最小卷积递归神经网络加速时空学习

Coku Can Horuz[1], Sebastian Otte[1], Martin V. Butz[2], and Matthias Karlbauer[2]

[1] Adaptive AI Research Group, University of Lübeck, Germany
[2] Neuro-Cognitive Modeling Group, University of Tübingen, Germany

摘要 我们介绍了 MinConvLSTM 和 MinConvGRU，这两种新颖的时空模型结合了卷积递归网络的空间归纳偏置与最小可并行化 RNN 的训练效率。我们的方法扩展了 MinLSTM 和 MinGRU 的对数域前缀和公式到卷积架构中，实现了完全并行训练的同时保留局部空间建模。这消除了在教师强迫过程中更新顺序隐藏状态的需要——这是传统 ConvRNN 模型中的主要瓶颈。此外，我们还借鉴了 xLSTM 架构中的指数门控机制整合到了 MinConvLSTM 中，进一步简化了对数域计算。我们的模型结构上是最简化的且计算效率高，参数数量减少并且提高了可扩展性。我们在两个时空预测任务上评估了我们的模型：Navier-Stokes 动力学和现实世界地势数据。在训练速度方面，我们的架构显著优于标准的 ConvLSTM 和 ConvGRU。此外，在封闭循环自回归模式下，我们的模型在两个领域中也实现了更低的预测误差。这些发现表明，当最小递归结构与卷积输入聚合结合时，为时空序列建模提供了一个有吸引力且高效的替代方案，弥合了递归简单性和空间复杂性之间的差距。

Keywords: RNN · 卷积 LSTM · 卷积 GRU · 最小卷积 LSTM · 最小卷积 GRU · 地形位势 · Navier-Stokes.

1 介绍

时空序列预测一直是视频预测 [25] 和天气临近预报 [23] 等多种应用中的关键挑战。这些任务通常涉及基于先前帧的历史条件来预测高维空间观测的序列。特别是对于像降水临近预报这样的场景，及时和局部化的预测至关重要，因此有效地准确建模空间和时间相关性是必不可少的 [8]。

先前，循环神经网络的卷积变体，尤其是卷积长短期记忆（ConvLSTM）网络 [24] 及其类似方法 [1] 已被证明在这些任务中非常有效。通过将卷积操作整合到输入到状态和状态到状态的转换中，ConvLSTM 捕捉局部空间依

赖关系，同时通过递归建模时间演变。当堆叠在编码器-预测器架构中时，这些模型已经在各种基准 [24,16] 上表现出强大的性能。然而，它们本质上是顺序的性质对训练效率造成了显著限制，特别是在预测范围增加时。

同时，更广泛的序列学习社区开始质疑现有范式（特别是 Transformer）的可扩展性，这激发了对经典递归模型的新兴趣，并启发了强调并行化、效率和简洁性的新架构。在这种情况下，高度简化的架构如 MinLSTM 和 MinGRU[7] 去除了门控机制中的隐藏状态依赖关系，通过前缀求和（也称为并行扫描算法 [3]）实现了完全的并行训练。这些模型展示了有效的序列建模可以通过最小、结构良好的递归实现，而不是增加架构复杂性，这一原则同样被各种现代 RNN 方法 [12,26,20] 所采用。

在这篇论文中，我们通过提出两种形式的最小化卷积 RNN——MinConvLSTM 和 MinConvGRU，将线性和扫描兼容的循环架构的效率转移到时空领域。这些模型扩展了 MinRNNs 的对数域扫描基础公式到 ConvRNN 框架，消除了在教师强制过程中需要进行序列隐藏状态更新的需求，同时通过其卷积结构保持局部空间敏感性。此外，我们借鉴 [2] 的指数门控概念来替换我们在 MinConvLSTM 架构中的传统 sigmoid 门控，进一步简化了模型的对数空间计算。

经验结果显示，在两个基准测试（流体和大地势动力学）中，我们的模型相较于传统的 ConvRNNs 显著加快了训练速度（快 3 到 5 倍），同时在两个领域内都实现了预测精度的提升，甚至在闭环自回归模式下也是如此。这些发现表明，极简且并行的循环结构为未来的时空序列建模提供了一个有吸引力的方向。

2 基础

在本节中，我们回顾递归序列建模的相关关键技术发展，即带门控的 RNN、它们的卷积扩展以及最近关于线性、扫描兼容架构的工作。

2.1 经典门控记忆单元

几十年前的长短期记忆（LSTM）[15] 和门控循环单元（GRU）[4] 仍然是许多现代架构的基础，并被广泛应用于各个领域 [2,7,9]。这些模型引入了门机制来调节信息流动，有效地解决了梯度消失和梯度爆炸问题。其设计包含了受控的非线性：用于调制隐藏状态动态的 sigmoid 门。值得注意的是，

