TimeMAE_zh

Melmaphother

December 2024

目录

摘要

通过自监督预训练增强基于深度学习的时间序列模型的表达能力在时间序列分类中变得日益普遍。尽管已经付出了大量努力开发自监督模型用于时间序列数据，但我们认为当前方法不足以学习到最优的时间序列表示，原因在于仅通过稀疏的逐点输入单元进行单向编码。在这项工作中，我们提出了 TimeMAE，一种基于 Transformer 网络的用于学习可传递的时间序列表示的新型自监督范式。TimeMAE 的独特特点在于通过窗口切片分区将每个时间序列处理成一系列不重叠的子序列，然后通过对局部子序列的语义单元进行随机掩码策略。这种简单而有效的设置可以帮助我们一举三得，即（1）通过双向编码方案学习丰富的时间序列上下文表示;（2）增加基本语义单元的信息密度;（3）使用 Transformer 网络高效编码时间序列表示。然而，在这种新颖的建模范式下执行重构任务是一个非平凡的问题。为了解决新注入的掩码嵌入引起的差异问题，我们设计了一个解耦的自编码器架构，它通过两个不同的编码器模块分别学习可见（未掩码）位置和掩码位置的表示。此外，我们构建了两种信息丰富的目标以完成相应的预文本任务。一种是创建一个分配给每个掩码区域的代码词的分词器模块，从而有效完成掩码代码词分类（MCC）任务。另一种是采用孪生网络结构为每个掩码输入单元生成目标表示，旨在执行掩码表示回归（MRR）优化。经过全面的预训练，我们的模型可以高效地学习可传递的时间序列表示，从而有益于时间序列的分类。我们在五个基准数据集上进行了广泛的实验，验证了 TimeMAE 的有效性。实验结果表明，TimeMAE 可以显著超越先前的竞争基线。此外，我们还通过进行迁移学习实验展示了所学表示的普适性。为了保证结果的可重现性，我们将实验代码公开共享，以促进在 https://github.com/Mingyue-Cheng/TimeMAE 中进行时间序列的自监督表示。

1 介绍

时间序列数据涉及一组随时间同步变化的变量，对智能应用如临床监测、交通分析、气候科学和工业检测等具有关键作用。其中，时间序列分类作为一种基础技术受到了显著关注，从早期的经典方法到最新的深度学习方法都提出了一系列模型。

深度学习方法相较于经典模型表现出较大的优势，因为它们更容易扩展到大量不同的时间序列数据集，且对先前知识的依赖较少。然而，直接使用这些强大的神经架构（如 Transformer 网络）进行时间序列分类可能无法取得令人满意的结果。这一不理想的结果部分归因于有限的带标签数据，因为 Transformer 架构对大量的训练标签依赖较重。尽管先进的传感器设备使得收集时间序列数据变得容易，但在某些实际场景中，获取大量准确的带标签时间序列数据仍然是一个耗时、容易出错甚至不可行的任务。

面对这一难题，研究者们致力于挖掘未标记的时间序列数据。其中，自监督学习是一种吸引人的方法，通过从未标记数据中学习可传递的模型参数，在语言和视觉领域已取得显著成功。自监督学习的基本方案是首先获得经过预训练的模型，然后将其应用于目标任务，通过表示特征或完全微调的方式。尽管自监督技术在时间序列领域取得了成功，其中对比学习范式几乎成为最流行的解决方案。然而，这些对比方法的不变性假设在实际场景中并不总是成立，同时可能在制定数据增强策略时引入归纳偏见，并在负采样时引入额外的偏见。更糟糕的是，这些方法本质上采用了单向编码器方案来学习时间序列的表示，从而在很大程度上限制了上下文表示的提取。

