Spectral Filtering for Deep Learning

Hazan Lab
Princeton University

January 3rd, 2025

Motivation

Motivation
Spectral filtering
STU
Experiments
Current work

Architectures

RNNs

Transformers

Spectral Filtering

Recurrent Neural Networks

Motivation

Spectral filtering

STU

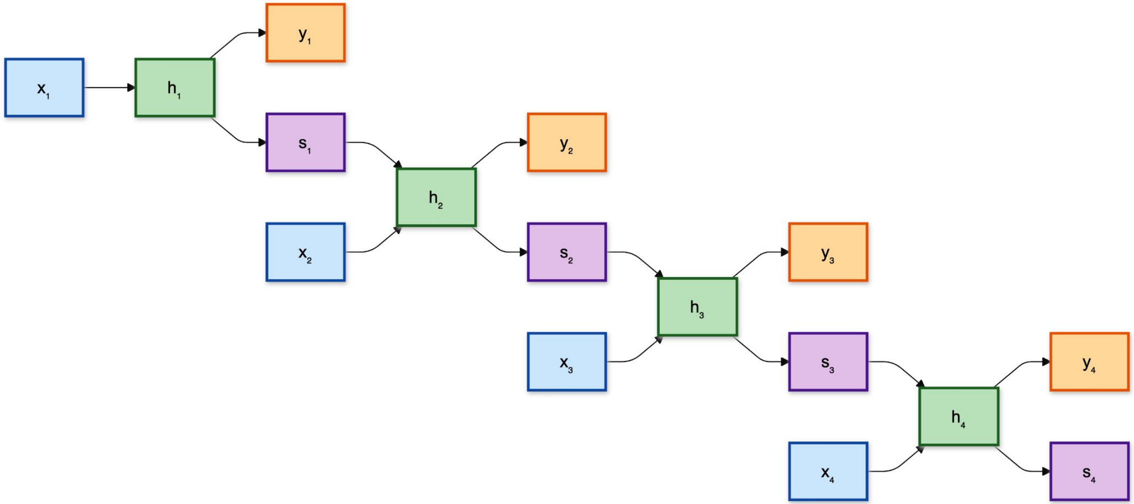
Experiments

Current work

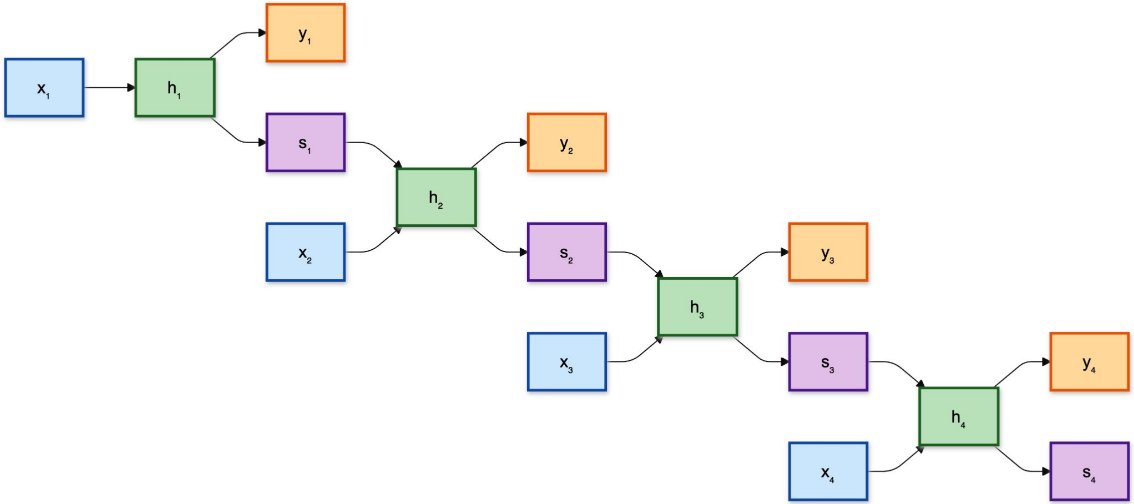
- O(L) time complexity

- O(L) time complexity
- But sequential

- O(L) time complexity
- But sequential
    - Need s_{t-1} to compute y_t



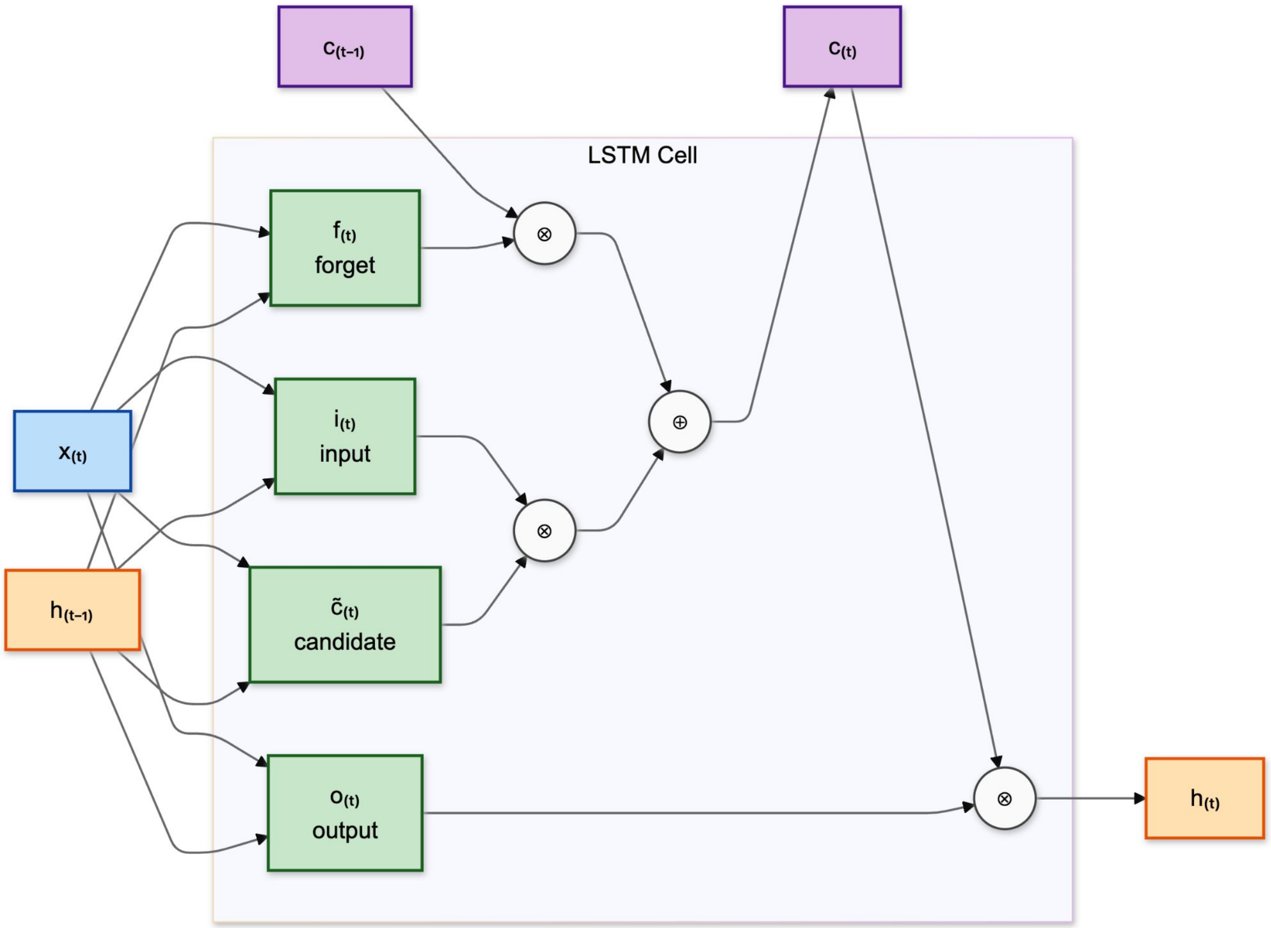
- O(L) time complexity
- But sequential
    - Need s_{t-1} to compute y_t
- Unstable (exploding/vanishing gradients)



- People tried other variants

- People tried other variants
  - LSTMs (long short-term memory)

- People tried other variants
    - LSTMs (long short-term memory)
    - Same idea, just more bells and whistles



c(t−1)
c(t)
LSTM Cell
f(t)
forget
⊗
i(t)
input
⊕
x(t)
⊗
h(t−1)
c̃(t)
candidate
o(t)
output
⊗
h(t)

RNNs are *slower* than a Transformer in practice.

Transformers

Motivation
Spectral filtering
STU
Experiments
Current work

Self-attention

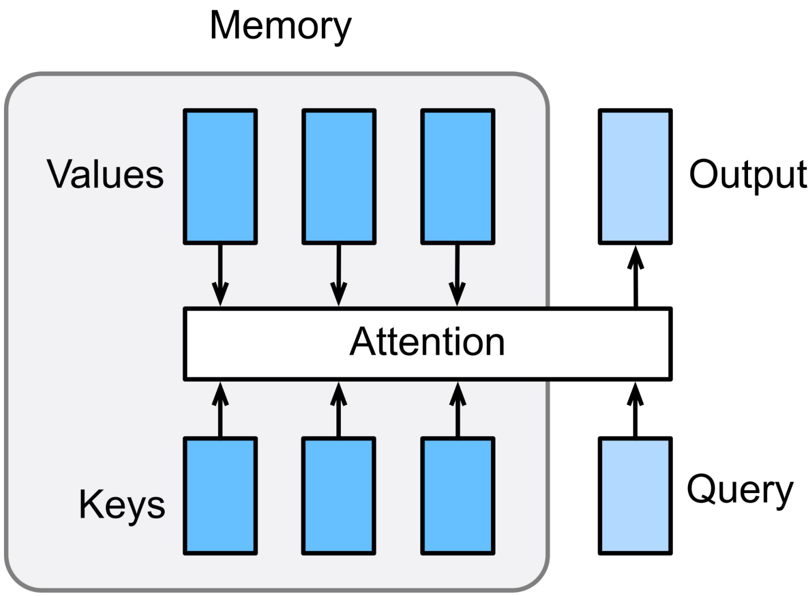


Image source: https://github.com/d2l-ai/d2l-en

Self-attention scales as $O(L^2)$ in sequence length.