这些门本身依赖于前一时刻的隐藏状态，从而在状态演化中引入了非线性依赖关系。这种门机制不仅稳定了时间步长间的状态传播，还使模型能够追踪和保持相关的时序上下文，称为（隐式）状态跟踪 [5,9,19]。

LSTM

$$
\begin{aligned}
\boldsymbol{\phi}^t &= \sigma\left(\text{Linear}_{\phi}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{\iota}^t &= \sigma\left(\text{Linear}_{\iota}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\tilde{\boldsymbol{s}}^t &= \tanh\left(\text{Linear}_{s}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{s}^t &= \boldsymbol{\phi}^t \odot \boldsymbol{s}^{t-1} + \boldsymbol{\iota}^t \odot \tilde{\boldsymbol{s}}^t \\
\boldsymbol{\omega}^t &= \sigma\left(\text{Linear}_{\omega}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{h}^t &= \boldsymbol{\omega}^t \odot \tanh\left(\boldsymbol{s}^t\right)
\end{aligned}
$$

GRU

$$
\begin{aligned}
\boldsymbol{z}^t &= \sigma\left(\text{Linear}_{z}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{r}^t &= \sigma\left(\text{Linear}_{r}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\tilde{\boldsymbol{h}}^t &= \tanh\left(\text{Linear}_{h}[\boldsymbol{x}^t, \boldsymbol{r}^t \odot \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{h}^t &= \left(1 - \boldsymbol{z}^t\right) \odot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}
$$

在 LSTM 中，ϕ, ι, ω 是遗忘门、输入门和输出门，控制通过单元状态 $\boldsymbol{s}$ 和隐藏状态 $\boldsymbol{h}$ 的信息流。GRU 简化了这种表述，并仅使用重置门和更新门（即 $\boldsymbol{r}, \boldsymbol{z}$）以及指数移动平均（EMA）方案来控制隐藏状态的流动。

2.2 卷积 RNNs

ConvLSTM [24] 和 ConvGRU [1] 通过卷积建模空间结构并通过非线性递归建模时间动态。这是通过将 RNN 部署为在空间域上的一系列局部连接模块来实现的，每个模块应用相同的计算和共享权重，同时根据输入和邻近状态的局部感受野发展不同的隐藏状态。技术上，这一转变是通过用卷积替换线性操作来实现的。符号类似于 LSTM 和 GRU 方程：

ConvLSTM

$$
\begin{aligned}
\boldsymbol{\phi}^t &= \sigma\left(\text{Conv}_{\phi}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{\iota}^t &= \sigma\left(\text{Conv}_{\iota}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\tilde{\boldsymbol{s}}^t &= \tanh\left(\text{Conv}_{s}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{s}^t &= \boldsymbol{\phi}^t \odot \boldsymbol{s}^{t-1} + \boldsymbol{\iota}^t \odot \tilde{\boldsymbol{s}}^t \\
\boldsymbol{\omega}^t &= \sigma\left(\text{Conv}_{\omega}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{h}^t &= \boldsymbol{\omega}^t \odot \tanh\left(\boldsymbol{s}^t\right)
\end{aligned}
$$

ConvGRU

$$
\begin{aligned}
\boldsymbol{z}^t &= \sigma\left(\text{Conv}_{z}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{r}^t &= \sigma\left(\text{Conv}_{r}[\boldsymbol{x}^t, \boldsymbol{h}^{t-1}]\right) \\
\tilde{\boldsymbol{h}}^t &= \tanh\left(\text{Conv}_{h}[\boldsymbol{x}^t, \boldsymbol{r}^t \odot \boldsymbol{h}^{t-1}]\right) \\
\boldsymbol{h}^t &= \left(1 - \boldsymbol{z}^t\right) \odot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}
$$

尽管这些方程使用相同的符号，变量张量获得了一个额外的空间维度。

ConvLSTM 和 ConvGRU 在时空基准测试中表现出具有竞争力的性能，并在各种领域找到应用 [17,23,16]。虽然集成了强大的工具，类似于非线性循环网络，但由于其仅限于顺序处理的特性，它们注定只能保持小规模。

2.3 线性递归模型

近期引入的状态空间模型（SSMs）解决了非线性递归模型的扩展限制，并提供了线性解决方案。从 HiPPO 理论 [11] 开始，该理论为线性递归引入了结构化的权重矩阵，并通过递归-卷积等价性 [13] 提升了模型效率，SSMs 拉开了线性 RNN 时代的帷幕。这些模型，如 S4 [12] ，S5 [26] ，S6 即 Mamba [10]，和 S6.2 即 Mamba2 [6] 在序列建模中表现出色，达到了并超过了 Transformer[27] 的性能。事实上，SSMs 可以使用线性数量的处理器在对数并行时间内计算，并且只需要线性内存，而 Transformers 由于其全局自注意力机制，在使用二次数量的处理器时表现出对数并行时间复杂度，并导致二次内存使用。在自回归推理过程中，SSMs（因为它们是 RNNs）只需常量内存。