Masked autoencoders [23], [24]，可以很好地克服先前提到的这些限制，此前已经被提出。它们的主要理念是将掩码样式的损坏输入编码到潜在空间，然后通过编码器和解码器对原始输入进行恢复。其中一个成功的例子是基于 Transformer 网络构建的 BERT [16]，在语言表示方面取得了重要的里程碑。受此启发，我们注意到基于 Transformer 网络的一项开创性工作 [25]，直接将原始时间序列视为输入，然后通过逐点回归优化来恢复这些完整的输入。然而，这种方法通常由于忽略了由自注意机制引起的二次计算复杂性而需要非常昂贵的计算成本。最近的方法 [21]，[28]，[29] 报告说，这种方法由于其弱的泛化性能而产生有限的表示结果。在我们看来，这种差劲的泛化性能主要是因为忽视了时间序列的独特属性。首先，与语言数据不同，时间序列是具有时间冗余的自然序列信号，即每个时间步可以轻松地从其相邻点推断出来。因此，由于恢复任务的天真属性，预训练表示编码器模型无法很好地训练。其次，由于时间序列中的判别性模式通常呈子序列形式（shapelet [5]，[30]），每个单一点值只能携带非常稀疏的语义信息，如图 1 左侧所示。最后，由于采用了掩码策略，自监督预训练和目标任务

优化之间会产生差异。在自监督预训练期间，某一比例的位置被替换为掩码嵌入，而在微调阶段通常缺乏这些人工符号。

为了解决上述问题，在这项工作中，我们提出了一种新颖的掩码自编码器架构，称为 TimeMAE，用于基于 Transformer 网络学习可转移的时间序列表示。具体而言，我们提出将每个时间步单独建模的思想，改为将每个时间序列分割成一系列不重叠的子序列，通过窗口切片操作实现。这种策略不仅增加了掩码语义单元的信息密度，还极大地节省了计算成本和内存消耗，因为序列长度变短。图 1 的右侧显示了窗口切片分区的示例。在窗口切片之后，我们对这些重新定义的语义单元执行掩码操作，目的是实现时间序列表示的双向编码。特别地，我们发现简单的随机掩码策略，配合 60% 的高比例，与时间序列的最佳表示相匹配。然而，这种窗口切片操作和掩码策略也容易引发一些挑战。因此，为了解决掩码位置引起的差异，我们设计了一个解耦的自编码器架构，其中可见区域（未掩码）和掩码区域的上下文表示分别由两个不同的编码器模块提取。为了完成恢复任务，我们制定了两个预文本任务来引导预训练过程。一个是掩码密码分类（MCC）任务，将每个掩码区域分配给其独立的离散密码，由另外设计的分词器模块生成。另一个是掩码表示回归（MRR）任务，旨在将掩码位置的预测表示与提取的目标表示对齐。在这里，我们采用孪生网络架构来实现为每个掩码位置生成目标表示。我们在五个公共数据集上进行了广泛的实验。实验结果清楚地表明，预训练的 TimeMAE 在性能上优于从头开始的编码器和先前的竞争性基线。

总的来说，我们描述了提出的 TimeMAE 的贡献如下：

1. 我们提出了一个在时间序列表示上概念简单但非常有效的自监督范式，该范式从点粒度到局部子序列粒度促使基本语义元素的提取，并且同时实现了从单向到双向的上下文信息提取。2. 我们为时间序列表示提出了一个端到端的解耦自编码器架构，其中：(1) 我们解耦了掩码和可见输入的学习，以消除由掩码策略引起的差异问题；(2) 我们制定了两个基于可见输入的恢复缺失部分的预文本任务，包括 MCC 和 MRR 任务。3. 我们在五个公开可用的数据集上进行了广泛的实验，以验证 TimeMAE 的有效性。实验结果清楚地展示了新提出的自监督范式和解耦自编码器架构的有效性。

