experiments on randomly generated binary images with a size of 512×512. The generation function MIX(p) [20] had a parameter $p = 0.9$. The results are shown in Table 3. Our method continues to demonstrate advantages even in the presence of high data randomness.

Table 3. The comparison of time and error for randomly generated binary images under different methods.

	SampEn2D	MCSampEn2D	UCBMCSampEn2D
time (s)	230.546	1.4046	0.9773
error	/	5.016×10^{-3}	3.362×10^{-3}

4. Discussion

4.1. Analysis of the UCB Strategy

In Section 2.2, we observed that the MCSampEn2D algorithm utilizes a random sampling method to select N_0 points and computes sample entropy across N_1 epoch numbers by averaging the outcomes. The chosen points' proximity to key image features affects their representativeness for the entire image, thereby influencing their relative importance. When sampled points accurately capture critical information, the error in that epoch number is reduced, resulting in sample entropy values that more closely approximate the true entropy. On the other hand, if the points fail to effectively capture information in the image, the resultant error in that round is magnified. The MCSampEn2D algorithm, which simply averages results across all rounds without weighing their importance, is adversely affected by these variations. This situation results in larger errors during convergence, particularly influenced by the number of sampled points N_0. Additionally, due to its inherent random sampling method, the standard deviation of the MCSampEn2D algorithm's results varies significantly with each epoch, leading to slower convergence and extended computation times.

We have addressed the MCSampEn2D algorithm's limitation in accurately reflecting the importance of epochs by integrating the UCB strategy. This strategy assigns significance to different epochs, thus modulating their individual impact on the final result. To compare the effectiveness of these approaches, we conducted 50 epochs each for MCSampEn2D and UCBMCSampEn2D using images from a natural dataset, specifically of size 3000×3000. We calculated the average and standard deviation of the error for the k-th round ($k \in \{1, 2, \ldots, N_1\}$). The results, displayed in Figures 4 and 5, reveal that the standard deviation of MCSampEn2D shows significant fluctuations across different rounds, while UCBMCSampEn2D maintains a more consistent performance. Furthermore, the error values for MCSampEn2D are consistently higher compared to those of UCBMCSampEn2D. This demonstrates that UCBMCSampEn2D not only achieves smaller errors than MCSampEn2D within the same timeframe but also effectively mitigates the issue of MCSampEn2D's inability to adequately capture the importance of each epoch.

Furthermore, we observed larger errors in the initial rounds of epochs with UCBMCSampEn2D. This can be attributed to the fact that the average entropy, serving as a temporary anchor, does not initially consider the historical context. Consequently, this leads to an overly high reward, $\hat{q}(i)$, in Equation (6) which, in turn, causes the confidence bound, ucb_i, in Equation (7) to be inconsistent. As a result, this inconsistency contributes to larger errors in the early rounds of epochs.

4.2. The Impact of Parameters on the UCB Strategy

In Section 2.3, where we introduced the formula for UCBMCSampEn2D, it was observed that the parameters a and b significantly impact the convergence speed and error of the algorithm. Since er_i in (6) reflects the proportion of bias, an unreasonable bias proportion could render the reward function ineffective. We conducted 50 experiments using wallpaper images and computed the standard deviation of the error for the k-th round ($k \in 1, 2, \ldots, N_1$), as shown in Figure 6. The results in Figure 6 demonstrate that the effectiveness of UCBMCSampEn2D is influenced by the parameters a and b. Appropriate selection of the values for a and b can reduce the standard deviation of the errors. Based on our experimental tests, we recommend using $a \in (7, 9)$ and $b \in (0.4, 0.6)$. However, adjustments may still be necessary based on the image size and the number of sampling points N_0.