自从 SSMs 问世以来，人们采用不同的方法开发了其他几种递归线性模型。xLSTM[2] 结合了 SSM 架构与注意力机制（mLSTM），同时也使用了顺序模式循环（sLSTM）。此外，一些线性 RNN 模型使用并行扫描算法 [3]，而不是利用卷积-递归等价性。这种方法产生了对数级的并行时间复杂度，并缓解了无法处理不规则采样时间序列 [26] 的非平凡卷积核计算问题。特别是 [18] 和 [21] 利用了线性 RNN 背景下并行扫描的特性，在许多领域中实现了高可扩展性和令人期待的表现。

2.4 最小的 RNN 变体

最近提出的 MinLSTM 和 MinGRU 模型 [7]，类似于早期的贡献 [18]，通过移除非线性递归关系来简化经典的 LSTM 和 GRU 架构，从而减少了参数的数量。结果得到的线性状态更新可以使用并行扫描计算，使全模型并行化成为可能。

MinLSTM

$$
\begin{aligned}
\boldsymbol{\Phi}^t &= \sigma\left(\text{Linear}_{\Phi}[\boldsymbol{x}^t]\right) \\
\boldsymbol{\iota}^t &= \sigma\left(\text{Linear}_{\iota}[\boldsymbol{x}^t]\right) \\
\hat{\boldsymbol{\Phi}}^t,\ \hat{\boldsymbol{\iota}}^t &= \frac{\boldsymbol{\Phi}^t}{\boldsymbol{\Phi}^t + \boldsymbol{\iota}^t},\ \frac{\boldsymbol{\iota}^t}{\boldsymbol{\Phi}^t + \boldsymbol{\iota}^t} \\
\tilde{\boldsymbol{h}}^t &= \text{Linear}_h[\boldsymbol{x}^t] \\
\boldsymbol{h}^t &= \hat{\boldsymbol{\Phi}}^t \odot \boldsymbol{h}^{t-1} + \hat{\boldsymbol{\iota}}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}
$$

MinGRU

$$
\begin{aligned}
\boldsymbol{z}^t &= \sigma\left(\text{Linear}_z[\boldsymbol{x}^t]\right) \\
\tilde{\boldsymbol{h}}^t &= \text{Linear}_h[\boldsymbol{x}^t] \\
\boldsymbol{h}^t &= \left(1 - \boldsymbol{z}^t\right) \odot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}
$$

符号表示与标准 LSTM 和 GRU 公式一致。MinLSTM 和 MinGRU 都省略了门中的隐藏状态反馈，移除了非线性递归并使更新可以并行化。在 MinLSTM 中，遗忘门和输入门被标准化以形成两个门控输出的凸组合。输出门也被丢弃。MinGRU 进一步通过使用单个更新门（省略重置门）来进行 EMA 样式的候选融合来简化 GRU 结构。这些更改减少了参数数量并允许高效训练，在对数域中实现，其中乘法变为加法，并且可以应用前缀和算法（详情见 [14]）。

3 最小卷积递归神经网络

在本节中，我们通过引入卷积变体将最小 RNNs[7] 扩展到时空域，即 MinConvLSTM 和 MinConvGRU。我们进一步使用指数门控优化 MinConvLSTM。

3.1 MinConvLSTM 和 MinConvGRU

我们将所有门和输入（新的隐藏状态候选者）的线性输入聚合替换为卷积，并得到以下方程：

MinConvLSTM

$$\begin{aligned}
\boldsymbol{\phi}^t &= \sigma\left(\text{Conv}_{\phi}[\boldsymbol{x}^t]\right) \\
\boldsymbol{\iota}^t &= \sigma\left(\text{Conv}_{\iota}[\boldsymbol{x}^t]\right) \\
\tilde{\boldsymbol{h}}^t &= \text{Conv}_{h}[\boldsymbol{x}^t] \\
\hat{\boldsymbol{\phi}}^t,\ \hat{\boldsymbol{\iota}}^t &= \frac{\boldsymbol{\phi}^t}{\boldsymbol{\phi}^t + \boldsymbol{\iota}^t},\ \frac{\boldsymbol{\iota}^t}{\boldsymbol{\phi}^t + \boldsymbol{\iota}^t} \\
\boldsymbol{h}^t &= \hat{\boldsymbol{\phi}}^t \odot \boldsymbol{h}^{t-1} + \hat{\boldsymbol{\iota}}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}$$

