采用双维压缩的时空预测有效数据集蒸馏

Taehyung Kwon*、Yeonje Choi*、Yeongho Kim、Kijung Shin
金载哲人工智能研究生院，*KAIST*，首尔，韩国{taehyung.kwon、yeonjechoi、yeongho、kijungs}@kaist.ac.kr

arXiv:2603.10410v1 [cs.LG] 11 Mar 2026

摘要——时空时间序列广泛应用于交通预测和天气预报等现实场景。它们是在较长时间跨度和多个位置上的观测值序列，自然表示为多维数据。预测是时空分析的核心任务，已有大量深度学习方法被开发用于解决该问题。然而，随着实际应用中的数据集规模和模型复杂度不断增加，训练深度学习模型所需的时间和资源成本也日益高昂。

数据集蒸馏是应对这一挑战的一种有前景的解决方案，它能够合成紧凑的数据集，从而有效替代原始数据用于模型训练。尽管数据集蒸馏已在包括时间序列分析在内的多个领域取得成功，现有的数据集蒸馏方法仅压缩一个维度，因此不太适用于时空数据集；在时空数据集中，空间维度和时间维度共同导致了庞大的数据量。

为解决这一局限，我们提出 STemDist，这是首个专门面向时空时间序列预测的数据集蒸馏方法。我们方案的核心思想是以均衡方式同时压缩时间和空间维度，从而减少训练时间和内存占用。我们进一步通过在簇级别而非单个位置级别进行蒸馏来降低蒸馏成本，并以基于子集的细粒度蒸馏技术补充这一粗粒度方法，从而提升预测性能。

在五个真实世界数据集上，我们通过实验表明，与通用蒸馏方法和时间序列数据集蒸馏方法相比，由我们的 STemDist 方法蒸馏得到的数据集能够使模型训练（1）更快（最高达 6×），（2）更节省内存（最高达 8×），以及（3）更有效（预测误差最多降低 12%）。

索引词—**数据集蒸馏，时空时间序列，预测**

I. 引言

时空时间序列数据集十分普遍，在真实世界应用中发挥着关键作用，包括交通预测 [1]–[5] 和天气预报 [6], [7]。这些数据集由多个位置随时间产生的观测值组成，自然地表示为多维数组，其中不同维度捕获空间和时间信息以及一个或多个测量特征（例如，天气数据集中的风、温度和气压）。由于此类时间序列通常在较长时期内从众多位置采集 [1], [2], [8], ，其规模往往十分庞大。使用时空时间序列的一项核心任务是预测 [3], [7],[9],，其目标是基于历史观测值预测各位置的未来时间序列值。为此，已发展出多种深度学习方法 [10]–[14] 。然而，随着数据集规模和模型复杂度持续扩大，在真实场景中，训练这些模型会带来巨大的存储需求和计算成本。

数据集蒸馏已成为一种有前景的方法，用以减轻训练深度学习模型所带来的高昂计算与存储成本。其核心思想是创建一个小而信息丰富的合成数据集，使其能够在训练过程中替代原始数据，同时保持相当的模型性能。几种广为人知的方法包括梯度匹配 [15],轨迹匹配 [16],和元学习 [17]。梯度匹配最小化在原始数据与合成数据上计算得到的梯度之间的差异，而轨迹匹配则使在两个数据集上训练的模型的训练动态保持一致。元学习侧重于提高在合成数据上训练的模型对原始数据集的可迁移性。尽管数据集蒸馏主要针对图像和图数据进行研究 [18]–[21], ，近期工作已探索其在时间序列中的应用 [22]–[24]。

然而，由于现有的数据集蒸馏方法通常仅压缩单一维度，它们不太适用于时空数据集，因为在这类数据集中，空间维度和时间维度共同导致了庞大的数据量。具体而言，时间序列蒸馏技术仅压缩时间维度，而保持空间维度（即位置数量）不变，这可能导致大量的训练时间和GPU内存占用，如下文的初步实验所示。

初步实验1（联合缩减的必要性）。在图*1*中，我们比较了两种具有相同总体缩减比例的随机采样策略：（*1*）仅将时间维度缩减*0.5%*；（*2*）同时将空间维度缩减*5%*、时间维度缩减*10%*。将这两种策略应用于五个真实世界的时空时间序列数据集（见第 *V-A* 节）后，我们测量了 *MT-GNN [25],*（一种用于时空预测的代表性深度学习模型）每次外层迭代的训练时间和*GPU*内存使用量。图*1*显示，即使总体缩减比例相同，与仅缩减时间维度相比，同时缩减空间维度和时间维度也能显著减少训练时间和*GPU*内存使用量。详细实验设置请参见第 *V-A* 节；相关分析另见第四节中的定理*1*和定理*2*。

在这项工作中，我们提出了**STemDist**（**S**patio-**Tem**poral Dataset **Dist**illation），这是首个专为时空时间序列设计的数据集蒸馏方法。压缩空间维度所需的关键步骤

*贡献相同。

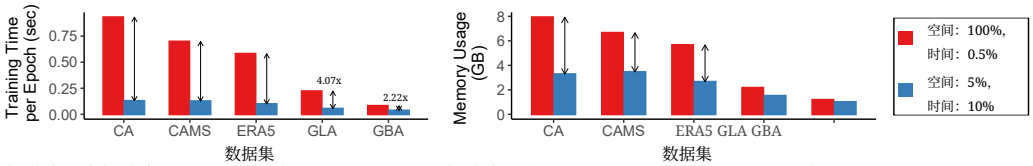

图1. 以均衡方式在时间维度和空间维度上同时压缩的合成数据，与仅沿时间维度压缩的同等规模数据集相比，训练时间和GPU内存使用量显著更低。详见初步实验1。

是设计一种代理模型，使其能够在较少的位置（即经过空间压缩的合成数据）上训练，同时仍适用于在更多位置上进行推理，而大多数现有时空预测模型都不支持这一点。为此，我们将位置编码器模块集成到时空图神经网络（STGNNs）中，该模块可生成与位置数量无关的嵌入。时空图神经网络是广泛使用的时空预测深度学习模型 [26]–[32]。

我们利用带有位置编码器模块的代理模型，在梯度匹配（如上所述）的基础上构建STemDist，并引入两项增强。首先，为提高效率，我们通过在簇级别进行蒸馏来缩短蒸馏时间。具体而言，将原始数据集中的位置分组为簇，并对每个簇内的平均数据进行蒸馏，而非对各个位置的数据进行蒸馏。其次，为弥补这种粗粒度方法可能带来的质量损失，我们采用基于子集的细粒度蒸馏。该技术使用不同的簇子集对数据进行蒸馏，确保数据的所有部分（即子集）都得到充分体现，从而提升输出合成数据集的质量。

我们使用五个真实世界时空数据集和九种基线方法，对 STemDist 进行了全面评估，结果表明，在相同压缩率下，STemDist 相较于基线方法具有以下优势：

- **更快：** 使用 STemDist 蒸馏得到的合成数据进行模型训练，速度最高可提升 6× **倍**，相比之下，使用基线方法蒸馏得到的数据进行训练则较慢。
- **内存效率更高：** 使用 STemDist 蒸馏得到的合成数据进行模型训练，所需的 **GPU 内存最多可减少**，相比之下，使用基线方法蒸馏得到的数据进行训练则需要更多内存。
- **效果更佳：** 使用 STemDist 蒸馏得到的合成数据训练的模型，其预测误差最高可**降低 12%**，相比之下，使用基线方法蒸馏得到的数据训练的模型误差更高。

本文其余部分安排如下：第 II 节介绍相关工作，第 III 节介绍预备知识，第 IV 节介绍我们提出的方法。第 V 节分享实验结果，第 VI 节总结全文。**可复现性：** 代码和数据集可在 https://github.com/kbrother/STemDist 获取。

II. 相关工作

时空时间序列预测： 时空时间序列预测是指根据一组位置的历史观测值 [33]，预测其未来时间序列值的任务。这是时间序列分析中的一项关键任务，需要有效捕捉时间动态和空间依赖关系。近期的进展主要依赖时空图神经网络 (STGNNs) [4]，[34]–[37]，，它们能够联合建模时空依赖关系，并实现最先进的预测性能。以往关于时空图神经网络的研究可分为以下两类：**(1)基于先验的模型** [10]，[38]–[40]，，这类模型使用预定义邻接矩阵（即显式结构信息）来建模位置之间的空间依赖关系；**(2)基于结构学习的模型** [11]，[12]，[25]，[41]，[42]，，这类模型通过专门的图学习模块以端到端的方式推断邻接矩阵。

在本文中，鉴于基于结构学习的模型具有灵活性和可扩展性，我们采用这类模型进行数据集蒸馏。由于邻接矩阵是动态学习的，因此训练或蒸馏过程中无需预定义或存储图结构。然而，大多数时空图神经网络无法直接用作我们蒸馏框架中的代理模型，因为它们要求训练和测试期间的位置数量相同，详见第四节 IV-A。

作为一种尤为相关的基于时空图神经网络的方法，Zhou 等人 [43] 提出了一种归纳式方法，该方法利用显式图结构和已观测位置的嵌入，为未观测位置推断嵌入。这与我们的位置编码器（第 IV-C 节）相关，该编码器同样能够生成位置嵌入。他们的嵌入将知识从已观测位置迁移到未观测位置，而我们的嵌入则在概念上采取相反方向，无需图结构即可实现从合成数据集向真实数据集的知识迁移。

通用数据集蒸馏： 数据集蒸馏旨在将原始数据集转换为小规模合成数据集，使其能够在机器学习模型训练中有效替代原始数据集 [44]。现有方法可分为：（1）**核心集选择方法** [21]，[45]–[48]，，从原始数据集中选取少量代表性样本；（2）**基于梯度匹配的方法** [15]，[20]，[49]–[51]，，缩小模型分别在原始数据集和合成数据集上训练时的梯度差异；（3）**基于轨迹匹配的方法** [16]，[52]，，最小化模型参数在原始数据集和合成数据集上训练所得轨迹之间的差异；（4）**基于元学习的方法** [53]，，直接优化在合成数据集上训练的模型向原始数据集迁移的能力；以及（5）**基于分布匹配的方法** [54]，[55]，，直接对齐以随机参数初始化的模型的输出分布。

我们提出的蒸馏方法 STemDist 基于梯度匹配，这是因为它简单易用，而且在时空时间序列数据集上的实证效果优于其他方法（实证比较请参阅第V-B节）。

时间序列数据集蒸馏： 已有多种专为时间序列数据集开发的数据集蒸馏方法。CondTSF [23] 通过加入一个插件，根据在原始数据集上训练的模型输出对合成数据集进行优化，从而将通用数据集蒸馏技术应用于时间序列预测

表I 常用符号说明

符号	说明
$Z \in \mathbb{R}^{M\times F\times N}$	时空时间序列
$z_t \in \mathbb{R}^{F\times N}$	Z中的第t个观测
M	时间序列总长度
F	（输入）特征数量
N	位置数量
$X_t \in \mathbb{R}^{L_{in}\times F\times N}$	时间 t的输入时间序列
$Y_t \in \mathbb{R}^{L_{out}\times F_{out}\times N}$	时间 t 的目标时间序列
L_{in}, L_{out}	输入窗口和目标窗口长度
F_{out}	目标特征数量
$\mathcal{T}=(\mathcal{X}_\mathcal{T},\ \mathcal{Y}_\mathcal{T})$	时空时间序列数据集
$\mathcal{X}_\mathcal{T} \in \mathbb{R}^{M_\mathcal{T}\times L_{in}\times F\times N_\mathcal{T}}$	$\mathcal{T}$中的输入时间序列列表
$\mathcal{Y}_\mathcal{T} \in \mathbb{R}^{M_\mathcal{T}\times L_{out}\times F_{out}\times N_\mathcal{T}}$	$\mathcal{T}$中的目标时间序列列表
$M_\mathcal{T}$	$\mathcal{T}$中的输入-目标对数量
$N_\mathcal{T}$	$\mathcal{T}$中的位置数量
$\mathcal{S}=\ (\mathcal{X}_\mathcal{S},\ \mathcal{Y}_\mathcal{S})$	合成时间序列数据集
$\mathcal{X}_\mathcal{S} \in \mathbb{R}^{M_\mathcal{S}\times L_{in}\times F\times N_\mathcal{S}}$	$\mathcal{S}$中的输入时间序列列表
$\mathcal{Y}_\mathcal{S} \in \mathbb{R}^{M_\mathcal{S}\times L_{out}\times F_{out}\times N_\mathcal{S}}$	$\mathcal{S}$中的目标时间序列列表
$M_\mathcal{S}$	$\mathcal{S}$中的输入-目标对数量
$N_\mathcal{S}$	$\mathcal{S}$中的位置数量
$\mathcal{X}_\mathcal{T}^{mat} \in \mathbb{R}^{N_\mathcal{T}\times(M_\mathcal{T}L_{in}F)}$	$\mathcal{X}_\mathcal{T}$的重塑版本
$\mathcal{C}$	聚类时间序列数据集
K	位置集合数量
$\{D_k\}_{k=0}^{K-1}$	划分后的位置集合

。TimeDC [24]将分解驱动的频率匹配与轨迹匹配相结合。CondTSC [22]通过感知频率的数据增强改进匹配过程，并联合利用时间信息和频域信息。

然而，值得注意的是，这些方法针对的是单变量或多变量时间序列，而非我们研究重点所在的时空数据集。因此，将它们应用于时空数据时，只会压缩时间维度，而空间维度（即位置数量）保持不变，导致即使经过蒸馏，训练成本仍然很高（见第I节中的初步实验1）。此外，这些方法是为不显式捕捉不同位置之间相关性的模型而设计并应用的 [56], [57] 。另外，CondTSC [22] 是为分类任务开发的，而我们的工作则以预测为目标。这些差异促使我们开发专门用于时空预测的数据集蒸馏方法。

