SparseTSF 模型的复现、跨平台迁移与优化探索

高景珩　　丛方昊　　章昊

Abstract

本文主要研究了一种新颖且轻量级的长期时间序列预测（*LTSF*）模型——*SparseTSF*。首先，在原作者研究的基础上，笔者重点探讨了如何在计算资源受限的条件下，利用 *SparseTSF* 模型高效实现长期时间序列预测。其次，笔者通过与其他模型的比较，并结合误差图和训练损失图，对 *SparseTSF* 模型的优势和不足进行了全面评估。在此基础上，笔者将代码迁移至其他数据集，进行了可迁移性分析。此外，笔者成功地将模型从 *PyTorch* 框架迁移至 *MindSpore* 框架，并对模型代码进行了精简与优化。通过去除冗余代码实现轻量化，并改进内存管理策略、采用 *PCA* 参数优化，提升了运行效率。实验结果表明，模型在 *MindSpore* 框架下的拟合效果与原框架基本一致，但运行速度得到了一定提升。最后，笔者通过可视化技术展示了不同 *epoch* 下的训练结果，并采用独热编码对模型参数进行了可视化分析，从而更直观地呈现了训练过程中的效果变化。

1. 引言

完成人：章昊、丛方昊、高景珩

1.1. 研究动机

1.1.1　框架迁移

近年来，由于人工智能技术迅猛发展，框架的选择已经成为开发者和研究人员的重要决策。PyTorch 和 TensorFlow 作为全球最主流的深度学习框架，它们在技术发展和社区支持方面具有巨大的优势。比如 SparseTSF [1] 就是基于 PyTorch 搭建的模型。然而，随着中国在科技自主创新方面的不断推进，国产深度学习框架逐渐崭露头角。其中，华为推出的 MindSpore 作为一款针对 AI 应用的开源深度学习框架，成为了国产化道路上的一大亮点。

MindSpore 是华为公司主导开发的 AI 计算框架，旨在为云端、边缘端和终端设备提供统一的 AI 计算平台。与 PyTorch 不同，MindSpore 注重硬件与软件的深度融合，特别是针对华为的自研芯片（如 Ascend 和 Kirin 芯片）进行了优化。此外，MindSpore 采用静态与动态图相结合的计算图架构，旨在提升性能、减少计算资源的浪费，同时提高开发效率。因为 Mindspore 具有的图模式、自动并行、跨平台支持、优化算子、数据增强、可解释性等优点，笔者决定将 SparseTSF 迁移到 Mindspore 框架上进一步测试器性能。

1.1.2　模型优化

在模型训练的过程中，笔者注意到训练的效率与运行时长和数据集的大小与特征数量息息相关，在运行大数据集时，会出现运行时间长、效率低下等问题。因此，笔者决定在数据维度方面进行改进。

例如在 traffic.csv 和 electricity.csv 数据集中，数据维度分别达到了 863 和 332，默认情况下会同时占用 40GB 以上内存，因此对于这种高维的数据直接进行运算会极大影响模型运行效率，甚至造成模型无法运行。

1.2. 成果概述

首先，笔者复现了 SparseTSF 模型在原始数据集上的表现，绘制了损失曲线以及测试结果图（图1），得到了与原文相似的结果。并进一步迁移至其他数据集，如 exchange_rate 数据集（图2），证明了其在不同场景下的有效性与泛化能力。

其次，笔者成功将 SparseTSF 模型从 PyTorch 框架迁移至 MindSpore 框架，通过一系列实验验证了迁移后模型在保持预测精度的同时，一定程度上提升了

