Evo-TFS：面向不平衡时间序列分类的时频域进化合成少数类过采样方法

裴文斌*，戴若豪*，薛冰，*IEEE*会士，张梦杰，*IEEE*会士，张强，张耀明，*IEEE*会士

arXiv:2601.01150v1 [cs.LG] 3 Jan 2026

摘要——时间序列分类是一项基础的机器学习任务，在现实世界中有着广泛的应用。尽管许多深度学习方法已被证明能够有效地学习用于分类的时间序列数据，但这些方法最初是在数据分布平衡的假设下开发的。一旦数据分布不均衡，这些方法往往会忽视少数类，而少数类通常具有更高的实际意义。为了解决这一问题，研究人员设计了过采样方法，通过生成少数类样本来改善数据分布；但这类方法依赖线性插值，往往难以保留时间动态特性，也难以生成多样化的样本。因此，本文提出了一种新颖的进化过采样方法 Evo-TFS，它融合了时域和频域特征。在 Evo-TFS 中，我们采用强类型遗传编程来进化多样且高质量的时间序列，并通过同时纳入时域和频域特征的适应度函数进行引导。在不平衡时间序列数据集上进行的实验表明，Evo-TFS 优于现有过采样方法，显著提升了时域和频域分类器的性能。

索引词—**不平衡时间序列分类、过采样、遗传编程和时频域。**

I. 引言

时间序列分类（TSC）已成为众多现实领域不可或缺的技术，包括金融风险评估 [1]，临床诊断 [2]，工业预测性维护 [3],和网络安全 [4]。深度学习模型，例如卷积神经网络（CNNs）和长短期记忆网络（LSTMs） [5]–[7]，已展现出强大的能力，能够捕捉时间依赖关系，并在复杂的时间序列数据上取得出色表现。然而，深度学习模型最初是在假设时间序列数据分布均衡的前提下开发的 [8]。遗憾的是，在许多现实场景中，例如医学诊断 [2]和欺诈检测 [

9],时间序列数据通常是不平衡的，某些类别（往往是重要类别）的样本数量显著少于其他类别，如图 1 (a) 所示。利用不平衡时间序列数据进行学习，对现有的时间序列分类深度学习方法构成了重大挑战 [10]。多数类样本占主导地位，导致这些模型的优化过度偏向多数类，往往以牺牲少数类的性能为代价。

解决类别不平衡的一种有效方法是通过采样技术重新平衡数据分布。其中，一种广泛使用的方法是基于插值的

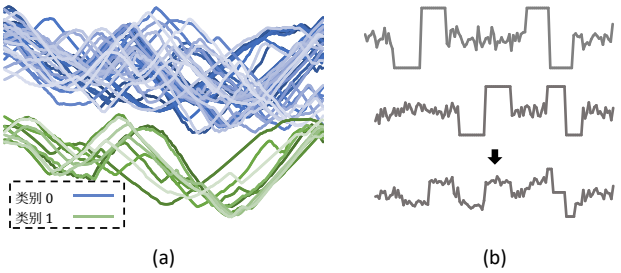


图 1. (a) 不平衡时间序列数据集的可视化，其中一类的样本数量显著较少；(b) 通过插值方法利用少数类样本生成样本。

过采样 [11]–[14], ，如图 1(b) 所示。需要注意的是，时间序列天然具有时间顺序，而这些插值方法通常仅关注点对点的值估计。这会破坏原有的时间模式和节律，导致合成样本无法准确反映数据真实的时间动态特性 [15]。为解决这一局限性，研究人员提出了基于时间的合成少数类过采样技术（T-SMOTE） [16]，该方法在数据生成过程中融入时间信息，以保留时间依赖关系。然而，尽管 T-SMOTE 强调时域特征，却往往忽略了在众多现实应用中至关重要的频域特征，包括医学诊断、语音处理和传感器数据分析。因此，有必要采用一种替代方法，以有效融合时域特征和频域特征，同时确保生成样本的多样性与结构完整性 [15]。

遗传编程（GP） [17]–[19] 是一种受自然选择启发的进化算法。它通过评估、选择和繁殖来进化程序，从而无需显式编程即可自动发现解决方案。通常，遗传编程个体以树结构表示，由多种可配置的函数节点和终端节点组成。这使遗传编程能够灵活地构建各种数学表达式，每个表达式都可视为生成的样本 [20]。遗传编程通过从现有数据中学习复杂变换，能够生成捕捉复杂时间模式的合成时间序列。此外，时间域信息和频域信息都可以纳入

适应度函数中，引导进化朝着更优的解决方案发展，同时保持原始数据的时间和频谱完整性。

然而，利用遗传编程生成时间序列的潜力尚未得到探索。因此，本文针对类别不平衡时间序列分类任务，提出一种新颖的进化式时频合成少数类过采样方法（Evo-TFS）。Evo-TFS 利用遗传编程为少数类生成多样且高质量的时间序列数据。在 Evo-TFS 中，每个遗传编程个体都表示一个待生成的时间序列样本。为增强生成时间序列的真实性，适应度函数纳入了时域和频域信息。这确保合成数据不仅保留时间依赖关系，也保留频谱特征。Evo-TFS 的主要贡献如下：

- 我们提出了一种新颖的个体表示，并专门设计了用于时间序列生成的终端集和函数集。该方法利用遗传编程自动且有效地合成时间序列。
- 我们设计了一种新的适应度函数，用于评估每个个体的优劣。在样本生成过程中，所提出的适应度函数纳入了时域和频域信息。这确保合成样本具有更好的时间序列特征。
- 我们提出了一种基于遗传编程的新型过采样方法，专门用于为类别不平衡的时间序列分类生成时间序列。实验结果表明，在生成时间序列方面，Evo-TFS 优于现有过采样技术，并显著提升了时域和频域分类器的性能。

II. 背景与相关工作

A. 类别不平衡时间序列分类

通常，解决类别不平衡问题的两种常用方法是代价敏感学习 [21], [22] 和采样技术 [23], [24]。

代价敏感学习通过明确考虑与不同类型分类错误相关的不同代价，促使学习模型区别对待不同的错误 [25]。不同于传统学习算法，代价敏感方法旨在最小化总误分类代价，而不仅仅是总体错误率 [26]。代价敏感学习可以通过修改训练数据分布或将代价矩阵纳入损失函数，提升模型在类别不平衡场景下的性能。迄今为止，许多传统分类算法已被改造成代价敏感版本，用于类别不平衡时间序列分类任务。例如，代价敏感卷积神经网络修改损失函数，以自适应地训练代价敏感模型，从而解决类别不平衡时间序列分类问题[27]。一种稳健的时间序列异常检测框架（RobustTAD） [28]将基于标签的权重和基于数值的权重同时引入损失函数，以调整类别标签的重要性以及每个样本邻域的贡献。然而，代价敏感学习方法存在若干局限性。首先，在实际应用中，即使是人类，甚至领域专家，也可能难以准确量化代价矩阵。此外，在纳入代价信息时，学习模型可能变得更加复杂；一旦代价信息发生变化或被错误识别，模型也难以实现良好的泛化。

与代价敏感学习不同，采样技术旨在重新平衡训练数据，以提升模型在类别不平衡场景中的性能。随机欠采样常因丢弃可能重要的样本而受到批评，而随机过采样则可能仅仅因为复制少数类样本而导致过拟合 [29]。为缓解这些问题，合成少数类过采样技术（SMOTE）通过在特征空间中的少数类样本之间进行线性插值来生成合成样本 [12]。其变体，例如不平衡学习自适应合成采样方法（ADASYN） [13]和 Borderline-SMOTE [30], , 通过专注于在少数类的特定区域生成合成样本，进一步提升生成样本的质量。对于时间序列数据，T-SMOTE [16] 专为在合成样本生成过程中保留时间连续性和依赖关系而设计，因此更适用于类别不平衡时间序列分类任务。然而，T-SMOTE 依赖于对现有少数类样本进行插值，因此当原始序列包含非线性或非平稳模式时，其效果可能较差。此外，T-SMOTE 根据局部邻域信息生成新样本，这限制了生成样本的多样性，并可能增加模型过拟合的风险。

B. 遗传编程

遗传编程是一种进化算法，已广泛应用于符号回归、分类、预测和数据生成任务 [20], [31], [32]。遗传编程的详细流程如图 2 所示。

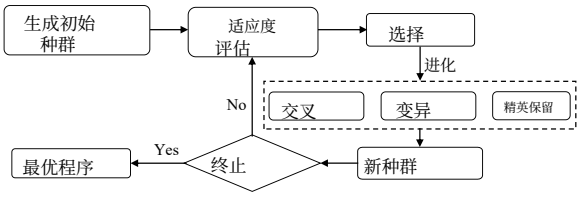


图 2. 遗传编程的流程图。

基于树的遗传编程是应用最广泛的版本。它只需要预先定义函数集、终端集和适应度函数。强类型遗传编程（STGP）通过引入数据类型约束、指定终端的输出类型，并要求函数的输入和输出类型相匹配，改进了标准遗传编程 [33], [34] 。这使 STGP 能够更有效地处理多种数据类型，同时缩小搜索空间。