图数据集蒸馏： 已有多项研究 [18], [19]，[58]–[60]探索了图数据集蒸馏。它们着重于减少预定义图拓扑中的节点数量。我们的问题与之不同，因为时空时间序列数据集缺乏预定义的图结构，而且我们的目标是同时压缩空间维度和时间维度。

III. 预备知识与问题定义

本节介绍基本概念和问题的形式化定义，并简要概述梯度匹配。常用符号列于表 I。

A. 基本概念

我们首先将时空时间序列定义如下：

定义 1 (时空时间序列)。 时空时间序列 $Z=(z_0,\cdots,z_{M-1})$ 是由 M 个观测值组成的序列，其中每个观测值 $z_t \in \mathbb{R}^{F\times N}$ 包含来自 N 空间位置的 F 个特征。

遵循先前的研究 [10]–[12], [25], ，我们将时空时间序列预测问题定义为：根据一段过去观测值窗口，预测目标特征的未来值，其中目标特征通常是输入特征的一个子集，如下所示：

问题 1。(时空时间序列预测)

- 给定： 长度为 L_{in} 的过去输入时空时间序列 $X_t=(z_{t-L_{in}},\cdots,z_{t-1}) \in \mathbb{R}^{L_{in}\times F\times N}$，
- 预测： 长度为 L_{out} 的未来目标时空时间序列 $Y_t=(z'_t,\cdots,z'_{t+L_{out}-1}) \in \mathbb{R}^{L_{out}\times F_{out}\times N}$，其中 $z'_t \in \mathbb{R}^{F_{out}\times N}$ 是时间 t 的 F_{out} 目标特征值。

使用上述定义的输入 X_t 和目标 Y_t，我们正式定义时空时间序列数据集如下。

定义2 (时空时间序列数据集)。时空时间序列数据集定义如下：

$$\mathcal{T}=(\mathcal{X}_\mathcal{T},\mathcal{Y}_\mathcal{T})=\big((X_t)_{t=0}^{M_\mathcal{T}-1},(Y_t)_{t=0}^{M_\mathcal{T}-1}\big),$$

其中 $\mathcal{X}_\mathcal{T}=(X_t)_{t=0}^{M_\mathcal{T}-1} \in \mathbb{R}^{M_\mathcal{T}\times L_{in}\times F\times N_\mathcal{T}}$表示输入时空时间序列列表， $\mathcal{Y}_\mathcal{T}=(Y_t)_{t=0}^{M_\mathcal{T}-1} \in\mathbb{R}^{M_\mathcal{T}\times L_{out}\times F_{out}\times N_\mathcal{T}}$ 表示目标时空时间序列列表， $M_\mathcal{T}$表示输入与目标时空时间序列对的数量， $N_\mathcal{T}$ 表示数据集中的位置数量。

从时空时间序列创建时空时间序列数据集的示例见附录 A [61]。

类似地，我们将通过数据集蒸馏得到的合成时空时间序列数据集（下文简称合成数据集）定义如下：

定义3 （合成数据集）。合成时空时间序列数据集定义为

$$\mathcal{S}=(\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{S})=\big((\tilde{X}_i)_{i=0}^{M_\mathcal{S}-1},(\tilde{Y}_i)_{i=0}^{M_\mathcal{S}-1}\big),$$

其中， $\mathcal{X}_\mathcal{S}=(\tilde{X}_i)_{i=0}^{M_\mathcal{S}-1} \in \mathbb{R}^{M_\mathcal{S}\times L_{in}\times F\times N_\mathcal{S}}$ 表示输入合成时间序列列表， $\mathcal{Y}_\mathcal{S}=(\tilde{Y}_i)_{i=0}^{M_\mathcal{S}-1} \in \mathbb{R}^{M_\mathcal{S}\times L_{out}\times F_{out}\times N_\mathcal{S}}$ 表示目标合成时间序列列表， $M_\mathcal{S}$ 表示输入-目标对的数量， $N_\mathcal{S}$表示合成数据集中位置的数量。

B. 问题定义

在基本概念的基础上，我们现在定义时空时间序列预测的数据集蒸馏问题。令 g_θ 表示一个（预定义或任意的）用于时空时间序列预测的机器学习模型，其参数为 θ；$\theta_\mathcal{S}$ 是在 $\mathcal{S}$上训练得到的参数集合。数据集蒸馏旨在构建一个合成

数据集 $\mathcal{S}$，使得在其上训练得到的模型 $g_{\theta_\mathcal{S}}$在真实数据分布 P上取得最佳性能，即

$$\min_{\mathcal{S}} \mathbb{E}_{(X,Y)\sim P}[\ell(g_{\theta_\mathcal{S}}(X), Y)], \tag{1}$$

其中，ℓ 是一个损失函数（例如平方误差和）。由于实际中未知真实数据分布 P，我们使用给定的数据集 $\mathcal{T}$将问题表述如下:

问题2。（时空时间序列预测的数据集蒸馏）

- 给定：一个时空时间序列数据集 $\mathcal{T}$,
- 求：一个合成数据集 $\mathcal{S}$,
- 使**.....**最小化： $g_{\theta_\mathcal{S}}$ 在 $\mathcal{T}$上的预测误差，即

$$\min_{\mathcal{S}} \mathcal{L}(g_{\theta_\mathcal{S}}, \mathcal{T}) \quad \textit{subject to} \quad \theta_\mathcal{S} = \arg\min_\theta \mathcal{L}(g_\theta, \mathcal{S}),$$

其中，$\mathcal{L}(g_{\theta_\mathcal{S}}, \mathcal{T}) = \frac{1}{M_\mathcal{T}} \sum_{t=0}^{M_\mathcal{T}-1} \ell(g_{\theta_\mathcal{S}}(X_t), Y_t)$是 $\frac{1}{M_\mathcal{S}} \sum_{i=0}^{M_\mathcal{S}-1} \ell(g_\theta(\tilde{X}_i), \tilde{Y}_i)$ 在 $\mathcal{T}$上的平均预测误差；而 $\mathcal{L}(\mathcal{S}, g_\theta)$则是.....在.....上的平均预测误差。

C. 梯度匹配

本节简要回顾梯度匹配 [15], ，这是我们的方法所基于的一种数据集蒸馏方法。

梯度匹配旨在学习合成数据，使得在其上训练得到的模型参数 $\theta_\mathcal{S}$ 能够收敛到一个接近于在原始数据上进行模型训练所获得的参数 $\theta_\mathcal{T}$ 的解。然而， $\theta_\mathcal{S}$ 通常会收敛到不同于 $\theta_\mathcal{T}$的局部极小值，因此直接与 $\theta_\mathcal{T}$对齐并不起作用。梯度匹配通过在 $\theta_\mathcal{S}$的每次更新时对齐在 $\mathcal{T}$和 $\mathcal{S}$ 上计算的梯度来解决这一问题，即使在收敛之前也是如此，而且不依赖于 $\theta_\mathcal{T}$的值。

为形式化梯度匹配目标，我们将代理机器学习模型 f_θ 经过 t个优化步骤后的参数记为 θ_t，其初始化参数为 θ_0。该目标是在从 $t=0$ 到 $T_{max}-1$的优化轨迹上，最小化在合成数据 $\mathcal{S}$ 和原始数据 $\mathcal{T}$ 上计算得到的梯度之间的距离，即:

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0\sim P_{\theta_0}} \left[\sum_{t=0}^{T_{max}-1} Dist(\nabla_\theta \mathcal{L}(\mathcal{S}, \theta_t), \nabla_\theta \mathcal{L}(\mathcal{T}, \theta_t)) \right],$$

$$\text{subject to} \quad \theta_t \leftarrow \theta_{t-1} - \eta \nabla_\theta \mathcal{L}(S, \theta_{t-1}),$$

其中， P_{θ_0} 是初始参数的分布， η 是学习率， $Dist$ 是两个梯度之间余弦距离的总和， $\mathcal{L}$是训练损失函数。

IV. 所提方法

本节介绍我们提出的用于时空时间序列预测的数据集蒸馏方法 STemDist（**时**空-**时间**数据集**蒸馏**）。我们首先概述解决问题2的关键挑战及相应的解决方案（第 IV-A 节）然后，介绍 STemDist 的总体情况（第 IV-B 节），并描述其三个关键组成部分：位置编码器（第 IV-C 节）、聚类（第 IV-D 节）和粒度蒸馏（第 IV-E 节）。最后，我们进行复杂度分析（第 IV-F 节）。

A. 关键挑战及相应的解决方案

解决问题2面临三个关键挑战:

- **C1. 现有方法无法充分降低训练成本**。现有数据集蒸馏方法仅专注于压缩单一维度，即本文语境中的时间维度（即定义3中的 $M_\mathcal{S}$ ）。然而，预测模型的训练成本还受到位置数量的影响（即定义3中的 $N_\mathcal{S}$ ），而位置数量会使计算成本呈二次增长。因此，在不压缩空间维度（即位置数量）的情况下，训练成本仍然十分高昂。

- **C2：数据集蒸馏本身耗时较长**。数据集蒸馏的最终目标是降低与模型训练相关的总体计算成本。因此，如果蒸馏过程本身成本过高，那么无论其性能如何，数据集蒸馏方法都无法实现其预期目的。因此，蒸馏成本通常会随着位置数量（即定义2中的$N_\mathcal{T}$）的增加而大幅增长，必须保持计算高效。

- **C3：合成数据集可能无法充分体现原始数据**。真实世界的时空时间序列通常采集自大量位置。每个位置的数据都应得到有效蒸馏，以便在合成数据上有效训练模型。然而，位置数量较多时，要在蒸馏过程中兼顾所有位置会变得颇具挑战，尤其是考虑到合成数据集的容量有限。

为应对这些挑战，我们提出的方法 STemDist 引入了三种相应的解决方案:

- **S1. 使用位置编码器同时压缩时间维度和空间维度**。为解决 C1，我们联合压缩时间维度和空间维度（即定义3中的$M_\mathcal{S}$ 和 $N_\mathcal{S}$ ）。据我们所知，我们是首个提出这种双维数据集蒸馏策略的团队。实现这一构想的关键是引入位置编码器（见第四节-C），使在较少位置上训练的深度学习模型能够应用于更多位置的推理。联合压缩两个维度可生成兼顾这两方面的合成数据集，不仅降低训练成本，还能提升模型性能。

- **S2. 原始数据集中的位置聚类**。为解决 C2，我们通过聚类减少原始数据集中的位置数量（即定义2中的 $N_\mathcal{T}$）。各个位置的时间序列将替换为相应聚类中心的时间序列，从而得到精简版原始数据集，详情见第四节-D。对该聚类数据集进行蒸馏可加快整个蒸馏过程。请注意，减少原始数据集中的位置数量（$N_\mathcal{T}$）这一做法不同于此前减少合成数据集中的位置数量（$N_\mathcal{S}$）的做法。

- **S3. 基于子集的细粒度蒸馏**。 为应对 C3，我们提出基于子集的细粒度蒸馏。蒸馏过程并非一次性使用全部位置，而是利用不断变化的位置子集。这使得不同部分（即

子集）的数据能够更有效地反映在合成数据集中，从而提升其整体质量。需要注意的是，这一思路通过缓解聚类可能造成的质量损失，对基于聚类的粗粒度方法形成补充。

这些解决方案体现了我们方法的核心思想，其细节见第 IV-C、IV-D 和 IV-E 节。

B. STemDist 概述

在本小节中，我们概述 STemDist，其整体流程总结于算法 1。

第 1–3 行描述蒸馏前的预处理步骤。首先，对原始数据集 $\mathcal{T}$ 中的位置进行聚类，以获得空间缩减版本 $\mathcal{C}=(\mathcal{X}_\mathcal{C},\mathcal{Y}_\mathcal{C})$，以及与每个聚类中位置数量成比例的聚类权重 $(w_i)_{i=0}^{N_\mathcal{S}-1}$（第 1 行）。聚类数据集 $\mathcal{C}$ 在后续步骤中替换$\mathcal{T}$，从而加速蒸馏过程（见第 IV-A 节中的**S2**）。聚类过程的细节见第四节（见算法 3）。然后，位置编码器的输入（见第 IV-A 节中的**S1**）$I_\mathcal{C}$ 预先由 $\mathcal{X}_\mathcal{C}$ 计算得到（第 2 行）并在后续步骤中使用。合成数据集 $\mathcal{S}=(\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{S})$ 通过从 $\mathcal{C}$ 随机采样进行初始化（第 3 行），从而自然地将每个聚类分配给 $\mathcal{S}$中的一个合成位置。[1]

第 5 至 17 行描述了蒸馏过程。给定预处理后的输入（$\mathcal{C}$、$I_\mathcal{C}$和初始化的 $\mathcal{S}$），该过程输出蒸馏后的合成数据 $\mathcal{S}$。由于聚类数据集 $\mathcal{C}$ 在时间维度上可能很大，我们使用沿时间维度从 $\mathcal{C}$ =（$\mathcal{X}_\mathcal{C}$、$\mathcal{Y}_\mathcal{C}$）中采样得到的小批量对（$\mathcal{X}_\mathcal{C}^B$、$\mathcal{Y}_\mathcal{C}^B$）进行蒸馏（第 7 行）。对于基于子集的细粒度蒸馏（见第四节中的**S3** 第 IV-A 节），我们将位置划分为 K 个互不相交的子集 $\{D_k\}_{k=0}^{K-1}$（第 8 行）。