Self-attention

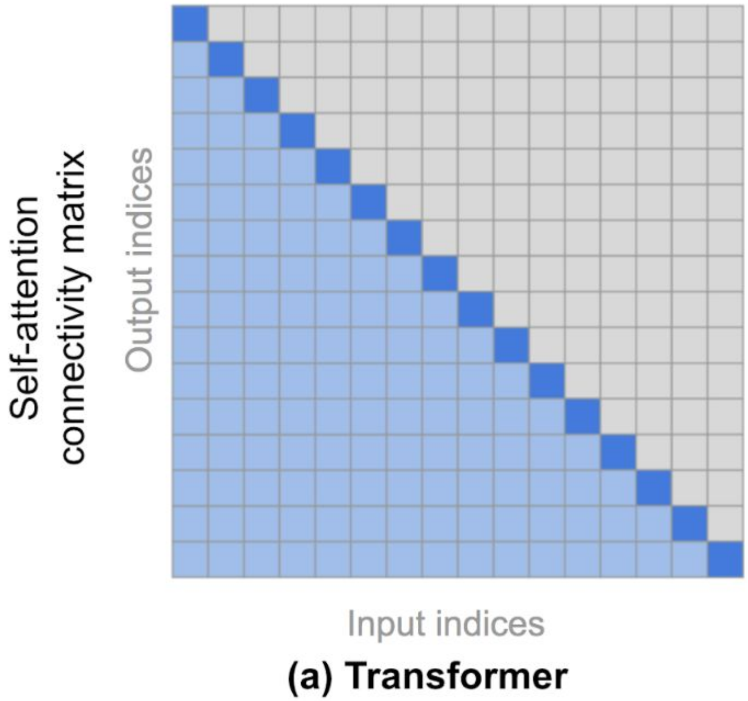


Image source: Child et al. (2019)

Can we do better?

Can we do better?

Efficient and **provable** RNNs
for long contexts?

Spectral filtering

Linear dynamical systems

$$x_t = Ax_{t-1} + Bu_t$$
$$y_t = Cx_t + Du_t$$

Learning optimal weights for factors of *A*, *B*, *C*, *D* is **hard** (non-convex)

$$y_t^{\mathrm{LDS}} = (CB + D)u_t + CABu_{t-1} + CA^2Bu_{t-2} + CA^3Bu_{t-3} + \dots$$

Learning optimal weights for relaxed parameterizations is **easy** (convex)

$$\hat{y}_t = M_0 u_t + M_1 u_{t-1} + M_2 u_{t-2} + \ldots$$

Intuition

- Consider the one-dimensional case (A = a, B,C = I, D = 0 for simplicity)

$$y_t^{\text{LDS}} = (CB + D)u_t + CABu_{t-1} + CA^2Bu_{t-2} + CA^3Bu_{t-3} + \ldots$$

$$y_t^{\text{LDS}} = 1 \cdot u_t + a \cdot u_{t-1} + a^2 \cdot u_{t-2} + a^3 \cdot u_{t-3} + \ldots$$

$$y_t^{\text{LDS}} = \underline{[1, a, a^2, \ldots a^L]}[u_t, u_{t-1}, u_{t-2} \ldots]^\top$$

- We are interested in vectors of the type $\mu(a) \triangleq \underline{[1, a, a^2, \ldots a^L]} \in \mathbb{R}^L \text{ for all } a \in [0, 1]$

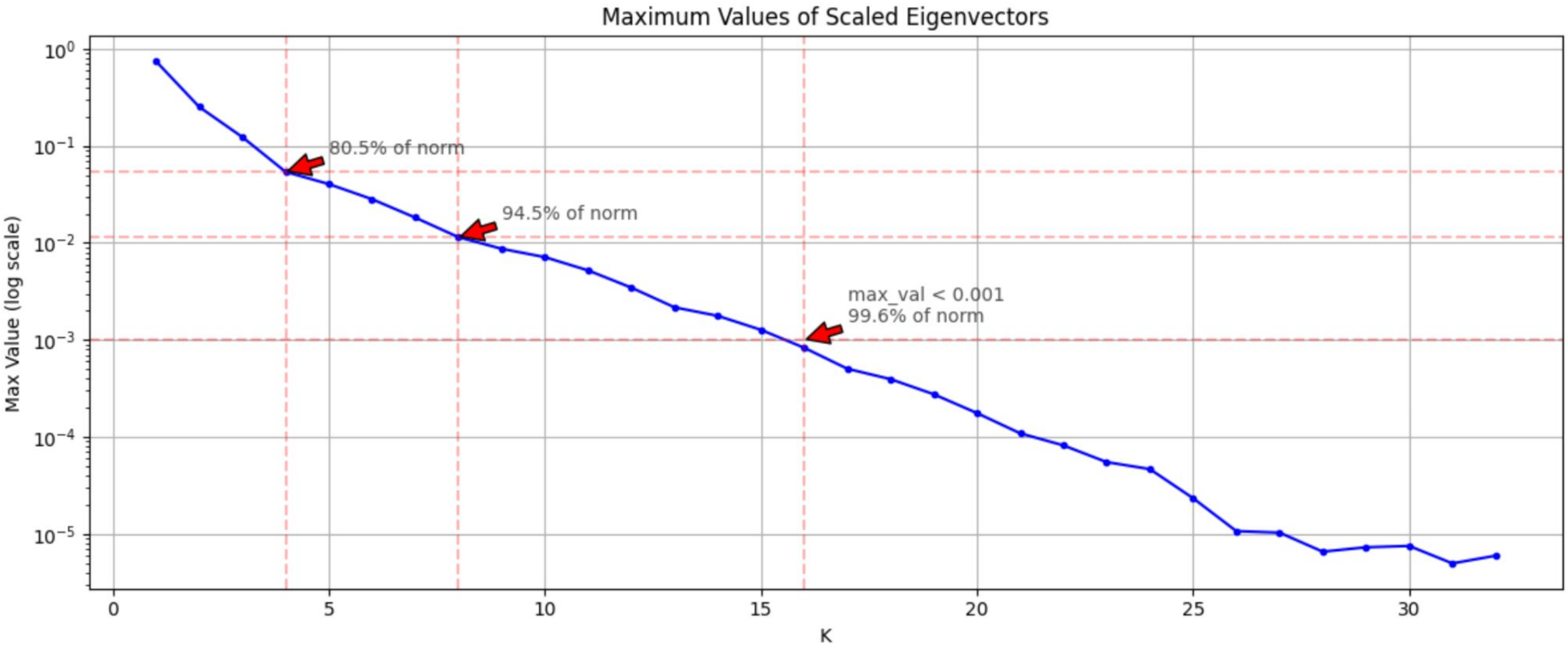
It's a Hankel matrix!

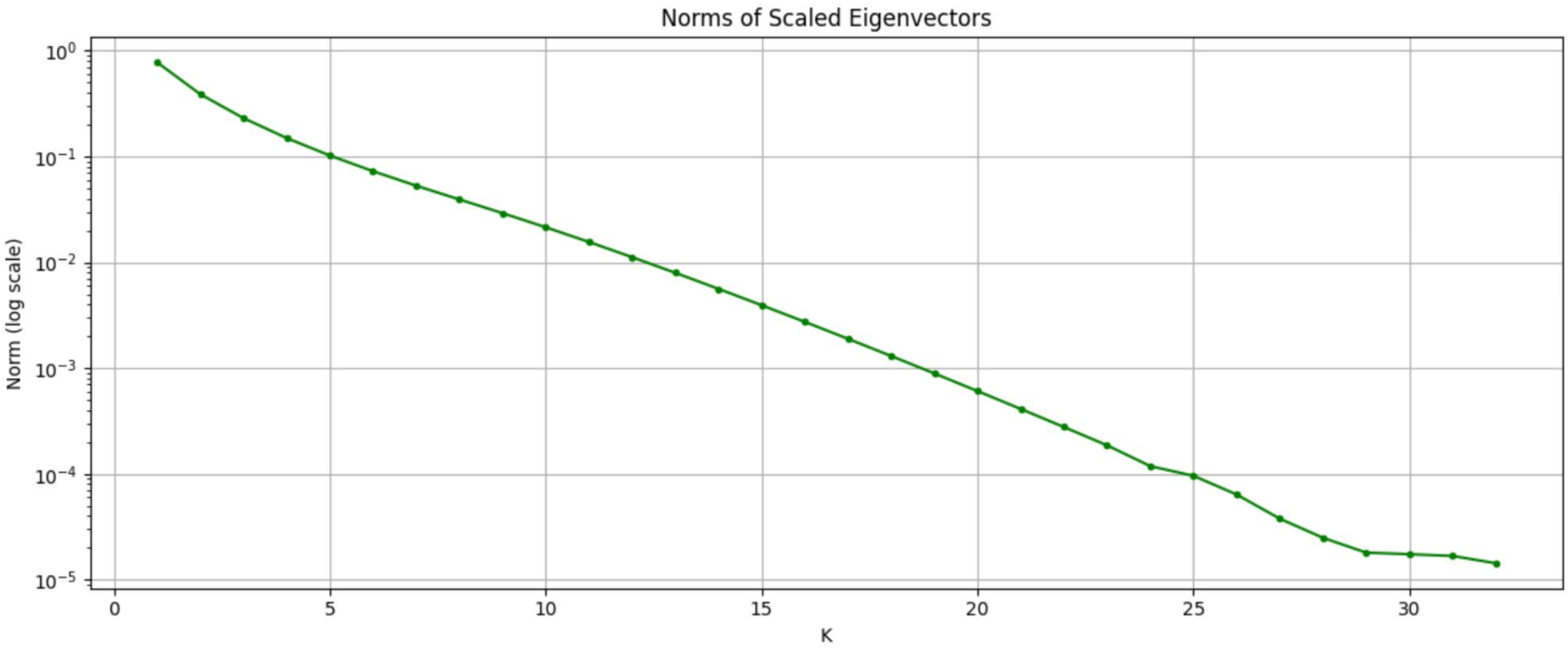
$$\rightarrow \quad \mu(a) \triangleq [1, a, a^2, \dots a^L]$$

