Unraveling Recurrent Dynamics: How Neural Networks Model Sequential Data

Guodong Li

Musketeers Foundation Institue of Data Science
Department of Statistics & Actuarial Science, CDS
University of Hong Kong

HKU IDS Workshop: Exploring the Foundations: Fundamental AI and Theoretical Machine Learning

May 29, 2025

Contents

Motivation

Motivation

Recurrent models have been a milestone in **sequence modeling** tasks:

- language modeling (Mikolov et al., 2010),
- machine translation (Sutskever et al., 2014),
- action and speech recognition (Chan et al., 2016),
- time series forecasting (Flunkert et al., 2017),
- dynamical system reconstruction (Hess et al., 2023).

Vanilla recurrent neural network (RNN) forms the basis for many important variants:

- long-short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997),
- gated recurrent units (GRU) (Cho et al., 2014),
- many others (Chang et al., 2019; Qiao et al., 2019; Gu et al., 2020; Erichson et al., 2021; Qin et al., 2023).

Motivation

RNNs have some disadvantages over Transformers (Vaswani et al., 2017):

- RNN cannot be trained in a parallel way.
- ...

RNNs have some advantages over Transformers:

- **Data modeling**: incorporating recurrent dynamics enables more sample-efficient extraction of sequential information (Shaw et al., 2018)
- **Algorithmic**: RNNs enjoys linear scaling in time and memory costs with respect to sequence length, in contrast to the quadratic complexity of attention mechanism in transformers.

Therefore, RNN's recurrent structures have been leveraged to address limitations of Transformer architectures, expanding the landscape of recurrent models.

Motivation

Current literature to theoretically explain RNNs:

- expressive power (Khrulkov et al., 2018),
- memory capacity (Collins et al., 2017; Haviv et al., 2019),
- generalization ability (Tu et al., 2020),
- training dynamics (Alemohammad et al., 2021; Farrell et al., 2022; Cohen-Karlik et al., 2023).

Yet, a fundamental question remains unaddressed: *What elementary temporal patterns can these models capture at a granular level?*

Motivation

Our contribution:

- Provide a quantification for recurrent dynamics by block diagonalizing recurrent matrices and introducing a new concept of recurrence features;
- Utilizing the prevalence of low-order features, we propose a parallelized network compring small-sized units, each having as few as two hidden states;
- We show both theoretically and numerically that the parallelized network accelerates computation dramatically while achieving comparable performace.

Recurrent Dynamics and Its Quantification

Recurrent Dynamics and Its Quantification

Recurrent Matrix Shaping Recurrent Dynamics

- Consider a general RNN layer with inputs $\boldsymbol{x}_t \in \mathbb{R}^{d_{in}}$ and hidden states $\boldsymbol{h}_t \in \mathbb{R}^d$ where $1 \leq t \leq T$. It has the form of

$$\boldsymbol{h}_t = \sigma_h(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}), \tag{1}$$

  where $\sigma_h(\cdot)$ is an element-wise activation function, $\boldsymbol{W}_h \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_x \in \mathbb{R}^{d \times d_{in}}$ are weight matrices, and $\boldsymbol{b}$ is the bias term.
- Crucially, matrix $\boldsymbol{W}_h$ determines temporal recurrent dynamics because it controls how subsequent hidden states are shaped by previous ones. In light of its important role, we refer to it as the *recurrent matrix*.
- Under the special case $\sigma_h(\boldsymbol{x}) = \boldsymbol{x}$ and $\boldsymbol{b} = \boldsymbol{0}$ which results in a linear RNN without bias, the output at (1) can be clearly rewritten into an explicit form: $\boldsymbol{h}_t = \sum_{j=0}^{t-1} \boldsymbol{W}_h^j \boldsymbol{W}_x \boldsymbol{x}_{t-j}$.

Recurrent Dynamics and Its Quantification

Decouple Recurrent Dynamics of RNN: A Special Scenario

When recurrent matrix is block diagonal, i.e., $\boldsymbol{W}_h = \oplus_{k=1}^{K} \boldsymbol{W}_h^{(k)} \in \mathbb{R}^{d\times d}$, with $\oplus$ denoting the matrix direct sum, $\boldsymbol{W}_h^{(k)} \in \mathbb{R}^{d_k \times d_k}$ and $d = \sum_{k=1}^{K} d_k$, the RNN at (1) can be decomposed into a series of smaller RNNs $\{\boldsymbol{h}_t^{(k)}\}_{k=1}^{K}$ with

$$\boldsymbol{h}_t^{(k)} = \sigma_h(\boldsymbol{W}_h^{(k)} \boldsymbol{h}_{t-1}^{(k)} + \boldsymbol{W}_x^{(k)} \boldsymbol{x}_t + \boldsymbol{b}^{(k)}) \in \mathbb{R}^{d_k} \text{ and } \boldsymbol{h}_t = \text{Concat}[\boldsymbol{h}_t^{(1)}, ..., \boldsymbol{h}_t^{(K)}],$$

where Concat[·] denotes the concatenation, and $\boldsymbol{W}_x^{(k)}$ and $\boldsymbol{b}^{(k)}$ are obtained by partitioning $\boldsymbol{W}_x$ and $\boldsymbol{b}$ along the rows, i.e., $\boldsymbol{W}_x^{(k)} = (\boldsymbol{W}_x)_{a_{k-1}+1:a_k}$, $\boldsymbol{b}^{(k)} = \boldsymbol{b}_{a_{k-1}+1:a_k}$, $a_0 = 0$, and $a_k = \sum_{i=1}^{k} d_k$ for $k > 0$.