对于每个位置子集 D_k，我们执行梯度匹配（见第 III-C 节），以更新对应合成位置的合成时间序列（第 10 至 16 行），并采用配备了我们的位置编码器模块（见第 IV-C 节）的时空图神经网络作为代理模型 f_θ。将聚类数据集 $\mathcal{C}$ 中的输入时间序列小批次 $\mathcal{X}_\mathcal{C}^B$以及合成数据集 $\mathcal{S}$ 中的输入时间序列 $\mathcal{X}_\mathcal{S}$ 连同位置编码器的输入 $I_\mathcal{C}$ 和 $I_\mathcal{S}$一起提供给模型f_θ（第 11 和 13 行）。然后，考虑分配给每个合成位置的簇权重 $(w_i)_{i=0}^{N_\mathcal{S}-1}$，计算 $\mathcal{C}$ 和 $\mathcal{S}$ 的损失。之后，测量梯度之间的差异，得到梯度匹配损失（第 15 行），并通过反向传播更新合成数据集 $\mathcal{S}$（第 16 行）。更新完所有位置子集对应的合成数据集后，使用当前合成数据集更新 f_θ的参数（第 17 行）。

代理模型 f_θ 在合成数据集 $\mathcal{S}=(\mathcal{X}_\mathcal{S},\ \mathcal{Y}_\mathcal{S})$ 上的详细训练过程见算法2。给定 $\mathcal{S}$，我们首先计算位置编码器的输入 $I_\mathcal{S}$（第 1 行）。对于每个训练轮次 t，将合成数据集中的位置随机划分为 K

[1]随着蒸馏的推进，合成位置的语义可能会变得模糊，因为蒸馏旨在优化训练等效性，而非可解释性。因此，应将合成位置视为训练效率高的代理，而非具有语义意义的位置。

算法 1： STemDist 中的数据集蒸馏

输入： (1) 原始数据集 $\mathcal{T}=(\mathcal{X}_\mathcal{T},\ \mathcal{Y}_\mathcal{T})$,
(2) 外层迭代次数 T_{outer}，
(3) 蒸馏迭代次数 $T_{distill}$，
(4) 用于模型训练的轮次数量 T_{model}，
(5) 位置集合的数量 K，
(6) 学习率 η，用于 θ_t。
输出： 合成数据集 $\mathcal{S}=(\mathcal{X}_\mathcal{S},\ \mathcal{Y}_\mathcal{S})$。

1 对 $\mathcal{T}$ 中的位置进行聚类，得到聚类数据集 $\mathcal{C}=(\mathcal{X}_\mathcal{C},\ \mathcal{Y}_\mathcal{C})$ 和聚类权重 $(w_i)_{i=0}^{N_\mathcal{S}-1}$ ▷算法3和第IV-D节
2 根据 $\mathcal{X}_\mathcal{C}$计算位置编码器的输入 $I_\mathcal{C}$ ▷ 第 IV-C
3 从（$\mathcal{X}_\mathcal{C}$, $\mathcal{Y}_\mathcal{C}$）中随机采样，初始化 $\mathcal{S}=$（$\mathcal{X}_\mathcal{S}$, $\mathcal{Y}_\mathcal{S}$）
4 **对于** $t_{outer}\leftarrow 0,\ \cdots,\ T_{outer}-1$，**执行**
5 使用 $\theta_0\sim P_{\theta_0}$初始化代理模型 f_θ
6 **对于** $t\leftarrow 0,\ \cdots,\ T_{distill}-1$ **执行**
7 采样一个小批量对$\mathcal{X}_\mathcal{C}^B\sim\mathcal{X}_\mathcal{C}$ 和 $\mathcal{Y}_\mathcal{C}^B\sim\mathcal{Y}_\mathcal{C}$
8 位置分区 $\{0,\cdots,N_\mathcal{S}-1\}$ 为 $\{D_k\}_{k=0}^{K-1}$ ▷第 IV-E 节
9 **对于** $k\leftarrow 0,\ \cdots,\ K-1$ **执行**
10 计算位置编码器的输入 $I_\mathcal{S}$ 来自 $\mathcal{X}_\mathcal{S}$ ▷第 IV-C 节
11 $\tilde{\mathcal{Y}}_\mathcal{C}\leftarrow f_{\theta_t}(\mathcal{X}_\mathcal{C}^B(:,\ :,\ :D_k)\ I_\mathcal{C}(D_k,\ :))$
12 $\mathcal{L}(\mathcal{C},\theta_t)\leftarrow\sum_{i\in D_k}w_i\|(\tilde{\mathcal{Y}}_\mathcal{C}-\mathcal{Y}_\mathcal{C}^B)(:,:,:i)\|_2^2$
13 $\tilde{\mathcal{Y}}_\mathcal{S}\leftarrow f_{\theta_t}(\mathcal{X}_\mathcal{S}(:,\ :,\ :D_k)\ I_\mathcal{S}(D_k,\ :))$
14 $\mathcal{L}(\mathcal{S},\theta_t)\leftarrow\sum_{i\in D_k}w_i\|(\tilde{\mathcal{Y}}_\mathcal{S}-\mathcal{Y}_\mathcal{S})(:,:,:i)\|_2^2$
15 $\mathcal{L}^{grad}\leftarrow Dist(\nabla_\theta\mathcal{L}(\mathcal{C},\theta_t)\ \nabla_\theta\mathcal{L}(\mathcal{S},\theta_t))$
16 反向传播 $\mathcal{L}^{grad}$并更新 $\mathcal{X}_\mathcal{S}(:,\ :,\ :D_k)$ 并 $\mathcal{Y}_\mathcal{S}(:,\ :,\ :D_k)$ ▷ 节。 IV-E
17 $\theta_{t+1}\leftarrow$ 使用（$f_\theta\ \mathcal{X}_\mathcal{S},$）训练代理模型 $\mathcal{Y}_\mathcal{S},T_{model},K,\eta,\theta_t,$和$(w_i)_{i=0}^{N_\mathcal{S}-1}$） ▷算法2
18 **返回**$(\mathcal{X}_\mathcal{S},\ \mathcal{Y}_\mathcal{S})$

算法2： STemDist中的（代理）模型训练

输入： （1）一个合成数据集 $\mathcal{S}=$ （$\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{S}$），
（2）训练轮次数量 T，
(3) 若干组位置集合，
(4) 学习率 η、初始参数 θ_0,
(5) 聚类权重（w_i）$_{i=0}^{N_\mathcal{S}-1}$。
输出：（代理）模型的训练后参数 θ_T f_{θ_0}。

1 计算位置编码器的输入 $I_\mathcal{S}$ $\mathcal{X}_\mathcal{S}$ ▷第 IV-C 节
2 **为** $t\leftarrow 0,\cdots,T-1$ **执行**
3 $\{D_k\}_{k=0}^{K-1}\leftarrow$位置分区 $\{0,\cdots,N_\mathcal{S}-1\}$ ▷ 第 IV-E 节
4 **对于** $k\leftarrow 0,\cdots,K-1$ **执行**
5 $\tilde{\mathcal{Y}}_\mathcal{S}\leftarrow f_{\theta_t}(\mathcal{X}_\mathcal{S}(:,\ :,\ :D_k)\ I_\mathcal{S}(D_k,\ :))$
6 $\mathcal{L}(\mathcal{S},\ \theta_t)\leftarrow\sum_{i\in D_k}w_i\|(\tilde{\mathcal{Y}}_\mathcal{S}-\mathcal{Y}_\mathcal{S})(:,:,:i)\|_2^2$
7 $\theta_t\leftarrow\theta_t-\eta\nabla_\theta\mathcal{L}(\mathcal{S},\ \theta_t)$
8 $\theta_{t+1}\leftarrow\theta_t$
9 **返回** θ_T

互不相交的子集（第 3 行），以与蒸馏阶段所采用的子集级粒度保持一致。对于每个子集D_k，模型 f_θ 会为合成数据集对应的输入生成预测（第 5 行）。随后计算蒸馏过程中同样使用的加权损失（第 6 行），最后通过梯度下降更新模型参数（第 7 行）。

C. 位置编码器

在本小节中，我们详细介绍了于第 IV-A 节提出的第一种方案，**S1**。具体而言，我们介绍了位置编码器模块，该模块用于增强时空图神经网络（见 II 节），这些网络在蒸馏过程中充当代理模型 f_θ，更广泛地说，也是用于时空时间序列预测的强大模型。

如第 IV-A 节的**C1** 所述，尽管在数据集蒸馏中压缩空间维度（以及时间维度）是可取的，但这并非易事，因为在位置数量较少的（合成）数据集上训练的时空图神经网络（以及大多数其他方法）无法在位置数量更多的情况下进行推理。为了解决这一问题，我们引入一种位置编码器，使时空图神经网络能够针对位置数量进行归纳式处理，具体如下所述。

时空图神经网络的传导性：我们首先介绍时空图神经网络中的相关组成部分，这些部分导致当训练与推理时的位置不同时，模型无法适用。在没有预定义图（即位置之间的空间依赖关系）的情况下，时空图神经网络 [11], [12], [25] 通过传导式（即可直接学习的）位置嵌入来学习图结构。例如，在 AGCRN [12] 和 Graph WaveNet [11],中，邻接矩阵 A 的推导方式如下：

$$A \leftarrow \mathrm{Softmax}(\mathrm{ReLU}(E_1^T E_2)), \tag{2}$$

其中，$E_1 \in \mathbb{R}^{N\times R}$ 和 $E_2 \in \mathbb{R}^{N\times R}$ 是维度为 R 的传导式嵌入，适用于 N 个位置；Softmax 和 ReLU 是激活函数。请注意，训练后的参数与训练数据集中的位置直接相关（即，在我们的数据集蒸馏场景中，是合成数据集中的位置），因此时空图神经网络具有传导性。也就是说，当位置的语义发生变化，或在推理过程中引入新位置时，学得的嵌入及其导出的邻接矩阵便不再适用。

位置编码器概述：为了解决这一局限，我们采用位置编码器来替代传导式位置嵌入。具体而言，对于位置数量任意的时空时间序列，该编码器旨在为数据集中的每个位置生成嵌入。尤其是，即使仅使用合成数据集 $\mathcal{S}$中较少数量的（合成）位置进行训练，该编码器也应能够在推理时泛化到原始数据集 $\mathcal{T}$中的更多位置。

为此，我们将位置编码器设计为序列到序列架构，因为输入（来自多个位置的时间序列）和输出（对应的位置嵌入）都可以自然地表示为按位置索引的序列。编码器还必须满足**（1）长度不敏感性：**它应能处理任意长度的序列，从而能够处理位置数量不同的合成数据集和原始数据集；以及**（2）参数共享：**它使用共享参数生成每个位置的嵌入，从而确保在位置数量变化时保持一致性和泛化能力。请注意，尽管我们侧重于序列到序列的设计，但满足这些性质的其他架构也可能适用。

位置编码器的输入和输出：为了获得位置编码器的输入，我们在使用原始数据 $\mathcal{T}$时对 $\mathcal{X}_\mathcal{T} \in\mathbb{R}^{M_\mathcal{T}\times L_{in}\times F\times N_\mathcal{T}}$ 进行预处理，在使用合成数据 $\mathcal{S}$ 时对 $\mathcal{X}_\mathcal{S} \in\mathbb{R}^{M_\mathcal{S}\times L_{in}\times F\times N_\mathcal{S}}$ 进行预处理（ $\mathcal{X}_\mathcal{T}$ 和 $\mathcal{X}_\mathcal{S}$请参见定义 2 和定义 3）。由于它们可能过长，我们首先沿时间¯ ¯维度对其取平均值。也就是说， $\mathcal{X}_\mathcal{T}$和 $\mathcal{X}_\mathcal{S}$ 的计算方式如下：

$$\bar{\mathcal{X}}_{(\mathcal{T},\mathcal{S})} \leftarrow \frac{1}{M_{(\mathcal{T},\mathcal{S})}} \sum\nolimits_{m=0}^{M_{(\mathcal{T},\mathcal{S})}-1} \mathcal{X}_{(\mathcal{T},\mathcal{S})}(m,:,:,:). \tag{3}$$

然后，我们将 $\bar{\mathcal{X}}_\mathcal{T} \in \mathbb{R}^{L_{in}\times F\times N_\mathcal{T}}$和 $\bar{\mathcal{X}}_\mathcal{S} \in \mathbb{R}^{L_{in}\times F\times N_\mathcal{S}}$分别重塑为矩阵 $I_\mathcal{T} \in \mathbb{R}^{N_\mathcal{T}\times L_{in}F}$和 $I_\mathcal{S} \in \mathbb{R}^{N_\mathcal{S}\times L_{in}F}$，并将其用作位置编码器的输入。

给定输入 $I_\mathcal{T}$（或 $I_\mathcal{S}$），位置编码器会输出矩阵 $E_\mathcal{T} \in \mathbb{R}^{N_\mathcal{T}\times R}$（或 $E_\mathcal{S} \in \mathbb{R}^{N_\mathcal{S}\times R}$），其中每一行表示对应位置的一个 R 维嵌入。这些输出矩阵用于构建邻接矩阵，以替代传导式位置嵌入，并遵循时空图神经网络的结构学习范式。**位置编码器的详细架构：**位置编码器由一个单头自注意力层 [62]及其前后连接的线性层组成。在注意力层周围应用残差连接，随后进行层归一化 [63]。为清晰起见，我们使用$I_\mathcal{T}$来表示方程，但也可以用 $I_\mathcal{S}$表示。

$$\begin{aligned} H_1 &= \mathrm{Linear}_1(I_\mathcal{T}),\ H_2 = \mathrm{SelfAttn}(H_1),\\ E_\mathcal{T} &= \mathrm{Linear}_2\big(\mathrm{LayerNorm}(H_1 + H_2)\big). \end{aligned} \tag{4}$$