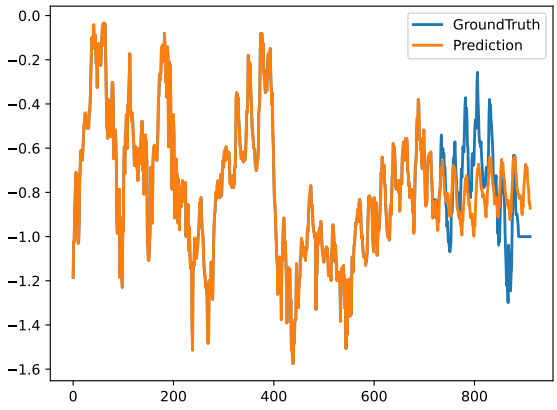

图 1. etth1 结果复现

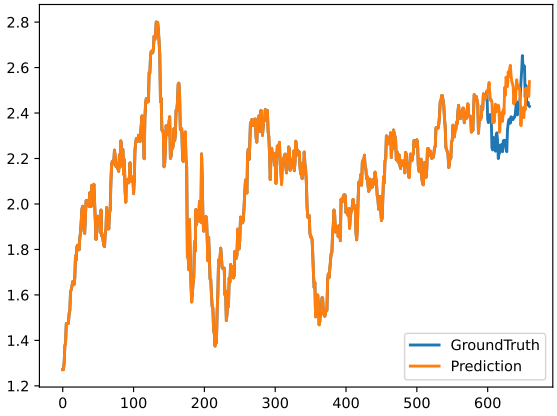

图 2. exchange_rate 数据集迁移结果

计算效率，展示了其在新平台上的潜力。

此外，为了深入理解 SparseTSF 的工作机制，笔者对模型的参数进行了独热编码可视化分析，通过热力图清晰展示了等效变换矩阵的等间距条带分布。这一分析进一步验证了 SparseTSF 通过稀疏机制加强周期性特征的提取，极大优化了模型参数，提高了预测效率，为其在 LTSF 任务中的出色表现提供了新的理论支持。

最后，笔者尝试利用 PCA 降维，对模型进行一定改进，使之适合配置较低的环境。改进后的模型成功在内存不足的情况下跑通了 electricity 数据集，并得到了优秀的预测结果。

2. 相关工作

完成人：章昊、丛方昊

2.1. 迁移的准备工作

迁移准备是代码迁移工作中至关重要的一环，主要包括环境搭建与依赖安装、数据集与实验配置准备，以及目标代码功能的分析与标记。

首先，为了避免依赖冲突、确保迁移工作顺利进行，笔者在 conda 中创建了一个新的隔离环境，基于 Python3.11，并安装了 MindSpore 框架 2.4.0CPU 版本，以及 numpy、matplotlib、scikit-learn 等必要的科学计算和数据分析库。同时，为了便于代码迁移的测试与验证，笔者选择了 Jupyter Notebook 作为初步开发平台，并在其中初始化了基础框架代码，以便验证 MindSpore 接口的兼容性和环境配置的正确性，可以逐块验证和配置代码，方便调试。

其次，笔者保留了原模型所使用的数据集，以确保迁移前后实验结果的可比性。通过仔细审查原模型的数据加载逻辑和相关 sh 脚本文件，笔者记录了包括数据预处理步骤、模型超参数以及训练设置等在内的关键实验配置参数，确保迁移性能与原模型相符。

此外，为明确迁移工作的重点和优先级，笔者使用 MindSpore 提供的 API 扫描插件（MindSpore Dev Toolkit）[2] 在 VSCode 中快速识别模型中的 PyTorch 接口，进而对原模型的 PyTorch 代码进行了详细的功能分析，在此基础上，笔者手动审查了数据处理、训练过程以及模型构建等关键模块，详细记录了未能自动转换的接口，在 MindSpore 官方文档 [3] 中查阅，作为后续手动调整的重点。这种系统化的分析与标记，不仅明确了迁移任务的框架，也极大提高了迁移工作的效率，为后续代码迁移指明了方向。

2.2. PCA 降维相关资料

在测试模型泛化能力时，面对维度较大的数据集，计算资源的限制常常成为挑战。为此，笔者在 MindSpore 并行计算框架的基础上，希望通过合理的数据预处理方法减少空间资源的损耗。高维数据处理中，降维是常用的技术手段，包括主成分分析（Principal Component Analysis, PCA）、线性判别分析（Linear Discriminant Analysis, LDA）、多维尺度变换（Multi-dimensional Scaling, MDS）以及 t-SNE 等方法。