在 STGP 中，使用类型约束初始化来初始化种群。类型保持交叉和类型约束变异是主要的遗传算子，旨在生成适应度更高、表现更好的后代，同时遵守类型约束。

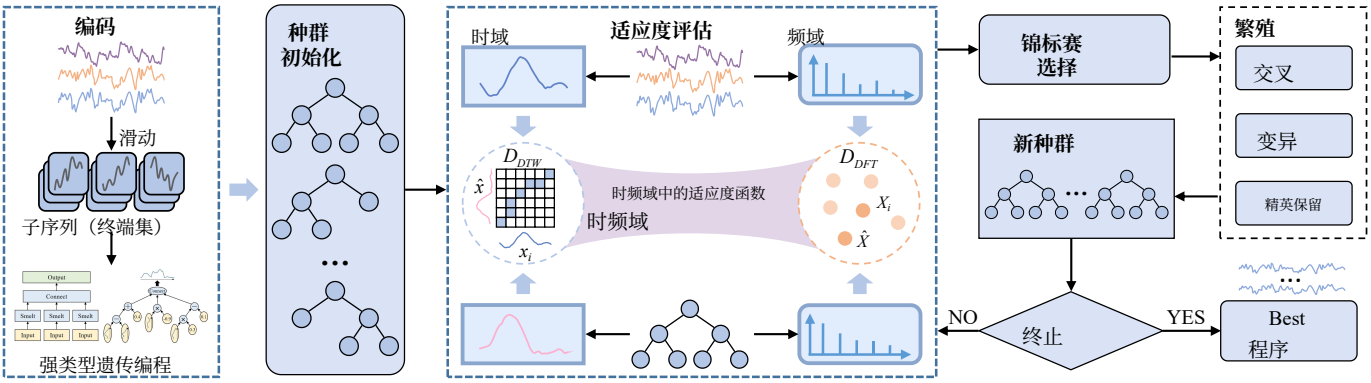


图3。Evo-TFS的框架。

- **类型约束初始化：** 从根节点开始构建树，选择返回类型与所需类型匹配的函数，并递归地为其参数选择类型匹配的函数或终端。这确保初始种群中的所有个体都遵循类型约束，因此都是有效的。

- **类型保持交叉：** 从父代 P_1 中随机选择一个子树，再从父代树P_2中随机选择另一个具有相同输出类型的子树。随后交换这两个子树以生成两个后代，从而确保类型一致性并避免类型冲突。

- **类型约束变异：** 从父代树中随机选择一个子树，然后随机生成一个具有相同输出类型的新树来替换该子树。

需要注意的是，一个遗传编程个体由可配置的函数节点和终端节点组成。这使得遗传编程具备灵活构建各种数学表达式的巨大潜力，而每个数学表达式都可视为一个生成的样本。通过从现有数据中学习复杂变换，遗传编程能够生成捕捉复杂时间模式的合成时间序列。然而，据我们所知，现有文献尚未探索使用遗传编程生成动态时间序列数据。因此，本研究旨在探究如何利用遗传编程为类别不平衡时间序列分类任务生成高质量的时间序列数据集。

III. 所提方法

本节介绍 Evo-TFS，这是一种基于时频域的进化过采样方法，用于类别不平衡时间序列分类任务。首先介绍 Evo-TFS 的整体框架。随后，详细说明专为生成时间序列数据而设计的遗传编程程序表示。接着阐述基于时频域的适应度函数。最后，全面说明 Evo-TFS 的整体流程。

A. Evo-TFS 的框架

Evo-TFS 的框架如图 3 所示。Evo-TFS 包含多个遗传编程过程，每个过程都分配给一个特定的

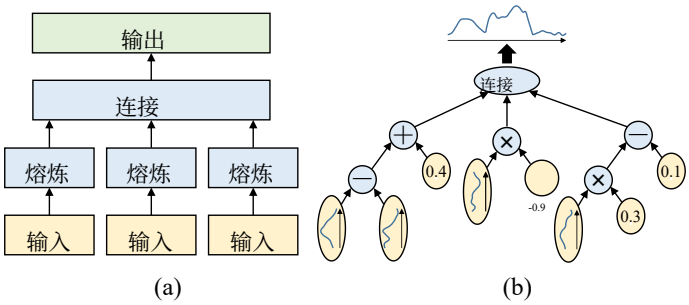


图4. (a) STGP 的结构；(b) 一个 STGP 程序示例。

目标时间序列样本，该样本从原始训练集中选取。在进化学习过程开始之前，使用滑动窗口对训练集中的所有时间序列样本进行分段，以提取可能的子序列，作为遗传编程的终端集。在每个进化过程中，适应度函数会结合该目标样本的时频特征，评估个体的质量。一旦满足终止准则，进化过程便会停止，并选取一部分高质量个体作为最终的合成样本。随后，将所有遗传编程过程生成的样本与原始训练数据集结合，形成用于训练分类器的最终训练集。

B. 树表示

我们设计了一种基于 STGP 生成时间序列数据的新表示。如图4所示，一个个体由四层组成： *Input*、 *Smelt*、 *Connect* 和 *Output*。 *Input* 层中的节点从终端集中选取，而 *Smelt* 层和 *Connect* 层中的节点从函数集中选取。 *Output* 层生成最终样本及其对应的标签。终端集和函数集介绍如下：

*1)*终端集： 在每个 GP 过程中，终端集包含从所有训练时间序列数据中使用滑动窗口提取的所有可能子序列，以及一个在 -1 到 1 范围内均匀分布的随机常数，如表 I 所示。

表 I 终端集和函数集

	Name	输入类型	
终端集	子序列	*Iput* (数组)	
	随机常数	*Iput* (浮点数)	
	Name	输入类型	输出类型
函数集	+	[*Iput*, *Iput*]	*A_Iput*(数组)
	−	[*Iput*, *Iput*]	*A_Iput*(数组)
	×	[*Iput*, *Iput*]	*A_Iput*(数组)
	÷	[*Iput*, *Iput*]	*A_Iput*(数组)
	连接	[*A_Iput*, *A_Iput*, *A_Iput*]	*Oput* (数组)

注：*Iput*、*A_Iput*和 *Oput* 表示进化树程序中的不同数据类型，其后括号中显示的数据类型表示它们的实际数据类型。

时间序列样本表示为 $x_i = \{v_1, v_2, \ldots, v_T\}$，其中 T 表示样本中的时间步数量。对于每个样本，使用滑动窗口方法提取子序列 $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,K}\}$，其中 K 表示子序列数量，且每个子序列的长度为 L，即 $s_{i,\ j} \in \mathbb{R}^L$。具体而言，对于样本 x_i，第 j 个子序列$S_{i,\ j} = \{v_j, v_{j+1}, \ldots, v_{j+L-1}\}$通过应用长度为 L 的滑动窗口获得，该窗口从位置 j 开始。

对于整个训练集，子序列集可表示为 $S = \{S_1, S_2, \ldots, S_n\}$，其中 n 是训练样本的数量。每个子序列集 S_i 包含 K 个子序列，每个子序列的长度为 L。因此，子序列集 S 的维度为 $S \in \mathbb{R}^{N\times K\times L}$，其中每个训练样本生成 K 个子序列，每个子序列的长度为 L。为便于后续处理，将子序列集合 S 的维度重塑为 $S \in \mathbb{R}^{(N\times K)\times L}$，从而有效地将所有样本中的子序列展平为一个新的集合，其中包含 $N\times K$ 个子序列，每个子序列的长度为 L。

2) 函数集：表 I 报告了函数集，其中包含五个函数，即加法（+）、减法（–）、乘法（×）、保护除法（÷）和 *Connect* 函数。请注意，当分母为 0 时，保护除法返回 1。四种算术函数（即 +、-、 ×和保护 ÷）均有两个参数，每个参数可以是浮点数，也可以是长度为 L 的数组。当两个参数都是数组时，算术函数会根据对应的逐元素运算输出一个数组。当一个参数为数组类型、另一个为浮点数时，算术函数会将该浮点数与数组的每个元素进行运算，并返回结果数组。*Connect* 函数有三个参数，每个参数都是来自 *Smelt* 层、类型为 *A Iput* 的子序列。该函数返回一个新的数组类型时间序列。

基于终端集和函数集，下面介绍 Evo-TFS 中遗传编程个体的各层:

- *Input* 层：该层的节点从终端集中选取，作为个体的输入。
- *Smelt* 层：该层可以选择 +、–、 × 的函数节点以及受保护的 ÷。该层的功能是构建子序列。
- *Connect* 层：此层使用 *Connect* 函数。该层的功能是将 *Smelt*层的输出进行拼接。

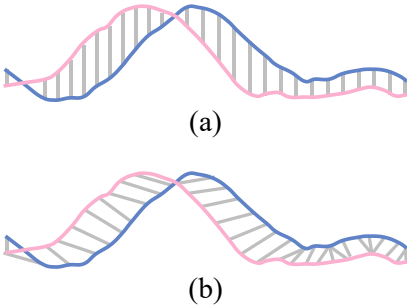


图 5. (a) 计算时间序列之间欧几里得距离的示意图；(b) 计算时间序列之间DTW距离的示意图。

- *Output*层：此层生成最终样本及其对应的标签。

C. 个体评估

在遗传编程中，适应度函数用于评估个体解决给定问题的质量，从而引导进化朝着更优解的方向发展。为了全面评估个体质量并确保其满足预期要求，我们设计了一种适应度函数，将时间序列数据的局部相似性与全局特征结合起来，从而兼顾时域和频域的信息。

1) 动态时间规整（***DTW***）距离：动态时间规整 [35] 是一种用于衡量两个时间序列相似度的方法，尤其适用于时间轴上存在非线性失真的情况。如图 5 (a) 所示，传统的欧几里得距离假设两个时间序列沿时间轴严格对齐，即只有在对应时间点一一匹配时才能评估相似度。因此，欧几里得距离对时间错位和局部失真 [36]非常敏感。结果是，即使两个序列之间只有轻微的时间偏移，也可能导致计算出的相似度出现显著偏差，使欧几里得距离在现实世界的时间序列比较中不够稳健和可靠。如图 5 (b) 所示，动态时间规整采用动态规划方法，寻找使累积总距离最小的最优对齐路径，从而适应非线性时间失真。