隐藏向量记为 $H_1 \in \mathbb{R}^{N_\mathcal{T}\times d}$ 和 $H_2 \in\mathbb{R}^{N_\mathcal{T}\times d}$，其中 d 表示隐藏维度。通过自注意力层（SelfAttn），计算 H_2的第 i行，即第 i个位置的隐藏向量，计算如下：

$$H_2(i,:) = \sum\nolimits_{j=1}^{N_\mathcal{T}} \alpha(i,j) H_1(j,:) W_V W_O, \text{ where} \tag{5}$$

$$\alpha(i,j) = \frac{\exp\big((H_1(i,:)W_Q)\cdot(H_1(j,:)W_K)^T/\sqrt{d}\big)}{\sum_{n=1}^{N_\mathcal{T}} \exp\big((H_1(i,:)W_Q)\cdot(H_1(n,:)W_K)^T/\sqrt{d}\big)}.$$

注意，W_Q，W_K，W_V和 $W_O \in \mathbb{R}^{d\times d}$ 均为可学习参数。自注意力层计算每个第 j个位置相对于第 i个位置的重要性分数 $\alpha(i, j)$。随后，使用 $\alpha(i, j)$ 作为权重，对从每个第 j个位置提取的信息 $H_1(j, :)$ W_V进行聚合，从而生成第 i个位置的隐藏向量 $H_2(i, :)$。因此，得到的隐藏向量 $H_2(i, :)$ 汇集了来自所有位置的信息，并根据这些位置与第 i个位置的学习相关性进行加权。

这种架构满足位置编码器所需的上述两个属性。式 (4) 和式 (5) 中的所有组件均**不受序列长度影响**，能够处理任意长度的序列，且参数规模与序列长度无关。此外，所有参数（即线性层、层归一化以及 W_Q、W_K、W_V 和 W_O 的参数）均在所有位置间共享。因此，**参数共享**得以实现，从而确保在位置数量不同时保持一致性和泛化能力。因此，我们的位置编码器

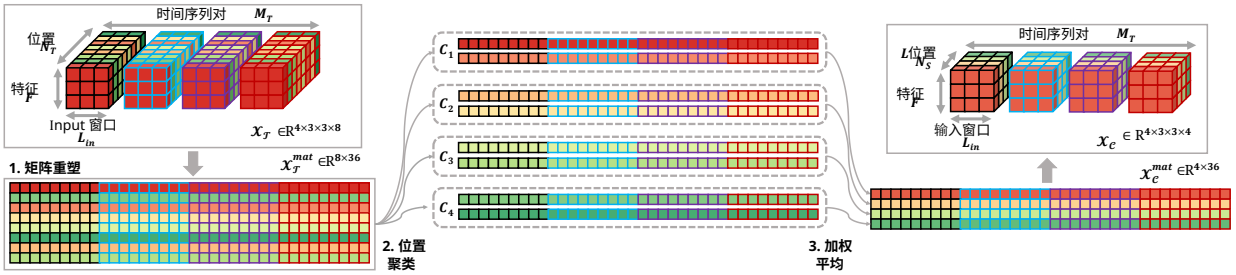

图2。 位置聚类示例 ，其中 $\mathcal{X}_T$ 中的八个位置被缩减为 $\mathcal{X}_C$中的四个位置（聚类中心）。颜色相同的条目对应于 同一位置的数据。详情请参阅示例1。

算法 3： STemDist 中的位置聚类

输入： (1) 原始数据集 $(\mathcal{X}_T, \mathcal{Y}_T)$
(2) T_{kmeans} 次K-means迭代
输出： (1) 聚类数据集 $(\mathcal{X}_C,\ \mathcal{Y}_C)$
(2) 聚类权重 $(w_i)_{i=0}^{N_S-1}$

1 将 $\mathcal{X}_T$ 重塑为$\mathcal{X}_T^{mat}$，并将 $\mathcal{Y}_T$ 重塑为 $\mathcal{Y}_T^{mat}$
2 $\{C_i\}_{i=0}^{N_S-1} \leftarrow$ K-means($\mathcal{X}_T^{mat}$, T_{kmeans})
3 **对于** $i \leftarrow 0$ 到 N_S-1 **执行**
4 $(\mathcal{X}_C^{mat},\ \mathcal{Y}_C^{mat})(i,\ :\ ) \leftarrow \frac{1}{|C_i|}\sum_{l\in C_i}(\mathcal{X}_T^{mat},\ \mathcal{Y}_T^{mat})(l,\ :\ )$
5 $w_i \leftarrow \dfrac{|C_i|}{N_T}$
6 将 $\mathcal{X}_C^{mat}$ 重塑为 $\mathcal{X}_C$，并将 $\mathcal{Y}_C^{mat}$ 重塑为 $\mathcal{Y}_C$
7 **返回** $(\mathcal{X}_C,\ \mathcal{Y}_C)$，$(w_i)_{i=0}^{N_S-1}$

使时空图神经网络能够适用于位置数量不同的数据集，并在这些数据集上有效运行，其中包括位置数量多于合成训练数据集的原始数据集。

D. 位置聚类

如第 IV-A 节**C2**所述，为了高效蒸馏，有必要减少原始数据集中的位置数量。我们通过算法 3 中描述的位置聚类（**A2**）来解决这一问题，该方法包含三个步骤：

1) 我们将原始数据集的输入-目标时间序列对（$\mathcal{X}_T, \mathcal{Y}_T$）$\in$（$\mathbb{R}^{M_T\times L_{in}\times F\times N_T}, \mathbb{R}^{M_T\times L_{out}\times F_{out}\times N_T}$）重塑为矩阵（$\mathcal{X}_T^{mat}$, $\mathcal{Y}_T^{mat}$）$\in$（$\mathbb{R}^{N_T\times(M_T L_{in} F)}\ \mathbb{R}^{N_T\times(M_T L_{out} F_{out})}$）（第 1 行）。$\mathcal{X}_T^{mat}$、$\mathcal{Y}_T^{mat}$的每一行对应于 $\mathcal{T}$中的一个位置。

2) 然后，我们将 $\mathcal{X}_T^{mat}$的每一行视为某个位置的特征向量，并在 $\mathcal{X}_T^{mat}$上应用 K-means 聚类，将N_T个位置划分为 N_S 个簇（第 2 行）。对于每个簇 C_i，我们对分配到该簇 C_i的所有位置的特征向量取均值，以计算其代表性特征向量，得到 $\mathcal{X}_C^{mat}$（i,：）和 $\mathcal{Y}_C^{mat}$（i,：）（第 4 行）。

3) 最后，我们将获得的簇特征（$\mathcal{X}_C^{mat}, \mathcal{Y}_C^{mat}$）$\in$（$\mathbb{R}^{N_S\times(M_T L_{in} F)}\ \mathbb{R}^{N_S\times(M_T L_{out} F_{out})}$）重塑为原始维度（$\mathcal{X}_C, \mathcal{Y}_C$）$\in$（$\mathbb{R}^{M_T\times L_{in}\times F\times N_S}, \mathbb{R}^{M_T\times L_{out}\times F_{out}\times N_S}$），从而得到聚类数据集 $\mathcal{C}$。我们将（$\mathcal{X}_C, \mathcal{Y}_C$）视为来自缩减后位置的输入—目标时间序列对，每一对代表一个簇。

示例1。图 *2* 展示了输入时间序列 $\mathcal{X}_T$的聚类过程。对目标时间序列 $\mathcal{Y}_T$应用相同的步骤，以获得 $\mathcal{Y}_C$。颜色相同的条目对应同一位置。首先，将 $\mathcal{X}_T \in \mathbb{R}^{4\times3\times3\times8}$重塑为 $\mathcal{X}_T^{mat} \in \mathbb{R}^{8\times36}$。我们对 $\mathcal{X}_T^{mat}$的行进行聚类，并利用结果构建 $\mathcal{X}_C^{mat}$。在此示例中，将 $\mathcal{X}_T^{mat}(0,:)$和 $\mathcal{X}_T^{mat}(2,:)$求平均，得到 $\mathcal{X}_C^{mat}(0,:)$；将 $\mathcal{X}_T^{mat}(3,:)$和 $\mathcal{X}_T^{mat}(6,:)$求平均，得到 $\mathcal{X}_C^{mat}(1,:)$；将 $\mathcal{X}_T^{mat}(4,:)$和$\mathcal{X}_T^{mat}(7,:)$求平均，得到 $\mathcal{X}_C^{mat}(2,:)$；将 $\mathcal{X}_T^{mat}(1,:)$和 $\mathcal{X}_T^{mat}(5,:)$求平均，得到 $\mathcal{X}_C^{mat}(3,:)$。最后，将大小为4×36 的 $\mathcal{X}_C^{mat}$重塑为大小为 $4\times3\times3\times4$的 $\mathcal{X}_C$。

$\mathcal{C}$ 中的簇（即缩减后的位置）可能对应于 $\mathcal{T}$中数量不同的位置。为了解决这种不平衡，我们分配权重 $\{w\}_{i=0}^{N_S-1}$，其中每个权重定义为相应簇中包含的位置数量与 $\mathcal{T}$中位置总数之比（第 5 行）。这些权重用于后续蒸馏过程中的损失计算（算法 1 的第 12 行和第 14 行，以及算法2 的第 6 行）。

如第四节-F 节的复杂度分析所示，蒸馏成本随位置数量呈二次增长。以聚类后的数据集替代原始数据集，可以减少位置数量，从而提升蒸馏的速度和效率。除了有效之外，基于子集的粒度蒸馏还通过减少每次处理的位置数量，降低了 STemDist 的时间复杂度和空间复杂度（详情参见第四节-F 节中的定理 3）。**K-means 聚类的替代方法：** K-means 可以替换为其他聚类方法，例如 K-medoids [64]。[2] 由于简单的聚类方法可能因忽视地理关系而模糊局部拓扑结构，因此，当位置的地理信息（例如经度和纬度）已知时，可以改用空间信息聚类方法 [65],[66]，如第五节-E 节所探讨。

E. 基于子集的细粒度蒸馏

在本小节中，我们介绍基于子集的细粒度蒸馏（第 IV-A 节中的**S3** ）。我们的方法会在更少且随机选取的位置子集上重复进行蒸馏。这样一来，同时对所有位置进行蒸馏（第 IV-A 节中的**C3** ）时常被忽略的较弱但重要的位置间相关性，也能有效地反映在合成数据集中。因此，合成数据集的质量得以提升，正如我们在第五节 V-E 中通过实验证明的那样。下面，我们将介绍基于子集的细粒度蒸馏的具体细节。

在每次蒸馏迭代中，我们会将聚类数据集 $\mathcal{C}$ 中的位置集合 $\{0, ..., N_S-1\}$ 随机划分为（或

[2]Zorrilla 等人 [64] 使用 K-medoids 对位置进行聚类，并为每个聚类训练一个代表性模型（具体而言为 ARIMA）。在线阶段，会为每个查询位置选择合适的代表性模型来进行预测。这种策略无需为每个位置单独训练模型，从而降低了总体训练成本。他们通过训练多个特定于聚类的模型以供选择，这与我们的方法形成对比：我们使用规模缩减后的合成数据集训练单个模型（具体而言为时空图神经网络），并将所有位置及其空间关系纳入模型。

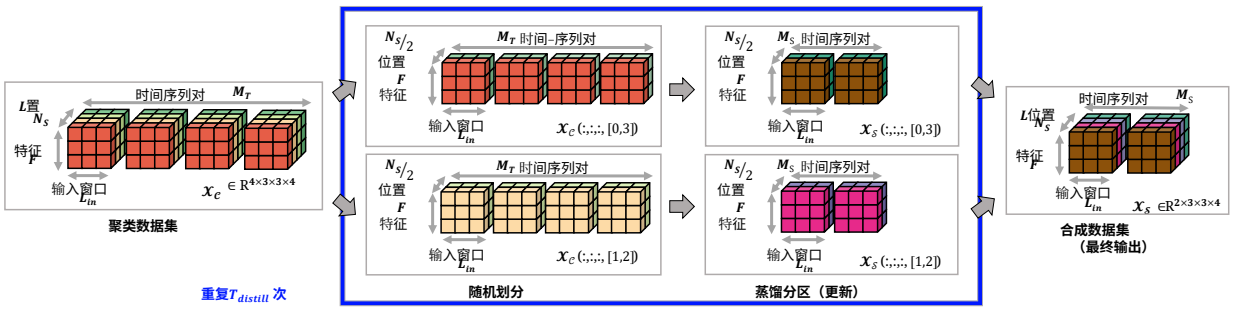

图 3. 示例说明聚类时空数据集（在每个聚类内取平均值）如何被划分为两个位置子集，并在子集层面进行蒸馏。颜色相同的条目对应同一位置的结果。详情请参阅示例 2。

合成数据集 $\mathcal{S}$划分为 K 大小相同且互不重叠的子集 $\{D_k\}_{k=0}^{K-1}$ $\frac{N_\mathcal{S}}{K}$—(算法 1 第 8 行；另见示例 2）。每次迭代时，我们都会重新划分位置，形成一组新的不同子集，以便在蒸馏过程中捕获位置之间多样的空间相关性。