⋆ *Block diagonality of the recurrent matrix offers a way to separate the recurrent dynamics.*

Recurrent Dynamics and Its Quantification

Decouple Recurrent Dynamics of RNN: General Cases

Proposition 1 (Real Jordan Decomposition (Horn and Johnson, 2012)).

Suppose a matrix $\boldsymbol{W}_h \in \mathbb{R}^{d\times d}$ has r distinct nonzero real eigenvalues $\{\lambda_j\}_{j=1}^r$ and s distinct conjugate pairs of nonzero complex eigenvalues $\{(\lambda_{r+2k-1}, \lambda_{r+2k}) = (\gamma_k e^{i\theta_k}, \gamma_k e^{-i\theta_k})\}_{k=1}^s$ with $\lambda_j \in \mathbb{R}$, $\gamma_k > 0$ and $\theta_k \in (-\pi/2, \pi/2)$. Assume the geometric multiplicities of all nonzero eigenvalues to be one, then it has a real Jordan canonical form $\boldsymbol{W}_h = \boldsymbol{B}\boldsymbol{J}\boldsymbol{B}^{-1}$ where $\boldsymbol{B} \in \mathbb{R}^{d\times d}$ is an invertible matrix and $\boldsymbol{J}$ is a real block diagonal matrix formed by a direct sum of real Jordan blocks

$$\boldsymbol{J} = \boldsymbol{J}_{n_1}(\lambda_1) \oplus \cdots \oplus \boldsymbol{J}_{n_r}(\lambda_r) \oplus \boldsymbol{C}_{n_{r+1}}(\gamma_1, \theta_1) \oplus \cdots \oplus \boldsymbol{C}_{n_{r+s}}(\gamma_s, \theta_s) \oplus \mathbf{0}_{d-r-2s}.$$

The subscript $n_k \geq 1$ is the corresponding algebraic multiplicity of λ_k or $(\gamma_{k-r}e^{i\theta_{k-r}}, \gamma_{k-r}e^{-i\theta_{k-r}})$ and $\mathbf{0}_{d-r-2s} \in \mathbb{R}^{(d-r-2s)\times(d-r-2s)}$ is a zero matrix.

Recurrent Dynamics and Its Quantification

Decouple Recurrent Dynamics of RNN: General Cases

The two types of real Jordan blocks have the forms of:

$$\boldsymbol{J}_n(\lambda) = \begin{pmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \lambda & 1 \\ & & & \lambda \end{pmatrix} \in \mathbb{R}^{n\times n}$$

$$\text{and } \boldsymbol{C}_n(\gamma,\theta) = \begin{pmatrix} \boldsymbol{C}(\gamma,\theta) & \boldsymbol{I}_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \boldsymbol{I}_2 \\ & & & \boldsymbol{C}(\gamma,\theta) \end{pmatrix} \in \mathbb{R}^{2n\times 2n},$$

where $\boldsymbol{C}(\gamma,\theta) = \gamma \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{I}_2 \in \mathbb{R}^{2\times 2}$ is identity matrix.

Recurrent Dynamics and Its Quantification

Decouple Recurrent Dynamics of RNN: General Cases

- Putting this decomposition back into (1), we obtain

$$\begin{aligned}\boldsymbol{h}_t &= \sigma_h[\boldsymbol{B}(\boldsymbol{J}\boldsymbol{B}^{-1}\boldsymbol{h}_{t-1} + \boldsymbol{B}^{-1}\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{B}^{-1}\boldsymbol{b})] \\ &\approx \boldsymbol{B}\sigma_h(\boldsymbol{J}\boldsymbol{B}^{-1}\boldsymbol{h}_{t-1} + \boldsymbol{B}^{-1}\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{B}^{-1}\boldsymbol{b}),\end{aligned}$$

  where the approximation of moving $\boldsymbol{B}$ outside the activation function becomes equivalence if $\sigma_h(\boldsymbol{x}) = \boldsymbol{x}$.
- As a result, it leads to an interpretable RNN surrogate:

$$\tilde{\boldsymbol{h}}_t = \sigma_h(\boldsymbol{J}\tilde{\boldsymbol{h}}_{t-1} + \tilde{\boldsymbol{W}}_x\boldsymbol{x}_t + \tilde{\boldsymbol{b}}) \quad \text{and} \quad \boldsymbol{h}_t = \boldsymbol{B}\tilde{\boldsymbol{h}}_t,$$

  where $\tilde{\boldsymbol{W}}_x = \boldsymbol{B}^{-1}\boldsymbol{W}_x$ and $\tilde{\boldsymbol{b}} = \boldsymbol{B}^{-1}\boldsymbol{b}$. Note that its recurrent matrix has the desirable block diagonal form, allowing for a decomposition similar to the special case.

Recurrent Dynamics and Its Quantification

Recurrence Features

- The RNN surrogate has two types of constituent RNNs,

$$\boldsymbol{h}_t = \sigma_h\big(\boldsymbol{J}_n(\lambda)\boldsymbol{h}_{t-1} + \boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{b}\big) \in \mathbb{R}^n$$
$$\text{and} \quad \boldsymbol{h}_t = \sigma_h\big(\boldsymbol{C}_n(\gamma,\theta)\boldsymbol{h}_{t-1} + \boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{b}\big) \in \mathbb{R}^{2n}.$$

  The recurrent matrices $\boldsymbol{J}_n(\lambda)$ and $\boldsymbol{C}_n(\gamma,\theta)$ cannot be further decomposed by real Jordan decomposition.
- Recognizing the two RNNs are the **fundamental units** to constitute the recurrent dynamics of a general RNN, we formally define their dynamics as the *recurrence feature*.
- The corresponding dynamics are categorized as Types R-n and C-n features, respectively.