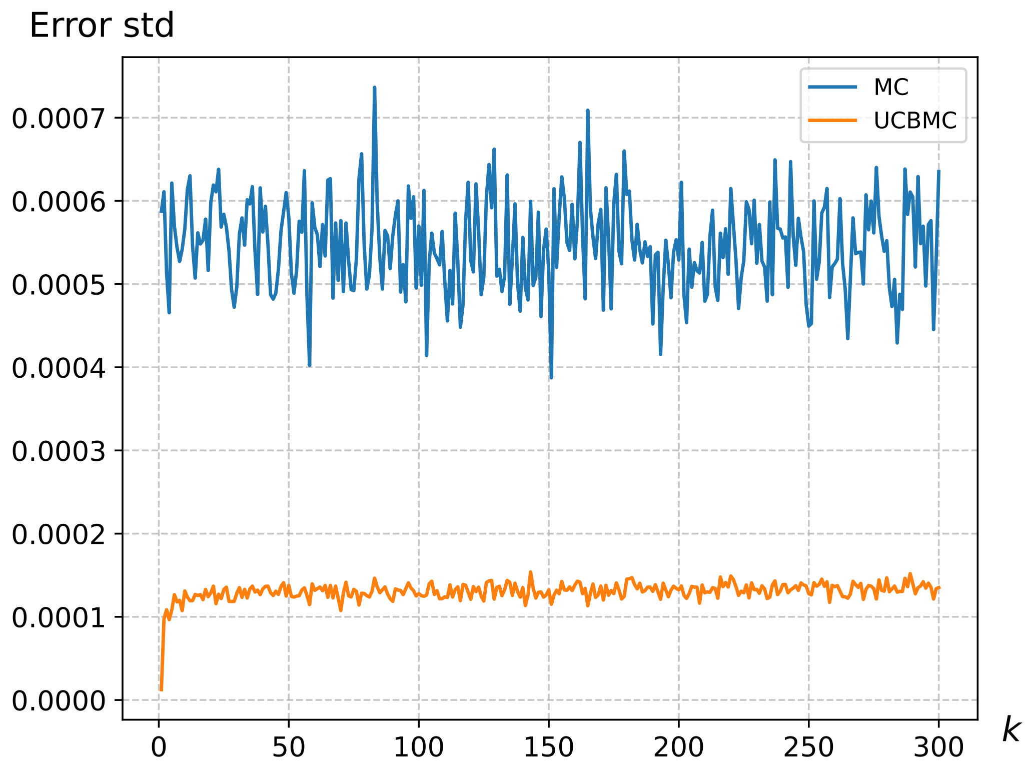

Figure 4. The error standard deviation variation for the wallpaper, where $N_0 = 128$, $N_1 = 300$, where the UCB parameters were set at $a = 8$ and $b = 1$.