MinConvGRU

$$\begin{aligned}
\boldsymbol{z}^t &= \sigma\left(\text{Conv}_{z}[\boldsymbol{x}^t]\right) \\
\tilde{\boldsymbol{h}}^t &= \text{Conv}_{h}[\boldsymbol{x}^t] \\
\boldsymbol{h}^t &= \left(1 - \boldsymbol{z}^t\right) \odot \boldsymbol{h}^{t-1} + \boldsymbol{z}^t \odot \tilde{\boldsymbol{h}}^t
\end{aligned}$$

所有卷积操作都可以并行执行，因为它们通过隐藏状态反馈不具有时间依赖性。唯一的顺序组件是线性递归，它仅取决于输入序列和门激活的序列。如 [7] 中所建议，在对数域中操作时，这种结构可以通过并行扫描实现高效计算。也就是说，使用 $\log(\sigma(x)) = -\,\text{Softplus}(-x)$ 我们可以得到：

$$\log(\hat{\boldsymbol{\phi}}) = -\,\text{Softplus}(\text{Softplus}(-\text{Conv}_{\phi}[\boldsymbol{x}]) - \text{Softplus}(-\text{Conv}_{\iota}[\boldsymbol{x}])) \tag{1}$$

$$\log(\hat{\boldsymbol{\iota}}) = -\,\text{Softplus}(\text{Softplus}(-\text{Conv}_{\iota}[\boldsymbol{x}]) - \text{Softplus}(-\text{Conv}_{\phi}[\boldsymbol{x}])) \tag{2}$$

可以在时间独立的情况下无需昂贵的除法或 sigmoid 函数来计算对数归一化门。在下面的内容中，我们将进一步简化门控机制。

3.2 最小卷积指数 LSTM

受对数域计算的启发，并受到 [2] 的激励，我们提出了 MinConvLSTM 中指数门控的一种变体以进一步提高模型效率。为了实现这一点，我们将 sigmoid 替换为指数函数：

$$\boldsymbol{\phi} = \exp\left(\text{Conv}_{\phi}[\boldsymbol{x}]\right) \tag{3}$$

$$\boldsymbol{\iota} = \exp\left(\text{Conv}_{\iota}[\boldsymbol{x}]\right) \tag{4}$$

然后，通过利用 sigmoid 函数和指数函数之间的关系，标准化的遗忘门和输入门采取以下形式：

$$\hat{\boldsymbol{\phi}} = \frac{\boldsymbol{\phi}}{\boldsymbol{\phi} + \boldsymbol{\iota}} = \sigma\left(\text{Conv}_{\phi}[\boldsymbol{x}] - \text{Conv}_{\iota}[\boldsymbol{x}]\right) \tag{5}$$

$$\hat{\boldsymbol{\iota}} = \frac{\boldsymbol{\iota}}{\boldsymbol{\phi} + \boldsymbol{\iota}} = \sigma\left(\text{Conv}_{\iota}[\boldsymbol{x}] - \text{Conv}_{\phi}[\boldsymbol{x}]\right) = 1.0 - \hat{\boldsymbol{\phi}} \tag{6}$$

这种表达方式使两个门可以从单次 sigmoid 计算中得出，从而提高计算效率。相应地，我们可以对这些表达式取对数以获得 $\log(\hat{\boldsymbol{\phi}})$ 和 $\log(\hat{\boldsymbol{\iota}})$。

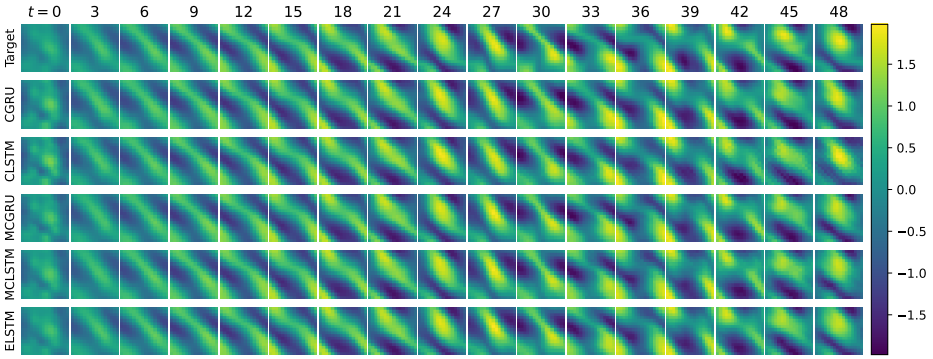

图 1. 不同模型（行）在不同预测提前期（列）下对 30 个时间步长的 Navier-Stokes 动力学进行闭合循环预测，前 20 个步骤使用教师强迫。第一行显示真实值，后续行依次为 ConvGRU、ConvLSTM、MinConvGRU、MinConvLSTM 和 MinConvExpLSTM。