Recurrent Dynamics and Its Quantification

Recurrence Features

Intriguingly, these two types of features align with two well-established temporal patterns of ARMA models in time series literature (Cryer and Chan, 2008).

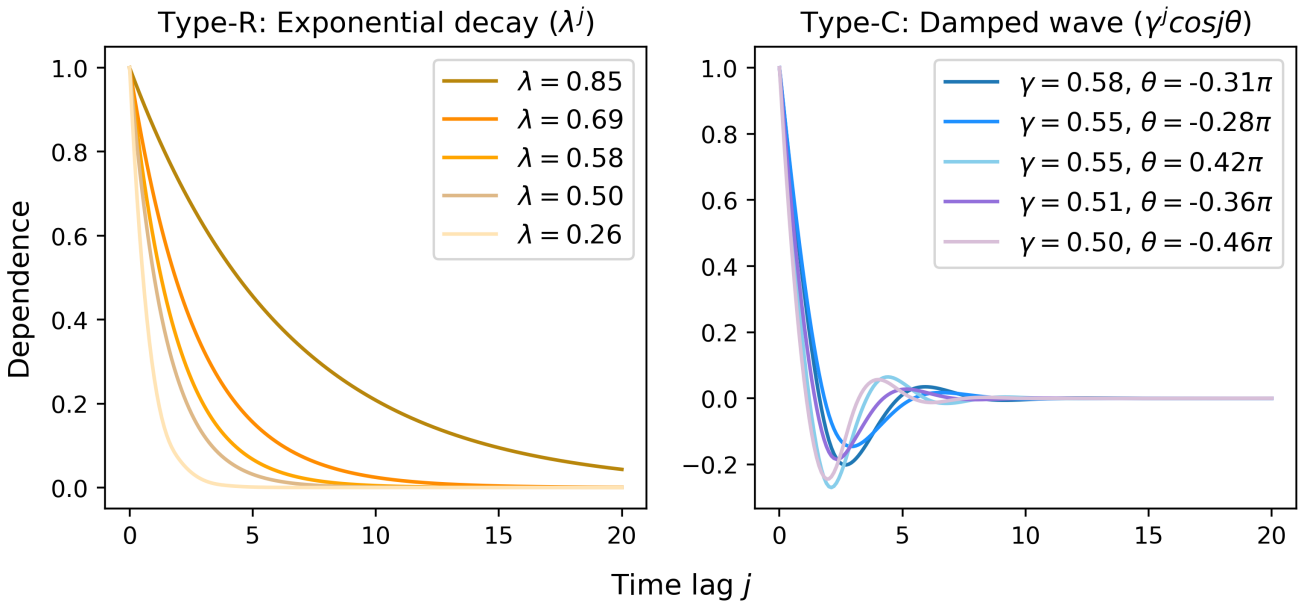
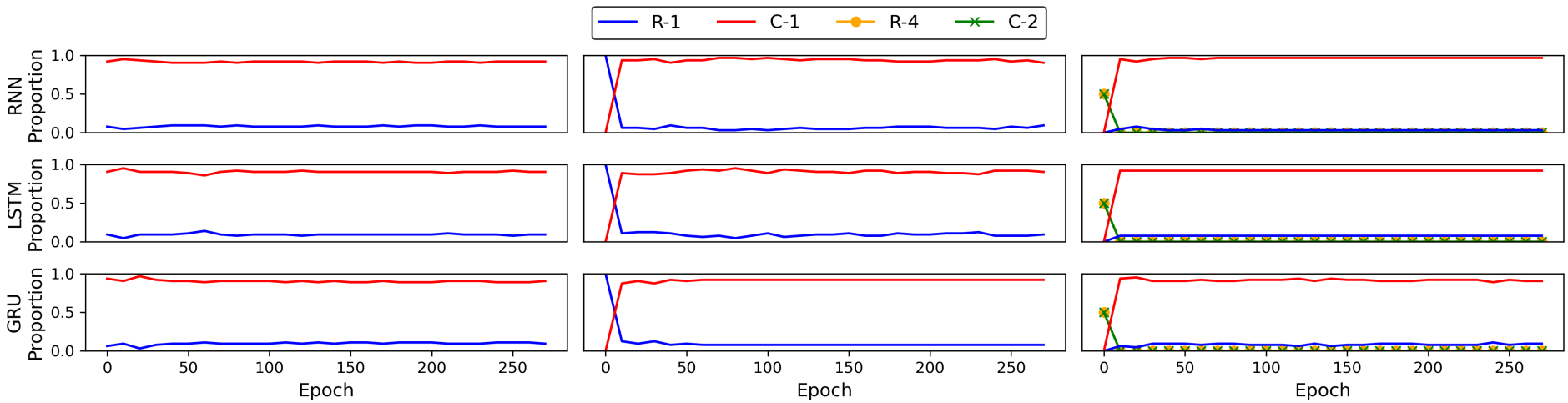


Figure: Two patterns of recurrence features that the vanilla RNN learns from real data: exponential decay (Type R) and damped wave decay (Type C).

Recurrent Dynamics and Its Quantification

Some Dominating Features

Empirical evidence show that recurrent models after gradient updates tend to learn low-order (R-1 and C-1) features, and a majority of them are complex types.



(a) R-1 and C-1 initialization (b) R-1 initialization (c) R-4 and C-2 initialization

Figure: Evolution of recurrence feature types across training for vanilla RNN, LSTM, and GRU models from the permuted sequential MNIST task. Types other than R-1, C-1, R-4, or C-2 have zero occurrence.