2 相关工作

在这一节中，我们简要介绍与我们的工作密切相关的两个方向的先前研究：时间序列分类和时间序列的自监督学习。

2.1 时间序列分类

过去几年中，已经有大量的研究致力于时间序列的分类。一般而言，先前的研究可以粗略地分为两类：基于模式和基于特征。前者通常提取模式的集合，然后将其输入到分类器中 [5]，[31]。这些方法的有效性在许多相关进展中得到了证明 [32]。然而，这些方法的最大局限性在于它们的计算成本严格与时间序列示例的数量和序列长度耦合，这是在实际应用中的主要瓶颈 [33]。与基于模式的方法相比，基于特征的方法通常更高效，但其性能在很大程度上取决于特征表示的质量 [34]，[35]。近年来，基于深度学习的方法在特征表示学习方面的优越性得到了证明，其中使用的先验知识较少。许多复杂的神经网络，如循环神经网络、卷积神经网络 [4]，[6]，[36] 和自注意力模型，已经应用于时间序列分类任务。此外，使用图卷积网络在时间序列之间挖掘图结构进行全面分析也吸引了一些研究者 [37]。然而，这些基于深度学习的方法通常需要大量的训练数据才能实现其最佳表达能力。由于很难获得大量的标记数据，直接使用深度神经架构可能不会带来令人满意的结果。

自监督时间序列 目前，自监督预训练范式几乎已经成为自然语言处理（NLP）[15]，[16]，[23] 和计算机视觉（CV）[17]，[18]，[24] 领域的默认配置。由于时间序列具有独特的特性，时间序列的自监督学习面临与 CV 和 NLP 中不同的特殊挑战。因此，许多研究致力于开发时间序列的自监督方法，其中当前的研究主要遵循两种范式：重建型 [16]，[38]，[39] 和判别型 [40]，[41]，[42]。重建型方法的思想是通过依赖自动编码器来尝试恢复完整的输入。例如，TimeNet [43] 首先利用编码器将时间序列转换为低维向量，然后使用循环神经网络的解码器来基于这些向量重建原始时间序列。另一种代表性的重建方法是 TST [25]，其主要思想是通过基于 Transformer 架构的去噪自动编码器来恢复这些被掩码的时间序列点。然而，重建优化是在点级别建模的，导致计算消耗非常昂贵且泛化能力有限 [28]。

相比之下，判别型方法需要更少的计算消耗 [28]，其主要解决方案是将正例聚集在一起，同时将负例推开 [44]。这些方法中共同的潜在主题是它们都通过数据增强策略 [45] 学习表示，然后使用孪生网络架构最大化正例的相似性。关键工作集中在通过各种负采样策略解决平凡常数解的问题 [17]。例如，在 T-Loss [46] 中，利用三元损失进行基于时间的负采样。TNC [8] 利用时间序列信号的局部平滑性，将子序列邻域视为正例，将非邻域视为负例。TS-TCC [28] 使用点级别和实例级别对比优化目标来对齐相应的表示。TS2Vec [29] 的显着特征是在多个尺度上层次执行对比优化。此外，在 [21] 中使用了时间和频域之间一致性的假设。总的来说，这些对比学习方法的目标是学习对各种增强输入扭曲不变的特征 [13]，[20]。然而，不仅数据增强策略需要许多归纳偏差，而且不变性假设并非总是成立。

在本节中，我们首先正式描述所使用的符号和研究的问题。之后，我们简要介绍 TimeMAE 的整体预训练架构。然后，我们详细介绍从特征编码器层、表示层到预文本优化任务的预训练架构。

3 提出 TimeMAE 模型

3.1 问题定义

我们介绍这些使用的符号和问题定义。$D = (X_1, y_1), (X_2, y_2), ..., (X_n, y_n)$ 是一个包含一系列对 (X_i, y_i) 的数据集，其中 n 表示示例的数量，X_i 表示具有相应标签 y_i 的单变量或多变量时间序列。我们将每个时间序列信号表示为 $X = [x_1, x_2, ..., x_T] \in R^{m \times T}$，其中 m 是通道数，T 是随时间测量的次数。给定一组未标记的时间序列数据，时间序列的自监督学习旨在获得可转移的时间序列表示，这可以促进时间序列分类任务的能力。

3.2 预训练架构的综述

正如图 2 所示，我们展示了 TimeMAE 模型中预训练过程的整体架构。首先，我们将输入系列投影到潜在表示中，然后通过掩码策略将整个输入分为可见（未掩码）输入和掩码子系列。然后，我们采用了一个解耦的自编码器架构来学习可见和掩码位置的表示，其中有两个相应的编码器。在编