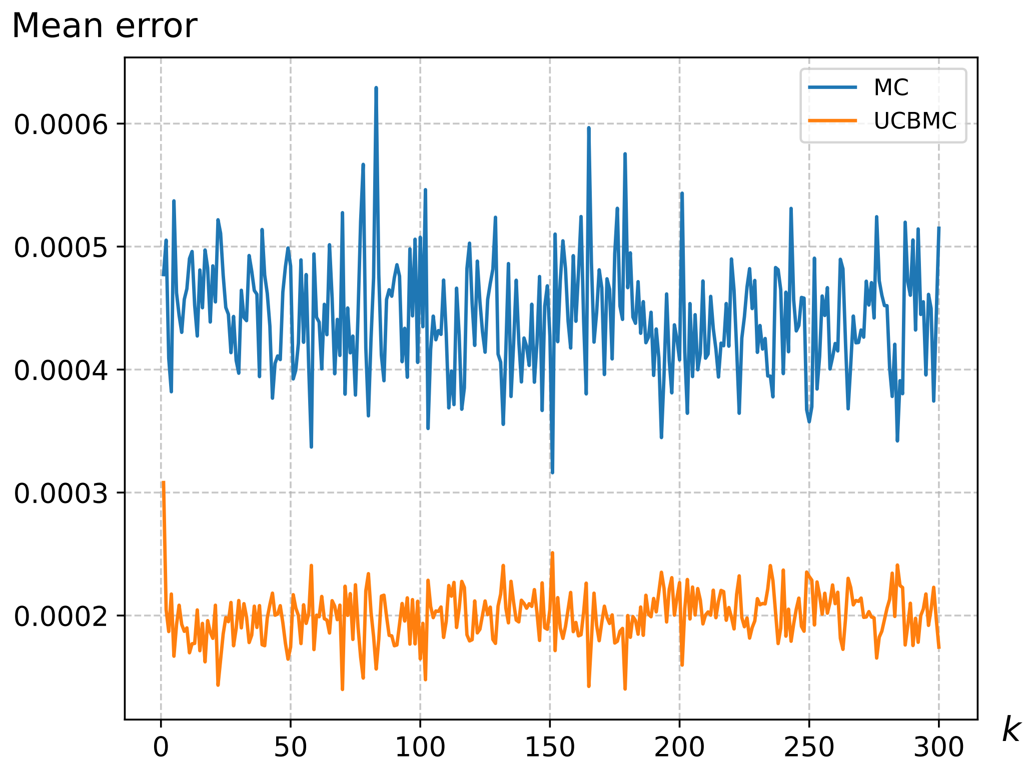

Figure 5. The mean error variation for the wallpaper, where the UCB parameters were set at $a = 8$ and $b = 1$.

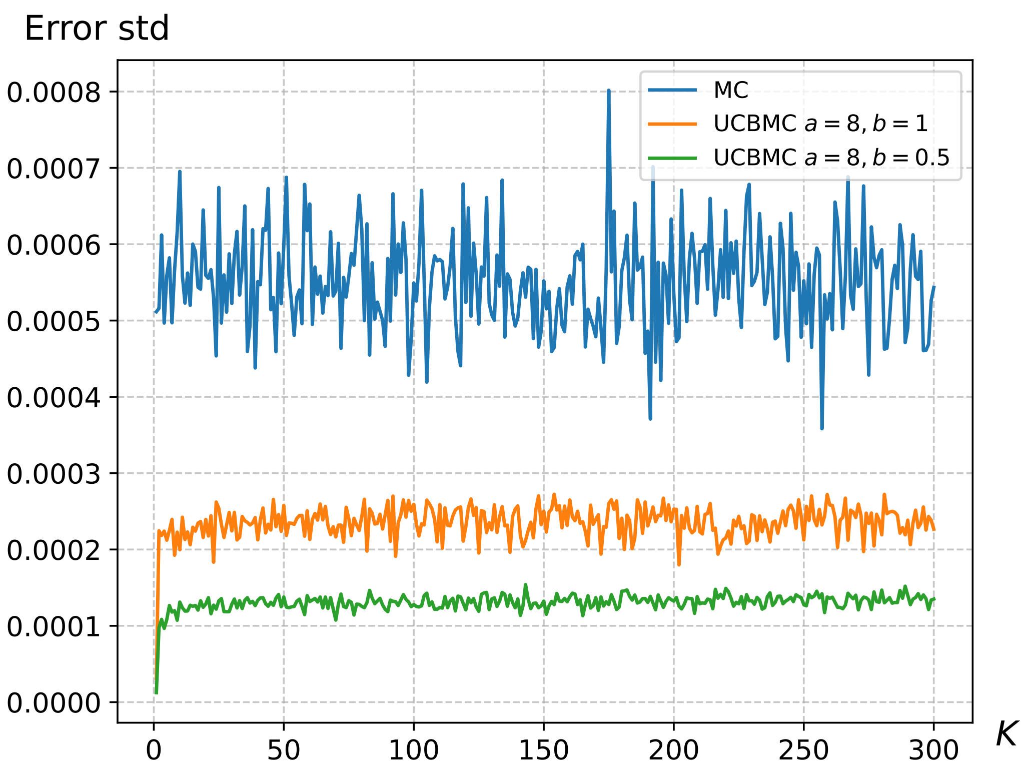

Figure 6. The error standard deviation variation for the wallpaper, where $N_0 = 128$ and $N_1 = 300$.

4.3. The Application of Sample Entropy in Medical Image Dataset

In the Warwick QU dataset, all pathological slices can be categorized into two classes: Benign and Malignant. We computed the entropy using the UCBMCSampEn2D algorithm for the dataset and utilized an SVM for training and classification, yielding the results presented in Table 4. It is evident that the entropy calculated by the UCBMCSampEn2D algorithm exhibits distinct trends for the two types of pathological slices, demonstrating potential in pathological slice diagnosis and positioning it as a viable feature for aiding future work in this field. The findings suggest that sample entropy can serve as a valuable supplementary characteristic in the context of pathological diagnosis.