4 实验与结果

我们进行了两组实验，以将我们的 MinConvGRU 和 MinConvLSTM 架构与已有的 ConvGRU 和 ConvLSTM 模型进行基准测试。在第一组实验中，我们在周期性域内对不可压缩 Navier-Stokes 动力学进行训练。第二，我们研究了这些模型学习和预测称为地势的真实世界天气动力学数据集的能力。随后，我们讨论了运行时间差异，并提及在张量处理单元（TPUs）上生成的结果，在这里我们将 PyTorch 的卷积操作运行时与不同大小的张量进行了基准测试。

4.1 纳维-斯托克斯方程

Navier-Stokes 方程通常应用于流体模拟，并可用于生成高度非线性的动力学。典型示例包括大气天气动力学的模拟和预测，以及用于确定物体设计中空气阻力的流动模拟。

数据生成 我们在模拟 50 个时间步长的时空动力学时，遵循 [16]，并将区域离散化为一个 64×64 网格，雷诺数为 $Re = 1 \times 10^3$ 且具有周期性边界条件。为了减少计算开销，我们进一步将空间域下采样四倍，并获得每时间步 16×16 像素的分辨率。我们生成 1000、50 和 200 个样本用于训练、验证和测试。

模型配置 由于每个模型每单位实现的权重矩阵数量不同，我们调整每一层的隐藏通道数，使得所有模型大约有 175 k 个参数。经过一个 1×1 卷积作

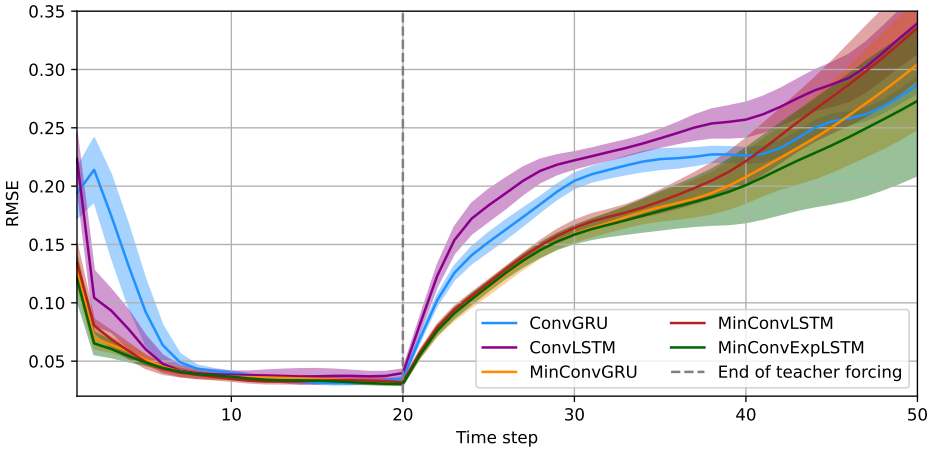

图 2. 不同模型在预测纳维-斯托克斯动力学时随时间变化的均方根误差。虚线位于 $t = 20$ 处，表示从教师强制到闭环的过渡，在此过程中模型开始自回归地对未来进行预测。实线和阴影区域表示用五个具有不同随机种子训练的模型的平均值和标准差。

为编码器将输入投影到深度为 c 的潜空间中，我们构建了四个隐藏层，这些隐藏层分别具有 ConvGRU、ConvLSTM、MinConvGRU、MinConvLSTM 或 MinConvExpLSTM 单元，数量分别为 $c = 28$、25、49、40 和 40。另一个 1×1 卷积最终将演化的潜在状态投影回到具有通道深度为 1 的数据域。我们应用跳跃连接、层归一化和组归一化进行进一步的正则化，特别是为了在提议模型中的线性层之间引入非线性。

训练配置 我们在使用 Adam 训练所有模型时，采用的批处理大小为一，并且权重衰减设置为 1×10^{-2}。我们使用余弦学习率调度器，在 30 个周期内将初始学习率 $\eta_0 = 5 \times 10^{-4}$ 衰减至零。我们在 NVIDIA RTX 1060 GPU 上进行训练时，从数据中随机裁剪长度为 25 的序列，并使用 20 步教师强制，接着在训练过程中使用五步闭合循环。6 GB

结果 在测试我们的模型时，我们使用了 20 步的教师强制后接 30 步的闭环模式。我们在 Figure 1中定性地可视化了一个 Navier-Stokes 动力学示例序列以及每个模型预测的结果。乍一看，所有模型都能成功预测出稳定且现实的 Navier-Stokes 动力学。在最后的时间步骤中，我们发现 ConvGRU 和 ConvLSTM 偏离了真实值，失去了轮廓（我们认为这是平滑导致的），如 Figure 1的第 44 帧所示。这些趋势在 Table 1以及 Figure 2中得到了定量支持，后者详细说明了所有模型在测试集上随时间变化的 RMSE。实线表示