SparseTSF 模型通过线性层在稀疏化约束下对时序数据进行映射，因此笔者更倾向于选择同为线性方法的 PCA 对数据进行预处理。PCA 以其较高的计算效

率和线性降维的特性，成为笔者重点关注的技术之一。

PCA 通过协方差矩阵的特征值分解，将数据投影到保留最大方差的方向，从而实现降维。特征值的大小反映了投影方向所包含的方差信息，而对应的特征向量则定义了这一方向。笔者深入分析了 PCA 降维的基本原理，理解其通过最大化数据方差实现信息保留的设计思想。

然而，仅凭 PCA 的降维机制无法保证降维后的数据仍符合 SparseTSF 模型的需求。PCA 在实践中的局限性主要体现在以下方面：其假设数据满足线性分布，对非线性关系较难处理；并且对数据分布有一定要求（例如近似正态分布）。这些限制可能对 SparseTSF 的性能产生影响。[5]

由于 SparseTSF 接收到的数据是符合线性特征的，假设输入数据具有完美的周期性，则每个周期内部在 PCA 投影之后保持的主要特征是可与下一周期的同方位数据相协调的，在这个过程中，面对并不完美的周期性数据，笔者通过结合主成分分析与聚类分析的方法，可以消除这一影响。因此，笔者最终采用对每段周期数据进行 PCA 降维的方式，来解决空间消耗过大的问题。

3. 方法介绍

完成人：章昊、丛方昊

3.1. 迁移过程

为了高效完成代码迁移，笔者简化了 SparseTSF 模型的源代码，移除了未使用的工具类和函数，确保代码聚焦核心功能，降低了迁移复杂度。迁移后的代码框架如图3所示。各部分的迁移过程如下：

1. 数据集加载与划分：迁移过程中，数据加载模块从 PyTorch 的 `Dataset` 和 `DataLoader` 替换为 MindSpore 的 `mindspore.dataset.GeneratorDataset`，以直接生成数据集。数据标准化和分类划分依然依赖 `sklearn` 完成，并引入 `MultiTimeSeriesDataset` 类管理输入输出形式和参数。

2. SparseTSF 模型类：迁移后模型核心模块保留了 `Conv1d`、`Dense` 等功能，使用 MindSpore 的 `nn.Cell` 构建。主要调整包括：

- **构建方法修改：**按照 MindSpore 中 nn.Cell 的规定，模型需要重写 `construct` 方法，类似 PyTorch 中的 `forward`，作为模型的运算方法。
- **上下采样的转置操作：**将 PyTorch 的 `tensor.permute(shape)` 替换为 MindSpore 的 `tensor.ops.permute(tensor, shape)`。
- **滑动聚合的维度一致性：**使用 MindSpore 的 `ops.mean` 参数 `keep_dims=True` 替代 PyTorch 的 `unsqueeze`，简化了升维和降维操作。

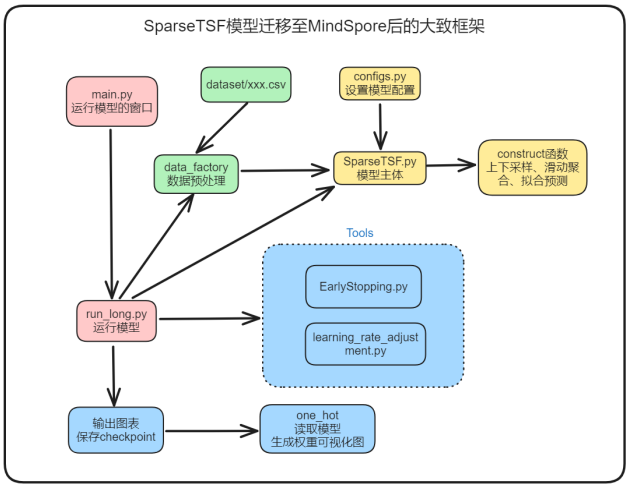

图 3. 迁移后的大致框架

3. tools 模块：迁移后删除未使用的模块，仅保留 `adjust_learning_rate` 和 `earlystopping` 功能：

- **学习率调整：**使用 MindSpore 的学习率 Tensor 机制替代 PyTorch 的 `schedulers`，通过生成与步数匹配的 Tensor 实现学习率的动态调整。
- **早停机制：**用 MindSpore 的 `save_checkpoint` 方法替代 PyTorch 的 `save_checkpoint`，实现模型保存与调用。对于早停的判断与实现，均与 PyTorch 框架中类似。