为了计算两个时间序列样本 $x = \{v_1, v_2, \ldots, v_m\}$ 和 $\hat{x} = \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n\}$之间的 DTW距离，我们首先构建一个代价矩阵 $D \in \mathbb{R}^{m\times n}$，其中每个元素定义如下:

$$D(i,j) = \|v_i - \hat{v}_j\|. \tag{1}$$

接下来，我们定义累积代价矩阵 $C \in \mathbb{R}^{m\times n}$，其中第一个元素初始化为 $C(1,\ 1) = D(1,\ 1)$。对于 $i \geq 2$ 和 $j \geq 2$，$C(i,\ j)$ 的递推关系为:

$$C(i,j) = D(i,j) + \min \begin{cases} C(i-1,j), \\ C(i,j-1), \\ C(i-1,j-1). \end{cases} \tag{2}$$

x与 $\hat{x}$之间的 DTW距离定义为:

$$D_{\text{DTW}}(x, \hat{x}) = C(m, n). \tag{3}$$

在本研究中，生成的样本 $\hat{x}$ 与目标样本 x在长度上严格对齐，确保 $m = n$。

在 Evo-TFS 中使用动态时间规整（DTW），可以灵活地进行逐点对齐，从而捕捉生成样本与目标样本之间细微的时间差异。与欧几里得距离或其他相似性度量相比，DTW距离对时间序列数据中的局部失真和时间偏移更加稳健。这一能力使 DTW 能够有效对齐存在时间错位的序列，并更准确地衡量样本之间的相似性。特别是，DTW 能够识别不同时间步之间的变化模式并捕捉局部相似性，因此非常适合处理真实世界时间序列数据的复杂分布。

2)傅里叶距离：离散傅里叶变换（DFT）[37] 是分析有限长度离散时间序列频率分量的重要数学工具。通过使用 DFT 将时域样本转换为频域表示，可以有效识别样本所包含的频率分量。具体而言，如图 6 所示，DFT 将样本分解为一组正弦或余弦分量，每个分量都具有特定的频率、振幅和相位。

对于长度为 T 的离散时间序列 x，离散傅里叶变换（DFT）会将样本转换为其频域表示 $X[k]$，其中 $k = 0$，1，2，……，$T - 1$ 表示不同的频率分量。DFT 的公式如下 [37]:

$$X[k] = \sum_{n=0}^{T-1} x(n) e^{-j\frac{2\pi}{T}kn}, \quad k = 0, 1, \ldots, T-1 \tag{4}$$

其中:

- $x(n)$ 是原始时域样本的第 n 个时间点,
- $X[k]$ 是样本在频域中第 k 个频率分量的复数表示,
- $e^{-j\frac{2\pi}{T}kn}$ 是复指数函数，用于表示正弦和余弦分量的频率响应。

傅里叶距离通常用于比较两个时间序列在频域中的相似度。计算两个时间序列 x 和 $\hat{x}$ 之间距离的公式如下 [38]:

$$D_{DFT}(x, \hat{x}) = \sqrt{\sum_{k=0}^{T-1} (|X[k]| - |\hat{X}[k]|)^2 + \sum_{k=0}^{T-1} (\arg(X[k]) - \arg(\hat{X}[k]))^2}, \tag{5}$$

其中:

- $X[k]$ 和$\hat{}$ $X[k]$分别是 x 和$\hat{x}$的频域值；$\hat{}$
- $|X[k]|$ 和 $|X[k]|$分别是频率点 k处样本 x和$\hat{x}$的幅值；$\hat{}$
- arg($X[k]$)和arg($X[k]$)分别是频率点 k处样本x和$\hat{x}$的相位。

在 Evo-TFS 中使用傅里叶距离，可以捕捉生成样本与目标样本之间的频谱差异。

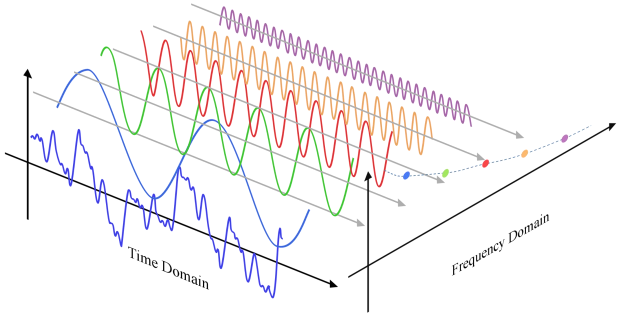


图6. 离散傅里叶变换示意图。

总傅里叶距离通过同时考虑样本之间的幅度谱和相位谱差异来衡量频域相似性，尤其适用于时间序列呈现周期性或准周期性变化的情况。

3) 适应度函数：基于动态时间规整和傅里叶距离指标，适应度函数定义如下:

$$Fitness = \alpha Q(D_{\text{DTW}}(x, \hat{x})) + (1 - \alpha) Q(D_{\text{DFT}}(x, \hat{x})), \tag{6}$$

$$Q(x) = e^{\frac{-x^2}{2\sigma^2}}, \tag{7}$$

其中，x 表示目标样本，$\hat{x}$ 表示生成的样本，$\alpha \in [0, 1]$ 是用于平衡时域距离和频域距离重要性的加权参数。D_{DTW} 表示生成样本与目标样本之间的DTW距离，而D_{DFT} 表示生成样本与目标样本之间的总傅里叶距离。最后，$Q(x)$表示高斯函数的部分函数，用于对 D_{DTW} 和 D_{DFT} 进行归一化，使其处于相同范围内。

适应度函数整合了DTW和傅里叶距离指标，从而更全面地评估生成样本与目标样本之间的相似性。DTW距离能够有效捕捉时间窗口内的局部变化，尤其适用于存在时间轴畸变的情况。这提供了一种灵活的匹配方法，确保生成的时间序列样本在局部特征层面与目标样本保持一致。另一方面，傅里叶距离在频域中运行，揭示时间序列数据中频率分量的变化，并有助于捕捉全局结构特征。通过结合这两种距离指标，适应度函数能够同时捕捉时间序列中的局部波动并保留其整体模式，从而增强生成样本的多样性、合理性和代表性。

D. 进化过程

Evo-TFS 采用多个相互独立的遗传编程过程，每个过程使用一个少数类目标时间序列样本，并在若干代中进化一组候选解种群。这使 Evo-TFS 能够利用多样化的现有少数类时间序列数据生成多样化的样本

来生成新样本。在每个进化过程中，适应度函数会评估个体的质量，判断其时频特征与目标时间序列样本的时频特征有多接近。

Evo-TFS 中遗传编程过程的数量定义如下：

$$N_p = \sum_{i \in C} \min(N_{gi}, N_i), \tag{8}$$

其中，C 表示标签集合，N_i 表示类别 i中的样本数量，N_{gi} 表示少数类 i需要生成的样本数量。

不平衡比率（IR）可用于计算少数类[20]需要生成的实例数量。令 N_{pi} 表示用于为类别 i生成样本的过程数量。

- 如果 $IR = 2$，则 $N_{pi} = N_{gj}$。在这种情况下，所有现有样本都用作目标样本，并分别分配给一个遗传编程过程，由该过程返回最优个体作为生成样本。
- 如果 $IR > 2$，则所有现有样本都用作目标样本，并分别分配给一个遗传编程过程，该过程需要返回排名前 $\frac{N_{gi}}{N_{pi}}$ 的最优解作为生成样本。
- 如果 $IR < 2$，则 $N_{pi} < N_i$。在这种情况下，选择最接近类中心（稍后将详细介绍）的 N_{pi} 个样本作为目标样本，并将每个样本分配给一个独立的遗传编程过程，最终返回适应度最优的个体作为生成样本。

某一类别的类中心定义为属于同一类别的所有时间序列在各元素上的均值。对于类别 i，类中心（记为 X_c）计算如下：

$$X_c(t) = \frac{1}{N_i} \sum_{j=1}^{N_i} x_j(t), \quad \forall t = 1, \ldots, T \tag{9}$$

其中，$x_j(t)$ 表示 x_j 的第 t 个时间点，T 表示一个样本中的时间步数。

类中心的详细计算步骤如下。首先，我们对时间序列数据应用最小-最大归一化，以统一所有样本的尺度。随后，由于随机波动和高频噪声在逐元素求和过程中往往会相互抵消，因此得到的均值向量可作为稳定且平滑的原型。

请注意，所提出的方法并不局限于周期性时间序列，因为它不依赖于任何关于周期性或循环结构的先验假设。我们所提出方法采用的分割策略，只是使用固定长度的滑动窗口将每个时间序列数据划分为子序列，并未尝试与任何特定周期进行对齐。请注意，对于非周期性数据，将其划分为子序列不会影响生成样本的质量。这是因为每个独立的遗传编程过程都维护着一个候选解种群（即生成样本），其中每个候选解都会根据适应度函数进行评估，以表明其质量，即生成样本是否能有效匹配目标时间序列样本的时频特征。适应度值较低的生成实例表明其质量较差，在进化过程中不太可能存活。

本文专门关注时间序列分类中的类别不平衡问题。将所提出的方法应用于含有缺失值的数据集时，必须将适当的插补技术作为预处理步骤。所提出的方法在应用于包含异常的不平衡数据集时具有内在的鲁棒性，原因如下。首先，该方法在生成合成实例时会明确优先选择最接近类中心的样本。其次，遗传编程能够自动确定训练集中某个样本的哪些子序列将用于生成新样本。如果遗传编程生成的合成样本是基于离群值或噪声数据的子序列构造的，那么它很可能获得较低的适应度分数。因此，该样本被选中存活的概率会大幅降低，从而在内在上减轻离群值或噪声数据对生成数据整体质量造成的负面影响。