$$Z = \int_0^1 \mu(\alpha) \otimes \mu(\alpha)\, d\alpha$$

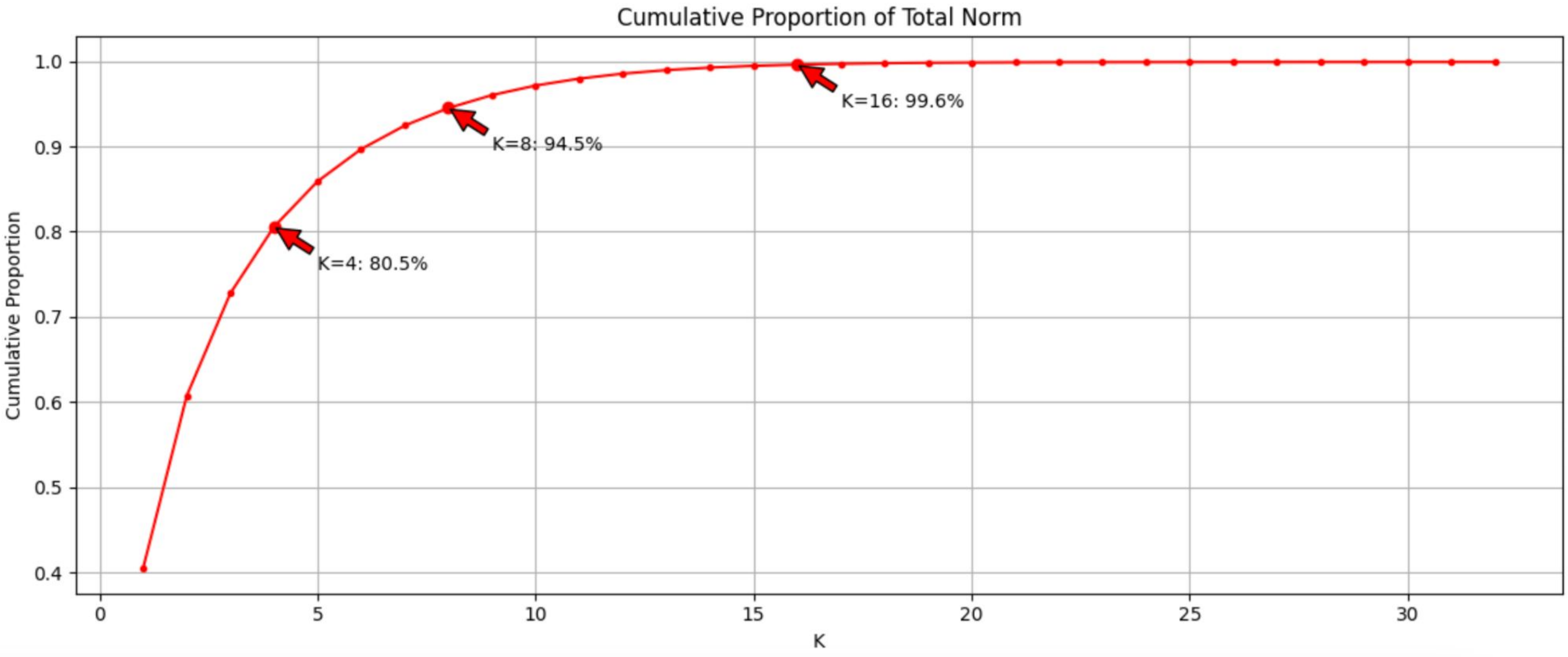
$$Z = \int_0^1 \begin{bmatrix} 1 & \alpha & \alpha^2 & & \\ \alpha & \alpha^2 & \alpha^3 & & \\ \alpha^2 & \alpha^3 & \alpha^4 & & \\ & & & \ddots & \\ & & & & \alpha^{2T-2} \end{bmatrix} d\alpha$$

Eigenvalues of Hankel matrices decay **exponentially.**


Maximum Values of Scaled Eigenvectors
Max Value (log scale)
K
80.5% of norm
94.5% of norm
max_val < 0.001
99.6% of norm
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5
10
15
20
25
30



Norms of Scaled Eigenvectors
Norm (log scale)
K
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5
10
15
20
25
30



Cumulative Proportion of Total Norm
Cumulative Proportion
K
K=4: 80.5%
K=8: 94.5%
K=16: 99.6%
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30

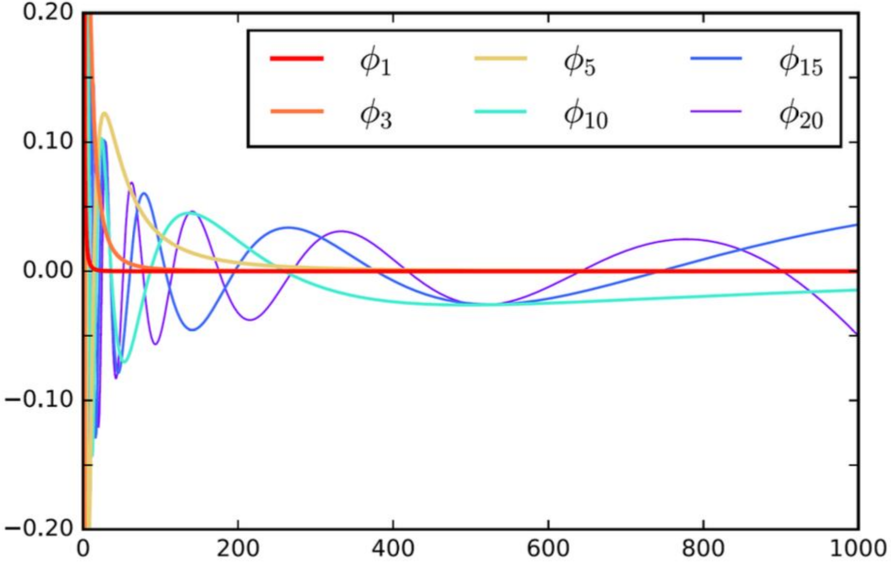


Figure 1: The filters obtained by the eigenvectors of Z.

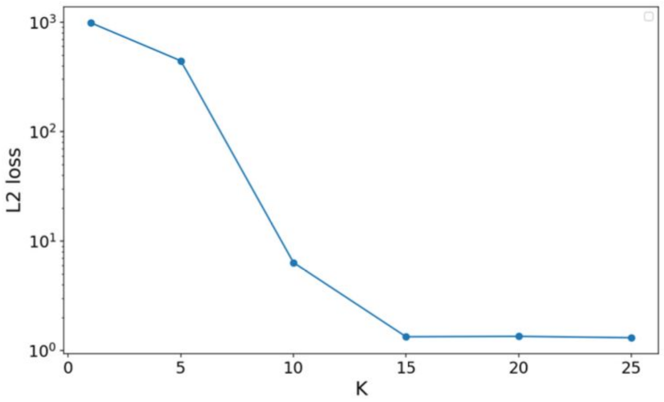


Figure 4: Error obtained by an STU layer as a function of the model parameter K. We observe an exponential drop in the reconstruction loss as predicted by the analysis.

†Figures from the Spectral SSM paper (arXiv: 2312.06837)

Featurization.

- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z

- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z
- Featurization is the **convolution** of input sequence $u_1 \ldots u_L$ with filters $\Phi_1 \ldots \Phi_K$

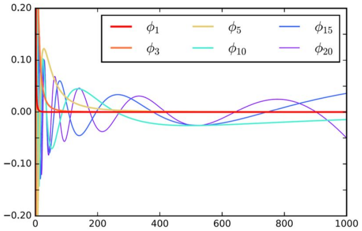
- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z
- Featurization is the **convolution** of input sequence $u_1 \ldots u_L$ with filters $\Phi_1 \ldots \Phi_K$
- These filters are universal – they work for **any sequence prediction task!**

- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z
- Featurization is the **convolution** of input sequence $u_1 \ldots u_L$ with filters $\Phi_1 \ldots \Phi_K$
- These filters are universal – they work for **any sequence prediction task!**
- Convolutions using FFT run in O(L log L) time
  - Featurizing with *k* filters runs in **O(k·L log L) time**
  - Suffices that k ~ O(log L), so we get roughly O(L $\log^2$ L) time

- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z
- Featurization is the **convolution** of input sequence $u_1 \ldots u_L$ with filters $\Phi_1 \ldots \Phi_K$
- These filters are universal – they work for **any sequence prediction task!**
- Convolutions using FFT run in O(L log L) time
  - Featurizing with *k* filters runs in **O(k·L log L) time**
  - Suffices that k ~ O(log L), so we get roughly O(L $\log^2$ L) time
- Convolutions = **GPU-friendly!**
  - NOT sequential, unlike RNNs
  - ==> Asymptotic analysis is meaningful

- Set $\Phi_1 \ldots \Phi_K$ to be the *top-k* eigenvectors of the system matrix Z
- Featurization is the **convolution** of input sequence $u_1 \ldots u_L$ with filters $\Phi_1 \ldots \Phi_K$
- These filters are universal – they work for **any sequence prediction task!**
- Convolutions using FFT run in O(L log L) time
  - Featurizing with *k* filters runs in **O(k·L log L) time**
  - Suffices that k ~ O(log L), so we get roughly O(L $\log^2$ L) time
- Convolutions = **GPU-friendly!**
  - NOT sequential, unlike RNNs
  - ==> Asymptotic analysis is meaningful
- Theoretical guarantees
  - **Sublinear regret** on the order of $\sqrt{L}$ (near-optimal!)

$$\hat{y}_t = M_0 u_t + M_1 u_{t-1} + M_2 u_{t-2} + \dots$$



$$y_t^{\text{LDS}} \sim \sum_{i=1}^{K} \underbrace{\langle \mu(\alpha), \Phi_i \rangle}_{\text{Learned params } M,} \cdot \underbrace{\langle \Phi_i, [u_t, u_{t-1}, u_{t-2} \dots] \rangle}_{\text{Fixed featurization}}$$

Learned params *M*,
scales as $e^{-i/\log(L)}$
so k ~ log(L)

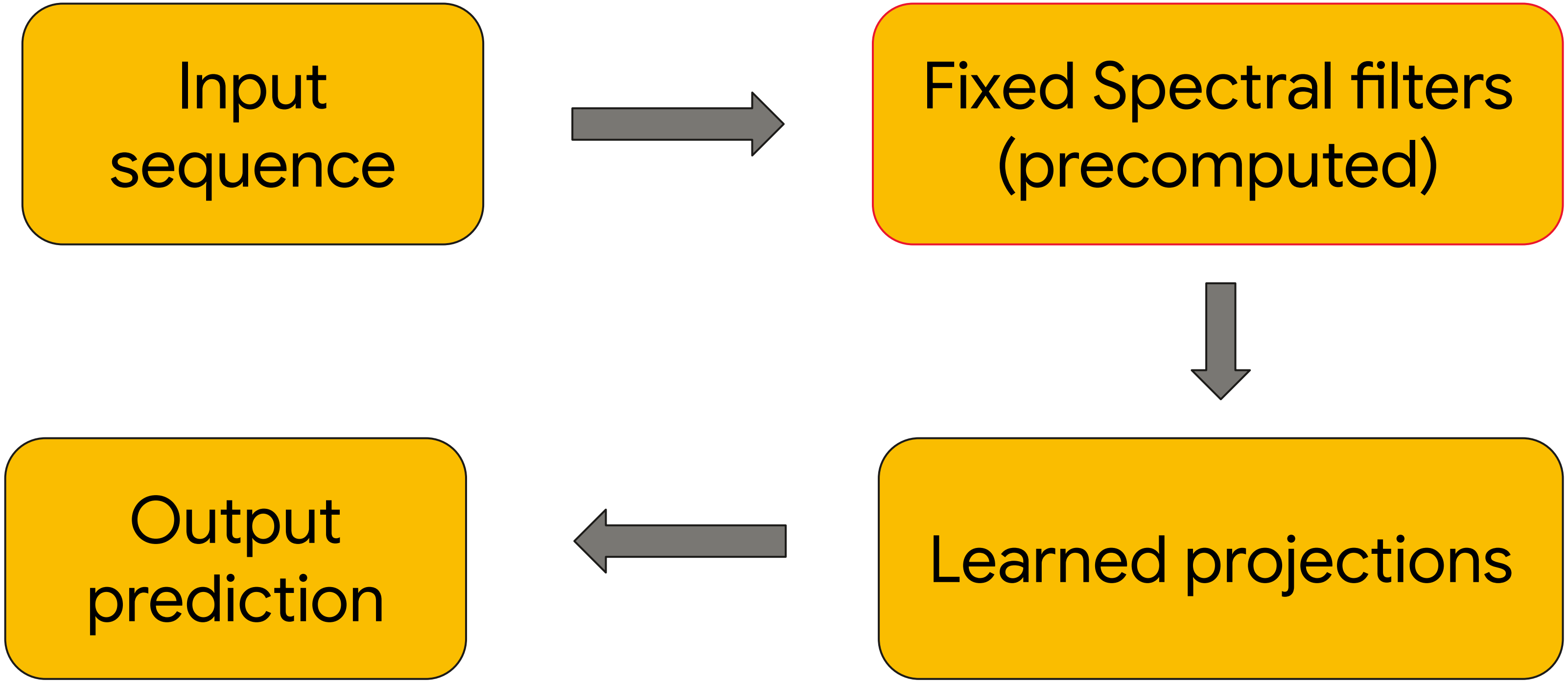
Fixed featurization

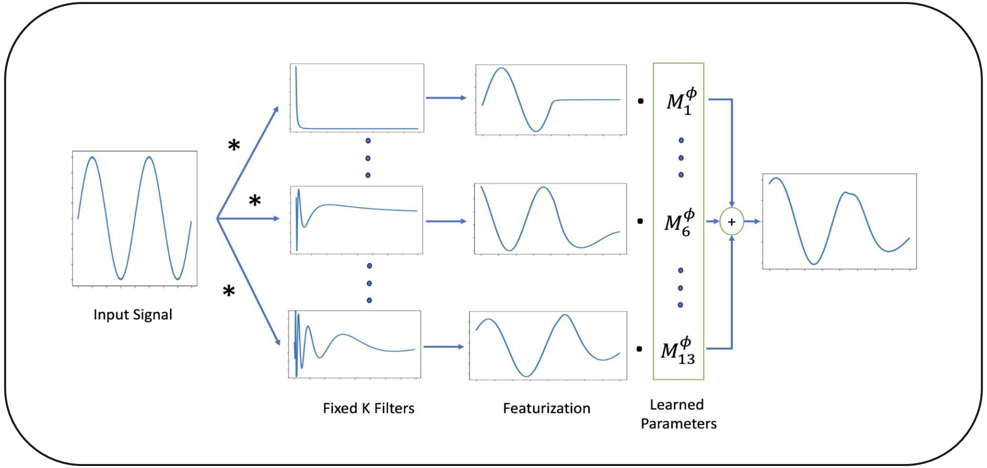
*$\mu(\alpha)$ for any α has all but ϵ mass concentrated on the top eigenvectors of Z.

$$\boxed{y_t^{\text{LDS}} \sim \sum_{i=1}^{K} M_i \cdot \underbrace{\langle \Phi_i, [u_t, u_{t-1}, u_{t-2} \dots] \rangle}_{\text{Fixed Featurization}}}$$

Spectral Transform Unit

Motivation
Spectral filtering
STU
Experiments
Current work


Input sequence
Fixed Spectral filters (precomputed)
Learned projections
Output prediction

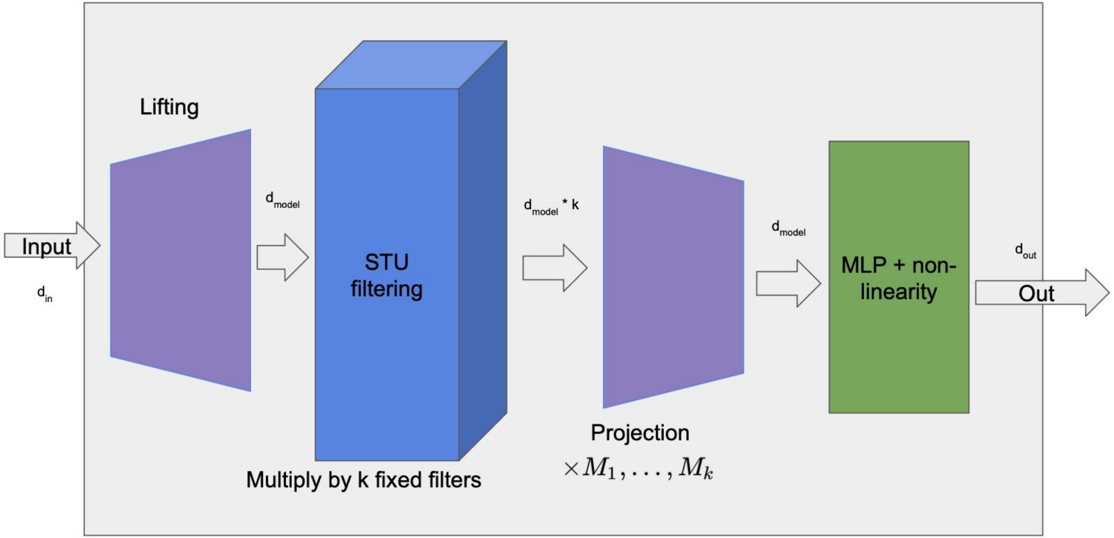


*
*
*
M_1^{ϕ}
M_6^{ϕ}
M_{13}^{ϕ}
+
Input Signal
Fixed K Filters
Featurization
Learned Parameters

Models

STU

Flash STU



STU Block

†Figures from the Flash STU paper (arXiv: 2409.10489)


FLASH STU

Flash STU Model Architecture

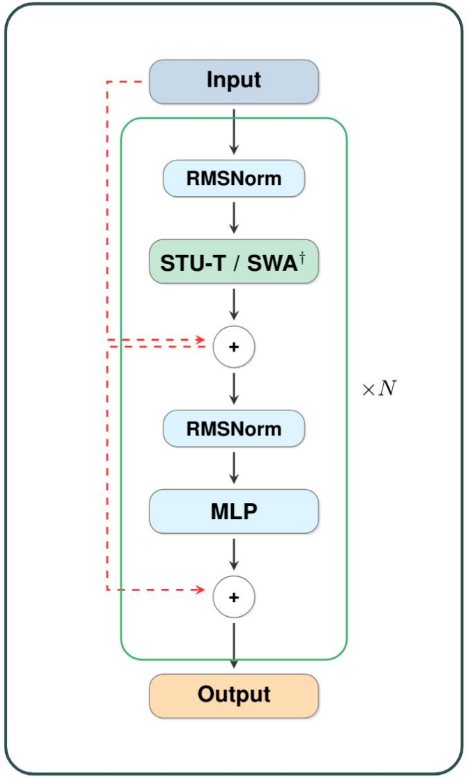


Figure 12: Flash STU Model Architecture, alternating between STU-T and (sliding window) attention†.