示例 2。图 *3* 展示了一个划分示例，其中位置被划分为两个子集（$K = 2$）：$D_0 = \{0, 3\}$ 和 $D_1 = \{1, 2\}$。 $\mathcal{X}_\mathcal{C}$ 和 $\mathcal{Y}_\mathcal{C}$ 也相应地被拆分，每对子集 $(\mathcal{X}_\mathcal{C}, \mathcal{Y}_\mathcal{C})(:, :, : D_k)$ 用于蒸馏对应的合成子集$(\mathcal{X}_\mathcal{S}, \mathcal{Y}_\mathcal{S})(:, :, : D_k)$。

划分后的子集有四种用途。首先，它们用于从原始输入（即 $\mathcal{X}_C^B$、 $\mathcal{X}_\mathcal{S}$、 I_C 和 I_S）构造代理模型的子集级输入。其次，接收这些子集-˜ ˜级输入的代理模型 f_θ 会生成子集级输出， $\mathcal{Y}_\mathcal{C}$ 和 $\mathcal{Y}_\mathcal{S}$（第 11 行和第 13 行）。第三，它们用于计算子集级损失。对于第 k 个子集 D_k，损失是在 D_k 内计算的，方法是将其组成位置 $i \in D_k$ 的重构误差相加（第 12 行和第 14 行）。最后，根据这些损失计算出的合成数据集和聚类后的原始数据集的梯度，将用于更新各子集对应的合成时间序列 $\mathcal{X}_\mathcal{S}(:, :, : D_k)$ 和 $\mathcal{Y}_\mathcal{S}(:, :, : D_k)$（第 15 行和第 16 行）。

F. 复杂度分析

在本小节中，我们分析上述算法的时间和空间复杂度：（a）在合成数据集上进行模型训练（算法2），以及（b）数据集蒸馏（算法 1）。位置聚类的复杂度在附录 B [61]中分析。以下回顾分析中常用的关键符号。我们用 $N_\mathcal{T}$ 和$N_\mathcal{S}$ 分别表示原始数据集和合成数据集中的位置数量。原始数据集和合成数据集中的时间序列数量分别表示为 $M_\mathcal{T}$ 和$M_\mathcal{S}$。输入窗口大小表示为 L_{in}，目标窗口大小表示为 L_{out}，输入特征数量表示为 F，位置子集的数量表示为 K。**我们假设， [25]配备我们的位置编码器的 MTGNN 同时用于蒸馏（作为代理模型 f_θ）和训练；为简洁起见，我们省略视为常数的超参数**。

在合成数据集上进行模型训练： 我们首先分析使用 STemDist 生成的合成数据集进行模型训练的复杂度。详细证明和分析见附录 B [61]。

定理1和定理2给出了在合成数据集上进行模型训练的时间和空间复杂度，其中 T_{model} 表示算法2中的训练轮次。

定理 1（模型训练的时间复杂度）。 算法*2*需要 $O($ $T_{model} M_\mathcal{S}($ $\frac{N_\mathcal{S}^2}{K}$ $+ N_\mathcal{S} L_{in}))$ 时间。

定理 2（模型训练的空间复杂度）。 算法*2*需要 $O(N_\mathcal{S} F$ $(L_{in} + L_{out}) M_\mathcal{S} +$ $\frac{N_\mathcal{S}^2}{K^2}$ $)$ 空间。 —

值得注意的是，时间与空间复杂度均随合成数据集中的位置数量 $N_\mathcal{S}$ 呈二次增长，这表明我们的关键思路——降低 $N_\mathcal{S}$ （**S1** 在第 IV-A 节中）的重要性。

STemDist 的数据集蒸馏： 我们通过定理 3 分析 STemDist 的整体数据集蒸馏成本。回顾算法 1， T_{outer} 表示外层迭代次数， $T_{distill}$ 表示蒸馏迭代次数， T_{model} 表示代理模型的训练轮次数。聚类数据集的批次大小记为 B。详细的证明和分析见附录 B [61]。

定理 3（STemDist 的时间复杂度）。算法 *1*（即整个蒸馏过程）的时间复杂度为O $N_\mathcal{T} N_\mathcal{S}$ $M_\mathcal{T} L_{in} F$ $T_{kmeans} + T_{outer} T_{distill}$ $T_{model} M_\mathcal{S} +$ B $\frac{N_\mathcal{S}^2}{K}$ $+ L_{in}$ $T_{model} M_\mathcal{S} + B$ $+ M_\mathcal{S} + B$ $L_{out} F_{out}$ $N_\mathcal{S}$ ((()(()()(()())))。

在空间复杂度方面，位置聚类步骤需要 $O((N_\mathcal{T} + N_\mathcal{S})$ $M_\mathcal{T} L_{in} F)$ 的空间，其余蒸馏步骤需要 $O($ $\frac{N_\mathcal{S}^2}{K^2}$ $+ N_\mathcal{S} F(L_{in} + L_{out})$ $(M_\mathcal{S} + M_\mathcal{T}))$。 —

STemDist 的时间与空间复杂度都表明了空间维度缩减的重要性（**第 IV-A 节中的 S1** ），它会降低 $N_\mathcal{S}$；基于子集的蒸馏（**S3**）则会增大 K。我们的位置聚类策略（**S2**）使复杂度仅随 $N_\mathcal{T}$ 线性增长。

V. 实验

我们开展实验来回答以下问题：Q1.**蒸馏性能：** 与其他方法相比，使用 STemDist 生成的合成数据集进行训练有多有效、高效？ Q2. **跨模型性能：** 使用 STemDist 生成的合成数据集训练各种模型的效果如何？ Q3. **可扩展性与速度：** 随着数据集中时间序列、位置和特征数量的增加，STemDist 的扩展能力如何？ 与其他方法相比，它的速度有多快？

表 II 数据集统计。

	数据维度 M	N	F	# 总计数据点
真实世界数据集				
GBA [2]	1,977	2,352	1	4,649,904
GLA [2]	1,977	3,834	1	7,579,818
ERA5 [67]	2，137	6,561	6	84,125,142
CAMS [68]	2，556	7，070	6	108,425,520
元胞自动机 [2]	1,977	8,600	1	17,002,200
生成的人工数据集				
最小值	262,144	2,352	1	616,562,688
最大值	8,388,608	262,144	8,192	5,050,881,540,096

Q4. **消融研究：** STemDist 的各个组件如何影响蒸馏时间和性能？Q5. **超参数分析：** STemDist 在不同超参数设置下的表现如何?

A. 实验设置

机器： 所有实验均在一台配备四块 RTX A6000 GPU 和 512GB 内存的工作站上进行。

数据集： 我们使用了五个真实世界数据集，其统计信息汇总于表 II。GBA、GLA 和 CA 数据集是多个位置的交通测量数据。我们使用了在 [2], 中预处理并发布的版本，其中交通流量同时作为输入和目标变量。此外，我们还使用了 [69]提供的ERA5和CAMS数据集。ERA5 [67]提供由欧洲中期天气预报中心制作的全球天气再分析数据，而 CAMS [68] 则提供哥白尼大气监测服务的气象分析和预报数据。对于这两个数据集，我们选取最常用的六个变量作为输入；目标变量分别为ERA5中的10U（10米向东风）和CAMS中的CO（一氧化碳浓度）。我们还生成了不同规模的人工数据集，用于评估可扩展性。我们按时间顺序将每个数据集划分为训练集（60%）、验证集（20%）和测试集（20%）。输入序列和目标序列的长度均固定为12。

评估指标： 为了评估使用合成数据集训练的模型的性能，我们采用相对均方根误差（Relative RMSE）和相对平均绝对误差（Relative MAE）。它们的定义如下：

$$\underset{\text{RMSE}}{\text{Relative}} = \sqrt{\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}}, \underset{\text{MAE}}{\text{Relative}} = \frac{\sum_{i=1}^{N}|y_i-\hat{y}_i|}{\sum_{i=1}^{N}|y_i-\bar{y}|}, \quad (6)$$

其中， y_i 是真实值，$\hat{}y_i$ 是预测值， $\bar{y}$ 是真实值的均值。相对指标用于归一化不同数据集之间的尺度差异；对于这些指标，值越小表示准确性越高。

基线方法： 我们采用九种基线方法，分类如下：

- **核心集选择方法：随机采样**从原始时空时间序列数据集 $\mathcal{T}$ 中随机抽取样本（即输入-目标时间序列对）。

K-Center [47] 从 K-means 聚类的每个聚类中选取代表性样本。**Herding** [46] 通过贪心方式选择样本，以尽可能准确地逼近原始数据集的均值。

CRAIG [48] 选择带权重的训练数据子集，以便准确估计完整梯度。

- **通用数据集的蒸馏方法：DC** [15],**DM** [54], **MTT** [16], **DATM** [52], **IDM** [55],和**Frepo** [53] 分别采用梯度匹配、分布匹配、轨迹匹配和元学习，详见第II节。
- **时间序列数据集的蒸馏方法：CondTSF** [23] 和 **TimeDC** [24],（详见第II节）分别是针对单变量和多变量时间序列数据集的专用数据集蒸馏方法。

蒸馏详情： 对于所有基线方法，我们使用或调整作者提供的实现。对于 STemDist 及其变体，我们使用由我们提出的位置编码器增强的 MTGNN [25]作为代理模型 f_θ。对于 DC、DM、MTT 和 Frepo，我们也使用 MTGNN，但不使用位置编码器，因为位置数量未减少时并不需要它。对于 CondTSF 和 TimeDC，我们采用其论文中指定的相同代理模型。在 TimeDC 和 CondTSF 中，我们将目标特征作为唯一输入特征，因为它们仅支持单个输入变量。

对于所有方法，GBA/GLA 数据集的外层迭代次数（T_{outer}）设为 200，CA/ERA5/CAMS 数据集的外层迭代次数设为 100。在 STemDist 中，我们使用 20 次蒸馏迭代（$T_{distill}$）、10 个模型训练轮次（T_{model}），并将 K-means 聚类的最大迭代次数设为 300（T_{kmeans}）。对于会减少位置数量的蒸馏方法，当压缩率为 0.5% 时，我们分别将时间序列数量（M_S）和位置数量（N_S）设为原始数据集的 10% 和 5%；当压缩率为 1% 时，则分别设为 10% 和 10%（相关分析见第V-F节）。对于第V-B节中的基线方法，为匹配压缩率，我们仅压缩时间序列数量（M_S），因为这些方法本身不支持压缩位置数量（N_S）。我们将位置子集数量（K）设为 4，并将位置编码器模型的嵌入维度设为 32。在第V-B节和第V-C节中，每种方法对每个数据集使用不同随机种子进行五次蒸馏；对每份蒸馏后的数据，使用五组随机初始参数训练模型，因此实验总数为 25。代理模型（η）的训练学习率和合成数据的更新学习率通过在 $\{10^{-2}, 10^{-3}, 10^{-4}\}$ 上进行网格搜索来调整。有关基线方法的更多详情和超参数设置，请参见 [61] 。

推理详情： 训练后的推理过程中，每个批次包含 8 个时间序列。当位置数量超过 2^{15},时，我们会进一步将数据划分为每批次包含 2^{15} 个位置的批次，以避免时空图神经网络中的内存溢出错误。

B. 蒸馏性能

我们考察了使用 STemDist 及其竞争方法生成的合成数据集进行训练的有效性和效率。为了评估这些合成数据集，我们训练 MTGNN [25] ，并在需要时引入我们的位置编码器。

就有效性而言，STemDist 在各种压缩率下始终优于所有竞争方法，如表III所示。表中报告了预测误差的均值和标准差

表III 数据集蒸馏方法的有效性。使用各方法生成的合成数据集训练 MTGNN，并在必要时配备位置编码器，以进行评估。所提方法的性能以粗体标出；各设置下的最佳性能以黄色标出。内存溢出：内存不足。