IV. 实验设计

为了验证 Evo-TFS 的有效性，我们在十个时间序列数据集上开展了大量实验，并与六种具有代表性的采样方法进行了全面比较分析。

A. 数据集

在实验中，我们使用了来自 UCR档案库 的 12 个时间序列数据集[1]，，详细信息见表 II（DistalPhalanxOutlineAgeGroup 简写为 DistalP-AgeGrp，MiddlePhalanxOutlineCorrect 简写为 MiddleP-Corr）。这些数据集涵盖了多个领域，包括医疗保健、地球科学、工业检测、生物学、能源管理和动作识别。由于所有数据集都已划分为训练集和测试集，我们保留了原始划分。按照 [16]，我们对所选数据集的训练集进行分层抽样，以生成具有不同 IR的新数据集。所得 IR范围为 2.00 至 38.80，涵盖了广泛的不平衡程度。在抽样得到的数据集中，多数类标记为**多数类**，少数类标记为**少数类**。

表 II 实验中的数据集

数据集	样本数量	类别	IR	长度	索引
ShapeletSim	15	2	2.00	500	D1
ECG200	100	2	2.23	96	D2
Ham	76	2	3.00	431	D3
Herring	52	2	3.00	512	D4
WormsTwoClass	135	2	3.50	270	D5
ItalyPowerDemand	41	2	4.86	24	D6
DistalP-AgeGrp	317	3	8.57	80	D7
PowerCons	99	2	10.00	144	D8
屏幕类型	149	3	10.42	720	D9
Yoga	147	2	13.70	426	D10
计算机	131	2	20.83	720	D11
MiddleP-Corr	398	2	38.80	80	D12

[1]这些数据集可在以下网址获取：https://www.cs.ucr.edu/~eamonn/timeseries_data/ _

B. 基线方法

基线方法描述如下：

- **SMOTE [12]：** 这是最常用的过采样方法，通过在一个**少数类** 样本及其邻居之间进行插值来合成新样本，从而平衡数据。
- **ADASYN [13]：** 与 SMOTE 不同，它优先处理**少数类**中难以学习的样本，以生成更多合成样本来平衡数据。
- **Borderline-SMOTE1（简称 B1SMOTE）[30]**：它通过对**少数类**中的边界样本（标记为 $danger$ 样本）与其同类（少数类）邻居进行插值来合成新样本。
- **Borderline-SMOTE2（简称 B2SMOTE）[30]**：与 Borderline-SMOTE1 相同，它基于**少数类** 中的边界样本及其同类邻居合成新样本。不过，在某些情况下，它还会在插值时考虑来自**多数类**的邻居，从而生成通常更靠近**少数类**、但位于决策边界区域内的合成样本。
- **T-SMOTE [16]**：该方法在合成样本的生成过程中融入时间信息，以确保生成数据保留其时间依赖性。它还会根据每个**少数类**样本的重要性赋予不同权重，从而生成新样本。
- **Diffusion-TS [39]**：这是一种新颖的基于扩散的方法，利用具有解耦时间表示的编码器-解码器 Transformer 生成多变量时间序列样本。在分解技术的引导下，该方法捕获数据的整体语义，同时由 Transformer 从带噪输入中提取细粒度的序列模式。

C. 参数设置

表 III 展示了所提出的 Evo-TFS 方法的参数配置。Evo-TFS 通过 N_p 个遗传编程过程生成 N_g 个合成样本，每个过程生成 $Max(1, \frac{N_g}{N_p})$ 个样本。为平衡模型性能与计算效率之间的权衡，我们根据 IR 为特定数据集设置种群规模。如果满足 $IR < 15$，则每个遗传编程过程中的种群规模设为 30；否则设为 50。世代数设为 50，以确保进行足够次数的评估。在每个遗传编程过程中，个体采用递增半数-半数方法进行初始化。对于遗传算子的配置，交叉率和变异率分别设为 80% 和 20%[40], 。采用锦标赛选择，锦标赛规模为 3。为避免学习过程中出现树膨胀，最大树深度设为 10。对于 Evo-TFS 中涉及的高斯函数 $Q(x)$，标准差参数 σ 设为 10。

Evo-TFS 使用 DEAP 包 [41],实现，基线采样方法则使用 Imbalanced-learn 包 [42]实现。为确保实验间的公平性，本研究采用了一个统一的评估框架，具体如下。首先，使用由两种方法生成的重平衡训练集训练分类器，

表 III 参数设置

参数	值
世代数	50
锦标赛规模	3
交叉率	80%
变异率	20%
精英个体数	2
初始化	递增半与半初始化法
最大树深度	10
α	0.5

这两种方法分别是 Evo-TFS 和基线方法，随后在相应的测试集上评估性能。

具体而言，我们采用两个分类器来评估和分析时间序列数据：

(1) 鉴于卷积神经网络在时间序列分类任务中的优异表现，我们使用卷积神经网络学习时域特征 [5]。需要注意的是，我们研究中使用的卷积神经网络模型是一维卷积神经网络（1D-CNN） [43], , 专为时间序列数据处理而设计。1D-CNN 的架构利用一维卷积核沿时间轴滑动，从而有效捕捉局部时间模式和趋势。

(2) 为了进一步研究数据采样对频域分类的影响，我们提出一种结合傅里叶变换和随机森林（RF）的方法，以学习频域特征 [44]。

最后，我们提出一个指标，用于评估重平衡数据集中不同类别之间的密度一致性，其定义如下：

$$U = |\frac{\sum_{i \in Maj} \sum_{j \in n_i^k} D_{DTW}(i,j)}{k \times |Maj|} - \frac{\sum_{i \in Min} \sum_{j \in n_i^k} D_{DTW}(i,j)}{k \times |Min|}|, \tag{10}$$

其中 n_i^k 表示样本 i 的 k 个最近邻。 U 的值范围为 0 到 $+\infty$，值越小表示重平衡后多数类与少数类之间的分布越均匀。

V. 结果与分析

在每个数据集上，Evo-TFS 和基线方法（Diffusion-TS 除外）使用不同的随机种子各自独立运行 30 次。使用不同采样方法重新平衡后的训练集训练两个分类器，并在同一个测试集上进行测试。采用 F1 分数 [45], [46], 几何平均数 [11] 和曲线下面积 [47] 指标评估性能（曲线下面积的详细结果见**附录D**）。为评估 Evo-TFS 与某个基线方法之间的性能差异是否具有统计显著性 [49]，采用显著性水平为 0.05 的 Wilcoxon符号秩检验，并针对多重比较进行 Holm–Bonferroni校正 [48] 。

表 IV：12 个 UCR 数据集上时域中使用卷积神经网络的采样方法结果。最优结果以**粗体**显示，次优结果带下划线

指标	方法	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
F1 分数	SMOTE	0.110±0.181 (+)	0.852±0.019 (=)	0.704±0.015 (+)	0.461±0.077 (+)	0.680±0.020(=)		0.878±0.007 (+)	0.642±0.023(+)	0.852±0.046(+)	0.252±0.022(+)	0.355±0.023 (-)	0.106±0.028(+) 0.767±0.005(+)
	ADASYN	0.048±0.106 (+)	0.838±0.018 (+)	0.711±0.019(+)	0.420±0.061 (+)	0.694±0.017(=)		0.875±0.007 (+)	0.641±0.023 (+)	0.774±0.042 (+)	0.251±0.031(+)	0.259±0.018 (-)	0.099±0.025(+) 0.777±0.007 (=)
	B1SMOTE	0.125±0.228 (+)	0.834±0.013 (+)	0.698±0.029(+)	0.448±0.085 (+)	0.695±0.024(=)		0.876±0.007 (+)	0.649±0.030 (+)	0.805±0.047 (+)	0.311±0.028(+)	0.191±0.029 (=)	0.138±0.036(=) **0.787±0.009 (-)**
	B2SMOTE	0.048±0.1006(+)	0.821±0.015(+)	0.710±0.023(+)	0.454±0.081 (+)	0.687±0.017 (=)		0.875±0.007 (+)	0.650±0.025(+)	0.765±0.040(+)	0.241±0.017(+)	**0.360±0.026 (-)**	0.122±0.026 (+) 0.773±0.009 (+)
	T-SMOTE	0.249±0.185 (+)	0.843±0.014 (+)	0.777±0.027 (-)	0.312±0.056 (+)	0.697±0.035(=)	0.867±0.0104 (+)	0.588±0.042 (+)	**0.925±0.026 (=)**	0.317±0.023 (+)	0.255±0.027 (-)	0.117±0.036 (+)	0.727±0.001 (+)
	Diffusion-TS	0.261 (+)	0.839 (+)	**0.803 (-)**	0.495 (+)	**0.727 (-)**	0.907 (=)	0.668 (+)	0.883 (=)	0.349 (=)	0.229 (=)	0.145 (=)	0.727 (+)
	Evo-TFS	**0.390±0.220**	**0.857±0.024**	0.753±0.041	**0.563±0.074**	0.693±0.025	**0.910±0.018**	**0.699±0.042**	0.902±0.051	**0.351±0.028**	0.208±0.063	**0.158±0.054**	0.780±0.008
	总计	49 +, 15 =, 8 -											
几何平均数	SMOTE	0.136±0.160 (+)	0.804±0.021 (=)	0.649±0.031 (+)	0.551±0.060 (+)	0.496±0.036 (=)		0.885±0.006 (+)	0.769±0.016 (=)	0.862±0.041 (+)	0.299±0.034 (+)	0.461±0.017 (-)	0.234±0.036 (+) 0.508±0.033 (+)
	ADASYN	0.089±0.130 (+)	0.805±0.022 (=)	0.652±0.029(+)	0.519±0.048 (+)	0.524±0.033(=)		0.881±0.006 (+)	0.770±0.013 (=)	0.795±0.036 (+)	0.299±0.049(+)	0.382±0.015 (-)	0.226±0.031(+) 0.523±0.030 (+)
	B1SMOTE	0.079±0.150 (+)	0.808±0.018 (=)	0.653±0.040 (+)	0.538±0.065(+)	0.529±0.036 (-)		0.883±0.007 (+)	0.774±0.017 (=)	0.821±0.040 (+)	0.387±0.037 (+)	0.323±0.027 (=)	0.270±0.039 (=) **0.568±0.036** (=)
	B2SMOTE	0.094±0.137 (+)	0.794±0.027 (+)	0.651±0.026 (+)	0.545±0.063 (+)	0.526±0.032 (=)		0.882±0.006 (+)	0.773±0.016 (=)	0.788±0.034 (+)	0.281±0.027(+)	**0.465±0.020 (-)**	0.254±0.030(+) 0.530±0.037 (=)
	T-SMOTE	**0.323±0.170 (=)**	0.801±0.017(=)	0.734±0.040 (=)	0.436±0.048 (+)	**0.535±0.034 (=)**		0.875±0.009 (+)	0.721±0.028 (+)	0.897±0.024 (=)	0.400±0.031 (+)	0.380±0.022(-)	0.246±0.041 (=) 0.510±0.015 (+)
	Diffusion-TS	0.295 (=)	0.795 (+)	**0.755 (=)**	0.581 (+)	0.528 (=)		0.911 (=)	0.768 (=)	0.904 (=)	0.403 (+)	0.342 (=)	0.265 (=) 0.545 (=)
	Evo-TFS	0.315±0.183	**0.816±0.028**	0.745±0.047	**0.626±0.061**	0.503±0.045		**0.914±0.017**	**0.775±0.030**	**0.907±0.046**	**0.443±0.041**	0.336±0.055	**0.288±0.058** 0.562±0.085
	总计	38 +, 29 =, 5 -											