Table 4. The UCBMCSampEn2D results for some different categories of pathological slices in Warwick QU dataset.

Image Name	UCBMCSampEn2D	Benign or Malignant
testB_1	0.317812	Benign
train_15	0.529283	Benign
train_47	0.672241	Benign
testA_17	0.762295	Benign
testA_24	1.06252	Benign
testA_57	2.37266	Malignant
testA_59	2.1973	Malignant
testA_8	2.36954	Malignant
testA_19	2.10283	Malignant
testB_7	2.07255	Malignant

5. Conclusions

This paper introduces two accelerated algorithms for estimating two dimensional sample entropies, termed MCSampEn2D and UCBMCSampEn2D. These algorithms were rigorously tested on both medical and natural datasets. The study's significance is manifold: firstly, the MCSampEn2D algorithm, an extension of the MCSampEn algorithm, substantially improves the computational efficiency for two-dimensional sample entropy. Further, we delve into the convergence challenges faced by the MCSampEn2D algorithm and adopt the UCB strategy to mitigate these issues. This strategy, as applied in our study, prioritizes the varying significance of different epochs, with its upper confidence bounds effectively mirroring this importance. The experiments detailed in Section 3 validate the efficacy of both the MCSampEn2D and UCBMCSampEn2D algorithms.

Overall, due to the UCBMCSampEn2D algorithm's impressive performance in computing sample entropy, it demonstrates considerable promise for analyzing diverse images while minimizing computational time and error.

Author Contributions: Conceptualization, Y.J.; methodology, Y.J. and Z.Z.; software, Z.Z. and W.L.; validation, Y.J. and Z.Z.; formal analysis, Y.J. and Z.Z.; investigation, Y.J.; writing—original draft preparation, Z.Z.; writing—review and editing, Y.J., Z.Z., R.W. and W.G.; visualization, Z.Z. and Z.L.; supervision, Y.J.; project administration, Y.J.; funding acquisition, Y.J. and W.G. All authors have read and agreed to the published version of the manuscript.

Funding: Project supported by the Key-Area Research and Development Program of Guangdong Province, China (No. 2021B0101190003), the Natural Science Foundation of Guangdong Province, China (No. 2022A1515010831) and the National Natural Science Foundation of China (No. 12101625).