†Figures from the Flash STU paper (arXiv: 2409.10489)

Experiments

Motivation
Spectral filtering
STU
Experiments
Current work

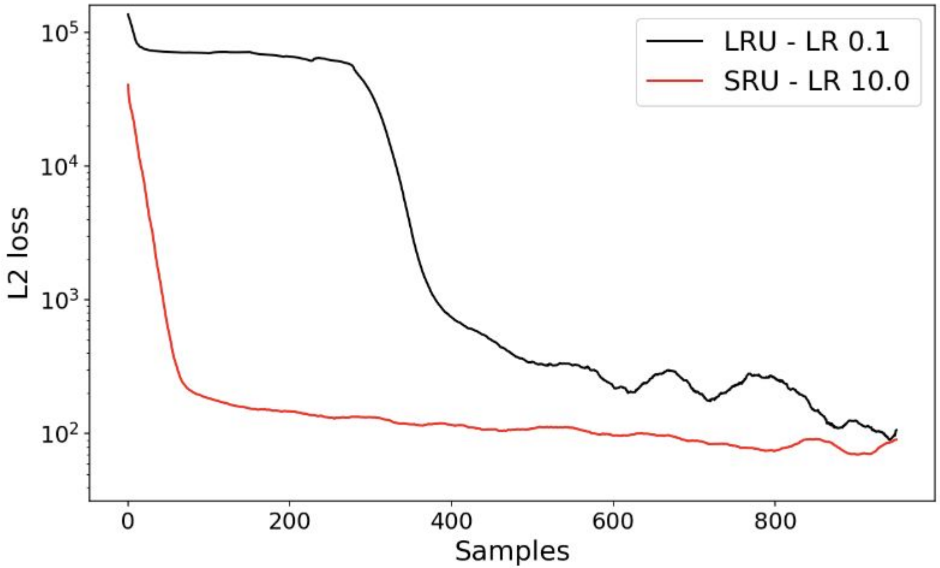
Tasks

LDS

Robotics

Language Modeling

Linear Dynamical Systems



LRU (Orvieto et al. 2023)

- Diagonal A (complex)
- Stable Exponential Param.
  - $A_{ii} = \exp(-\exp(\log(\nu))i + \theta j)$
- Ring Initialization
  - Ensure $A_{ii} \in [r_{\min}, r_{\max}]$
- γ-normalization
  - Multiplier on B adapted to A
  - Prevents loss blowup at init

LRU required **all** interventions to train.

STU trained out of the box with zero init.

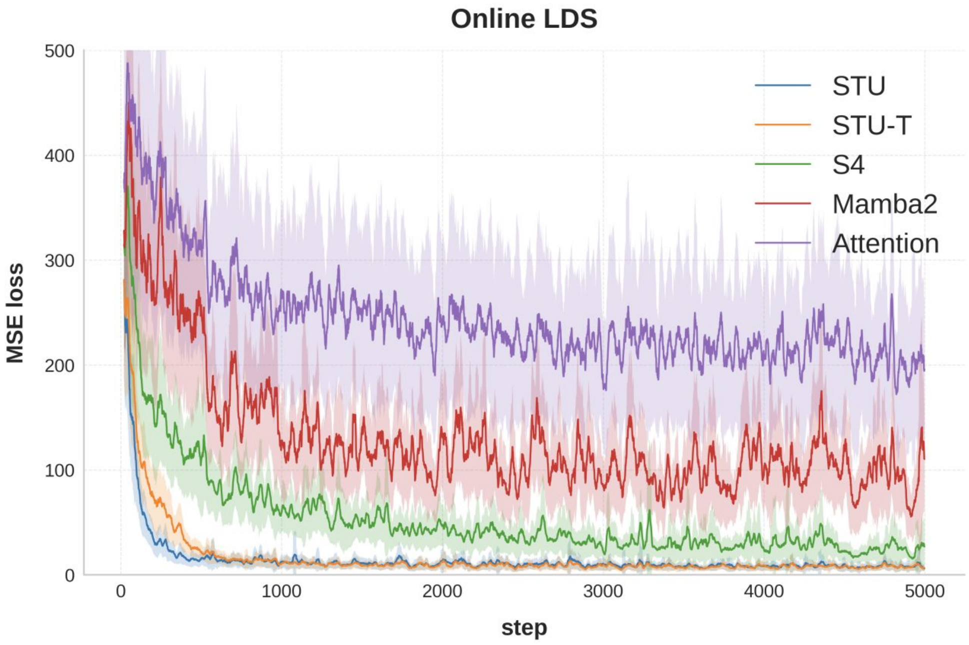


Figure 2: Mean squared error $\|\hat{y}_{t+1} - y_{t+1}\|^2$ of the different layers on a single sequence from an LDS.

Robotics

- Zero hyperparameter tuning needed for filters
- Stable training out-of-the-box

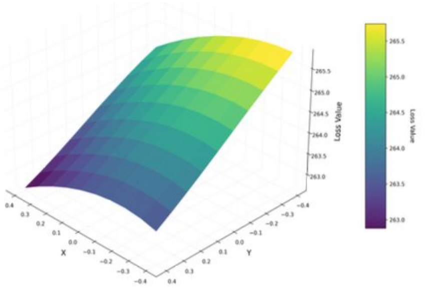


Figure 3: Local loss landscape of the **STU** layer.

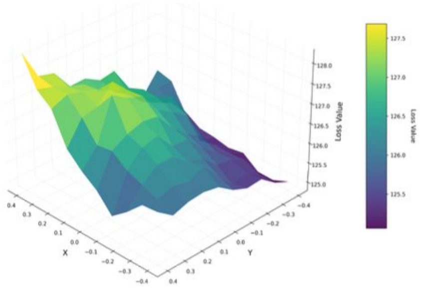


Figure 4: Local loss landscape of the **S4** layer.

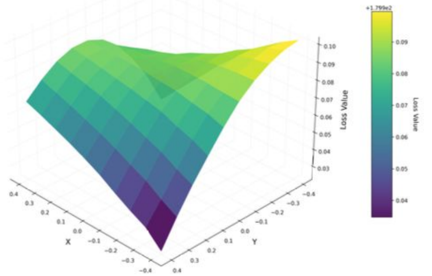
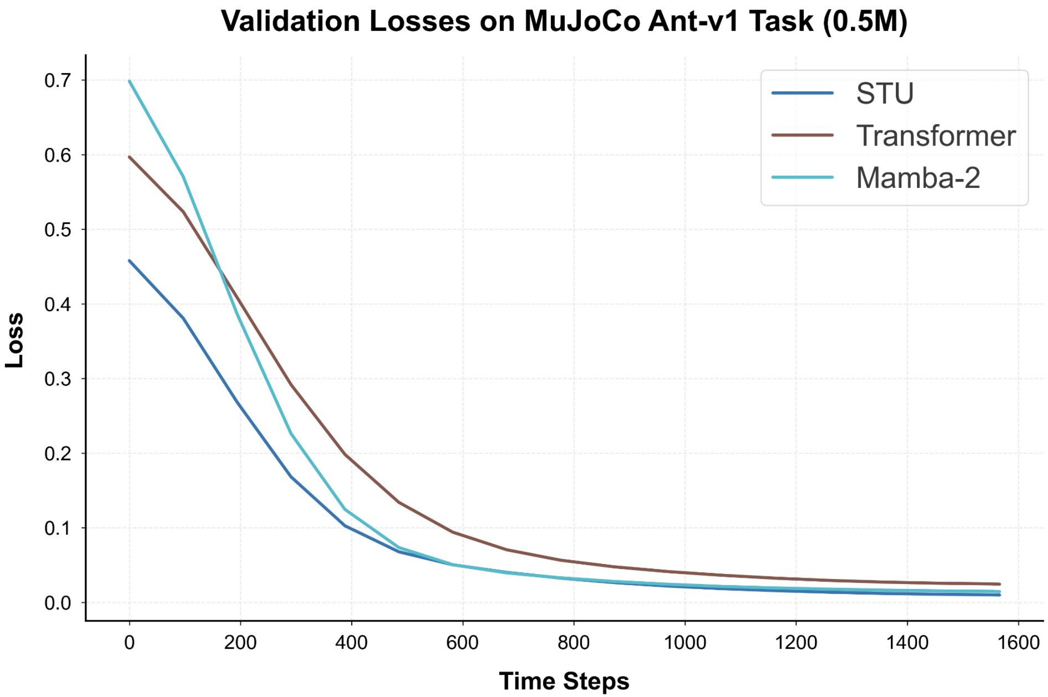
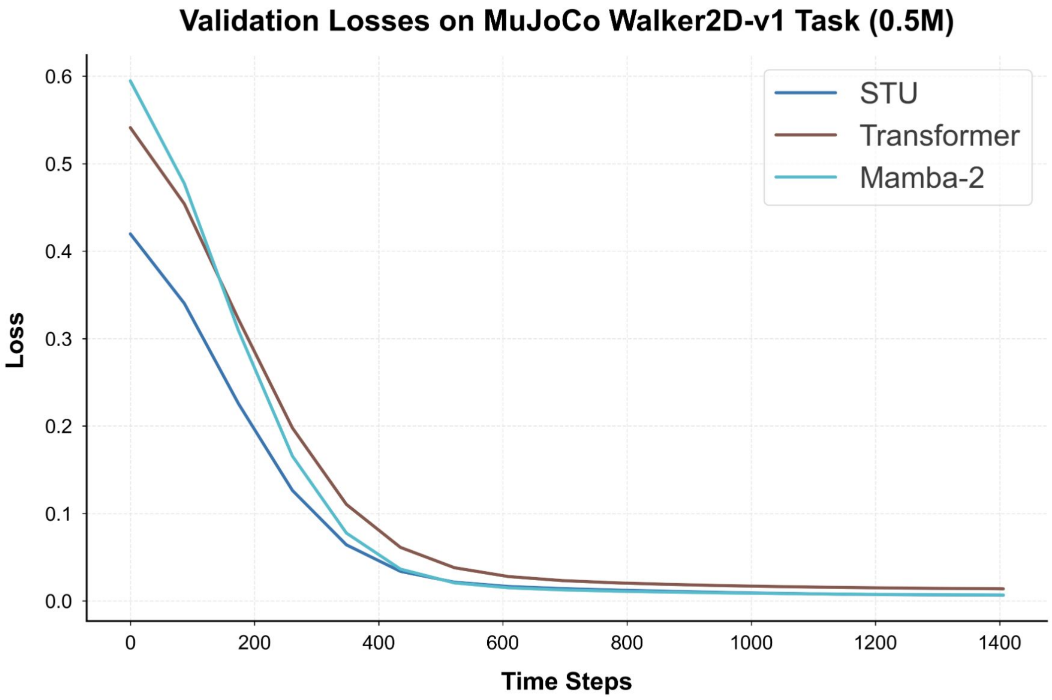