方法	GBA		GLA		ERA5		CA		CAMS	
	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE
随机	0.380 ± 0.095	0.453 ± 0.086	0.313 ± 0.050	0.410 ± 0.051	0.482 ± 0.036	0.520 ± 0.028	0.323 ± 0.030	0.431 ± 0.065	0.660 ± 0.104	0.796 ± 0.109
K中心 [47]	0.319 ± 0.001	0.400 ± 0.042	0.269 ± 0.002	0.356 ± 0.028	0.461 ± 0.011	0.524 ± 0.012	0.289 ± 0.051	0.355 ± 0.034	0.883 ± 0.836	0.883 ± 0.073
Herding [46]	0.467 ± 0.008	0.506 ± 0.004	0.331 ± 0.013	0.428 ± 0.004	0.471 ± 0.000	0.522 ± 0.001	0.472 ± 0.000	0.521 ± 0.004	0.584 ± 0.000	0.730 ± 0.002
CRAIG [48]	0.291 ± 0.007	0.354 ± 0.007	0.258 ± 0.015	0.323 ± 0.017	0.430 ± 0.004	0.470 ± 0.003	0.287 ± 0.020	0.354 ± 0.021	0.563 ± 0.006	0.685 ± 0.008
直流 [15]	0.281 ± 0.009	0.348 ± 0.006	0.243 ± 0.009	0.311 ± 0.009	0.392 ± 0.004	0.442 ± 0.004	0.254 ± 0.011	0.331 ± 0.014	0.630 ± 0.005	0.735 ± 0.006
DM [54]	0.338 ± 0.021	0.409 ± 0.051	0.298 ± 0.023	0.380 ± 0.025	0.420 ± 0.007	0.467 ± 0.008	0.321 ± 0.027	0.405 ± 0.028	0.528 ± 0.020	0.674 ± 0.011
MTT [16]	0.339 ± 0.051	0.414 ± 0.055	0.343 ± 0.058	0.439 ± 0.066	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
DATM [52]	0.430 ± 0.012	0.517 ± 0.007	0.332 ± 0.016	0.402 ± 0.022	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
IDM [55]	0.358 ± 0.062	0.420 ± 0.050	0.300 ± 0.017	0.381 ± 0.024	0.438 ± 0.018	0.481 ± 0.018	0.315 ± 0.037	0.394 ± 0.047	0.567 ± 0.054	0.709 ± 0.049
Frepo [53]	0.397 ± 0.008	0.465 ± 0.075	0.350 ± 0.087	0.439 ± 0.084	0.482 ± 0.036	0.510 ± 0.031	0.361 ± 0.085	0.436 ± 0.084	0.688 ± 0.116	0.802 ± 0.118
CondTSF [23]	0.337 ± 0.040	0.407 ± 0.041	0.311 ± 0.027	0.394 ± 0.031	0.449 ± 0.017	0.489 ± 0.017	0.322 ± 0.050	0.403 ± 0.051	0.555 ± 0.053	0.692 ± 0.037
TimeDC [24]	0.350 ± 0.031	0.422 ± 0.030	0.344 ± 0.061	0.445 ± 0.068	0.437 ± 0.015	0.480 ± 0.015	0.360 ± 0.073	0.448 ± 0.074	0.585 ± 0.105	0.729 ± 0.078
STemDist-K-Center	0.305 ± 0.014	0.367 ± 0.006	0.248 ± 0.007	0.313 ± 0.008	0.396 ± 0.003	0.440 ± 0.005	0.291 ± 0.012	0.348 ± 0.009	0.903 ± 0.093	0.884 ± 0.077
STemDist-DC	0.263 ± 0.007	0.325 ± 0.006	0.227 ± 0.002	0.292 ± 0.006	0.392 ± 0.002	0.431 ± 0.002	0.247 ± 0.006	0.314 ± 0.005	0.545 ± 0.007	0.668 ± 0.008
STemDist-MTT	0.263 ± 0.007	0.325 ± 0.004	0.227 ± 0.004	0.300 ± 0.004	0.399 ± 0.009	0.437 ± 0.007	0.250 ± 0.004	0.311 ± 0.006	0.570 ± 0.010	0.685 ± 0.010
STemDist-CondTSF	0.260 ± 0.006	0.335 ± 0.010	0.225 ± 0.010	0.298 ± 0.005	0.407 ± 0.002	0.452 ± 0.002	0.247 ± 0.008	0.319 ± 0.013	0.514 ± 0.003	0.648 ± 0.002
STemDist **STemDist（1个特征）**	**0.251 ± 0.005**	**0.317 ± 0.005**	**0.211 ± 0.008**	**0.277 ± 0.002**	**0.385 ± 0.007** **0.400 ± 0.003**	**0.426 ± 0.004** **0.447 ± 0.003**	**0.226 ± 0.001**	**0.292 ± 0.002**	**0.522 ± 0.005** **0.520 ± 0.007**	**0.653 ± 0.005** **0.649 ± 0.007**
原始数据	0.207 ± 0.008	0.264 ± 0.010	0.182 ± 0.001	0.250 ± 0.005	0.368 ± 0.004	0.424 ± 0.010	0.120 ± 0.002	0.253 ± 0.003	0.500 ± 0.005	0.661 ± 0.004

(a) 压缩率：0.5%

方法	GBA		GLA		ERA5		CA		CAMS	
	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE
随机	0.291 ± 0.016	0.364 ± 0.017	0.271 ± 0.023	0.355 ± 0.033	0.453 ± 0.012	0.494 ± 0.009	0.289 ± 0.031	0.374 ± 0.045	0.617 ± 0.039	0.737 ± 0.041
K中心 [47]	0.294 ± 0.001	0.366 ± 0.003	0.240 ± 0.001	0.309 ± 0.002	0.434 ± 0.000	0.480 ± 0.000	0.256 ± 0.001	0.326 ± 0.002	0.682 ± 0.010	0.841 ± 0.009
Herding [46]	0.312 ± 0.001	0.387 ± 0.001	0.273 ± 0.000	0.342 ± 0.001	0.441 ± 0.000	0.478 ± 0.001	0.243 ± 0.001	0.313 ± 0.001	0.567 ± 0.000	0.696 ± 0.000
CRAIG [48]	0.272 ± 0.007	0.339 ± 0.006	0.243 ± 0.005	0.311 ± 0.005	0.417 ± 0.011	0.457 ± 0.009	0.261 ± 0.005	0.323 ± 0.005	0.551 ± 0.010	0.679 ± 0.012
直流 [15]	0.272 ± 0.005	0.340 ± 0.006	0.236 ± 0.003	0.304 ± 0.001	0.393 ± 0.002	0.434 ± 0.002	0.253 ± 0.003	0.318 ± 0.002	0.520 ± 0.009	0.652 ± 0.008
DM [54]	0.296 ± 0.013	0.361 ± 0.010	0.279 ± 0.028	0.356 ± 0.032	0.417 ± 0.017	0.461 ± 0.016	0.285 ± 0.022	0.361 ± 0.031	0.538 ± 0.018	0.678 ± 0.021
MTT [16]	0.289 ± 0.015	0.357 ± 0.013	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
DATM [52]	0.376 ± 0.014	0.463 ± 0.014	0.267 ± 0.008	0.336 ± 0.003	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
IDM [55]	0.290 ± 0.018	0.356 ± 0.018	0.280 ± 0.009	0.358 ± 0.008	0.411 ± 0.013	0.457 ± 0.012	0.283 ± 0.021	0.349 ± 0.019	0.527 ± 0.008	0.666 ± 0.008
Frepo [53]	0.283 ± 0.002	0.356 ± 0.006	0.256 ± 0.012	0.330 ± 0.019	0.419 ± 0.012	0.460 ± 0.013	0.259 ± 0.011	0.329 ± 0.013	0.653 ± 0.007	0.748 ± 0.002
CondTSF [23]	0.294 ± 0.014	0.364 ± 0.012	0.274 ± 0.024	0.356 ± 0.033	0.413 ± 0.005	0.461 ± 0.005	0.279 ± 0.018	0.359 ± 0.029	0.534 ± 0.032	0.680 ± 0.028
TimeDC [24]	0.295 ± 0.017	0.367 ± 0.016	0.274 ± 0.028	0.358 ± 0.040	0.426 ± 0.013	0.472 ± 0.014	0.295 ± 0.036	0.377 ± 0.052	0.560 ± 0.054	0.689 ± 0.021
STemDist-K-Center	0.280 ± 0.003	0.340 ± 0.006	0.243 ± 0.007	0.302 ± 0.009	0.395 ± 0.004	0.436 ± 0.005	0.274 ± 0.008	0.332 ± 0.011	0.872 ± 0.106	0.866 ± 0.066
STemDist-DC	0.258 ± 0.010	0.328 ± 0.009	0.223 ± 0.004	0.295 ± 0.007	0.392 ± 0.003	0.432 ± 0.003	0.240 ± 0.004	0.305 ± 0.003	0.537 ± 0.008	0.663 ± 0.002
STemDist-MTT	0.261 ± 0.005	0.321 ± 0.004	0.224 ± 0.007	0.293 ± 0.003	0.403 ± 0.003	0.444 ± 0.004	0.248 ± 0.012	0.317 ± 0.014	0.550 ± 0.016	0.681 ± 0.005
STemDist-CondTSF	0.261 ± 0.010	0.321 ± 0.003	0.220 ± 0.005	0.289 ± 0.004	0.405 ± 0.002	0.449 ± 0.002	0.243 ± 0.006	0.313 ± 0.004	0.513 ± 0.007	0.645 ± 0.002
STemDist **STemDist（1个特征）**	**0.242 ± 0.001**	**0.307 ± 0.002**	**0.205 ± 0.001**	**0.272 ± 0.001**	**0.384 ± 0.005** **0.399 ± 0.003**	**0.426 ± 0.004** **0.445 ± 0.004**	**0.222 ± 0.002**	**0.287 ± 0.002**	**0.518 ± 0.008** **0.501 ± 0.006**	**0.652 ± 0.006** **0.644 ± 0.003**
原始数据	0.207 ± 0.008	0.264 ± 0.010	0.182 ± 0.001	0.250 ± 0.005	0.368 ± 0.004	0.424 ± 0.010	0.120 ± 0.002	0.253 ± 0.003	0.500 ± 0.005	0.661 ± 0.004

(b) 压缩率：1%

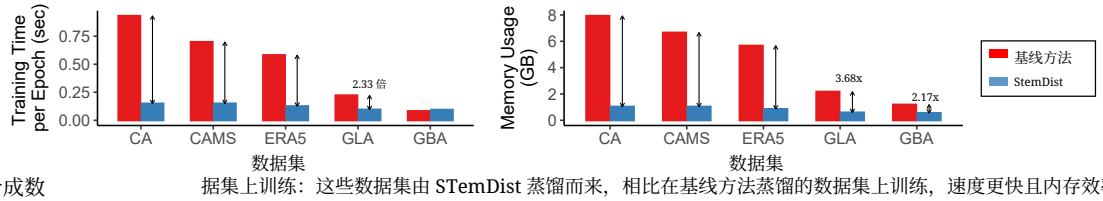

图4. 在合成数据集上训练：这些数据集由 STemDist 蒸馏而来，相比在基线方法蒸馏的数据集上训练，速度更快且内存效率更高。

。具体而言，当压缩率为 0.5% 时，与最佳竞争方法相比，使用 STemDist 蒸馏得到的合成数据训练的模型在 CA数据集上的相对均方根误差低 12%。这表明，STemDist 能够在合成数据集中以均衡的方式有效保留时间和空间信息。

使用 STemDist 蒸馏得到的合成数据集训练模型也比使用基线方法蒸馏得到的数据集训练更高效。如图4所示，当压缩率为 0.5% 时，在 STemDist 蒸馏的数据集上训练最多可快 6.3×，内存占用最多可减少 7.7× 。[3] 这些结果与第四节 IV-F 节中的定理1和定理2一致。

此外，我们评估了 STemDist 的若干变体：它们将我们的位置编码器和空间缩减整合到四种基线蒸馏方法中，在表III中记为 STemDist-[基线名称]。这些变体优于各自对应的基线方法，表明了我们思路的有效性；然而，在大多数数据集上，这些变体的表现仍不及 STemDist。

我们进一步考察了输入仅限于单个特征的情况，具体而言，即每个数据集的目标特征。对于 GBA、GLA 和 CA 数据集，这与其原始设置相对应，因为它们仅包含一个特征。如表III所示，即使在这一限制下，STemDist 仍优于大多数基线方法。

C. 跨模型性能

我们通过训练广泛使用的时空

[3]为清楚起见，本实验与第I节中的初步实验1有所不同，因为它采用了不同的模型（使用我们的位置编码器的 MTGNN）和训练策略（算法2中的基于子集的训练）。相比之下，初步实验1遵循了 [25] 中的设置。

表 IV 数据集蒸馏方法的跨模型泛化。每种方法生成的合成数据集均用于训练不同的机器学习模型，以进行评估。我们的方法的性能以粗体标出；各设置下的最佳性能以黄色标出。

Graph WaveNet [11]

方法	GBA		GLA		ERA5		CA		CAMS	
	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE
K中心 [47]	0.336 ± 0.027	0.427 ± 0.032	0.279 ± 0.021	0.367 ± 0.018	0.466 ± 0.017	0.510 ± 0.012	0.318 ± 0.032	0.396 ± 0.030	0.763 ± 0.055	0.796 ± 0.028
直流 [15]	0.321 ± 0.036	0.395 ± 0.033	0.288 ± 0.031	0.312 ± 0.009	0.424 ± 0.007	0.470 ± 0.004	0.292 ± 0.043	0.369 ± 0.038	0.550 ± 0.006	0.688 ± 0.004
CondTSF [23]	0.346 ± 0.065	0.416 ± 0.066	0.302 ± 0.030	0.387 ± 0.028	0.464 ± 0.008	0.514 ± 0.007	0.322 ± 0.038	0.413 ± 0.031	0.531 ± 0.026	0.695 ± 0.022
STemDist	**0.241 ± 0.007**	**0.310 ± 0.007**	**0.212 ± 0.006**	**0.273 ± 0.003**	**0.385 ± 0.004**	**0.431 ± 0.003**	**0.222 ± 0.004**	**0.293 ± 0.006**	**0.528 ± 0.006**	**0.665 ± 0.007**
原始数据	0.181 ± 0.005	0.227 ± 0.009	0.166 ± 0.007	0.212 ± 0.011	0.376 ± 0.007	0.419 ± 0.005	0.169 ± 0.009	0.215 ± 0.007	0.488 ± 0.014	0.631 ± 0.007

STGCN [10]

方法	GBA		GLA		ERA5		CA		CAMS	
	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE
K中心 [47]	0.606 ± 0.091	0.714 ± 0.085	0.643 ± 0.094	0.746 ± 0.062	0.791 ± 0.081	0.800 ± 0.046	0.625 ± 0.046	0.731 ± 0.073	1.155 ± 0.045	1.075 ± 0.019
直流 [15]	0.767 ± 0.077	0.812 ± 0.053	0.767 ± 0.078	0.866 ± 0.031	0.733 ± 0.044	0.753 ± 0.060	0.655 ± 0.088	0.736 ± 0.068	0.936 ± 0.035	0.986 ± 0.018
CondTSF [23]	0.628 ± 0.065	0.697 ± 0.038	0.622 ± 0.070	0.734 ± 0.087	0.762 ± 0.065	0.796 ± 0.052	0.616 ± 0.095	0.730 ± 0.094	0.896 ± 0.069	0.950 ± 0.037
STemDist	**0.441 ± 0.038**	**0.501 ± 0.041**	**0.392 ± 0.032**	**0.459 ± 0.034**	**0.632 ± 0.032**	**0.667 ± 0.037**	**0.443 ± 0.050**	**0.504 ± 0.045**	**0.748 ± 0.022**	**0.844 ± 0.011**
原始数据	0.247 ± 0.008	0.294 ± 0.008	0.240 ± 0.005	0.295 ± 0.003	0.511 ± 0.006	0.546 ± 0.012	0.371 ± 0.348	0.268 ± 0.006	0.667 ± 0.020	0.743 ± 0.010