码器级别，我们使用一组普通的 Transformer 编码器块来提取可见输入的上下文表示，同时利用基于交叉注意力的 Transformer 编码器网络的多层来学习掩码位置的表示。在解码器级别，我们通过基于可见输入的两种类型的新构建的目标信号的帮助，执行重建任务，预测所有缺失的部分。一方面，我们添加了一个标记模块，为每个掩码区域分配其自己的代码字，从而训练代码字分类任务。另一方面，我们使用孪生网络架构生成连续的目标表示信号，旨在执行目标表示回归任务。接下来，我们详细介绍 TimeMAE 模型的设计。

3.3 解码层特征

在这个小节中，我们主要回答掩码操作期间的两个关键问题：如何选择信息丰富的基本语义单元进行预训练，以及如何实现时间序列的双向编码器目标？

3.3.1 窗口切片策略

对于大多数现有的工作 [28], [47]，以逐点方式处理时间序列几乎已经成为将时间序列编码到表示空间的默认范式，其中每个时间序列点都以捕捉序列依赖性的方式进行建模。在我们看来，我们认为这种逐点建模方式在时间序列的自监督学习能力上存在很大的限制。更具体地说，这种设置粗略地忽视了掩码词和掩码时间序列点的信息密度差异。相反，前者是由人生成的，通常保留非常丰富的语义信息，而后者只涉及稀疏的语义特征。与此同时，在时间序列数据的相邻点之间存在很大的时间冗余，这意味着每个掩码点可以在不太具有挑战性的情况下从其相邻点中轻松推断出来。因此，在这样一个琐碎的重建过程中，预训练的编码器只能携带有限的信息。这些分析自然地激发了我们挖掘通过增加基本语义单元的信息密度来进行输入建模的潜力的动机。因此，我们提出利用时间序列的局部模式属性，允许子系列作为基本建模元素而不是逐点输入。因此，每个时间序列可以被处理成一个子系列单元的序列，其中每个局部子系列区域保留更丰富的语义信息并确保了重建任务的挑战。注意，这样的论点也可以得到基于 shapelet 的模型的支持 [48]，在这些模型中，与类标签相关的许多有辨识力的子系列

被提取为用于时间序列分类的有用的模式特征。例如，我们需要通过波形的许多局部区域而不是单个点来识别异常的心电图信号 [49]。

要更加精确，我们可以使用切片窗口操作将原始时间序列分割为连续的、不重叠的子系列。形式上，对于给定的时间序列 $X = \{x_1, x_2, ..., x_T\} \in R^{T \times m}$，一个切片是原始时间序列的一部分，定义为 $s_{i:j} = \{x_i, x_{i+1}, ..., x_{i+\sigma}\}$。这里，表示切片窗口的大小。假设给定时间序列 X 的长度为 T，我们的切片操作将其减少到新的长度 $\lceil \frac{T}{\sigma} \rceil$。将零填充操作进行以确保固定的序列长度。在我们的实现中，我们将整个时间序列处理成一系列常规的、不重叠的子系列补丁。这样设置的主要原因是这样可以确保可见区域不包含有关重建区域的信息。在对这些子系列单元的序列依赖性进行建模之前，我们需要通过特征编码器将每个元素编码为潜在表示。在这里，我们使用一个一维卷积层来提取跨通道的局部模式特征。值得注意的是，在特征编码器层中，不同的子系列共享相同的投影参数。输入 X 的投影隐藏表示由 $Z = [z_1, z_2, ..., z_{\lceil \frac{T}{\sigma} \rceil}] \in R^{\lceil \frac{T}{\sigma} \rceil \times d}$ 表示，其中 d 是隐藏表示的维数。我们使用 S 表示新的序列长度 $\lceil \frac{T}{\sigma} \rceil$ 同时使用 S_v/S_m 表示可见/掩码位置的数量。