Recurrent Dynamics and Its Quantification

Some Dominating Features

Theorem 1.

Let $\mathbb{M}_d(\mathbb{R})$ be the set of all $d \times d$ real matrices with rank at most R, where $1 < R \leq d$. Define $\mathbb{M}_d^1(\mathbb{R})$ and $\mathbb{M}_d^2(\mathbb{R})$ as the sets of matrices in $\mathbb{M}_d(\mathbb{R})$ whose nonzero real Jordan blocks are only of the form $\boldsymbol{J}_1(\lambda)$ or exclusively of the forms $\boldsymbol{J}_1(\lambda)$, $\boldsymbol{C}_1(\gamma, \theta)$, or $\boldsymbol{J}_2(\lambda)$. Then $\mathbb{M}_d^2(\mathbb{R})$ is dense in $\mathbb{M}_d(\mathbb{R})$, while $\mathbb{M}_d^1(\mathbb{R})$ is not dense in $\mathbb{M}_d(\mathbb{R})$.

- Give a partial account of why RNNs usually concentrate on low-order features from the perspective of approximation capability.
- Underscore the limitations of a diagonal recurrent matrix belonging to $\mathbb{M}_d^1(\mathbb{R})$, a design considered by Li et al. (2018); Martin and Cundy (2018); Rusch and Mishra (2021)

Recurrent Dynamics and Its Quantification

Some Dominating Features

From a probabilistic view:

Theorem 2.

Consider a general RNN with recurrent matrix $\boldsymbol{W}_h = (\xi_{ij})_{1\le i,j\le d} \in \mathbb{R}^{d\times d}$, and the entries $\{\xi_{ij}\}$ have a continuous joint distribution. Then with probability one, the RNN contains recurrence features with Types R-1, R-2 and C-1 only. Moreover, if $\{\xi_{ij}\}$ are independent standard normal random variables, then with probability at most $1/2^{d(d-1)/4}$, the RNN contains recurrence features with Type R-1 only.

These simplify our understanding of recurrent dynamics and will be leveraged to guide a novel network design.

Parallelized Recurrent Neural Network

Parallelized Recurrent Neural Network

A New Network

We propose to approximate the RNN at (1) by parallelizing K small RNNs of equal hidden size d_s to maximize the computational efficiency:

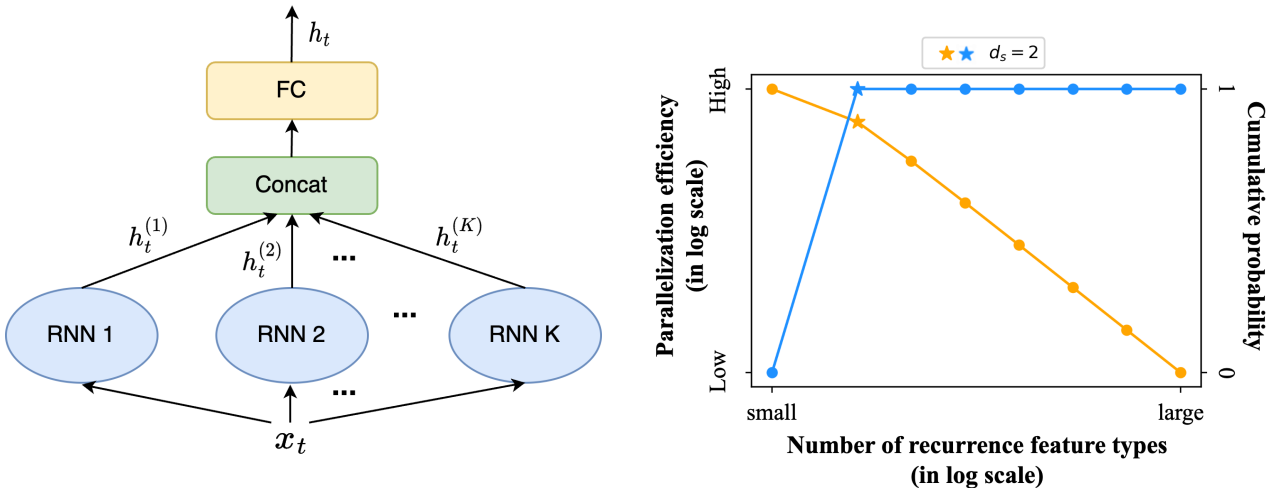
$$\boldsymbol{h}_t^{(k)} = \text{Recurrent-cell}(\boldsymbol{h}_{t-1}^{(k)}, \boldsymbol{x}_t) \in \mathbb{R}^{d_s} \text{ and } \boldsymbol{h}_t = \text{FC}(\text{Concat}[\boldsymbol{h}_t^{(1)}, ..., \boldsymbol{h}_t^{(K)}]).$$

Here d_s is hyperparameter, and $d = Kd_s$ is divisible by d_s without loss of generality. Especially, its recurrent matrix $\boldsymbol{W}_h^{(k)} \in \mathbb{R}^{d_s \times d_s}$ is freely learnable rather than being parametrized by a specific form of real Jordan block.

Parallelized Recurrent Neural Network

A New Network

We recommend setting $d_s = 2$ first because it covers the most probable recurrence feature types of a vanilla RNN while maintaining high computational efficiency.



(a) (b)

Figure: (a) A ParaRNN layer at time t. (b) Number of recurrence feature types ($d = 128$): its cumulative distribution from Theorem 2 (in blue) and the trade-off with parallelization efficiency (in orange).