FourierGNN [70]

方法	GBA		GLA		ERA5		CA		CAMS	
	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE	相对 MAE	相对 RMSE
K中心 [47]	0.353 ± 0.032	0.419 ± 0.027	0.310 ± 0.027	0.395 ± 0.025	0.432 ± 0.005	0.480 ± 0.005	0.360 ± 0.020	0.437 ± 0.021	0.651 ± 0.012	0.795 ± 0.021
直流 [15]	0.335 ± 0.015	0.403 ± 0.015	0.301 ± 0.020	0.376 ± 0.021	0.426 ± 0.005	0.469 ± 0.003	0.337 ± 0.010	0.409 ± 0.011	0.561 ± 0.024	0.703 ± 0.028
CondTSF [23]	0.348 ± 0.021	0.419 ± 0.036	0.313 ± 0.018	0.421 ± 0.009	0.469 ± 0.022	0.511 ± 0.019	0.351 ± 0.028	0.441 ± 0.023	0.557 ± 0.059	0.715 ± 0.070
STemDist	**0.309 ± 0.003**	**0.384 ± 0.002**	**0.263 ± 0.003**	**0.339 ± 0.004**	**0.412 ± 0.001**	**0.459 ± 0.001**	**0.293 ± 0.007**	**0.371 ± 0.006**	**0.534 ± 0.008**	**0.658 ± 0.001**
原始数据	0.248 ± 0.006	0.307 ± 0.012	0.216 ± 0.009	0.284 ± 0.006	0.396 ± 0.002	0.444 ± 0.002	0.219 ± 0.009	0.288 ± 0.006	0.496 ± 0.003	0.642 ± 0.002

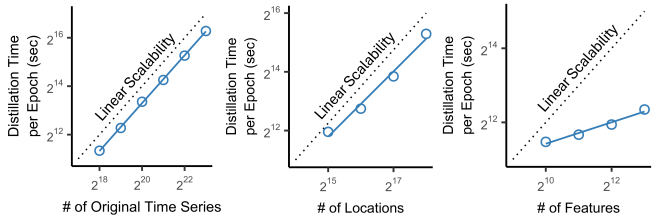

图 5. STemDist 的可扩展性。其实际蒸馏时间随（a）原始时间序列的数量、（b）原始数据集中的位置数量以及（c）原始数据集中的特征数量呈（次）线性增长。

时间序列预测模型，评估合成数据集的跨模型泛化能力：Graph WaveNet [11],、STGCN [10]，和 FourierGNN [70]，，并在必要时为其配备我们的位置编码器。[4] K中心 [47],、DC [15]，和 CondTSF [23]，分别是各自基线类别中的最佳方法，因此被用作基线；压缩率设为 0.5%。如表 IV 所示，STemDist 在所有测试模型和数据集上均表现最佳，展现出很强的跨模型泛化能力。这表明，STemDist 生成的合成数据集捕获了能够泛化至代理模型之外的关键信息。其他模型（具体而言，AGCRN [12]和 STNORM [36]）的结果见附录 E [61]。

D. 可扩展性与速度

我们使用人工生成的大规模数据集，评估 STemDist 针对各种输入/输出因素的可扩展性。我们逐步将每个因素加倍，同时将其他因素保持为默认值[5]，，总结如下：

[4]我们修改了 STGCN，使其能够通过公式 (2) 学习图，而不依赖预定义邻接矩阵。请注意，某些模型（例如 [36], [70]）在合成数据集上训练时不需要位置编码器。

[5]262, 144 条原始时间序列、64 条合成时间序列、2，352 个位置和 1 个特征。

表 II，并测量每轮遍历数据集所需的蒸馏时间。虽然我们控制原始数据集中的位置数量，但合成数据集的大小相应设定为 $N_{\mathcal{S}} = 0.05 \times N_{\mathcal{T}}$。每项实验均进行三次，并报告均值。

如图 5 所示，STemDist 的蒸馏时间随原始时间序列数量和位置数量线性增长（双对数坐标下斜率为 1），随特征数量和合成时间序列数量次线性增长（双对数坐标下斜率小于 1；关于合成时间序列数量的结果见附录 F [61]）。原始时间序列数量方面的结果与第四节 IV-F 节中的定理 3 一致。对于其他因素，得益于每个批次内的 GPU并行化，STemDist 在实测可扩展性方面优于定理 3 中的理论上界。请注意，原始时间序列数量增加时，并行化不会发挥作用，因为这会增加批次数量，而非批次大小。

我们还在图 6 中比较了压缩率为 0.5% 时 S TemDist 与基线方法的蒸馏时间。基于轨迹匹配的方法（MTT [16], DATM [52],CondTSF [23]，和 TimeDC [24]）通常需要较少的蒸馏时间。不过，在蒸馏过程开始前，它们还需要额外的时间来生成专家轨迹。与 DM [54] 和 IDM [55],相比，在位置数量较多的数据集上，STemDist 的时间效率明显更高。与 DC [15], 相比，STemDist 在所有数据集上始终表现出更高的时间效率。

位置编码器产生的推理延迟可以忽略不计，在真实世界数据集上最多为 0.0002 秒（表 II），在人工生成的数据集上最多为 0.002 秒，占

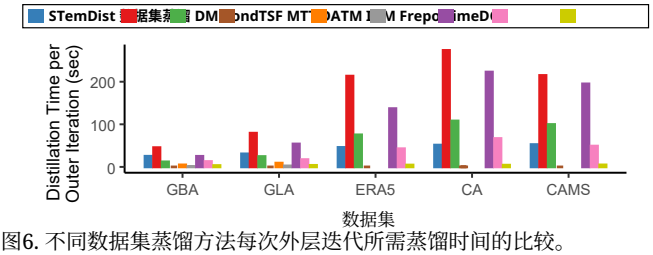

图6. 不同数据集蒸馏方法每次外层迭代所需蒸馏时间的比较。

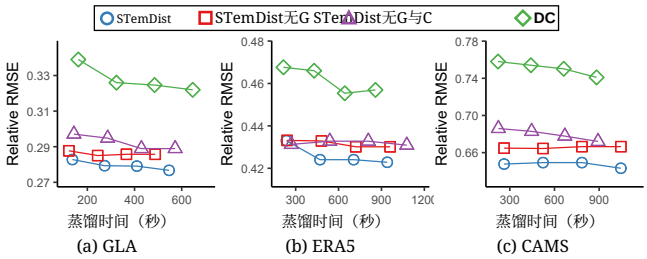

图7. 消融研究。GLA、ERA5、CAMS数据集上STemDist及其变体的蒸馏性能随蒸馏时间的变化

总推理时间的比例分别仅为 0.06% 和 0.004%。

E. 消融研究

为确认STemDist各个组件的有效性，我们将STemDist与三个变体进行比较：

- **STemDist无G**：不采用基于子集的**G**粒度蒸馏的STemDist。
- **STemDist 无G与C**：不采用基于子集的**G**粒度蒸馏和**C**位置聚类的STemDist。
- **DC** [15]：一种朴素的梯度匹配方法。

在 0.5% 的压缩率下，我们对每种方法进行三次蒸馏，并对每个合成数据集评估三次。我们报告均值。

图 7 展示了 GLA、ERA5 和 CAMS 数据集上各变体的性能随蒸馏时间的变化情况（其他数据集的结果请参见附录 H [61] ）。值得注意的是，STemDist 的每个组成部分都有助于更快地实现有效蒸馏（即降低预测误差）。

我们评估了 STemDist 的变体：在位置聚类中，使用空间约束聚类（RedCap [65]）或图粗化（METIS [66]）替代 K-means。对于 ERA5 和 CAMS，我们根据经度和纬度坐标构建基于网格的空间图；由于缺少空间信息，交通数据集不在评估范围内。与使用 K-means（ERA5 上的相对均方根误差为 0.426，CAMS 上为 0.653）相比，使用 METIS（ERA5 上为 0.421，CAMS 上为 0.656）或 RedCap（ERA5 上为 0.429，CAMS 上为 0.672）的蒸馏性能相近，没有明显提升。

F. 超参数分析

我们研究了在 GBA、GLA 和 ERA5 数据集上，总压缩率固定为 1% 时，超参数对 STemDist 蒸馏性能的影响。除受控超参数外，其余参数均固定为第 V-A 节所述的默认值。我们评估了四种时间–空间比率组合，并观察到，当时间比率与空间比率

表 V 在总压缩率为 1% 时，不同超参数对 STemDist 蒸馏性能的影响（以相对均方根误差衡量）。

数据集	时间—空间压缩比对（$\frac{M_S}{M_\mathcal{T}}, \frac{N_S}{N_\mathcal{T}}$）			
	(2%, 50%)	(5%, 20%)	(10%, 10%)	(25%, 4%)
GBA	0.319 ± 0.008	0.307 ± 0.004	0.307 ± 0.002	0.324 ± 0.007
GLA	0.291 ± 0.004	0.278 ± 0.002	0.272 ± 0.001	0.279 ± 0.003
ERA5	0.426 ± 0.004	0.424 ± 0.003	0.426 ± 0.004	0.424 ± 0.003
CA	0.302 ± 0.009	0.292 ± 0.002	0.287 ± 0.002	0.293 ± 0.003
CAMS	0.656 ± 0.007	0.652 ± 0.003	0.652 ± 0.006	0.653 ± 0.003
Avg.	0.399 ± 0.006	0.391 ± 0.003	0.389 ± 0.003	0.395 ± 0.004

(a) 时空压缩比的影响

数据集	子集数量 K			
	2	4	8	16
GBA	0.352 ± 0.007	0.307 ± 0.002	0.313 ± 0.009	0.310 ± 0.011
GLA	0.274 ± 0.006	0.272 ± 0.001	0.275 ± 0.004	0.276 ± 0.004
ERA5	0.425 ± 0.005	0.426 ± 0.004	0.434 ± 0.005	0.429 ± 0.003
CA	0.297 ± 0.007	0.287 ± 0.002	0.281 ± 0.002	0.280 ± 0.006
CAMS	0.664 ± 0.003	0.652 ± 0.006	0.652 ± 0.003	0.654 ± 0.003
Avg.	0.402 ± 0.006	0.389 ± 0.003	0.391 ± 0.006	0.390 ± 0.005

（b）子集数量的影响

保持平衡时，STemDist 往往更有效（表 V(a)）。我们进一步研究了基于子集的细粒度蒸馏中子集数量 K 的影响。如表 V(b) 所示，当子集数量为 4 个或更多时，STemDist 的表现同样出色。回顾第 V-E 节，我们观察到，当子集数量为 1 时，性能会大幅下降（图 7 中的 STemDist无G）。总压缩率为 0.5% 时的超参数分析结果见附录 I [61]。

VI. 结论与未来方向

在这项工作中，我们提出了 STemDist，这是一种专为时空时间序列设计的新型数据集蒸馏方法。STemDist 的核心思想是双维压缩，即同时缩减合成数据集的时间维度和空间维度，从而实现更有效、更高效的模型训练。这通过三个核心组件（位置编码器、位置聚类和基于子集的细粒度蒸馏）实现，它们提升了蒸馏过程的有效性和效率。在我们的实验中，STemDist 生成的合成数据使模型训练**更快**（最高可达 6×）、**更节省内存**（最高可达 8×），且**效果更好**（预测误差最多降低 12%）。未来的一个潜在方向是以成本敏感的方式扩展 STemDist，优先保障高风险或罕见事件（例如气象数据中的极端天气事件）的性能。

致谢： 本研究获得由韩国政府（MSIT）资助的信息通信技术规划与评估研究院（IITP）项目支持（编号：RS-2024-00457882，人工智能研究中心项目，40%）（编号：RS-2022-II220157，稳健、公平、可扩展的数据中心持续学习，40%）（编号：RS-2024-00438638，EntireDB2AI：面向完整关系数据库的全面深度表征学习与预测的基础和软件，10%）（编号：RS-2019-II190075，人工智能研究生院项目（KAIST），10%）。