表 V 随机森林在频域上对 12 个 UCR 数据集使用采样方法的结果。最优结果以**粗体**显示，次优结果带下划线

指标	方法	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
F1 分数	SMOTE	0.492±0.057(+)	0.808±0.010(+)	0.212±0.059(+)	0.547±0.014 (=)	0.728±0.052 (-)	**0.749±0.020 (-)**	0.641±0.013 (=)	0.509±0.009 (=)	0.174±0.005 (-)	0.300±0.001 (-)	0.348±3E-04 (+)	0.734±0.003 (+)
	ADASYN	0.492±0.057 (+)	0.798±0.011 (+)	0.212±0.059 (+)	**0.550±0.015 (=)**	0.729±0.050 (-)	0.706±0.004 (-)	**0.654±0.013 (=)**	0.435±0.004 (+)	0.174±0.006 (-)	0.300±6E-04(-)	0.348±5E-04 (+)	0.737±0.002 (+)
	B1SMOTE	0.514±0.061(+)	0.787±0.009(+)	0.168±0.057(+)	0.546±0.013 (=)	0.651±0.066 (=)	0.707±0.006(-)		0.645±0.017 (=)	0.509±0.004(=)	0.166±1E-04 (=)	0.292±7E-04(+)	0.347±0.001(+) **0.740±0.002 (=)**
	B2SMOTE	0.512±0.066(+)	0.845±0.011 (-)	0.235±0.054(+)	**0.550±0.014 (=)**	**0.745±0.032 (-)**	0.705±0.004 (-)	0.642±0.013(=)	0.418±0.004 (+)	**0.177±0.008 (-)**	0.290±0.001 (+)	0.347±3E-04 (+)	0.731±0.004(+)
	T-SMOTE	0.452±0.086 (+)	0.846±0.008(-)	0.224±0.028 (+)	0.545±2E-06 (=)	0.731±0.036 (-)	0.746±0.017(-)	0.547±0.019 (+)	0.490±0.016 (+)	0.166±0.002 (=)	0.291±3E-04 (+)	0.345±4E-06 (+)	0.726±6E-06 (+)
	Diffusion-TS	0.555 (+)	**0.884 (-)**	0.127 (+)	0.525 (=)	0.409 (+)	0.658 (-)	0.641 (+)	0.531 (=)	0.166 (=)	**0.302 (-)**	**0.352 (=)**	0.726 (+)
	Evo-TFS	**0.595±0.052**	0.822±0.017	**0.421±0.013**	0.533±0.052	0.677±0.038	0.637±0.045	0.653±0.019	**0.533±0.060**	0.169±0.006	0.298±3E-4	0.351±0.005	**0.740±0.8E-4**
	总计	34 +, 19 =, 19 -											
几何平均数	SMOTE	0.568±0.043 (=)	0.803±0.019 (+)	0.308±0.054(=)	0.306±0.084 (=)	0.664±0.042 (-)	0.585±0.071 (=)	**0.769±0.011 (-)**	0.424±0.039 (=)	0.025±0.047 (=)	**0.076±0.015** (=)	0.086±0.016 (+)	0.333±0.020 (+)
	ADASYN	0.568±0.043 (=)	0.782±0.015 (+)	0.238±0.054 (+)	0.305±0.080 (+)	0.585±0.041 (=)	0.420±0.022 (+)		0.767±0.011 (-)	0.280±0.028 (+)	**0.037±0.051 (-)**	0.071±0.010 (=)	0.080±0.027 (+) 0.274±0.019 (+)
	B1SMOTE	0.585±0.046(=)	0.775±0.015(+)	0.298±0.056(+)	0.256±0.101 (+)	0.601±0.053 (=)	0.423±0.026(+)		0.763±0.013 (-)	0.234±0.03 (+)	0.004±0.011 (=)	0.051±0.016 (+)	0.090±0.034 (+) 0.283±0.021 (+)
	B2SMOTE	0.584±0.050(=)	0.801±0.021(+)	0.270±0.046(+)	0.339±0.063 (=)	**0.678±0.027 (-)**	0.416±0.021(+)		0.764±0.010 (-)	0.305±0.029 (+)	0.022±0.044(=)	0.053±0.010 (+)	0.091±0.007 (+) 0.333±0.014 (+)
	T-SMOTE	0.539±0.067(+)	0.807±0.015(+)	0.262±0.024(+)	0.325±0.016 (=)	0.666±0.029 (-)	0.574±0.052(=)		0.683±0.014 (-)	0.418±0.038 (=)	0.008±0.020 (=)	0.066±0.001 (+)	0.001±5E-04 (+) 0.348±0.021 (=)
	Diffusion-TS	**0.616 (=)**	**0.832 (-)**	0.258 (+)	0.369 (=)	0.510 (+)	0.591 (=)	0.659 (=)	0.430 (=)	0.000 (=)	0.061 (+)	0.126 (=)	0.333 (+)
	Evo-TFS	0.590±0.055	0.819±0.022	**0.330±0.021**	**0.374±0.096**	0.589±0.053	**0.594±0.055**	0.663±0.017	**0.456±0.084**	0.003±0.013	0.075±0.001	**0.131±0.019**	**0.351±0.021**
	总计	34 +, 28 =, 10 -											

A. 总体结果

表 IV 和表 V 报告了基于 Evo-TFS 和六种基线方法训练的两个分类器在测试集上的 F1 分数和几何平均数。在这些表中，“+”表示 Evo-TFS 的性能显著优于基线方法，“–”表示性能显著较差，“=”表示与基线方法相比没有显著差异。在“Evo-TFS”这一行中，‘±”之后的值表示相应的标准差。

如表 IV 所示，在时域中使用 CNN 分类器时，Evo-TFS 在 F1 分数和几何平均数两项指标上均优于六种基线采样方法。具体而言，根据 30 次运行的平均结果，Evo-TFS 在每项指标上都在 7 个数据集上取得最优表现；在全部 12 个数据集中，除 F1 分数的 3 个数据集和几何平均数的 2 个数据集外，其余数据集的排名均为第一或第二。根据统计显著性检验，Evo-TFS 在总计 72 种情况下，有 49 种情况下的 F1 分数表现显著更优，而显著更差的仅有 8 种。同样，在 G 均值指标方面，Evo-TFS 在 72 种情况下有 39 种取得显著更优的结果。这些结果验证了 Evo-TFS 在生成时间序列数据以

表 VI F1 分数和几何平均数显著性检验的统计结果

指标	F1 分数		几何平均数	
方法	卷积神经网络 (+/=/–)	随机森林 (+/=/–)	卷积神经网络 (+/=/–)	随机森林 (+/=/–)
SMOTE	9/2/1	5/3/4	8/3/1	3/7/2
ADASYN	9/2/1	6/2/4	8/3/1	7/3/2
B1SMOTE	8/3/1	5/6/1	6/5/1	8/3/1
B2SMOTE	10/1/1	6/2/4	8/3/1	7/3/2
T-SMOTE	8/2/2	7/2/3	5/6/1	5/5/2
Diffusion-TS	5/5/2	5/4/3	3/9/0	4/7/1
总计	49/15/8	34/19/19	38/29/5	34/28/10

解决时间序列分类任务中卷积神经网络的类别不平衡问题。

如表 V 所示，在频域中使用 RF分类器时，Evo-TFS 同样表现出明显优势。在 F1 分数方面，Evo-TFS 在 12 个数据集中有 6 个取得最优或次优表现，比 Diffusion-TS 多 1 个（即 5 个）。在几何平均数方面，Evo-TFS 在 6 个数据集上优于基线方法，在 3 个数据集上排名第二。统计分析进一步证实了这些观察结果。在 F1 分数方面，Evo-TFS 显著