Parallelized Recurrent Neural Network

ParaRNN Function Class

Input: Sequence $\boldsymbol{X}_t = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) \in \mathbb{R}^{d_{\text{in}} \times t}$ with $1 \leq t \leq T$.

Network $\mathcal{N}$ consists of:

(i) Linear input layer $\mathcal{P} : \mathbb{R}^{d_{\text{in}} \times t} \to \mathbb{R}^{d \times t}$, with $\mathcal{P}(\boldsymbol{X}_t) = (\boldsymbol{P}\boldsymbol{x}_1, \ldots, \boldsymbol{P}\boldsymbol{x}_t)$, $\boldsymbol{P} \in \mathbb{R}^{d \times d_{\text{in}}}$;

(ii) L recurrent layers $\mathcal{R}_1, \ldots, \mathcal{R}_L : \mathbb{R}^{d \times t} \to \mathbb{R}^{d \times t}$ (ReLU, block size d_s);

(iii) Position-wise FC layer $\mathcal{F} : \mathbb{R}^{d \times t} \to \mathbb{R}^{d \times t}$ with ReLU;

(iv) Linear output layer $\mathcal{Q} : \mathbb{R}^{d \times t} \to \mathbb{R}^{d_{\text{out}} \times t}$ defined analogously to $\mathcal{P}$.

ParaRNN Function Class:

$$\begin{aligned}\mathcal{F}^{(t)}_{d_{\text{in}}, d_{\text{out}}, d, d_s, L, U} = \Big\{ & \mathcal{N}(\boldsymbol{X}_t)[t] \,:\, \mathbb{R}^{d_{\text{in}} \times t} \to \mathbb{R}^{d_{\text{out}}}, \\ & \mathcal{N}(\boldsymbol{X}_t) = \mathcal{Q} \circ \mathcal{F} \circ \mathcal{R}_L \circ \cdots \circ \mathcal{R}_1 \circ \mathcal{P}(\boldsymbol{X}_t) \in \mathbb{R}^{d_{\text{out}} \times t}, \\ & \text{with} \sup_{\boldsymbol{X}_t, t_0 \in \{1, \ldots, t\}} \|\mathcal{N}(\boldsymbol{X}_t)[t_0]\|_\infty \leq U \Big\}\end{aligned}$$

Parallelized Recurrent Neural Network

Definition 1 (Hölder class).

Let $\Omega \subset \mathbb{R}^{d_{\text{in}}}$ and $\beta > 0$ with $\beta = k + \omega$, where $k \in \mathbb{N}_0$ and $\omega \in (0, 1]$. A function is said to be β-smooth if all its partial derivatives up to order k exist and are bounded, and the partial derivatives of order k are ω-Hölder continuous. For $d_{\text{in}}, d_{\text{out}} \in \mathbb{N}$, the Hölder class with smoothness index β is defined as

$$\mathcal{H}^{\beta}_{d_{\text{in}},d_{\text{out}}}(\Omega, M) = \Big\{ f = (f_1, \ldots, f_{d_{\text{out}}})^\top : \Omega \mapsto \mathbb{R}^{d_{\text{out}}},$$
$$\sum_{\boldsymbol{n}:\|\boldsymbol{n}\|_1<\beta} \|\partial^{\boldsymbol{n}} f_i\|_{L^\infty(\Omega)} + \sum_{\boldsymbol{n}:\|\boldsymbol{n}\|_1=k} \sup_{\boldsymbol{x},\boldsymbol{y}\in\Omega,\boldsymbol{x}\neq\boldsymbol{y}} \frac{|\partial^{\boldsymbol{n}} f_i(\boldsymbol{x}) - \partial^{\boldsymbol{n}} f_i(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|^{\omega}} \leq M,$$
$$i = 1, \ldots, d_{\text{out}} \Big\},$$

where $\partial^n = \partial^{n_1} \ldots \partial^{n_{d_{\text{in}}}}$ with $\boldsymbol{n} = (n_1, \ldots, n_{d_{\text{in}}}) \in \mathbb{N}_0^{d_{\text{in}}}$ and $\|\boldsymbol{n}\|_1 = \sum_{i=1}^{d_{\text{in}}} n_i$.

Parallelized Recurrent Neural Network

Proposition 2 (Approximation Error Bound).

Let $t_0 \in \{1, \ldots, T\}$. Assume that $f^{(t_0)} \in \mathcal{H}^{\beta}_{d_{\text{in}} \times t_0, d_{\text{out}}}([0,1]^{d_{\text{in}} \times t_0}, U)$. Then for any $I, J \in \mathbb{N}^+$, there exists a ParaRNN-based function $\phi \in \mathcal{F}^{(t_0)}_{d_{\text{in}}, d_{\text{out}}, d, d_s, L, U}$ such that

$$\sup_{\boldsymbol{X} \in [0,1]^{d_{\text{in}} \times t_0}} \|\phi(\boldsymbol{X}) - f^{(t_0)}(\boldsymbol{X})\|_\infty \leq 19U(\lfloor\beta\rfloor + 1)^2 (d_{\text{in}} t_0)^{\lfloor\beta\rfloor + (\beta \vee 1)/2} (JI)^{-2\beta/(d_{\text{in}} t_0)},$$

Network Depth: $L = 42(\lfloor\beta\rfloor + 1)^2 I \lceil \log_2(8I) \rceil + 6 d_{\text{in}} T$,

Network Width: $d = 76(\lfloor\beta\rfloor + 1)^2 3^{d_{\text{in}} T} d_{\text{in}}^{\lfloor\beta\rfloor + 2} d_{\text{out}} T^{\lfloor\beta\rfloor + 1} J \lceil \log_2(8J) \rceil + d_s$.