人工智能生成内容声明

手稿的部分内容在 ChatGPT（GPT-5，OpenAI）的协助下进行了语言润色。作者已审阅并编辑所有人工智能生成内容，并对手稿的最终版本承担全部责任。

参考文献

[1] J. Han, W. Zhang, H. Liu, T. Tao, N. Tan, 和 H. Xiong, 《Bigst：用于大规模道路网络交通预测的线性复杂度时空图神经网络》, PVLDB, 第 17 卷, 1090, 2024 5, 1081, 第 – 页。[2] Y. Fang, Y. Liang, B. Hui, Z. Shao, L. Deng, X. Liu, X. Jiang, 和 K. Zheng, 《借助 Transformer 实现高效的大规模交通预测：空间数据管理视角》, 载于 KDD, 2025年。 [3] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, 和 S. Pan, 《迈向时空感知的交通时间序列预测》, 载于 ICDE, 2022年。 [4] D. Kieu, T. Kieu, P. Han, B. Yang, C. S. Jensen, 和 B. Le, 《Team：面向交通预测的拓扑演化感知框架》, PVLDB, 第 18 卷, 第 2 期, 2024年。 [5] Z. Lai, D. Zhang, H. Li, C. S. Jensen, H. Lu, 和 Y. Zhao, 《Lightcts：用于相关时间序列预测的轻量级框架》, SIGMOD, 2023。 [6] R. A. Angryk, P. C. Martens, B. Aydin, D. Kempton, S. S. Mahajan, S. Basodi, A. Ahmadzadeh, X. Cai, S. Filali Boubrahimi, S. M. Hamdi 等, 《用于空间天气数据分析的多变量时间序列数据集》, Scientific Data, 2020年。 [7] M. Ma, P. Xie, F. Teng, B. Wang, S. Ji, J. Zhang, 和 T. Li, 《Histgnn：用于天气预报的分层时空图神经网络》, Information Sciences, 第 648 卷, 119580 页, 2023。 [8] H. Miao, Y. Zhao, C. Guo, B. Yang, K. Zheng, F. Huang, J. Xie, 和 C. S. Jensen, 《面向流数据时空预测的统一基于回放的持续学习框架》, 载于 ICDE, 2024年。 [9] X. Wu, X. Wu, B. Yang, L. Zhou, C. Guo, X. Qiu, J. Hu, Z. Sheng, 和 C. S. Jensen, 《Autocts++：用于相关时间序列预测的零样本联合神经架构与超参数搜索》, The VLDB Journal, 第 33 卷, 第 5 期, 2024年。 [10] B. Yu, H. Yin, 和 Z. Zhu, 《时空图卷积网络：用于交通预测的深度学习框架》, 载于 IJCAI, 2018年。[11] Z. Wu, S. Pan, G. Long, J. Jiang, 和 C. Zhang, 《用于深度时空图建模的 Graph WaveNet》, 载于 IJCAI, 2019。 [12] L. Bai, L. Yao, C. Li, X. Wang, 和 C. Wang, 《用于交通预测的自适应图卷积循环网络》, 载于 NeurIPS, 2020年。[13] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, 和 C. Zhang, 《连点成线：利用图神经网络进行多变量时间序列预测》, 载于 KDD, 2020年。 [14] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, 和 C. S. Jensen, 《用于交通预测的解耦动态时空图神经网络》, PVLDB, 第 15 卷, 第 11 期, 第 2733–2746 页, 2022年。 [15] B. Zhao, K. R. Mopuri, 和 H. Bilen, 《利用梯度匹配进行数据集凝缩》, 载于 ICLR, 2021年。 [16] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, 和 J.-Y. Zhu, 《通过匹配训练轨迹进行数据集蒸馏》, 载于 CVPR, 2022年。 [17] T. Wang, J.-Y. Zhu, A. Torralba, 和 A. A. Efros, 《数据集蒸馏》, arXiv, 2018年。 [18] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, 和 N. Shah, 《面向图神经网络的图凝缩》, ICLR, 2021年。 [19] W. Jin, X. Tang, H. Jiang, Z. Li, D. Zhang, J. Tang, 和 B. Yin, 《通过单步梯度匹配凝缩图》, 载于 KDD, 2022年。 [20] X. Gao, T. Chen, Y. Zang, W. Zhang, Q. V. H. Nguyen, K. Zheng, 和 H. Yin, 《面向归纳式节点表示学习的图凝缩》, 载于 ICDE, 2024年。 [21] Y. Liang, W. Zhang, X. Gao, L. Yang, C. Chen, H. Yin, Y. Tong, 和 B. Cui, 《通过数据选择实现免训练的异质图凝缩》, 载于 ICDE, 2025年。 [22] Z. Liu, K. Hao, G. Zheng, 和 Y. Yu, 《通过双域匹配进行时间序列分类的数据集凝缩》, 载于 KDD, 2024年。 [23] J. Ding, Z. Liu, G. Zheng, H. Jin, 和 L. Kong, 《Condtsf：用于时间序列预测的数据集凝缩单行插件》, NeurIPS, 2024年。 [24] H. Miao、Z. Liu、Y. Zhao、C. Guo、B. Yang、K. Zheng 和 C. S. Jensen, 《少即是多：通过双重模态匹配实现高效时间序列数据集凝缩》, PVLDB, 第18卷第2期, 第226–238页, 2024年。 [25] Z. Wu、S. Pan、G. Long、J. Jiang、X. Chang 和 C. Zhang, 《连接各个点：利用图神经网络进行多变量时间序列预测》, 发表于 KDD, 2020年。 [26] K. Zhao、C. Guo、Y. Cheng、P. Han、M. Zhang 和 B. Yang, 《利用动态图建模进行多时间序列预测》, PVLDB, 第17卷第4期, 第753–765页, 2023年。 [27] M. Ma、J. Hu、C. S. Jensen、F. Teng、P. Han、Z. Xu 和 T. Li, 《学习用于空间相关时间序列预测的时间感知图结构》, 发表于 ICDE, 2024年。 [28] X. Wu、D. Zhang、C. Guo、C. He、B. Yang 和 C. S. Jensen, 《Autocts：自动化相关时间序列预测》, PVLDB, 第15卷第4期, 第971–983页, 2021年。 [29] X. Wu、X. Wu、D. Zhang、M. Zhang、C. Guo、B. Yang 和 C. S. Jensen, 《数分钟内实现全自动相关时间序列预测》, PVLDB, 第18卷第2期, 第144–157页, 2024年。 [30] Y. Cheng、P. Chen、C. Guo、K. Zhao、Q. Wen、B. Yang 和 C. S. Jensen, 《面向鲁棒时间序列预测的弱引导适应》, PVLDB, 第17卷第4期, 第766–779页, 2023年。 [31] Y. Jiang、X. Li、Y. Chen、S. Liu、W. Kong、A. F. Lentzakis 和 G. Cong, 《SAGDFN：用于多变量时间序列预测的可扩展自适应图扩散预测网络》, 发表于 ICDE, 2024年。 [32] R.-G. Cirstea、T. Kieu、C. Guo、B. Yang 和 S. J. Pan, 《EnhanceNet：用于增强相关时间序列预测的插件神经网络》, 发表于 ICDE, 2021年。 [33] G. Atluri、A. Karpatne 和 V. Kumar, 《时空数据挖掘：问题与方法综述》, 《ACM计算调查》, 第51卷第4期, 第1–41页, 2018年。 [34] G. Jin、Y. Liang、Y. Fang、Z. Shao、J. Huang、J. Zhang 和 Y. Zheng, 《面向城市计算预测学习的时空图神经网络：综述》, 《IEEE知识与数据工程汇刊》, 第36卷第10期, 第5388–5408页, 2023年。 [35] S. Lan、Y. Ma、W. Huang、W. Wang、H. Yang 和 P. Li, 《DSTAGNN：用于交通流预测的动态时空感知图神经网络》, 发表于 ICML, 2022年。 [36] J. Deng、X. Chen、R. Jiang、X. Song 和 I. W. Tsang, 《ST-Norm：用于多变量时间序列预测的时空归一化》, 发表于 KDD, 2021年。 [37] Z. Shao、Z. Zhang、F. Wang、W. Wei 和 Y. Xu, 《时空标识：一种简单而有效的多变量时间序列预测基线》, 发表于 CIKM, 2022年。 [38] Z. Fang、Q. Long、G. Song 和 K. Xie, 《用于交通流预测的时空图常微分方程网络》, 发表于 KDD, 2021年。 [39] Z. Fang、L. Pan、L. Chen、Y. Du 和 Y. Gao, 《MDTP：基于时空轨迹数据的多源深度交通预测框架》, PVLDB, 第14卷第8期, 第1289–1297页, 2021年。 [40] Y. Li、R. Yu、C. Shahabi 和 Y. Liu, 《扩散卷积循环神经网络：数据驱动的交通预测》, ICLR, 2018年。 [41] Z. Shao、Z. Zhang、F. Wang 和 Y. Xu, 《通过预训练增强的多变量时间序列预测时空图神经网络》, 发表于 KDD, 2022 年。 [42] J. Choi 、 H. Choi 、 J. Hwang 和 N. Park, 《用于交通预测的图神经控制微分方程》, 发表于 AAAI, 2022年。 [43] Q. Zhou、Y. Zhang、M. A. Makridis、A. Kouvelas、Y. Wang 和 S. Hu, 《Mogernn：用于未观测位置的归纳式交通预测器》, 《交通研究C辑：新兴技术》, 第174卷, 第105080页, 2025年。 [44] N. Sachdeva 和 J. McAuley, 《数据蒸馏：综述》, arXiv, 2023年。 [45] C. Chai、J. Liu、N. Tang、J. Fan、D. Miao、J. Wang、Y. Luo 和 G. Li, 《GoodCore：通过在不完整数据上进行核心集选择，实现数据有效且数据高效的机器学习》, SIGMOD, 2023年。 [46] M. Welling, 《通过集聚动态权重进行学习》, 发表于 ICML, 2009年。 [47] R. Z. Farahani 和 M. Hekmatfar, 《设施选址：概念、模型、算法与案例研究》, 施普林格科学与商业媒体, 2009年。 [48] B. Mirzasoleiman、J. Bilmes 和 J. Leskovec, 《用于机器学习模型高效训练的核心集》, 发表于 ICML, 2020年。 [49] B. Zhao 和 H. Bilen, 《采用可微孪生增强的数据集凝缩》, 发表于 ICML, 2021年。 [50] J. Wu、W. Fan、J. Chen、S. Liu、Q. Liu、R. He、Q. Li 和 K. Tang, 《通过轻量级策略梯度估计凝缩预增强推荐数据》, 《IEEE知识与数据工程汇刊》, 第37卷第1期, 第162–173页, 2025年。

[51] J. Gao、J. Wu 和 J. Ding，《异质图凝缩》，《IEEE知识与数据工程汇刊》，第36卷第7期，第3126–3138页，2024年。 [52] Z. Guo、K. Wang、G. Cazenavette、H. Li、K. Zhang 和 Y. You，《通过难度对齐轨迹匹配实现无损数据集蒸馏》，ICLR，2024年。 [53] Y. Zhou、E. Nezhadarya 和 J. Ba，《利用神经特征回归进行数据集蒸馏》，NeurIPS，2022年。 [54] B. Zhao 和 H. Bilen，《通过分布匹配进行数据集凝缩》，发表于冬季计算机视觉应用会议，2023年。 [55] G. Zhao、G. Li、Y. Qin 和 Y. Yu，《改进数据集凝缩的分布匹配》，发表于 CVPR，2023年。 [56] H. Wu、J. Xu、J. Wang 和 M. Long，《Autoformer：用于长期序列预测的分解变换器与自相关机制》，NeurIPS，2021年。 [57] H. Zhou、S. Zhang、J. Peng、S. Zhang、J. Li、H. Xiong 和 W. Zhang，《Informer：超越高效变换器的长序列时间序列预测方法》，发表于 AAAI，2021年。 [58] Y. Zhang、T. Zhang、K. Wang、Z. Guo、Y. Liang、X. Bresson、W. Jin 和 Y. You，《驾驭复杂性：通过扩展窗口匹配实现无损图凝缩》，发表于 ICML，2024年。 [59] M. Liu、S. Li、X. Chen 和 L. Song，《通过感受野分布匹配进行图凝缩》，arXiv，2022年。 [60] L. Wang、W. Fan、J. Li、Y. Ma 和 Q. Li，《基于结构的神经切线核实现快速图凝缩》，发表于万维网大会，2024年。 [61] T. Kwon、Y. Choi、Y. Kim 和 K. Shin。（2025）《采用双维压缩的时空预测有效数据集蒸馏》（代码、数据集和在线附录）。 [在线]。可访问：https://github.com/kbrother/STemDist [62] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin，《注意力就是你所需要的一切》，NeurIPS，2017年。 [63] J. L. Ba、J. R. Kiros 和 G. E. Hinton，《层归一化》，arXiv，2016年。[64] R. Zorrilla、E. Ogasawara、P. Valduriez 和 F. Porto，《一种面向时空预测的数据驱动模型选择方法》，载于《大型数据与知识中心系统汇刊》。施普林格，2024年，第98–118页。 [65] D. Guo，《采用动态约束凝聚聚类与划分的区域化（REDCAP）》，《国际地理信息科学杂志》，第22卷第7期，第801–823页，2008年。 [66] G. Karypis 和 V. Kumar，《面向非规则图的多层K路划分方案》，《并行与分布式计算杂志》，第48卷第1期，1998年。 [67] H. Hersbach、B. Bell、P. Berrisford、G. Biavati、A. Hor´anyi、J. Mu˜noz Sabater、J. Nicolas、C. Peubey、R. Radu、I. Rozum、D. Schep- ers、A. Simmons、C. Soci、D. Dee 和 J.-N. Th´epaut，《自1940年至今的ERA5单层逐小时数据》，哥白尼气候变化服务（C3S）气候数据存储库（CDS），2023年。 [68] A. Inness、M. Ades、A. Agust´ı-Panareda、J. Barr´e、A. Benedictow、A.- M. Blechschmidt、J. J. Dominguez、R. Engelen、H. Eskes、J. Flemming 等，《CAMS大气成分再分析》，《大气化学与物理》，第19卷第6期，2019年。 [69] C. Bodnar、W. P. Bruinsma、A. Lucic、M. Stanley、A. Allen、J. Brandstet- ter、P. Garvan、M. Riechert、J. A. Weyn、H. Dong 等，《面向地球系统的基础模型》，《自然》，2025年。 [70] K. Yi、Q. Zhang、W. Fan、H. He、L. Hu、P. Wang、N. An、L. Cao 和 Z. Niu，《FourierGNN：从纯图视角重新思考多变量时间序列预测》，发表于 NeurIPS，2023年。