4. 模型运行模块：训练和测试过程的迁移进行了精简与分层：

- **误差计算：**定义统一的 `evaluate` 函数，利用 MindSpore 的 `create_tuple_iterator` 获取数据对，高效计算损失。
- **训练与测试：**仿照 MindSpore 相关项目案例 [3]，通过 `train_one_step` 和 `train_one_epoch` 分

离逻辑，便于定位问题和调试。测试模块利用最佳模型的 checkpoint 直接调用 `evaluate` 函数完成评估。

- **工具调用：**在实例化模型前自定义 `Adam` 优化器和梯度计算器，使模型与 MindSpore 框架无缝衔接。

3.2. 优化方法介绍

本研究采用了 Scikit-learn 库中的 PCA 模块对数据进行降维处理，以下是具体实现过程的介绍：

笔者通过 `sklearn.decomposition.PCA` 模块实现了降维的核心步骤，结合 `StandardScaler` 进行预处理，完成了从高维特征到低维特征空间的映射。

具体而言，Scikit-learn 提供的 `n_components` 参数可以灵活调整降维目标，根据实验需要优化特征数量。这一参数直接决定了降维结果的特征数量，同时能够保留数据中主要的方差信息。对于 electricity 数据集，使用 `PCA(n_components=200)` 指定降维后的目标维度为 200。随后调用 `pca.fit_transform(X_scaled)`，即可完成对数据的 PCA 降维结果。其中，`fit` 阶段计算标准化数据的协方差矩阵，并提取特征值和特征向量；`transform` 阶段根据提取的主成分矩阵将数据投影到新的低维空间。

4. 实验及分析

完成人：章昊、高景珩

4.1. 迁移前后结果分析

4.1.1 预测结果

由于预测序列较长，因此笔者增大了图表的宽度，添加图例，以便展示更多细节。

Etth2 数据集预测结果如图4、5所示，分别展现了模型在 Torch、MindSpore 下的预测效果。分析图片可知，两者均准确把握了数据的趋势变化，对具体数值的误差也在合理范围内。因此可以说明迁移后的模型得到的正确的效果，MindSpore 框架下模型的预测能力并未下降。

Electricity 数据集的预测结果如图6、7所示，分别为迁移前与迁移后的结果。与上述分析类似，MindSpore 框架下模型的效果与 PyTorch 框架并无明显差异，预测效果优秀。