Figure 5: Local loss landscape of the **attention** layer.

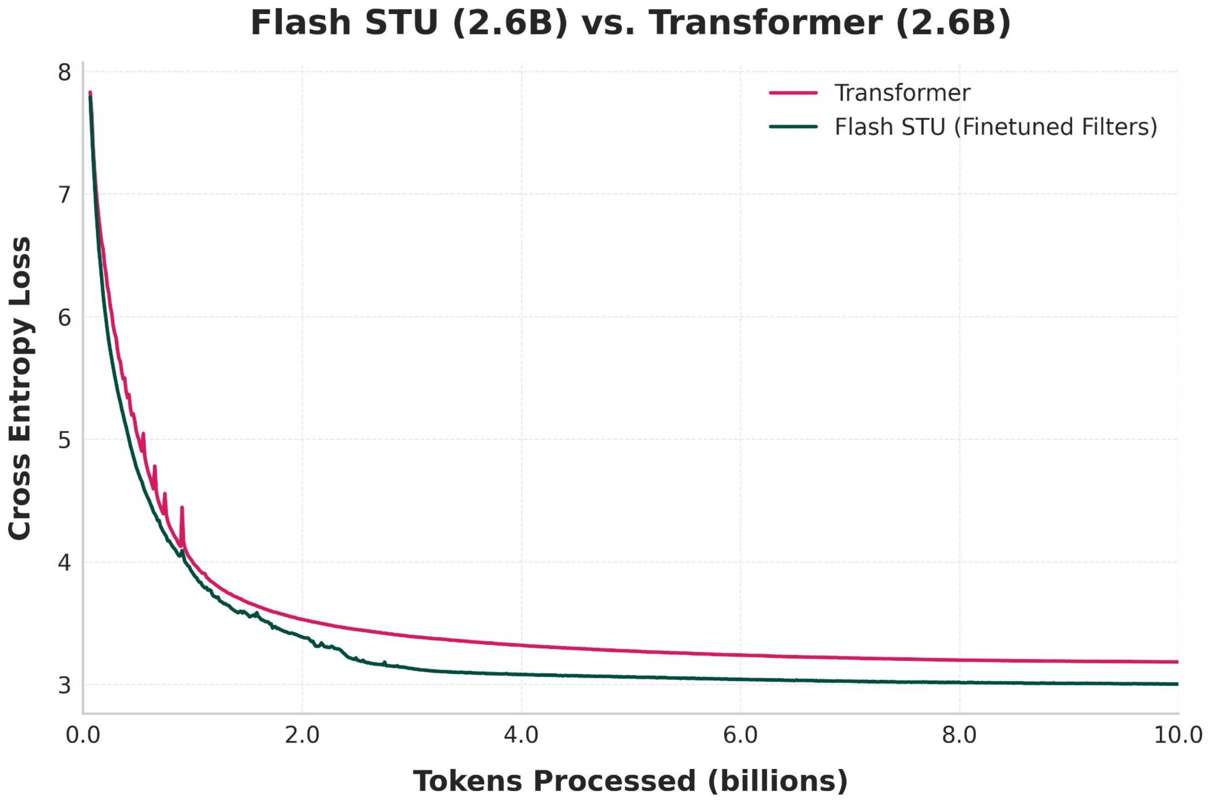
- Friendly “loss landscape” to optimize
- Theoretical guarantees on performance

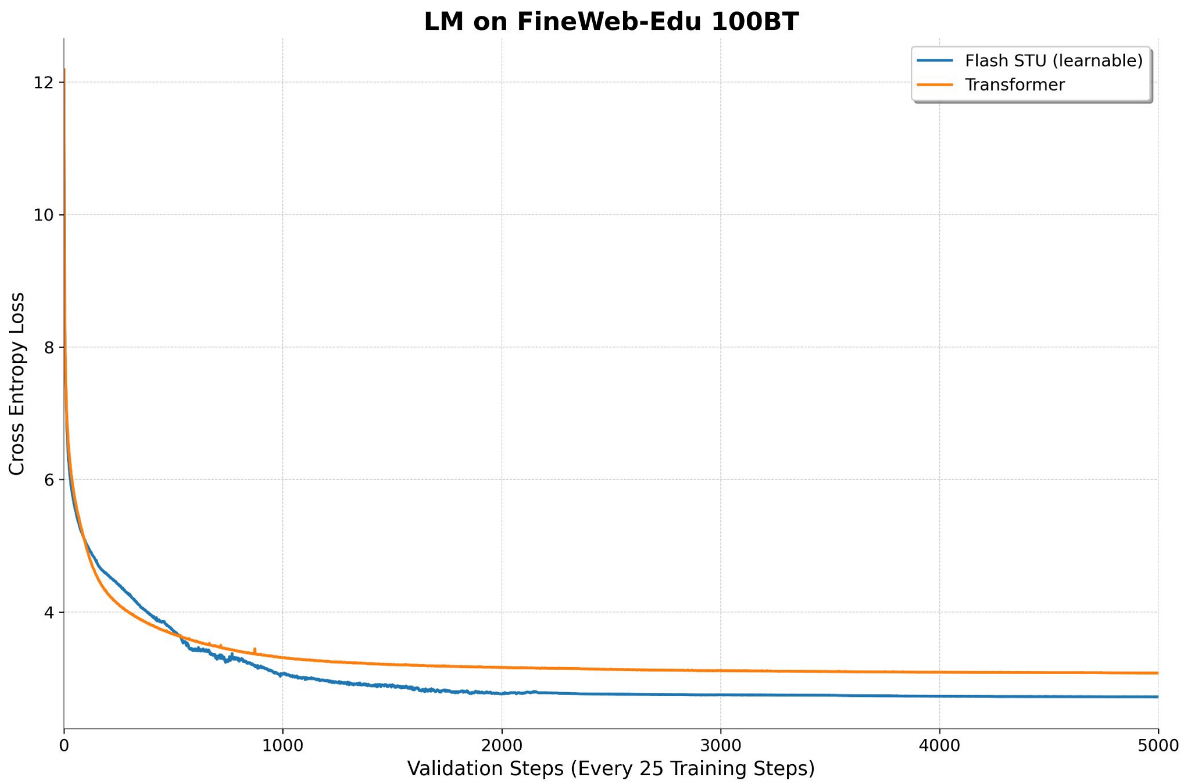


Model	Validation Loss
STU	**0.0092**
Transformer	0.0237
Mamba-2	0.0139



Model	Validation Loss
STU	**0.0062**
Transformer	0.0134
Mamba-2	0.0066


Flash STU (2.6B) vs. Transformer (2.6B)
Transformer
Flash STU (Finetuned Filters)
Cross Entropy Loss
Tokens Processed (billions)
8
7
6
5
4
3
0.0
2.0
4.0
6.0
8.0
10.0


LM on FineWeb-Edu 100BT
Flash STU (learnable)
Transformer
Cross Entropy Loss
Validation Steps (Every 25 Training Steps)
12
10
8
6
4
0
1000
2000
3000
4000
5000

Transformers	**RNNs / SSMs**	**STU**
$O(L^2)$ complexity	$O(L)$ complexity	$O(L \log L)$ complexity
Powerful but slow	Fast but unstable	Fast AND stable
Memory hungry	Training tricks needed	Simple to train

Current work

Motivation
Spectral filtering
STU
Experiments
Current work

Spectral State Space Models

Naman Agarwal
Google Deepmind
namanagarwal@google.com

Daniel Suo
Google Deepmind

Xinyi Chen
Princeton University
Google Deepmind

Elad Hazan
Princeton University
Google Deepmind

Abstract

This paper studies sequence modeling for prediction tasks with long range dependencies. We propose a new formulation for state space models (SSMs) based on learning linear dynamical systems with the spectral filtering algorithm [HSZ17]. This gives rise to a novel sequence prediction architecture we call a spectral state space model.