(Assume d divisible by d_s without loss of generality.)

Parallelized Recurrent Neural Network

ParaRNN for Nonparametric Regression

Problem Setup:

- Sequential data: $(\boldsymbol{x}_t, z_t)_{t=1}^T$, where $\boldsymbol{x}_t \in [0,1]^{d_{\text{in}}}$, $z_t \in \mathbb{R}$
- Goal: estimate $f_0^{(t)}$ with $f_0^{(t)}(\boldsymbol{X}_t) = \mathbb{E}[z_t \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t]$

Risk Definitions: (for any $\phi : \mathbb{R}^{d_{\text{in}} \times t} \to \mathbb{R}$)

- Population risk: $\mathcal{R}(\phi) = \mathbb{E}[(\phi(\boldsymbol{X}_t) - z_t)^2]$
- Empirical risk: $\mathcal{R}_N(\phi) = \frac{1}{N}\sum_{i=1}^N [\phi(\boldsymbol{X}_{i,t}) - z_{i,t}]^2$

Estimators:

- Empirical risk minimizer (ERM): $\widehat{f}^{(t)} = \arg\min_{\phi \in \mathcal{F}^{(t)}} \mathcal{R}_N(\phi)$
- Population risk minimizer: $\bar{f}^{(t)} = \arg\min_{\phi \in \mathcal{F}^{(t)}} \mathcal{R}(\phi)$

Excess Risk Decomposition:

$$\mathcal{R}(\widehat{f}^{(t)}) - \mathcal{R}(f_0^{(t)}) = \underbrace{\mathcal{R}(\widehat{f}^{(t)}) - \mathcal{R}(\bar{f}^{(t)})}_{\text{Estimation error}} + \underbrace{\mathcal{R}(\bar{f}^{(t)}) - \mathcal{R}(f_0^{(t)})}_{\text{Approximation error}}$$

Parallelized Recurrent Neural Network

Prediction Error Bound

Corollary 1.

Let $t_0 \in \{1, \ldots, T\}$. Suppose that $f_0^{(t_0)} \in \mathcal{H}^{\beta}_{d_{\text{in}} \times t_0, 1}\left([0,1]^{d_{\text{in}} \times t_0}, U\right)$ for some $U \geq 1$, and the function class $\mathcal{F}^{(t_0)}_{d_{\text{in}}, 1, d, d_s, L, U}$ has

- width $d \asymp N^{\eta} \log N$, and
- depth $L \asymp N^{\frac{d_{\text{in}} t_0}{2 d_{\text{in}} t_0 + 4\beta} - \eta} \log N$

for fixed $\eta \in [0, d_{\text{in}} t_0 / (2 d_{\text{in}} t_0 + 4\beta)]$. If $N \gtrsim d^2 L^2 \log \max\{d, L\}$, then the ERM $\widehat{f}^{(t_0)}$ satisfies

$$\mathbb{E}_{\mathcal{S}}\left[\mathcal{R}(\widehat{f}^{(t_0)}) - \mathcal{R}(f_0^{(t_0)})\right] \lesssim N^{-\frac{2\beta}{d_{\text{in}} t_0 + 2\beta}} (\log N)^{10}.$$

Note: the ERM $\widehat{f}^{(t_0)}$ based on deep and wide ParaRNNs achieves the optimal minimax rate $N^{-2\beta/(d_{\text{in}} t_0 + 2\beta)}$ established by Stone (1982) for nonparametric regression, up to a logarithmic factor.

Extension to LSTM Networks

Our theoretical framework extends naturally to gated recurrent architectures like LSTM, by identifying the recurrent matrices operating on previous hidden states.

LSTM cell computations:

$$\begin{bmatrix} \boldsymbol{f}_t \\ \boldsymbol{i}_t \\ \boldsymbol{o}_t \end{bmatrix} = \text{Sigmoid}\left(\begin{bmatrix} \boldsymbol{W}_f \\ \boldsymbol{W}_i \\ \boldsymbol{W}_o \end{bmatrix} \boldsymbol{h}_{t-1} + \begin{bmatrix} \boldsymbol{U}_f \\ \boldsymbol{U}_i \\ \boldsymbol{U}_o \end{bmatrix} \boldsymbol{x}_t + \begin{bmatrix} \boldsymbol{b}_f \\ \boldsymbol{b}_i \\ \boldsymbol{b}_o \end{bmatrix} \right)$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c \boldsymbol{h}_{t-1} + \boldsymbol{U}_c \boldsymbol{x}_t + \boldsymbol{b}_c)$$

Insight: The recurrent dynamics are jointly determined by four weight matrices, $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_o$ and $\boldsymbol{W}_c$, as they shape how the previous hidden state $\boldsymbol{h}_{t-1}$ is incorporated. Thus these matrices can be regarded as **recurrent matrices** as well.

Extension to Attention-Based Recurrent Cells

The **attention-based recurrent cell** serves as another example where the weights on the old memory play a comparable role as $\boldsymbol{W}_h$ in vanilla RNNs.