3.3.2 掩码嵌入策略

为实现具有双向编码方案的时间序列的上下文表示的目标，我们决定按照 [16] 的方法使用掩码策略构建损坏的输入。假设 Sv/Sm 是被掩码/可见位置的数量。通过这种方式，每个位置的更全面的上下文表示，包括先前和未来的上下文，可以同时进行编码。为了清晰起见，我们使用 Zv 表示可见（未掩码）位置的嵌入，同时让 Zm 成为被掩码位置的嵌入。尽管之前已经开发了一些启发式的掩码策略，例如块掩码 [23]，我们采用随机掩码策略来构建损坏的输入。这意味着在构建自监督信号时，每个子系列单元被掩码的概率相同。主要原因是这种策略有利于确保在重构优化过程中每个输入位置的表示质量得到充分提高。请注意，我们在每个预训练时期都会动态随机掩码时间序列，以进一步增加训练信号的多样性。除了随机掩码操作外，TimeMAE 还鼓励保留更高的掩码比例。这是因为掩码比例在决定恢复任务是否足够具有挑战性以帮助编码器携带更多信息方面起着关键作用。通常，较高的掩码比例意味着通过这些可见邻域解决恢复任务更加困难。因此，在预训练编码器网络中可以编码更具表现力的网络容量。在我们

的工作中，我们发现约为 60% 的掩码比例可以帮助我们实现最佳性能，与先前工作中的 15% 的掩码比例有很大不同 [16]，[25]。请注意，窗口切片和掩码策略对于时间序列输入是不可知的，与对比学习中的数据增强等不会引入太多归纳偏差。

3.4 时间序列表征学习

在这一部分，我们介绍用于复杂序列依赖建模的表示学习层。特别是，TimeMAE 的关键设计之一是采用两个不同的编码器模块分别学习可见位置和被掩码位置的表示。这样的解耦设置有助于减轻由于额外的被掩码嵌入在预训练优化和微调训练之间引起的差异问题。接下来，我们详细介绍它们。

3.4.1 可见位置表征

在 TimeMAE 中，我们采用了用 $\mathcal{H}_\theta$ 表示的 vanilla Transformer 编码器架构，包括多头自注意层和前馈网络层，以学习可见区域输入的上下文表示。通过这样的架构，每个输入单元的表示都可以通过自注意计算机制获得所有其他位置的语义关系。值得注意的是，通过我们的窗口切片操作，减轻了利用自注意架构处理长序列输入的瓶颈。由于置换等效的自注意计算，我们因此向子系列嵌入的每个位置添加相对位置嵌入 $P \in R^{S\times d}$ 以保留序列属性的顺序。因此，通过将位置编码 P 和投影表示 Z 相结合，输入嵌入可以重新组织，即 $Z = Z + P$。然后，我们将可见位置 Z_v 的输入嵌入发送到编码器 $\mathcal{H}_\theta$。在我们的模型中有 L_v 层 Transformer 块，最后一层的输出 $\mathcal{H}_\theta^{L_v} = \{h_1^{L_v}, h_2^{L_v}, \cdots, h_{S_v}^{L_v}\}$ 表示 S_v 个可见位置的全局上下文表示。与以前的工作不同，我们只将可见区域的表示馈送到编码器网络，而去除了被掩码位置的表示。通过这种方式，不使用被掩码嵌入来训练编码器网络，因此可以在很大程度上减轻由被掩码标记输入引起的预训练和微调任务之间的差异问题。也就是说，这些策略消除了在前向传递中将掩码标记馈送到编码器模块的困境。