表 VII Evo-TFS（平均值）与基线方法（平均值）在 12 个数据集上的两种分类器的性能排名。最优排名以**粗体**显示，次优排名带下划线

指标		F1 分数				几何平均数			
分类器	方法	D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 平均值				D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 平均值			
CNN	SMOTE	5.0 2.0 6.0 3.0 7.0 3.0 5.0 4.0 5.0	2.0	6.0	5.0	4.42	4.0 4.0 7.0 3.0 7.0 3.0 5.0 4.0 5.0	2.0	6.0	7.0	4.75
	ADASYN	7.0 5.0 4.0 6.0 4.0 6.0 6.0 6.0 6.0	3.0	7.0	3.0	5.25	6.0 3.0 5.0 6.0 5.0 6.0 4.0 6.0 6.0	3.0	7.0	5.0	5.17
	B1SMOTE	4.0 6.0 7.0 5.0 3.0 4.0 4.0 5.0 4.0	7.0	3.0	1.0	4.42	7.0 2.0 4.0 5.0 2.0 4.0 2.0 5.0 4.0	7.0	2.0	1.0	3.75
	B2SMOTE	6.0 7.0 5.0 4.0 6.0 5.0 3.0 7.0 7.0	1.0	4.0	4.0	4.92	5.0 7.0 6.0 4.0 4.0 5.0 3.0 7.0 7.0	1.0	4.0	4.0	4.75
	T-SMOTE	3.0 3.0 2.0 7.0 2.0 7.0 7.0 1.0 3.0	4.0	5.0	6.0	4.17	1.0 5.0 3.0 7.0 1.0 7.0 7.0 3.0 3.0	4.0	5.0	6.0	4.33
	Diffusion-TS	2.0 4.0 1.0 2.0 1.0 2.0 2.0 3.0 2.0	5.0	2.0	7.0	2.75	3.0 6.0 1.0 2.0 3.0 2.0 6.0 2.0 2.0	5.0	3.0	3.0	3.17
	Evo-TFS	1.0 1.0 3.0 1.0 5.0 1.0 1.0 2.0 1.0	6.0	1.0	2.0	**2.08**	2.0 1.0 2.0 1.0 6.0 1.0 1.0 1.0 1.0	6.0	1.0	2.0	**2.08**
RF	SMOTE	5.5 5.0 4.5 3.0 4.0 3.0 6.0 4.0 3.0	3.0	3.0	4.0	4.00	5.5 4.0 2.0 5.0 3.0 3.0 5.0 3.0 6.0	1.0	5.0	5.0	3.96
	ADASYN	5.5 6.0 4.5 1.0 3.0 6.0 1.0 6.0 2.0	2.0	4.0	3.0	3.67	5.5 6.0 7.0 6.0 6.0 6.0 6.0 6.0 1.0	3.0	6.0	7.0	5.46
	B1SMOTE	3.0 7.0 6.0 4.0 6.0 5.0 3.0 3.0 6.0	5.0	5.0	2.0	4.58	3.0 7.0 3.0 7.0 4.0 5.0 3.0 7.0 4.0	7.0	4.0	6.0	5.00
	B2SMOTE	4.0 3.0 2.0 2.0 1.0 7.0 4.0 7.0 1.0	7.0	6.0	5.0	4.08	4.0 5.0 4.0 3.0 1.0 7.0 4.0 5.0 2.0	6.0	3.0	4.0	4.00
	T-SMOTE	7.0 2.0 3.0 5.0 2.0 4.0 7.0 5.0 5.0	6.0	7.0	7.0	5.00	7.0 3.0 5.0 4.0 2.0 4.0 1.0 4.0 3.0	4.0	7.0	2.0	3.83
	Diffusion-TS	2.0 1.0 7.0 7.0 7.0 1.0 5.0 2.0 7.0	1.0	1.0	6.0	3.92	1.0 1.0 6.0 2.0 7.0 2.0 7.0 2.0 7.0	5.0	2.0	3.0	3.75
	Evo-TFS	1.0 4.0 1.0 6.0 5.0 2.0 2.0 1.0 4.0	4.0	2.0	1.0	**2.75**	2.0 2.0 1.0 1.0 5.0 1.0 2.0 1.0 5.0	2.0	1.0	1.0	**2.00**

更优或在统计上相当的情况有 53 例（共 72 例）。在几何平均数方面，Evo-TFS 显著更优或在统计上相当的情况有 62 例，仅有 10 例显著更差。这些结果表明，Evo-TFS 能有效地重新平衡时间序列数据，从而显著提升传统分类器在频域中的性能。

总之，Evo-TFS 在生成用于时间序列分类的时间序列数据方面展现出显著优势，无论是时域中的卷积神经网络分类器，还是频域中的随机森林分类器，均是如此。

B. 结果与分析总结

表 VI 总结了统计显著性检验结果。总体而言，Evo-TFS 在大多数情况下表现出明显优势。与基线采样方法相比，在总计 72 种情况中，Evo-TFS 帮助卷积神经网络在 63 种情况下（F1 分数）和 66 种情况下（几何平均数）取得显著更好或统计上相近的性能。同样，应用于随机森林时，在总计 72 种情况中，Evo-TFS 帮助其在 53 种情况下（F1 分数）和 62 种情况下（几何平均数）取得显著更好或统计上相近的性能。从具体方法来看，与随机森林相比，Evo-TFS 与卷积神经网络结合时表现更强，尤其是相较于 SMOTE、ADASYN 等基线方法。相比之下，Evo-TFS 与随机森林结合时，整体表现通常优于 Diffusion-TS。总之，就 F1 分数和几何平均数这两项指标而言，Evo-TFS 在大多数情况下都取得了显著更好的性能。这些结果凸显了所提方法在不同分类器和评估指标下的稳健性与稳定性。

如表 VI 所示，Evo-TFS 在大多数不平衡时间序列数据集上的表现优于基线方法。这主要是因为遗传编程能够从现有数据中学习复杂的变换，生成能够捕捉精细时间模式的合成时间序列。此外，适应度函数同时纳入了时域和频域信息，从而引导进化朝着更优的方向发展

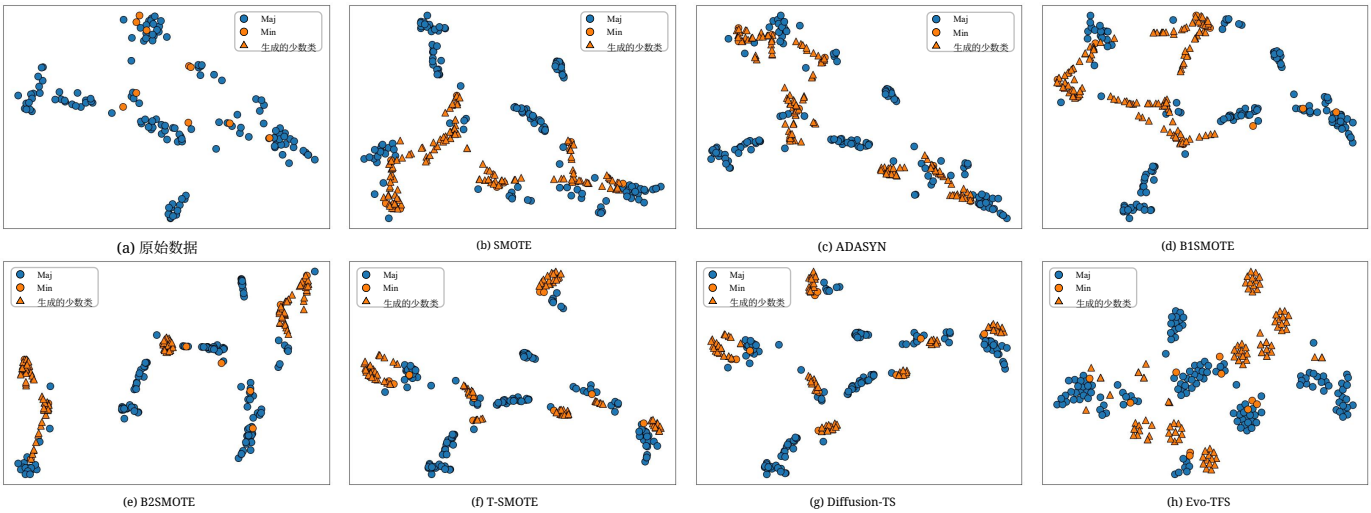
表 VIII 4 个数据集上的消融实验结果

数据集	指标	方法	不使用动态时间规整	不使用离散傅里叶变换	Evo-TFS
ECG200	F1 分数	CNN	0.855	0.854	**0.857**
		RF	**0.847**	0.802	0.822
	几何平均数	CNN	0.806	**0.819**	0.816
		RF	**0.827**	0.815	0.819
ItalyPowerDemand	F1 分数	CNN	**0.913**	0.901	0.910
		RF	0.630	0.632	**0.637**
	几何平均数	CNN	0.896	**0.924**	0.914
		RF	0.586	0.589	**0.594**
PowerCons	F1 分数	CNN	0.893	0.898	**0.902**
		RF	0.528	0.525	**0.533**
	几何平均数	CNN	**0.908**	0.897	0.907
		RF	**0.460**	0.419	0.456
计算机	F1 分数	CNN	**0.167**	**0.167**	0.158
		RF	0.333	0.333	**0.351**
	几何平均数	CNN	0.281	0.275	**0.288**
		RF	0.129	0.127	**0.131**

表 IX 4 个数据集上的参数敏感性结果

数据集	指标	方法	$\alpha = 0.3$	$\alpha = 0.7$	$\alpha = 0.5$
ECG200	F1 分数	CNN	0.851	0.854	**0.857**
		RF	0.819	**0.828**	0.822
	几何平均数	CNN	0.812	**0.819**	0.816
		RF	**0.824**	0.823	0.819
ItalyPowerDemand	F1 分数	CNN	**0.912**	0.903	0.910
		RF	0.633	**0.639**	0.637
	几何平均数	CNN	**0.919**	0.907	0.914
		RF	**0.602**	0.599	0.594
PowerCons	F1 分数	CNN	0.897	0.899	**0.902**
		RF	0.527	0.530	**0.533**
	几何平均数	CNN	**0.911**	0.909	0.907
		RF	0.453	**0.459**	0.456
计算机	F1 分数	CNN	**0.158**	0.157	**0.158**
		RF	0.333	0.333	**0.351**
	几何平均数	CNN	0.283	0.281	**0.288**
		RF	**0.133**	0.130	0.131