The recurrent cell first generates the query $\boldsymbol{Q}_t \in \mathbb{R}^{d\times 1}$ from $\boldsymbol{h}_{t-1}$. It also extracts from both $\boldsymbol{h}_{t-1}$ and $\boldsymbol{X}_t$ to get the key $\boldsymbol{K}_t = \begin{bmatrix}\boldsymbol{K}_{t,h} & \boldsymbol{K}_{t,x}\end{bmatrix} \in \mathbb{R}^{d\times(1+N)}$ and similarly for the value $\boldsymbol{V}_t = \begin{bmatrix}\boldsymbol{V}_{t,h} & \boldsymbol{V}_{t,x}\end{bmatrix} \in \mathbb{R}^{d\times(1+N)}$, where

$$\begin{bmatrix}\boldsymbol{Q}_t \\ \boldsymbol{K}_{t,h} \\ \boldsymbol{V}_{t,h}\end{bmatrix} = \begin{bmatrix}\boldsymbol{W}_Q \\ \boldsymbol{W}_K \\ \boldsymbol{W}_V\end{bmatrix} \boldsymbol{h}_{t-1}, \quad \text{and} \quad \begin{bmatrix}\boldsymbol{K}_{t,x} \\ \boldsymbol{V}_{t,x}\end{bmatrix} = \begin{bmatrix}\boldsymbol{W}_K \\ \boldsymbol{W}_V\end{bmatrix} \boldsymbol{X}_t.$$

Memory state update:

$$\boldsymbol{h}_t^\top = \text{softmax}\left(\frac{\boldsymbol{Q}_t^\top \boldsymbol{K}_t}{\sqrt{d}}\right)\boldsymbol{V}_t^\top$$

Insight: Recurrent dynamics are shaped by $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$, as they control how $\boldsymbol{h}_{t-1}$ influences the memory update. Hence we also consider them as the recurrent matrices.

Experiments

Simulation: Verifying the Trade-off in ParaRNN

Setup: We generate $N = 50,000$ samples by a vanilla RNN of size $d = 128$, with i-th sample represented as $\left(x_{i,1:T}\,, \boldsymbol{z}_i\right)$ and following the form:

$$\begin{aligned}
\boldsymbol{z}_i &= \boldsymbol{W}_y \boldsymbol{h}_{i,T} + \boldsymbol{b}_y + \boldsymbol{\epsilon}_i \in \mathbb{R}^{10} \\
\boldsymbol{h}_{i,t} &= \mathrm{Tanh}(\boldsymbol{W}_h \boldsymbol{h}_{i,t-1} + \boldsymbol{W}_x x_{i,t} + \boldsymbol{b}_h) \in \mathbb{R}^{128} \\
\boldsymbol{\epsilon}_i &\overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \text{ each entry of } \boldsymbol{W}\text{'s and } \boldsymbol{b}\text{'s } \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)
\end{aligned}$$

for $1 \le t \le T, 1 \le i \le N$. In particular, the inputs $x_{i,1:T}$ of length $T = 128$ are generated from the ARMA$(1,1)$ process with the AR and MA coefficients being 0.7 and 0.3, respectively.

Experiment:

- Train a ParaRNN with a fully connected output layer.
- Vary block size $d_s \in \{1, 2, 4, \ldots, 128\}$.

Simulation Results

ParaRNN with $d_s = 2$ effectively balances performance and speed, recovering recurrent dynamics of the vanilla model.

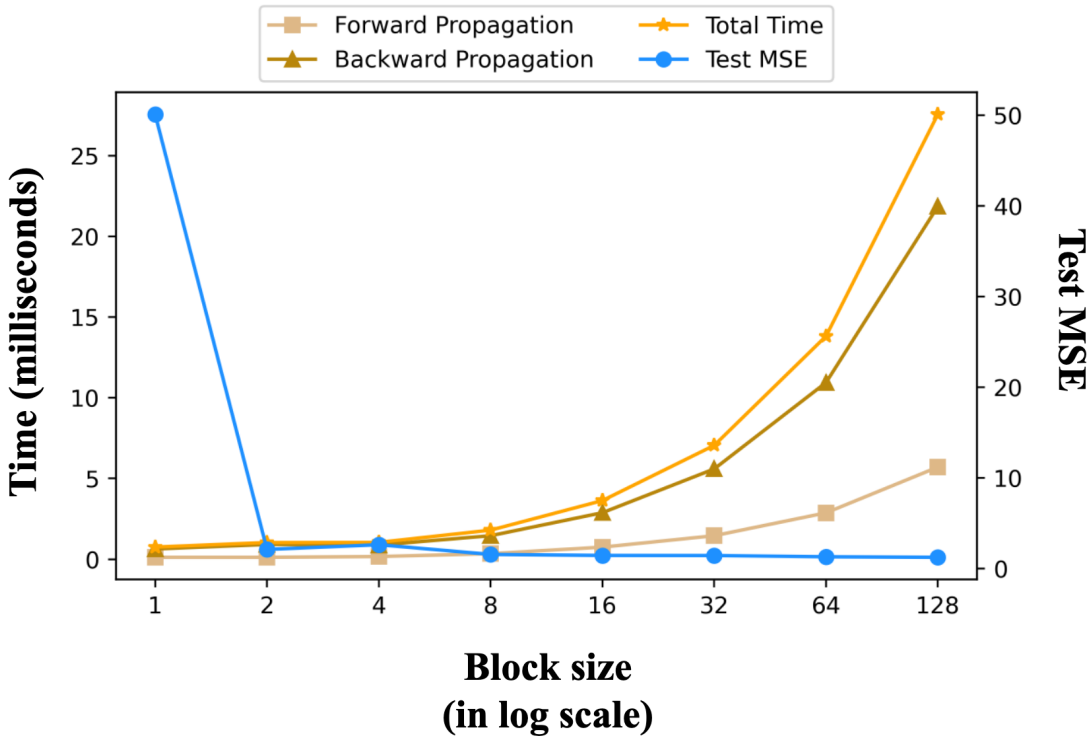


Figure: Test MSE averaged over 25 replicates of the simulated data (in blue), and average execution time over 100 replications of forward and backward propagation as well as their sum for a ParaRNN layer on a single V100 GPU