表 1. 不同模型在 Navier-Stokes 和地势动力学上的预测误差比较，分别在序列长度为 25 步和 24 步的情况下进行训练。两个数据集均使用了 20 步的教师强迫。$\mathrm{RMSE}_{\mathrm{TF}}$ 和 $\mathrm{RMSE}_{\mathrm{CL}}$，对于 Navier-Stokes 乘以 1×10^{2}，而对于地势动力学则乘以 1×10^{-2}，在序列长度为 50 且 Navier-Stokes 有 20 步教师强迫，以及地势动力学的 96 时间步中有 20 步教师强迫的一个独立测试集上进行计算。

	Geopotential		Navier-Stokes	
Model	$\mathrm{RMSE}_{\mathrm{TF}}$	$\mathrm{RMSE}_{\mathrm{CL}}$	$\mathrm{RMSE}_{\mathrm{TF}}$	$\mathrm{RMSE}_{\mathrm{CL}}$
ConvGRU	1.88 ± 0.12	21.52 ± 5.18	6.90 ± 0.96	20.95 ± 0.66
ConvLSTM	0.73 ± 0.09	12.76 ± 2.30	5.70 ± 0.82	23.81 ± 1.32
MinConvGRU	0.88 ± 0.23	12.61 ± 2.12	4.55 ± 0.36	18.57 ± 1.98
MinConvLSTM	$\mathbf{0.70 \pm 0.08}$	$\mathbf{10.80 \pm 1.49}$	4.57 ± 0.40	19.79 ± 1.59
MinConvExpLSTM	0.71 ± 0.09	10.94 ± 1.37	$\mathbf{4.37 \pm 0.28}$	$\mathbf{17.87 \pm 2.48}$

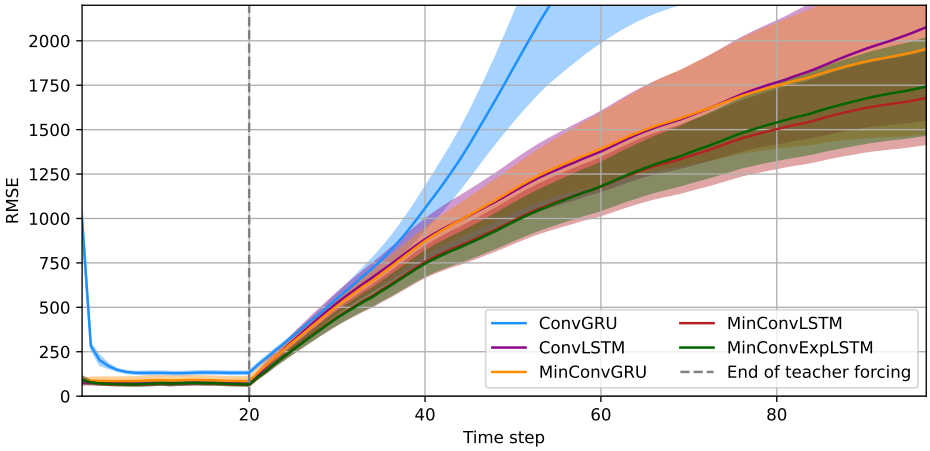

图 3. 不同模型在预测全球重力势动态时随时间变化的均方根误差。虚线位于 $t = 20$ 处，表示从教师强制到闭环的过渡，在此阶段，模型开始自回归地对未来进行预测展开。实线和阴影区域表示使用五个具有不同随机种子训练的模型的平均值和标准差。损失值按 10^{2} 缩放。

使用不同随机种子训练的五个独立模型的平均值，而阴影区域标记各自的标准差。我们发现 MinConvGRU、MinConvLSTM 和 MinConvExpLSTM 在教师强迫和闭合回路预测中均优于它们的 ConvGRU 和 ConvLSTM 对应模型。

4.2 地势位函数

作为一个实际基准，我们考虑 500 hPa 的地势高度，记为 Φ_{500}。这个数据集代表海平面以上的高度（单位：米），其中大气压力为 500 hPa（百帕斯卡），相当于海平面以上大约 5.5 km 的高度。它本质上是一种描述大气中间层的方式，广泛用于天气预报和气候研究。Φ_{500} 标志着大气科学中的一个核心变量，它描述了大规模区域和全球气象现象。

数据准备 我们从 WeatherBench 数据集 [22] 中下载所有可用的 Φ_{500} 字段。我们考虑 5.625° 的最低分辨率，该分辨率包含 32×64 个像素跨越我们的地球，并进一步将空间分辨率以两倍下采样，从而获得每小时时间步长 16×32 个像素的分辨率。遵循常见做法，我们使用 1979 年至 2014 年进行训练，2015 年至 2016 年用于验证，以及 2017 年至 2018 年用于测试。

模型配置 在这一系列实验中，我们遵循了用于学习 Navier-Stokes 动力学的参数配置，并做了一点修改：我们使用三层而不是四层，分别为 ConvGRU、ConvLSTM、MinConvGRU、MinConvLSTM 和 MinConvExpLSTM 使用 $c = 14, 12, 24, 20$ 和 20 个单位（通道）。