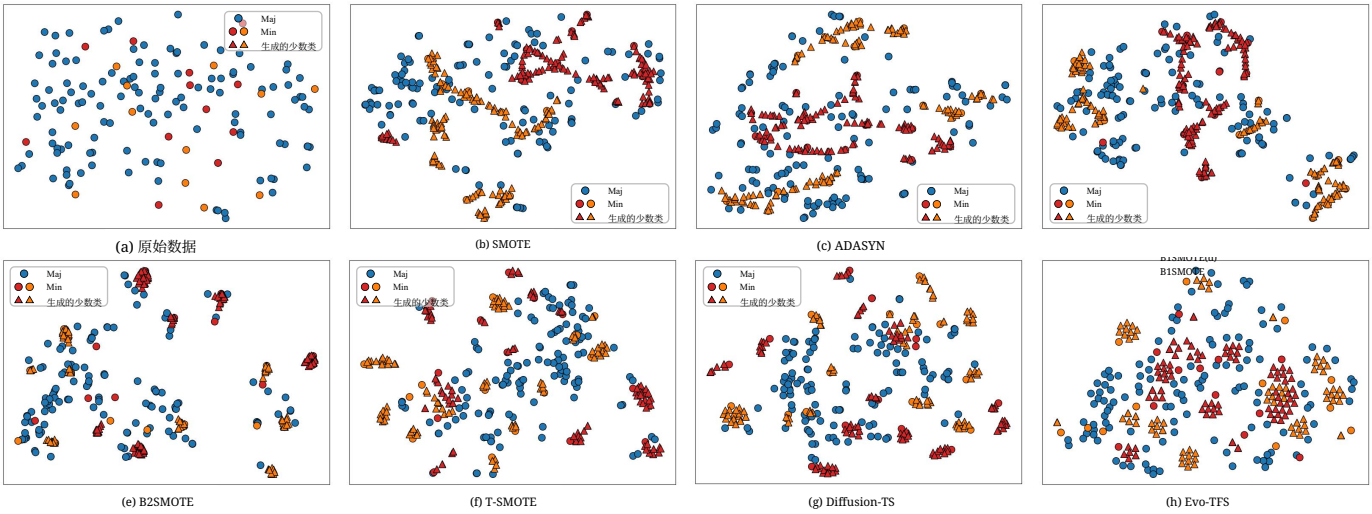
能够保持原始数据时间完整性和频谱完整性的解决方案。然而，在少数情况下，Evo-TFS 的表现不如基线方法。当少数类实例在特征空间中相对集中时，通常会出现这种情况。在此情况下，SMOTE（及其变体）凭借局部插值，以及 Diffusion-TS 凭借分布学习，都能够生成紧凑且具有代表性的样本，



(a) 原始数据 (b) SMOTE (c) ADASYN (d) B1SMOTE

(e) B2SMOTE (f) T-SMOTE (g) Diffusion-TS (h) Evo-TFS

图 7. 在屏幕类型数据集上使用 t-SNE 方法将可视散点图降至二维。



(a) 原始数据 (b) SMOTE (c) ADASYN

(e) B2SMOTE (f) T-SMOTE (g) Diffusion-TS (h) Evo-TFS

图 8. 在瑜伽数据集上使用 t-SNE 方法将可视散点图降至二维。

从而取得具有竞争力的性能。

为探究 Evo-TFS 是否能够生成高质量的平衡数据集以提升分类器的性能，表 VII 汇总了每种分类算法在 12 个数据集上各方法的详细排名。总计 48 种情况中，Evo-TFS 获得了 24 次第一名和 12 次第二名。此外，在两种分类器和两种评估指标上，它的平均排名均为最优。这些结果表明，所提出的 Evo-TFS 方法能够生成高质量的平衡数据集，从而有效提升不同领域类别不平衡时间序列分类任务中各类分类器的性能。

C. 消融研究与参数敏感性分析

我们进一步考察同时使用 DTW 和 DFT 距离是否是提升性能所必需的。在消融实验中，从适应度函数中移除 DTW 记为 w/o DTW，移除 DFT 记为 w/o DFT。

如表 VIII 所示，移除 DTW 或 DFT 中的任一项，都会损害模型在不同数据集和分类器上的性能稳定性。这是因为 DTW 和 DFT 捕获了互补信息。DTW 强调局部时间对齐，而 DFT 捕获全局频率模式。在适应度函数中同时纳入这两个组成部分，使 Evo-TFS 能够在不同指标和数据集上取得更加稳健的结果。

为进一步考察适应度函数中的参数 α 对结果的影响，表 IX 汇总了不同 α 设置下的结果。如表 IX 所示，在所有数据集和两种分类器上，不同 α 值对应的性能差异都非常小。这些观察结果表明，Evo-TFS 对参数 α 不敏感。

D. 数据可视化与进一步分析

图 7 和图 8 分别展示了 ScreenType 和 Yoga 数据集过采样后数据分布的 t-SNE可视化，

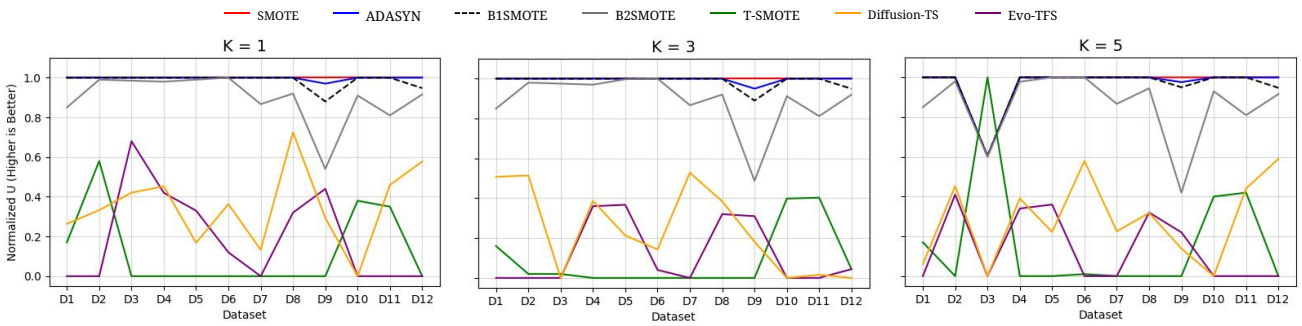


图 9. 不同方法在不同 K 值下的归一化密度一致性指标。

如图所示，基于插值的采样方法（如 SMOTE 和 ADASYN）往往会生成呈线性分布的样本，通常难以对复杂的时间动态特性进行建模。这主要是因为这些方法仅在特征空间中运作，忽略了时间序列数据的顺序和依赖关系。通过在随机选取的邻居之间进行线性插值来生成合成样本，很可能会破坏固有的时间结构（如趋势和自相关），从而产生噪声数据点。T-SMOTE 和 Diffusion-TS（图 7 和图 8(f-g)）会生成集中的簇状数据块，能更好地保留局部数据结构，但其全局覆盖能力不足，因而难以涵盖边界信息。如图 7 和图 8(h)所示，Evo-TFS 生成的合成样本结构更规整，分布也更均匀。这些样本不仅保持了类内一致性，还能有效扩展少数类的边界区域，以连续且结构一致的方式填充稀疏区域。

图 9 进一步定量证明了采样方法生成的重平衡数据集中不同类别之间的密度一致性。对于每个数据集，这些密度一致性得分（U）均已归一化至 $[0, 1]$，数值越小表示性能越好。理想情况下，均衡良好的数据集应当使多数类和少数类具有相似的密度，即 U 趋近于 0。结果表明，SMOTE 和 ADASYN 等传统的基于插值的采样方法，经常生成密度与多数类样本显著不同的少数类样本。相比之下，T-SMOTE、Diffusion-TS 和 Evo-TFS 生成的样本往往具有与多数类更相似的密度分布。值得注意的是，在大多数情况下，Evo-TFS 的密度一致性最高，证实了其在为类别不平衡时间序列分类任务生成高质量代表性样本方面的优势。

总之，综合 t-SNE 可视化和均匀性统计所揭示的空间分布信息，Evo-TFS 在保留数据结构、改善类别平衡以及支持稳健的下游分类性能方面展现出明显优势。

VI. 结论

本研究旨在生成一组多样化的时间序列样本，同时保留其时间特征，以平衡数据分布并提升分类器性能——为此，我们提出了一种基于时频域的进化过采样方法。具体而言，我们设计了一个用于过采样的遗传编程框架，并使用 STGP 来约束输入和输出类型。此外，我们将时频域信息融入适应度函数，以有效引导进化过程。

在不平衡时间序列数据集上的实验表明，所提出的 Evo-TFS 方法优于基线方法，能够生成质量更高、更多样且分布更均匀的时间序列样本。此外，在时域和频域分类器上，Evo-TFS 的表现均优于这些基线方法。尽管 Evo-TFS 展现出生成高质量、多样化样本的强大能力，但遗传编程的使用导致计算成本高、训练时间长。未来，我们将探索加速这一过程的策略。

参考文献

[1] X. Li、W. Li、X. Yu、Z. Han 和 Q. Jin，“基于非线性因果时间序列网络的不平衡数据金融风险评估”，Information Processing & Management，第62卷，第3期，104025页，2025年。 [在线]。获取地址：https://www.sciencedirect.com/science/article/pii/S0306457324003844