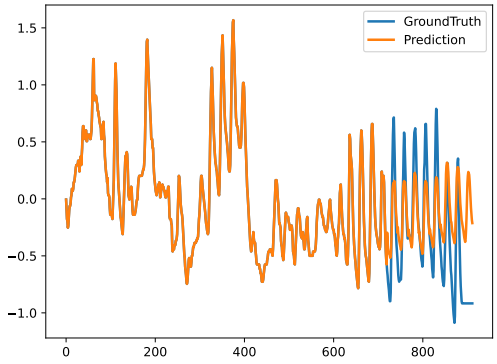

图 4. ETTh2 数据集 720-192 的预测结果（原框架）

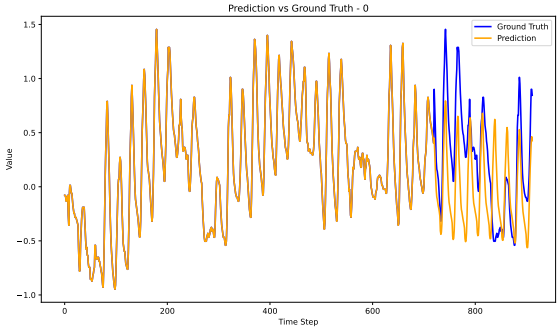

图 5. ETTh2 数据集 720-192 的预测结果（迁移后）

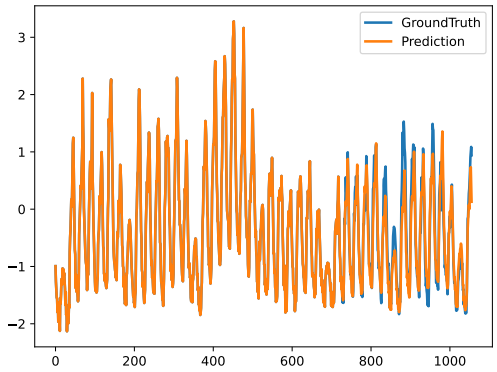

图 6. Electricity 数据集 720-336 的预测结果（原框架）

4.1.2 损失差异

对 ETTh1 数据集（图8 9）的损失进行分析绘图，并原模型的损失曲线对比，在数值上判断 MindSpore 框架下的预测效果。这里将**四次不同预测长度的曲线**绘制在同一张图中，便于观察。损失变化曲线如图。