Spectral state space models have two primary advantages. First, they have provable robustness properties as their performance depends on neither the spectrum of the underlying dynamics nor the dimensionality of the problem. Second, these models are constructed with fixed convolutional filters that do not require learning while still outperforming SSMs in both theory and practice.

The resulting models are evaluated on synthetic dynamical systems and long-range prediction tasks of various modalities. These evaluations support the theoretical benefits of spectral filtering for tasks requiring very long range memory.

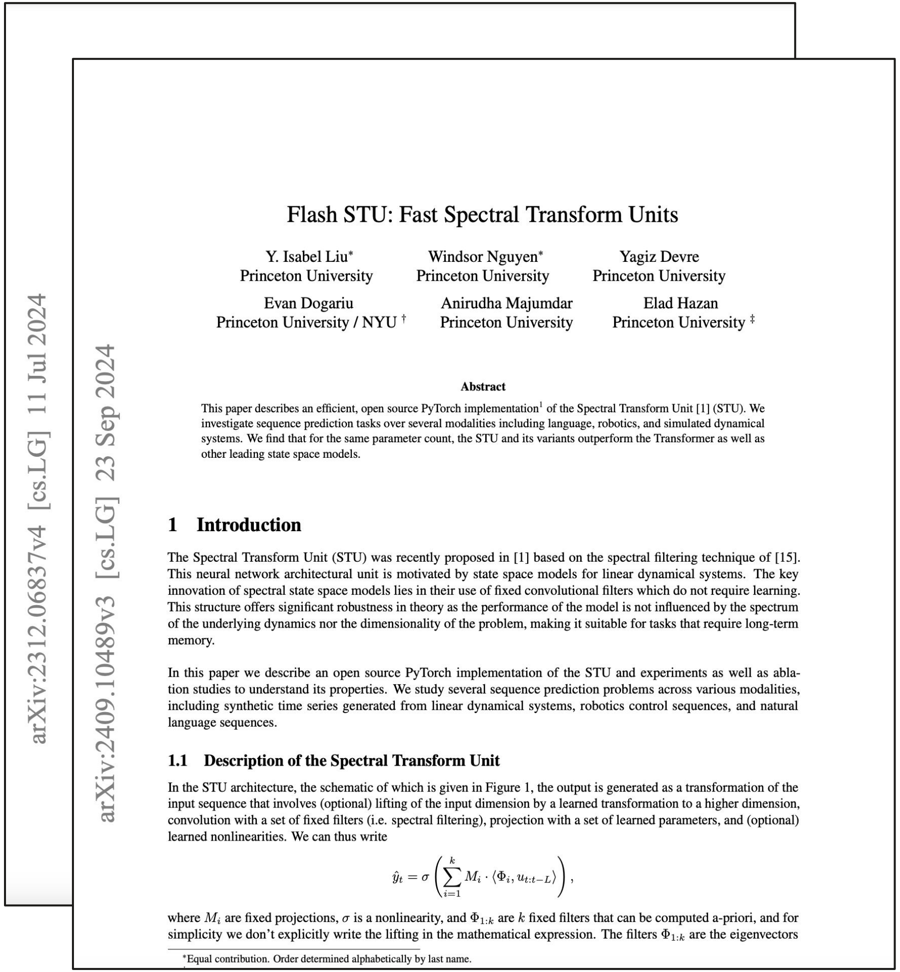
1 Introduction

Handling long-range dependencies efficiently remains a core problem in sequence prediction/modelling. Recurrent Neural Networks (RNN) [Hop82, RHW+85, Elm90] are a natural choice, but are notoriously hard to train; they often suffer from vanishing and exploding gradients [BSF94, PMB13] and despite techniques to mitigate the issue [HS97, CVMG+14, ASB16], they are also hard to scale given the inherently sequential nature of their computation.

In recent years, transformer models [VSP+17] have become the staple of sequence modelling, achieving remarkable success across multiple domains [BMR+20, DBK+20, JEP+21]. Transformer models are naturally parallelizable and hence scale significantly better than RNNs. However, attention layers have memory/computation requirements that scale quadratically with context length. Many approximations have been proposed (see [TDBM22] for a recent survey).

RNNs have seen a recent resurgence in the form of state space models (SSM) which have shown promise in modelling long sequences across varied modalities [GGR21, DFS+22, GGB22, OSG+23, PMN+23, GD23]. SSMs use linear dynamical systems (LDS) to model the sequence-to sequence transform by evolving the internal state of a dynamical system according to the dynamics equations

$$x_t = Ax_{t-1} + Bu_t \qquad y_t = Cx_t + Du_t.$$



Flash STU: Fast Spectral Transform Units

Y. Isabel Liu* Princeton University
Windsor Nguyen* Princeton University
Yagiz Devre Princeton University
Evan Dogariu Princeton University / NYU †
Anirudha Majumdar Princeton University
Elad Hazan Princeton University ‡

Abstract

This paper describes an efficient, open source PyTorch implementation[1] of the Spectral Transform Unit [1] (STU). We investigate sequence prediction tasks over several modalities including language, robotics, and simulated dynamical systems. We find that for the same parameter count, the STU and its variants outperform the Transformer as well as other leading state space models.

1 Introduction

The Spectral Transform Unit (STU) was recently proposed in [1] based on the spectral filtering technique of [15]. This neural network architectural unit is motivated by state space models for linear dynamical systems. The key innovation of spectral state space models lies in their use of fixed convolutional filters which do not require learning. This structure offers significant robustness in theory as the performance of the model is not influenced by the spectrum of the underlying dynamics nor the dimensionality of the problem, making it suitable for tasks that require long-term memory.

In this paper we describe an open source PyTorch implementation of the STU and experiments as well as ablation studies to understand its properties. We study several sequence prediction problems across various modalities, including synthetic time series generated from linear dynamical systems, robotics control sequences, and natural language sequences.

1.1 Description of the Spectral Transform Unit

In the STU architecture, the schematic of which is given in Figure 1, the output is generated as a transformation of the input sequence that involves (optional) lifting of the input dimension by a learned transformation to a higher dimension, convolution with a set of fixed filters (i.e. spectral filtering), projection with a set of learned parameters, and (optional) learned nonlinearities. We can thus write

$$\hat{y}_t = \sigma\left(\sum_{i=1}^{k} M_i \cdot \langle \Phi_i, u_{t:t-L} \rangle\right),$$

where M_i are fixed projections, σ is a nonlinearity, and $\Phi_{1:k}$ are k fixed filters that can be computed a-priori, and for simplicity we don't explicitly write the lifting in the mathematical expression. The filters $\Phi_{1:k}$ are the eigenvectors

*Equal contribution. Order determined alphabetically by last name.

Provable Length Generalization in Sequence Prediction via Spectral Filtering

Annie Marsden * Evan Dogariu † Naman Agarwal Xinyi Chen

Daniel Suo Elad Hazan ‡

November 5, 2024

Abstract

We consider the problem of length generalization in sequence prediction. We define a new metric of performance in this setting – the Asymmetric-Regret– which measures regret against a benchmark predictor with longer context length than available to the learner. We continue by studying this concept through the lens of the spectral filtering algorithm. We present a gradient-based learning algorithm that provably achieves length generalization for linear dynamical systems. We conclude with proof-of-concept experiments which are consistent with our theory.

1 Introduction

Sequence prediction is a fundamental problem in machine learning with widespread applications in natural language processing, time-series forecasting, and control systems. In this setting, a learner observes a sequence of tokens and iteratively predicts the next token, suffering a loss that measures the discrepancy between the predicted and the true token. Predicting future elements of a sequence based on historical data is crucial for tasks ranging from language modeling to autonomous control.

A key challenge in sequence prediction is understanding the role of *context length*—the number of previous tokens used to make the upcoming prediction—and designing predictors that perform well with limited context due to computational and memory constraints. These resource constraints become particularly significant during the training phase of a predictor, where the computational cost of using long sequences can be prohibitive. Consequently, it is beneficial to design predictors that can learn from a smaller context length while still generalizing well to longer sequences. This leads us to the central question of our investigation: Can we develop algorithms that learn effectively using short contexts but perform comparably to models that use longer contexts?

To address this question, we introduce a new performance metric—Asymmetric-Regret—which measures the difference in total prediction loss between an online predictor with limited context length and a benchmark predictor with a longer context. Unlike classical regret, which assumes both the learner and the benchmark operate under the same conditions, Asymmetric-Regret accounts for the asymmetry in context lengths, providing a more realistic assessment of performance in resource-constrained settings. With a formal and well-defined notion of Asymmetric-Regret in hand, we begin our investigation with the following question: are there algorithms that can attain non-trivial bounds on the Asymmetric-Regret for natural sequences?