训练配置 我们在使用 Adam 训练所有模型时，采用批量大小为 1 和权重衰减 1×10^{-2}。使用余弦学习率调度器时，在 20 个纪元内将初始学习率 $\eta_0 = 5 \times 10^{-4}$ 衰减至零。我们从数据中随机裁剪长度 24 的序列，并在配备 6 GB 的 NVIDIA RTX 1060 GPU 上进行训练，使用 20 步教师强制，随后进行四步闭环。

结果 与之前的实验结果一致，我们发现 MinConvRNN 变体优于其传统对应模型，如 Figure 3所示，并在 Table 1中进行了量化。

4.3 运行时间分析

训练时间 我们研究的主要动机是减少卷积递归单元的运行时间限制。在 Table 2中，我们报告了每个模型完成一个周期所需的训练时间（秒），这是基于前五个周期的平均值。我们也通过 `torch.compile` 报告了启用和不启用即时编译 (JIT) 的不同结果，这使得模型运行得更快。ConvGRU 需要最长时间，因为需要调用两次卷积操作才能完成一次前向传播。所有其他模型只需要一次卷积操作。在 Navier-Stokes 实验中，我们观察到使用 JIT 和不使用 JIT 时加速因子分别为 3 和 4.7。另一方面，最小模型通过不使用 JIT

表 2. 每个模型训练单个周期在 Navier-Stokes 和重力势实验中的运行时间（以秒为单位）。我们使用 `torch.compile` 进行了带有即时编译 (JIT) 和不带即时编译的相同实验。结果是第 2-6 个周期的平均值。第一个周期被省略，因为 `torch.compile` 有较长的预热阶段。

	Geopotential		Navier-Stokes	
Model	With JIT	No JIT	With JIT	No JIT
ConvGRU	232.74 ± 0.88	774.77 ± 2.46	26.87 ± 0.074	87.05 ± 0.162
ConvLSTM	182.41 ± 0.91	513.82 ± 0.89	26.46 ± 0.034	60.50 ± 0.317
MinConvGRU	71.944 ± 0.99	154.97 ± 1.83	9.62 ± 0.111	20.33 ± 0.063
MinConvLSTM	74.846 ± 0.63	170.25 ± 0.69	8.222 ± 0.32	19.74 ± 0.065
MinConvExpLSTM	$\mathbf{69.79 \pm 0.855}$	$\mathbf{154.29 \pm 0.67}$	$\mathbf{7.956 \pm 0.040}$	$\mathbf{18.49 \pm 0.015}$

表 3. 处理形状为 $4 \times 100 \times 1 \times \mathrm{H} \times \mathrm{W}$ 的张量十次所需的运行时间（毫秒），要么按时间维度（100）顺序进行，要么在 T4 GPU 或 Colab 中的 v2-8 TPU 上并行进行。$\mathrm{H} = \mathrm{W}$ 在表格列中变化。

		4×4	32×32	128^2	256^2	512^2	1024^2
GPU	Sequential	75.56	79.34	76.01	81.20	156.95	545.70
	Parallel	0.54	0.53	7.05	27.71	112.36	465.45
	Speed-up	×139	×149	×10	×3	×1.4	×1.2
TPU	Sequential	0.109	0.243	0.257	0.206	0.185	0.427
	Parallel	0.001	0.001	0.002	0.002	0.002	0.025
	Speed-up	×109	×243	×129	×103	×93	×17

显著提高了学习速度，加速因子为 5。这些发现证实了我们的最小卷积递归神经网络模型在计算上节省了大量的资源，使其对于大规模时空学习极具吸引力。

卷积操作基准测试 在我们的实验中，我们观察到在 PyTorch 中使用 GPU 对大型张量进行并行处理的局限性。因此，我们通过将不同形状的张量传递给卷积操作来进行运行时分析，这些操作要么完全并行，要么按时间维度顺序执行，使用的设备是 Colab 中的 T4 GPU 和 v2-8 TPU。

结果，如 Table 3所示，揭示了两种模式。首先，GPU 上并行处理相对于顺序处理的速度提升随着张量大小的增加而减少，并且对于形状为 $1\,024 \times 1\,024$ 的张量几乎趋于平稳。其次，在 TPU 上，即使对于形状为 512×512 的大张

量，平行处理的速度优势仍然保持不变，但超过该尺寸后也会减少。当我们比较顺序 ConvRNNs 与我们的 MinConvRNNs 时，观察到了类似的硬件约束效果。

5 结论

处理和学习递归神经网络中的时空动态通常会由于其内在的顺序性质而导致运行时间较长。在这项工作中，我们通过采用在 MinRNNs[7] 中引入的日志域扫描兼容结构来重新定义已建立的 ConvGRU 和 ConvLSTM 架构，从而产生两种最小卷积递归神经网络：MinConvGRU 和 MinConvLSTM，这两种模型能够并行处理整个时空序列。此外，我们在 MinConvLSTM 中集成了指数门控机制以进一步简化日志域中的计算，生成了 MinConvExpLSTM。