3.4.2 掩码位置表征

为了获得被掩码输入单元的表示，我们进一步将香草 Transformer 编码器中的自注意替换为交叉注意，形成解耦的编码器模块，表示为 $\mathcal{F}_\phi$，它表示可见和被掩码输入之间的表示学习解耦。具体而言，我们将这些可见位置和被掩码位置的表示发送到解耦编码器，以表示被掩码位置的输入单元。请注意，被掩码位置的嵌入被一个新初始化的向量 $z_{mask} \in R^d$ 替换，同时保持相应的位置嵌入不变。为了清晰起见，重新初始化的掩码区域的嵌入由 $\widetilde{Z}$ 表示。在解耦编码器中的表示学习过程中，我们将被掩码位置的表示视为查询嵌入，即 $\widetilde{Z}$，同时将可见位置的转换嵌入 $\mathcal{H}_\theta^{L_v}$ 视为帮助形成键和值的输入。形式上，被掩码查询的模型参数 $W^{Q_m} \in R^{d\times d}$ 对于所有被掩码查询都是相同的。可见键和值的投影参数，即 W^{K_v} 和 $W^{V_v} \in R^{d\times d}$ 也与可见输入共享。解耦编码器的第 L_m 层输出 $\mathcal{F}_\phi^{L_m} = \{f_1^{L_m}, f_2^{L_m}, \cdots, f_{S_m}^{L_m}\}$ 表示 S_m 个被掩码位置的转换上下文表示。请注意，解耦模块仅对被掩码位置的嵌入进行预测，而保持可见位置的嵌入不更新。主要原因是我们希望这样的操作有助于减轻反向传递中的差异问题。通过这个分割操作，可见输入的表示角色只由先前的编码器 $\mathcal{H}_\theta$ 负责。同时，解耦编码器主要集中在被掩码位置的表示上。此外，解耦模块 $\mathcal{F}_\phi$ 的另一个优势是防止解码器预测层进行可见位置的表示学习，从而使编码器模块 $\mathcal{H}_\theta$ 能够携带更多有意义的信息。

3.5 自监督优化

正如 [50] 中所述，直接制定自监督信号会在很大程度上由于原始空间的嘈杂和无约束性质而限制表示模型的容量 [50]。接下来，我们主要致力于回答如何为被掩码语义单元表示信息丰富的目标，以便完成相应的预文本任务。

3.5.1 掩码代码词分类

通过窗口切片操作，每个时间序列都被重新构造为一系列子序列，这些子序列可能展现出更丰富的语义。受到产品量化的启发 [51]，一个自然的想法是我们是否可以以一种新颖的离散视图表示这些重新构造的序列，即

为每个本地子序列分配自己的“代码词”。然后，这些分配的代码词可作为缺失部分的替代监督信号。对于当前大多数产品量化方法，其主要思想是使用基于簇操作的簇索引对每个密集向量进行编码，形成代码簿词汇表的所有索引组成短码。尽管其近似误差低，但实际上它是一种两阶段的方法，即独立分配聚类索引和学到的特征的提取。这种方式下，代码簿的表示能力可能与从 Transformer 编码器和解耦网络中提取的特征不兼容。由于这种不兼容性，如果朴素地采用这些技术来分配离散监督信号，自监督训练的性能将直接受到损害。因此，我们决定开发一个 Tokenizer 模块，该模块可以将被掩码位置的连续嵌入以端到端的方式转换为离散代码词。我们将这种类型的重构任务称为掩码代码词分类（MRR）。

3.5.2 掩码表示回归

除了对离散代码词进行预测之外，我们还对在潜在表示空间中直接执行回归优化感兴趣。我们将这样的预训练任务称为掩码表示回归（MRR）优化。这样的想法主要是受到表示学习基本原理的启发——表征数据基本分布的隐藏因素空间远低于原始空间，正如 [57] 中所述。同时，潜在表示空间允许集中关注原始时间序列的重要特征，同时对噪声的敏感性较低 [58]。

为了实现这一目标，我们选择使用广泛用于自监督对比学习优化的著名 siamese 网络架构。对比优化的主要思想是尝试将正相关实例的表示拉近，同时将负相关实例的表示推远。与典型的对比优化 [18]、[29] 不同，MRR 任务可以被视为在训练过程中不利用负实例的特殊情况。

为了生成目标表示，我们进一步采用了一个新颖的目标编码器模块。设 H_ξ 表示目标编码器，该编码器与 H_θ 的编码器具有相同的超参数设置，但其参数为 ξ。为了方便理解，我们将 vanilla transformer 编码器和解耦编码器模块的组合称为在线编码器，其中生成了对齐表示。依靠这样的 siamese 网络架构，目标编码器和在线编码器可以分别产生对掩码子序列表示的不同视图。因此，将两个不同视图的相应表示，即目标表示 $\mathcal{H}_\xi^{L_v}(Z_m)$ 和预测表示 $\mathcal{F}_\phi^{L_m}(\widetilde{Z})$ 进行对齐，自然形成了 MRR 任务的目标。鉴于连续表示作为回归信号，我们采用均方误差（MSE）损失来构建具体的优化目标。随后，我们定义确切的回归优化损失如下：