Real-World Datasets

1. **Time Series Forecasting** (Zhou et al., 2021)
   - ETT (ETTh_1, ETTh_2, ETTm_1): 7 power-related features, 2 years of data.
   - WTH: 12 weather features from 1,600 U.S. locations (2010–2013).
   - Goal: Predict "oil temperature" for ETT, "Wet Bulb" temperature for WTH.
2. **Sequential Image Classification**
   - Permuted MNIST (Lecun et al., 1998): Digit images flattened ($T = 784$, $d_{\text{in}} = 1$), randomly permuted.
   - Pixel-by-Pixel CIFAR-10 (Krizhevsky et al., 2009): Flattened RGB images ($T = 1024$, $d_{\text{in}} = 3$).
   - Noise-Padded CIFAR-10: Row-wise flattened RGB with noise padding ($T = 1000$, $d_{\text{in}} = 96$).

Experimental Results Across Tasks

Across all datasets, ParaRNN-based models demonstrate competitive or improved performance while preserving training efficiency.

RNN	ParaRNN	LSTM	ParaLSTM	GRU	ParaGRU	BRT	ParaBRT
(a) Time Series Forecasting – Total Wins							
7	**27**	7	**26**	8	**26**	15	**21**
(b) Permuted MNIST – Accuracy (%)							
90.31	**93.93**	**94.20**	94.10	94.44	**94.67**	97.87	**97.90**
(c) Pixel-by-Pixel CIFAR-10 – Accuracy (%)							
31.80	**36.37**	66.35	**66.40**	**70.61**	70.06	74.01	**74.06**
(d) Noise-Padded CIFAR-10 – Accuracy (%)							
–	–	56.71	**57.07**	52.90	**53.40**	68.85	**68.91**

Table: Performance comparison between **Vanilla** and **Para** variants for RNN, LSTM, GRU, and BRT across forecasting and classification tasks. Bold indicates best performance.

Conclusion

Theoretical Contributions

- Introduce a principled decomposition of recurrent matrices to decouple recurrent dynamics.
- Propose the concept of **recurrence features**, depicting the elementary temporal patterns that networks capture at the granular level.
- Provide an essential initial stride toward interpretable model behaviors, paving the way for enhancing the design of complex sequential data modeling.

Engineering Contributions: ParaRNN Framework

- Propose ParaRNN, a novel framework that serves as a computationally efficient alternative to many recurrent models.
- Provide a guideline for hyperparameter selection to balance the trade-off between recurrence feature richness and computational efficiency.
- Extensible to various RNN variants and modern architectures (e.g., Transformer-based models).

Thank you!

References I

Alemohammad, S., Wang, Z., Balestriero, R., and Baraniuk, R. (2021). The recurrent neural tangent kernel. In *International Conference on Learning Representations*.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 4960–4964. IEEE.

Chang, B., Chen, M., Haber, E., and Chi, E. H. (2019). AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Cohen-Karlik, E., Menuhin-Gruman, I., Giryes, R., Cohen, N., and Globerson, A. (2023). Learning low dimensional state spaces with overparameterized recurrent neural nets. In *The Eleventh International Conference on Learning Representations*.

Collins, J., Sohl-Dickstein, J., and Sussillo, D. (2017). Capacity and trainability in recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Cryer, J. and Chan, K. (2008). *Time Series Analysis: With Applications in R*. Springer Texts in Statistics. Springer New York.

Erichson, N. B., Azencot, O., Queiruga, A., Hodgkinson, L., and Mahoney, M. W. (2021). Lipschitz recurrent neural networks. In *International Conference on Learning Representations*.

References II

Farrell, M., Recanatesi, S., Moore, T., Lajoie, G., and Shea-Brown, E. (2022). Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion. *Nature Machine Intelligence*, 4(6):564–573.

Flunkert, V., Salinas, D., and Gasthaus, J. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110.

Gu, A., Gulcehre, C., Paine, T., Hoffman, M., and Pascanu, R. (2020). Improving the gating mechanism of recurrent neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Haviv, D., Rivkind, A., and Barak, O. (2019). Understanding and controlling memory in recurrent neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2663–2671. PMLR.

Hess, F., Monfared, Z., Brenner, M., and Durstewitz, D. (2023). Generalized teacher forcing for learning chaotic dynamics. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 13017–13049. PMLR.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Horn, R. A. and Johnson, C. R. (2012). *Matrix Analysis*. Cambridge University Press, USA, 2nd edition.

Khrulkov, V., Novikov, A., and Oseledets, I. (2018). Expressive power of recurrent neural networks. In *International Conference on Learning Representations*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.(2009).

References III

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. (2018). Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5457–5466. IEEE.

Martin, E. and Cundy, C. (2018). Parallelizing linear recurrent neural nets over sequence length. *arXiv.org*.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA.

Qiao, S., Wang, H., Liu, C., Shen, C., and Yuille, A. (2019). Stabilizing gradients for deep neural networks via efficient svd parametrization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11494–11503. IEEE.

Qin, Z., Yang, S., and Zhong, Y. (2023). Hierarchically gated recurrent neural network for sequence modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rusch, T. K. and Mishra, S. (2021). Unicornn: A recurrent model for learning very long time dependencies. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9168–9178. PMLR.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

References IV

Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10(4):1040–1053.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Tu, Z., He, F., and Tao, D. (2020). Understanding generalization in recurrent neural networks. In *International Conference on Learning Representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115.