分析损失变化可知，在 MindSpore 框架下，损失的变化更为平稳，这可能与 MindSpore 的图模式、Adam

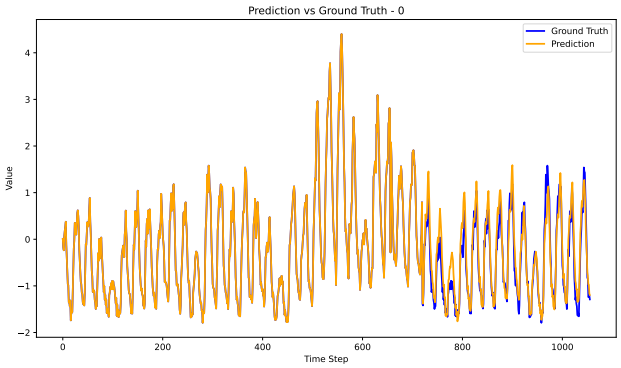

图 7. Electricity 数据集 720-336 的预测结果（迁移后）

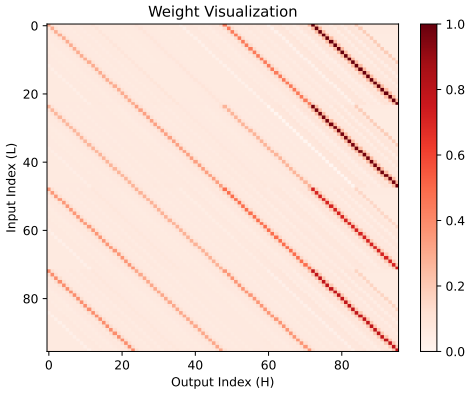

图 10. 迁移后 SparseTSF 参数独热编码可视化

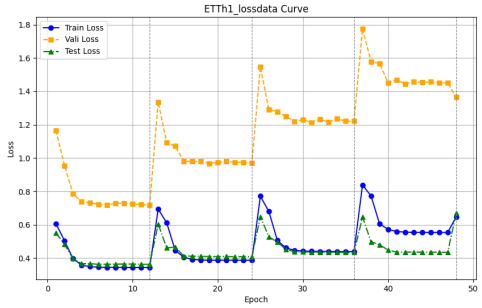

图 8. 四种预测长度下 ETTh1 数据集的损失曲线（原框架）

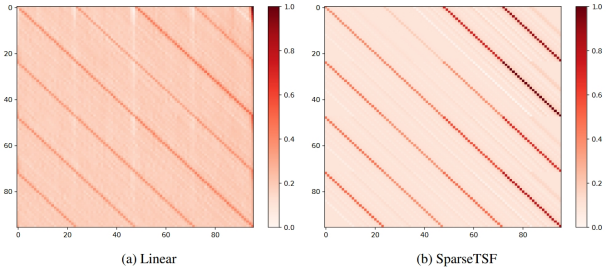

图 11. Linear 与 SparseTSF(Torch) 的参数可视化

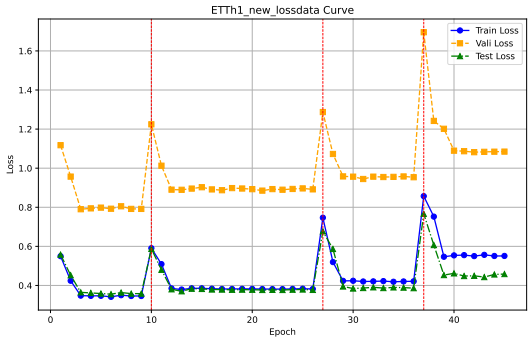

图 9. 四种预测长度下 ETTh1 数据集的损失曲线（迁移后）

优化器的配置有关。从总体上看，两者预测完成所用的 epoch 类似，得到的最终损失相差无几。可以认为，在 MindSpore 框架的模型迁移比较成功。

以下表格1，展示了 PyTorch 和 MindSpore 框架下模型的性能对比：与上述图表的分析结果类似，两者损失结果相差不大；但 MindSpore 在小数据集的运行速度上有一定优势，大大降低了运行时间。

4.1.3 权重可视化

根据 SparseTSF 论文中提到的等效特征矩阵 [1]

$$weight' = SparseTSF(\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix})^T$$

笔者也对迁移后的模型进行了独热编码可视化分析：

对 ETTh1 数据集进行序列长度、预测长度均为 96，周期设置为 24 的预测，可以得到如图10所示的权重可视化热力图。

对比论文中原图11可知，迁移后的模型同样得到了等间隔权重分布条带，在图中即体现为**间隔 24 的平行深色条带**，证明了 Sparse 技术可以通过加强模型从数据中提取周期性特征的能力，进而提升模型在 LTSF 任务中的预测能力。[1]

表 1. 模型迁移前后性能差异分析（以 Etth1 数据集为例）

性能指标	PyTorch	MindSpore	差异	说明
训练集损失	0.344	0.347	+0.8%	MindSpore 损失更高
验证集损失	0.718	0.794	+10.6%	MindSpore 损失更高
测试集损失	0.363	0.354	−2.5%	MindSpore 损失更低
训练时间 (秒/epoch)	20.5	5.8	−71.7%	MindSpore 小数据集下运行更快

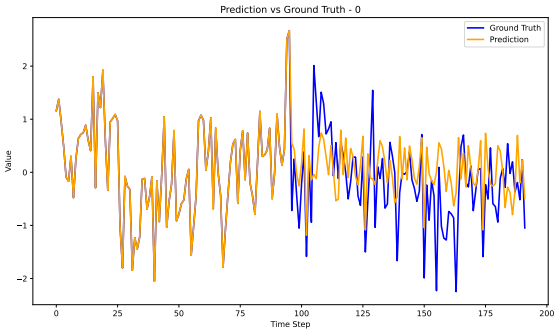

图 12. PCA 优化后的 electricity 数据集预测

4.2. PCA 降维测试

通过采用 PCA 对数据进行降维，我们成功地将数据维度压缩至约 200，从而在 16GB 内存的计算机上顺利完成了模型的运行。在 electricity 数据集上的实验结果如图12所示。尽管预测值与真实数据的贴合度未达到理想水平，但整体趋势基本保持一致。这表明，尽管 PCA 降维在一定程度上丢失了部分信息，但对周期性数据的核心特征影响较小，因此在资源受限的情况下，降维仍然是一种有效的解决方案。

值得注意的是，降维后的数据虽然能够支持模型正常运行，但降低了方法的可解释性。对于回归任务而言，模型依赖于数据中的变异信息，而 PCA 降维后的每个特征实际上是原始特征的线性组合。这导致降维后的特征难以直接映射到原始数据的物理意义上，进而限制了特征解释的直观性和可用性。