Data Availability Statement: The numerical experiments in this work utilize the following datasets: Kaggle Dataset Landscape Pictures is available at https://www.kaggle.com/datasets/arnaud58/landscape-pictures; Warwick QU Dataset is available at https://warwick.ac.uk/fac/cross_fac/tia/data/glascontest/download; Wall paper is available at https://t.bilibili.com/894092636358443014.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shannon, C.E. A Mathematical Theory of Communication. *Assoc. Comput. Mach.* **2001**, *5*, 1559–1662. [CrossRef]
2. Richman, J.S.; Moorman, J.R. Physiological time-series analysis using approximate entropy and sample entropy. *Am. J. Physiol. Heart Circ. Physiol.* **2000**, *278*, H2039–H2049. [CrossRef] [PubMed]
3. Pincus, S.M. Approximate entropy as a measure of system complexity. *Proc. Natl. Acad. Sci. USA* **1991**, *88*, 2297–2301. [CrossRef] [PubMed]
4. Tomčala, J. New fast ApEn and SampEn entropy algorithms implementation and their application to supercomputer power consumption. *Entropy* **2020**, *22*, 863. [CrossRef] [PubMed]
5. Rostaghi, M.; Azami, H. Dispersion entropy: A measure for time-series analysis. *IEEE Signal Process. Lett.* **2016**, *23*, 610–614. [CrossRef]
6. Li, Y.; Li, G.; Yang, Y.; Liang, X.; Xu, M. A fault diagnosis scheme for planetary gearboxes using adaptive multi-scale morphology filter and modified hierarchical permutation entropy. *Mech. Syst. Signal Proc.* **2017**, *105*, 319–337. [CrossRef]
7. Yang, C.; Jia, M. Hierarchical multiscale permutation entropy-based feature extraction and fuzzy support tensor machine with pinball loss for bearing fault identification. *Mech. Syst. Signal Proc.* **2021**, *149*, 107182. [CrossRef]
8. Li, W.; Shen, X.; Yaan, L. A comparative study of multiscale sample entropy and hierarchical entropy and its application in feature extraction for ship-radiated noise. *Entropy* **2019**, *21*, 793. [CrossRef] [PubMed]
9. Aboy, M.; Cuesta-Frau, D.; Austin, D.; Mico-Tormos, P. Characterization of sample entropy in the context of biomedical signal analysis. In Proceedings of the 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Lyon, France, 22–26 August 2007; pp. 5942–5945.
10. Jiang, Y.; Mao, D.; Xu, Y. A fast algorithm for computing sample entropy. *Adv. Adapt. Data Anal.* **2011**, *3*, 167–186. [CrossRef]
11. Mao, D. Biological Time Series Classification via Reproducing Kernels and Sample Entropy. Ph.D. Thesis, Syracuse University, Syracuse, NY, USA, 2008.
12. Schreiber, T.; Grassberger, P. A simple noise-reduction method for real data. *Phys. Lett. A* **1991**, *160*, 411–418. [CrossRef]
13. Theiler, J. Efficient algorithm for estimating the correlation dimension from a set of discrete points. *Phys. Rev. A Gen. Phys.* **1987**, *36*, 4456–4462. [CrossRef] [PubMed]
14. Manis, G. Fast computation of approximate entropy. *Comput. Meth. Prog. Bio.* **2008**, *91*, 48–54. [CrossRef] [PubMed]
15. Manis, G.; Aktaruzzaman, M.; Sassi, R. Low computational cost for sample entropy. *Entropy* **2018**, *20*, 61. [CrossRef] [PubMed]
16. Wang, Y.H.; Chen, I.Y.; Chiueh, H.; Liang, S.F. A Low-Cost Implementation of Sample Entropy in Wearable Embedded Systems: An Example of Online Analysis for Sleep EEG. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 9312616. [CrossRef]
17. Liu, W.; Jiang, Y.; Xu, Y. A Super Fast Algorithm for Estimating Sample Entropy. *Entropy* **2022**, *24*, 524. [CrossRef] [PubMed]

18. Garivier, A.; Moulines, E. On upper-confidence bound policies for switching bandit problems. In Proceedings of the International Conference on Algorithmic Learning Theory, Espoo, Finland, 5–7 October 2011; pp. 174–188.
19. Anderson, T. Towards a theory of online learning. *Theory Pract. Online Learn.* **2004**, *2*, 109–119.
20. Silva, L.E.V.; Senra Filho, A.C.S.; Fazan, V.P.S.; Felipe, J.C.; Murta, L.O., Jr. Two-dimensional sample entropy: Assessing image texture through irregularity. *Biomed. Phys. Eng. Express* **2016**, *2*, 045002. [CrossRef]
21. da Silva, L.E.V.; da Silva Senra Filho, A.C.; Fazan, V.P.S.; Felipe, J.C.; Murta, L.O., Jr. Two-dimensional sample entropy analysis of rat sural nerve aging. In Proceedings of the 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Chicago, IL, USA, 26–30 August 2014; pp. 3345–3348.
22. Audibert, J.-Y.; Munos, R.; Szepesvári, C. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.* **2009**, *410*, 1876–1902. [CrossRef]
23. Zhou, D.; Li, L.; Gu, Q. Neural contextual bandits with ucb-based exploration. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 11492–11502.
24. Gupta, N.; Granmo, O.-C.; Agrawala, A. Thompson sampling for dynamic multi-armed bandits. In Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops, Honolulu, HI, USA, 18–21 December 2011; Volume 1, pp. 484–489.
25. Cheung, W.C.; Simchi-Levi, D.; Zhu, R. Learning to optimize under non-stationarity. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019; pp. 1079–1087.
26. Xu, M.; Qin, T.; Liu, T.-Y. Estimation bias in multi-armed bandit algorithms for search advertising. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013.
27. Sarwinda, D.; Paradisa, R.H.; Bustamam, A.; Anggia, P. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. *Procedia Comput. Sci.* **2021**, *179*, 423–431. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.