在两个基准测试中，我们展示了 MinConvRNNs 在准确性方面具有竞争力的同时显著加速了训练。值得注意的是，我们的极简模型相比传统的 ConvRNN 模型实现了高达 5 倍的训练速度提升，而 MinConvExpLSTM 在流体动力学基准测试中达到了最低的预测误差。随后在 GPU 和 TPU 上的分析揭示了大规模张量并行处理相关的硬件限制。在大规模并行硬件上，当输入张量超过一定大小限制时，卷积操作可能会退回到不太高效的实现方式。

我们得出结论，MinConvRNNs 在时空动力学的完全并行化训练中具有巨大潜力，使递归架构在时空领域得以扩展。

未来的工作可以研究 MinConv 架构的多尺度或层次扩展，以更好地在多个分辨率下建模空间动态。在更高分辨率或更复杂的数据集（如降水短时预报或海洋动力学）上评估这些模型可能会揭示其可扩展性限制。此外，将最小循环结构与注意力机制相结合可能提供一种有前景的混合方法，在高效的扫描兼容动态内存和全局上下文建模之间取得平衡。

参考文献

1. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: ICLR (2016)
2. Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: xlstm: Extended long short-term memory. arXiv preprint arXiv:2405.04517 (2024)
3. Blelloch, G.E.: Prefix sums and their applications. In: Sythesis of parallel algorithms, pp. 35—60. Morgan Kaufmann Publishers Inc. (1990)

4. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: EMNLP. pp. 1724–1734 (2014)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning (2014)
6. Dao, T., Gu, A.: Transformers are ssms: Generalized models and efficient algorithms through structured state space duality (2024)
7. Feng, L., Tung, F., Ahmed, M.O., Bengio, Y., Hajimirsadeghi, H.: Were rnns all we needed? arXiv preprint arXiv:2410.01201 (2024)
8. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. In: NeurIPS. pp. 64–72 (2016)
9. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems **28**(10), 2222–2232 (2017)
10. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces (2024)
11. Gu, A., Dao, T., Ermon, S., Rudra, A., Ré, C.: Hippo: Recurrent memory with optimal polynomial projections. In: NeurIPS. vol. 33, pp. 1474–1487. Curran Associates, Inc. (2020)
12. Gu, A., Goel, K., Ré, C.: Efficiently modeling long sequences with structured state spaces. In: ICLR (2022)
13. Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., Ré, C.: Combining recurrent, convolutional, and continuous-time models with linear state-space layers. In: NeurIPS. Curran Associates Inc. (2021)
14. Heinsen, F.A.: Efficient parallelization of a ubiquitous sequential computation (2023)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Karlbauer, M., Maddix, D.C., Ansari, A.F., Han, B., Gupta, G., Wang, Y., Stuart, A., Mahoney, M.W.: Comparing and contrasting deep learning weather prediction backbones on navier-stokes and atmospheric dynamics. arXiv preprint arXiv:2407.14129 (2024)
17. Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., Butz, M.V.: A distributed neural network architecture for robust non-linear spatio-temporal prediction. arXiv preprint arXiv:1912.11141 (2019)

18. Martin, E., Cundy, C.: Parallelizing linear recurrent neural nets over sequence length. In: ICLR (2018)

19. Merrill, W., Petty, J., Sabharwal, A.: The illusion of state in state-space models. In: ICML (2024)

20. Orvieto, A., Smith, S.L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., De, S.: Resurrecting recurrent neural networks for long sequences. In: ICML. Proceedings of Machine Learning Research, vol. 202, pp. 26670–26698. PMLR (23–29 Jul 2023)

21. Qin, Z., Yang, S., Zhong, Y.: Hierarchically gated recurrent neural network for sequence modeling. In: NeurIPS (2023)

22. Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russell, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., et al.: Weatherbench 2: A benchmark for the next generation of data-driven global weather models. Journal of Advances in Modeling Earth Systems **16**(6), e2023MS004019 (2024)

23. Ravuri, S., Mohamed, S., et al.: Skilful precipitation nowcasting using deep generative models of radar. In: Nature. vol. 597, pp. 672–677. Nature Publishing Group (2021)

24. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: NeurIPS. vol. 28 (2015)

25. Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c.: Deep learning for precipitation nowcasting: A benchmark and a new model. NeurIPS **30** (2017)

26. Smith, J.T., Warrington, A., Linderman, S.W.: Simplified state space layers for sequence modeling. In: ICLR (2023)

27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NeurIPS. vol. 30. Curran Associates, Inc. (2017)