3.5.3 多任务优化

4 实验

在这一部分，我们首先介绍了我们 TimeMAE 的设置，包括预训练和微调阶段。然后，我们比较了所有比较方法在时间序列分类上的实验结果。此外，我们报告了一些有深刻见解的发现，以更深入地理解提出的 TimeMAE。

4.1 实验设置

4.1.1 数据集

我们在五个公开可用的数据集上进行实验，包括人体活动识别（HAR）、PhonemeSpectra（PS）、阿拉伯数字（AD）、Uwave 和癫痫，这些数据集由 [28]、[59]、[60]、[61] 提供。这五个数据集涵盖了各种变化：不同数量的通道、不同的时间序列长度、不同的采样率、不同的场景和各种类型的时间序列信号。值得注意的是，我们没有对这些数据集进行任何处理，因为原始的训练和测试集已经经过良好的处理。此外，表 1 总结了每个数据集的统计信息，例如实例数量（# Instance）、序列长度（# Length）、通道数量（# Channel）和类别数量（# Class）。

4.1.2 比较法

4.1.3 预训练和微调设置

在我们的实验中，我们通过两种主流的评估方式展示了预训练模型参数的有效性：线性评估和微调评估。前一种方式意味着预训练编码器的参数被冻结，只有新初始化的分类器层根据目标任务的标签进行微调。后一种方式表示预训练编码器的参数和新初始化的分类器层都在微调中进行，没有任何冻结操作。这两种评估方式分别用 FineLast 和 FineAll 表示。

相对而言，FineLast 方式的测量可以进一步反映预训练模型的质量。对于时间序列分类任务，我们需要整个实例级别的表示。因此，我们使用对所有时间点进行平均池化来表示整个时间序列，然后使用交叉熵损失来引导下游任务的模型优化。为了更好地评估不平衡的数据集，时间序列分类任

务的评估指标采用准确度和宏平均 F1 得分。最佳结果以粗体显示。在默认设置中，此表中所有准确度和 F1 得分的结果都以百分比（%）表示。报告的实验结果是在相同数据拆分上进行的三次独立运行的均值和标准差值。

4.1.4 执行细节

在预训练阶段，我们始终将嵌入大小设置为 64，对所有模型都保持一致。Adam 优化器是所有比较方法的默认优化器。学习率设置为 0.001，没有额外的复杂设置。所有模型的批处理大小设置为 64。默认情况下，所使用的 transformer 编码器架构的层深度设置为 8，其中注意力头的数量为 4，采用两层前馈网络，丢弃率为 0.2。对于一些竞争基线，我们严格遵循原始工作 [8] [25] [28] [29] 建议的相应超参数设置和数据增强策略。除了使用的 transformer 编码器外，我们还使用了一个六层的解耦编码器来提取被屏蔽位置的上下文表示。对于每个数据集，使用的切片窗口大小 δ 从 4, 8, 12, 16 中搜索。在默认设置中，我们采用了 TimeMAE 模型的 60% 的遮蔽比例。在 MCC 任务中，我们设计的 tokenizer 的代码书词汇量大小在特定数据集上从 64, 96, 128, 192, 256, 512 中搜索。至于 MRR 任务，τ 设置为 0.99。为了获得两个预文本任务之间的最佳权衡，我们将 α 固定为 1，同时搜索 β 的适当值范围为 1, 2, ..., 10。请注意，我们在微调阶段保持与预训练模型相同的超参数设置，同时涉及重叠的超参数。我们的 TimeMAE 的源代码可在 https://github.com/Mingyue-Cheng/TimeMAE 上获得。

4.2 实验结果

4.2.1 一对一预训练评估

4.2.2 迁移学习评估

4.2.3 使用不同比例的训练集进行微调

4.2.4 模型大小的可扩展性分析

4.2.5 不同比例的预训练集的分析

4.2.6 掩码策略的研究

4.2.7 消融研究

4.2.8 可视化分析

5 结论和未来工作