尽管如此，PCA 降维在节约计算资源方面展现了显著的优势。该方法有效保留了数据中最具区分度的信息，同时去除了冗余和噪声成分，从而降低了过拟合的风险。这不仅为模型的泛化能力提供了支持，也证明了 PCA 作为预处理步骤的实用性和可靠性。

4.3. 消融实验

为进一步验证 SparseTSF 模型中周期选择对拟合效果的影响，笔者通过对 ETTh1 数据集在序列长度、预测长度均为 720 的条件下，分别选取周期 48、24、12、2，进行模型拟合，得到测试集损失结果如表2所示。

可见，不同的周期选取在一定程度上会影响模型的拟合效果。在 ETTh1 数据集中，选取周期为 2、24 时效果最佳。可能的解释是，由于滑动聚合的存在，单个时间点会受到周围同周期数据的影响。周期较小时，模型更能把握数据间的变化关系；另一方面，周期较大时，模型能从整体上获取数据的内在联系。因此，周期的大或小相互制约，需要根据数据集的内在特征选取。

最终结果也表明，SparseTSF 模型正是通过把握数据集的内在特征——周期，从而简化预测过程，使用较少的参数即可达到优秀的效果。

表 2. 消融实验-周期选择（以 Etth1 数据集为例）

周期	测试集损失	说明
2	0.3560	无
12	0.3608	损失最高
24	0.3540	损失最低
48	0.3581	无

5. 总结

完成人：高景珩

本文中，笔者对 SparseTSF 模型进行了代码复现、框架迁移和优化数据集维度的进一步研究。首先，笔者完成了对源代码特点的学习并成功得到与原作相似的结果。其次，笔者将代码迁移到 Mindspore 框架上，并得到优秀的结果。接着，针对于数据集维度大、内存

占用过高的问题，笔者为 SparseTSF 模型增加了 PCA 降维方法，在保证预测结果的准确性的前提下，取得了更高的运行速率、降低了内存占用。

展望未来，笔者希望从以下几个方面进一步开展研究工作：

1. **提升模型精度：**探索更先进的特征提取与选择方法，例如结合深度特征学习与降维技术的混合方法，以进一步提升预测效果。
2. **优化计算效率：**针对大规模数据集，研究模型并行化与分布式计算方法，进一步学习 CUDA、Ascend 的调用方式，以适应实际场景中对高效处理的需求。
3. **探索更智能的训练机制：**研究探索**自适应周期**的优化策略，自动选择最佳周期，提升预测效果。

6. AI 工具的使用

完成人：章昊

在本研究过程中，我们充分利用了 AI 工具（ChatGPT [4]）来辅助代码开发、理论学习以及分析讨论。以下从三个方面详细介绍 AI 工具在研究中的应用场景和具体方式。

6.1. 名词解释与原理介绍

在研究中，我们经常需要深入理解新概念或方法的理论背景。例如，在 PCA 降维、滑动聚合等数学方法时，AI 工具提供了详细的名词定义、背景知识以及原理解释。这些信息不仅包括经典定义和公式，还通过直观的例子和类比进一步增强了我们的理解。此外，对于深度学习中的关键模块（如学习率调整机制），AI 提供了详细的实现方式与公式原理帮助理解。这种即时学习的方式显著提高了研究效率，避免了因繁琐查阅文献而造成的时间浪费。

6.2. 代码提问与修改

AI 工具在代码开发中为研究提供了高效的支持。特别是在模型实现和调试的过程中，对于复杂代码的逻辑错误或优化需求，我们通过 AI 工具进行交互式提问，例如检查逻辑漏洞、改进算法实现。当我们尝试 SparseTSF 中优化特定模块（如 EarlyStopping 机制）时，AI 工具快速提供了可行的代码片段，同时解释了其实现细节，帮助我们在短时间内完成从问题定位到解决的过程。此外，AI 工具还帮助我们优化了基于 MindSpore 框架的训练代码，通过简化冗余模块、改进内存管理策略等方式，使代码更加高效和稳定。