[2] Y. Wang、Y. Wei、H. Yang、J. Li 和 Q. Wu，“利用不平衡电子健康记录，通过集成学习和时间序列模型预测急性肾损伤”，BMC Medical Informatics and Decision Making，第20卷，第1期，238页，2020年。[3] W. Jiang、Y. Hong、B. Zhou、X. He 和 C. Cheng，“一种基于生成对抗网络的不平衡工业时间序列异常检测方法”，IEEE Access，第7卷，第143 608–143 619页，2019年。[4] H. R. Sayegh、W. Dong 和 A. M. Al-madani，“采用基于长短期记忆网络的模型、特征选择和SMOTE增强不平衡数据入侵检测”，Applied Sciences，第14卷，第2期，479页，2024年。[5] B. Zhao、H. Lu、S. Chen、J. Liu 和 D. Wu，“用于时间序列分类的卷积神经网络”，Journal of Systems Engineering and Electronics，第28卷，第1期，第162–169页，2017年。[6] F. Karim、S. Majumdar、H. Darabi 和 S. Harford，“用于时间序列分类的多变量LSTM-FCNS”，Neural Networks，第116卷，第237–245页，2019年。[7] N. Mohammadi Foumani、L. Miller、C. W. Tan、G. I. Webb、G. Forestier 和 M. Salehi，“时间序列分类与外部回归的深度学习：当前综述”，ACM Computing Surveys，第56卷，第9期，第1–45页，2024年。[8] J. M. Johnson 和 T. M. Khoshgoftaar，“类别不平衡条件下深度学习综述”，Journal of big data，第6卷，第1期，第1–54页，2019年。[9] X. Wang、Z. Liu、J. Liu 和 J. Liu，“通过不平衡交互式学习在多关系图上进行欺诈检测”，Information Sciences，第642卷，119153页，2023年。[10] K. Ghosh、C. Bellinger、R. Corizzo、P. Branco、B. Krawczyk 和 N. Japkowicz，“深度学习中的类别不平衡问题”，Machine Learning，第113卷，第7期，第4845–4901页，2024年。

[11] H. He 和 E. A. Garcia，“从不平衡数据中学习”，IEEE《知识与数据工程汇刊》，第21卷，第9期，第1263–1284页，2009年。[12] N. V. Chawla、K. W. Bowyer、L. O. Hall 和 W. P. Kegelmeyer，“SMOTE：合成少数类过采样技术”，《人工智能研究杂志》，第16卷，第321–357页，2002年。[13] H. He、Y. Bai、E. A. Garcia 和 S. Li，“ADASYN：不平衡学习自适应合成采样方法”，载于《2008年IEEE国际神经网络联合会议》（IEEE世界计算智能大会），2008年，第1322–1328页。[14] A. Fern´andez、S. Garcia、F. Herrera 和 N. V. Chawla，“面向不平衡数据学习的SMOTE：进展与挑战，纪念15周年”，《人工智能研究杂志》，第61卷，第863–905页，2018年。[15] Q. Wen、L. Sun、F. Yang、X. Song、J. Gao、X. Wang 和 H. Xu，“面向深度学习的时间序列数据增强：综述”，载于《第三十届国际人工智能联合会议论文集》，IJCAI-21，2021年8月，第4653–4660页，综述分会场。[在线]。获取地址：https://doi.org/10 24963 2021 631 。/ijcai。/ P. Zhao、C. Luo、B. Qiao、L. Wang、S. Rajmohan、Q. Lin 和 D. Zhang，“T-SMOTE：面向不平衡时间序列分类的时间导向合成少数类过采样技术”，载于国际人工智能联合会议，2022年，第2406–2412页。[16] [17] R. Poli、W. B. Langdon、N. F. McPhee 和 J. R. Koza，“遗传编程指南”，2008年。[18] W. B. Langdon 和 R. Poli，《遗传编程基础》。施普林格科学与商业媒体，2013年。[19] M. T. Ahvanooey、Q. Li、M. Wu 和 S. Wang，“遗传编程及其应用综述”，KSII《互联网与信息系统汇刊》（TIIS），第13卷，第4期，第1765–1794页，2019年。[20] W. Pei、Y. Cui、B. Xue、M. Zhang、J. Zhang、Y. Hou、G. Zou 和 Z. Qiang，“基于距离角度的遗传合成少数类过采样技术：一种用于不平衡数据学习的方法”，IEEE《进化计算汇刊》，2024年，doi:10.1109/TEVC.2024.3515485。[21] I. Araf、A. Idri 和 I. Chairi，“不平衡医疗数据的代价敏感学习：综述”，《人工智能评论》，第57卷，第4期，80页，2024年。[22] W. Pei、B. Xue、M. Zhang、L. Shang、X. Yao 和 Q. Zhang，“不平衡分类综述：进化计算能发挥什么作用？”，IEEE《进化计算汇刊》，第28卷，第2期，第353–373页，2024年。[23] R. Mohammed、J. Rawashdeh 和 M. Abdullah，“采用过采样与欠采样技术的机器学习：概述研究与实验结果”，载于《2020年第11届国际信息与通信系统会议》（ICICS），IEEE，2020年，第243–248页。[24] S. Bej、N. Davtyan、M. Wolfien、M. Nassar 和 O. Wolkenhauer，“LoRAS：一种面向不平衡数据集的过采样方法”，Machine Learning，第110卷，第279–301页，2021年。[25] W. Pei、B. Xue、M. Zhang、L. Shang、X. Yao 和 Q. Zhang，“不平衡分类综述：进化计算能发挥什么作用？”，IEEE《进化计算汇刊》，第28卷，第2期，第353–373页，2024年。[26] C. Elkan，“代价敏感学习基础”，载于《国际人工智能联合会议》，第17卷，第1期。劳伦斯·埃尔鲍姆联合公司，2001年，第973–978页。[27] Y. Geng 和 X. Luo，“用于不平衡时间序列分类的代价敏感卷积神经网络”，《智能数据分析》，第23卷，第2期，第357–370页，2019年。[28] J. Gao、X. Song、Q. Wen、P. Wang、L. Sun 和 H. Xu，“RobustTAD：通过分解和卷积神经网络实现稳健的时间序列异常检测”，arXiv预印本 arXiv:2002.09545，2020年。[29] K. Oksuz、B. C. Cam、S. Kalkan 和 E. Akbas，“目标检测中的不平衡问题：综述”，IEEE《模式分析与机器智能汇刊》，第43卷，第10期，第3388–3415页，2020年。[30] H. Han、W.-Y. Wang 和 B.-H. Mao，“Borderline-SMOTE：一种用于不平衡数据集学习的新型过采样方法”，载于《国际智能计算会议》。施普林格，2005年，第878–887页。[31] F. Zhang、Y. Mei、S. Nguyen、M. Zhang 和 K. C. Tan，“用于动态柔性作业车间调度的代理辅助进化多任务遗传编程”，IEEE《进化计算汇刊》，第25卷，第4期，第651–665页，2021年。[32] C. Wang、Q. Chen、B. Xue 和 M. Z hang，“用于多输出回归的语义引导多任务遗传编程”，《模式识别》，第161卷，111289页，2025年。[33] A. Fonseca 和 D. Poc,as，“比较强类型遗传编程与语法引导遗传编程的表达能力”，载于《……论文集》遗传与进化计算会议，2023年，第1100–1108页。[34] D. J. Montana，“强类型遗传编程”，Evolutionary Computation，第3卷，第2期，第199–230页，1995年。[35] A. Abanda、U. Mori和J. A. Lozano，“基于距离的时间序列分类综述”，数据挖掘与知识发现，第33卷，第2期，第378–412页，2019年。[36] R. J. Kate，“将动态时间规整距离用作特征以改进时间序列分类”，数据挖掘与知识发现，第30卷，第283–312页，2016年。[37] D. D. Sundararajan，“离散傅里叶变换”，载于《数字信号处理：导论》。施普林格，2024年，第67–106页。[38] C.-S. Yu，“用于距离继电保护应用的基于离散傅里叶变换的自适应仿相量估计器”，IEEE电力输送汇刊，第21卷，第4期，第1836–1846页，2006年。[39] X. Yuan和Y. Qiao，“Diffusion-TS：面向通用时间序列生成的可解释扩散”，载于第十二届国际学习表征会议，2024年。[在线]。网址：https: //openreview.net/forum?id =4 1 99 h apFjO B. Al-Helali、Q. Chen、B. Xue和M. Zhang，“采用基于特征的迁移学习、面向不完整数据符号回归的多树遗传编程”，IEEE控制论汇刊，第54卷，第7期，第4014–4027页，2024年。[41] F.-A. Fortin、F.-M. De Rainville、M.-A. G. Gardner、M. Parizeau和C. Gagn´e，“DEAP：让进化算法变得简单”，机器学习研究杂志，第13卷，第1期，第2171–2175页，2012年。[42] G. Lema ˜A ˇZtre、F. Nogueira和C. K. Aridas，“Imbalanced-learn：用于应对机器学习中不平衡数据集难题的Python工具箱”，机器学习研究杂志，第18卷，第17期，第1–5页，2017年。[43] Y. Zheng、Q. Liu、E. Chen、Y. Ge和J. L. Zhao，“使用多通道深度卷积神经网络进行时间序列分类”，载于《网络时代信息管理》，F. Li、G. Li、S.-w. Hwang、B. Yao和Z. Zhang编。施普林格国际出版：施普林格，2014年，第298–310页。[44] S. J. Rigatti，“随机森林”，保险医学杂志，第47卷，第1期，第31–39页，2017年。[45] C.-L. Liu和P.-Y. Hsieh，“面向不平衡数据的基于模型的合成采样”，IEEE《知识与数据工程汇刊》，第32卷，第8期，第1543–1556页，2019年。[46] J. Yin、C. Gan、K. Zhao、X. Lin、Z. Quan和Z.-J. Wang，“一种新颖的不平衡数据分类模型”，载于AAAI人工智能会议论文集，第34卷，第04期，2020年，第6680–6687页。[47] T. Fawcett，“ROC分析简介”，模式识别快报，第27卷，第8期，第861–874页，2006年。[48] S. Lee和D. K. Lee，“应用多重比较检验的正确方法是什么？”，韩国麻醉学杂志，第71卷，第5期，第353–360页，2018年。[49] J. Dem ˇsar，“多个数据集上分类器的统计比较”，机器学习研究杂志，第7卷，1月号，第1–30页，2006年。[40]