6.3. 思路讨论与分析

在方法选择和模型优化中，AI 工具为我们的分析提供了重要参考。例如，在探讨是否采用 PCA 降维时，AI 工具帮助我们分析了 PCA 的线性假设及其对时序数据的影响，并提供了对比分析其他降维方法（如 LDA 或 t-SNE）的优劣势。这使我们能够基于理论和实验数据做出更加合理有效的决策。此外，在对模型性能分析时，AI 工具通过快速生成图表、对实验结果进行可视化分析，帮助我们全面理解不同设置对模型预测能力的影响。通过与 AI 工具的交互讨论，我们能够更深入地探讨优化方法的合理性及其潜在改进方向。

7. 本研究的不足

完成人：丛方昊

在面对数据缺失比例较高的数据集时，SparseTSF 模型捕获周期信息的能力会受到显著影响，表现为周期性特征的模糊化。此外，本研究对数据进行 PCA 降维处理的方式进一步加剧了 SparseTSF 对缺失信息的敏感性，尤其在降维过程中对数据周期性特征的破坏较为明显，从而导致模型对原始数据依赖性的增强。

为缓解上述问题，我们计划在未来结合 KNN 插补方法与 PCA 降维对数据进行联合预处理，以弥补数据中缺失部分的信息损失，从而提升模型的泛化能力和对周期性特征的捕获效果。KNN 插补能够有效恢复缺失值的近似分布，为后续降维提供更高质量的输入数据，从而减小因缺失数据导致的周期性特征模糊化问题 [6]。

此外，对于多周期数据集（例如 traffic 数据集，其既包含日周期特征，也存在周周期特征），PCA 降维的使用可能带来额外的挑战。尽管原模型可通过下采样的方式在保留内部小周期特征的同时，重点捕获更大周期范围的信息，但 PCA 降维处理后特征的物理可解释性显著降低，这可能对数据的内部周期结构造成破坏，进而影响模型性能。

针对这一问题，我们计划在未来对 PCA 降维方法在更多多周期数据集上进行测试与优化。通过系统性实验，进一步分析 PCA 降维对周期性特征的影响，并

探索改进方案（如结合更适合周期性数据的降维方法），以在保证模型高效运行的同时，尽可能保留数据的周期性特征。

8. 小组分工

- **组长-高景珩**：引言编写，实验分析，总结成果
- **组员-丛方昊**：引言编写，优化分析，提出不足
- **组员-章昊**：迁移分析，实验分析，整理论文

参考文献

[1] Lin, S., Lin, W., Wu, W., Chen, H., and Yang, J. *SparseTSF: Modeling Long-term Time Series Forecasting with 1k Parameters*. 2024. Available on *arXiv preprint arXiv:2405.00946*. 1, 5

[2] MindSpore Development Team. *MindSpore Dev Toolkit Documentation*. 2024. Available at: https://www.mindspore.cn/devtoolkit/docs/zh-CN/r2.2/index.html. Accessed: 2024-12-12. 2

[3] MindSpore Development Team. *MindSpore Documentation*. 2024. Available at: https://www.mindspore.cn/docs/zh-CN/r2.4.0/index.html. Accessed: 2024-12-12. 2, 3

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, LamaAhmad, Ilge Akkaya, Florencia Leoni Aleman, DiogoAlmeida, Janko Altenschmidt, Sam Altman, ShyamalAnadkat, et al. *Gpt-4 technical report. arXiv preprint arXiv:2303.08774*. 2023. 7

[5] Greenacre, M., Groenen, P.J.F., Hastie, T. et al. *Principal component analysis. Nat Rev Methods Primers*. 2022. Available at: https://doi.org/10.1038/s43586-022-00184-w 3

[6] Cunningham, Pádraig and Delany, Sarah Jane. *k-Nearest Neighbour Classifiers - A Tutorial*. 2021. Available on*Association for Computing Machinery (ACM)* http://dx.doi.org/10.1145/3459665.

7