We explore this concept through the lens of spectral filtering algorithms (Hazan et al., 2017b, 2018). Spectral filtering has emerged as a robust method for learning linear dynamical systems when the system is unknown and the hidden state is unobserved. Beyond their theoretically sound properties, spectral filtering-based predictors have proven practical

Length Generalization

How to appropriately define length generalization?
New definition: **Asymmetric Regret**

<u>**New Result:**</u>

- **Spectral Transformers can achieve vanishing asymmetric regret** despite observing a small fraction of the context
- **Novel Tensored formulation** for Spectral Transformers
  - **arbitrary length extrapolation / length generalization**

Asymmetric Regret

How to define length generalization?
The asymmetric-regret:

$$\sum_t \|y_t - \hat{y}_t^{A(L')}\| - \mathbf{min}_{\pi \in \prod_L} \|y_t - \hat{y}_t^{\pi}\|$$

Prediction algorithm uses context of length L′ << L

Best predictor w. context length L

Tensored spectral filters

Tensor decomposition property of the Vandermonde vectors:

$$\mu_{\alpha}^{L^2} = (1, \alpha, \ldots, \alpha^{L^2}) \qquad \mu_{\alpha}^{L^2} = \mu_{\alpha}^{L} \otimes \mu_{\alpha^L}^{L}$$

⇒ Improper relaxation

$$\{\mu_{\alpha}^{L^2}\} \subseteq \{\mu_{\alpha}^{L} \otimes \mu_{\beta}^{L}\}$$

⇒ New Tensored filtering algorithm!

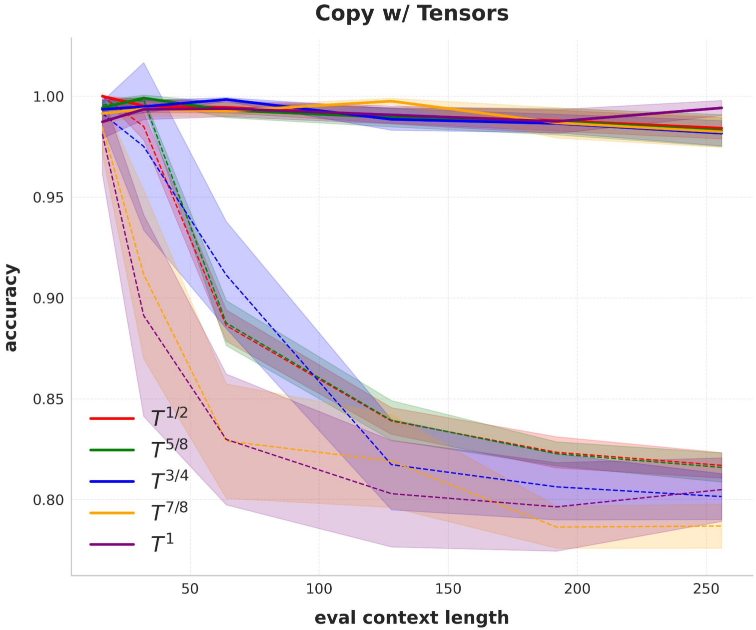
$$\{\phi_i^L \otimes \phi_j^L \ , \ i, j \in [k]\}$$

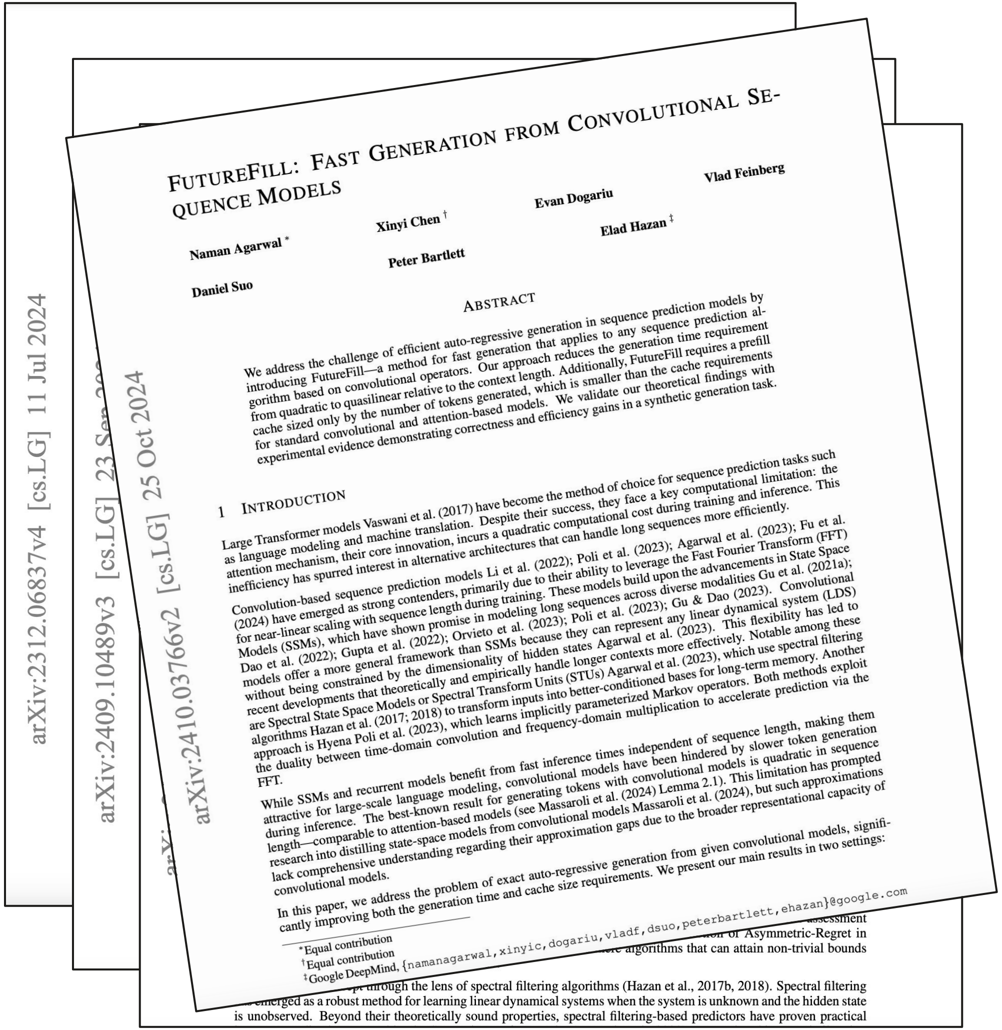
NEW: provable length generalization!

Theorem: let $y_1,\ldots,y_T$ be generated from marginally stable LDS, then STU algorithm guarantees:

$$\sum_t \|y_t - \hat{y}_t^{STU(\sqrt{T})}\| - \min_{\pi \in LDS_L} \|y_t - \hat{y}_t^{\pi}\| = O(\sqrt{T})$$

⇒ length generalization from root(T) to T !




arXiv:2312.06837v4 [cs.LG] 11 Jul 2024

arXiv:2409.10489v3 [cs.LG] 23 Sep 2024

arXiv:2410.03766v2 [cs.LG] 25 Oct 2024

FUTUREFILL: FAST GENERATION FROM CONVOLUTIONAL SEQUENCE MODELS

Naman Agarwal [*] Xinyi Chen [†] Evan Dogariu Vlad Feinberg
Daniel Suo Peter Bartlett Elad Hazan [‡]

ABSTRACT

We address the challenge of efficient auto-regressive generation in sequence prediction models by introducing FutureFill—a method for fast generation that applies to any sequence prediction algorithm based on convolutional operators. Our approach reduces the generation time requirement from quadratic to quasilinear relative to the context length. Additionally, FutureFill requires a prefill cache sized only by the number of tokens generated, which is smaller than the cache requirements for standard convolutional and attention-based models. We validate our theoretical findings with experimental evidence demonstrating correctness and efficiency gains in a synthetic generation task.

1 INTRODUCTION

Large Transformer models Vaswani et al. (2017) have become the method of choice for sequence prediction tasks such as language modeling and machine translation. Despite their success, they face a key computational limitation: the attention mechanism, their core innovation, incurs a quadratic computational cost during training and inference. This inefficiency has spurred interest in alternative architectures that can handle long sequences more efficiently.

Convolution-based sequence prediction models Li et al. (2022); Poli et al. (2023); Agarwal et al. (2023); Fu et al. (2024) have emerged as strong contenders, primarily due to their ability to leverage the Fast Fourier Transform (FFT) for near-linear scaling with sequence length during training. These models build upon the advancements in State Space Models (SSMs), which have shown promise in modeling long sequences across diverse modalities Gu et al. (2021a); Dao et al. (2022); Gupta et al. (2022); Orvieto et al. (2023); Poli et al. (2023); Gu & Dao (2023). Convolutional models offer a more general framework than SSMs because they can represent any linear dynamical system (LDS) without being constrained by the dimensionality of hidden states Agarwal et al. (2023). This flexibility has led to recent developments that theoretically and empirically handle longer contexts more effectively. Notable among these are Spectral State Space Models or Spectral Transform Units (STUs) Agarwal et al. (2023), which use spectral filtering algorithms Hazan et al. (2017; 2018) to transform inputs into better-conditioned bases for long-term memory. Another approach is Hyena Poli et al. (2023), which learns implicitly parameterized Markov operators. Both methods exploit the duality between time-domain convolution and frequency-domain multiplication to accelerate prediction via the FFT.

While SSMs and recurrent models benefit from fast inference times independent of sequence length, making them attractive for large-scale language modeling, convolutional models have been hindered by slower token generation during inference. The best-known result for generating tokens with convolutional models is quadratic in sequence length—comparable to attention-based models (see Massaroli et al. (2024) Lemma 2.1). This limitation has prompted research into distilling state-space models from convolutional models Massaroli et al. (2024), but such approximations lack comprehensive understanding regarding their approximation gaps due to the broader representational capacity of convolutional models.

In this paper, we address the problem of exact auto-regressive generation from given convolutional models, significantly improving both the generation time and cache size requirements. We present our main results in two settings:

[*]Equal contribution
[†]Equal contribution
[‡]Google DeepMind, {namanagarwal,xinyic,dogariu,vladf,dsuo,peterbartlett,ehazan}@google.com

...assessment ...of Asymmetric-Regret in ...algorithms that can attain non-trivial bounds

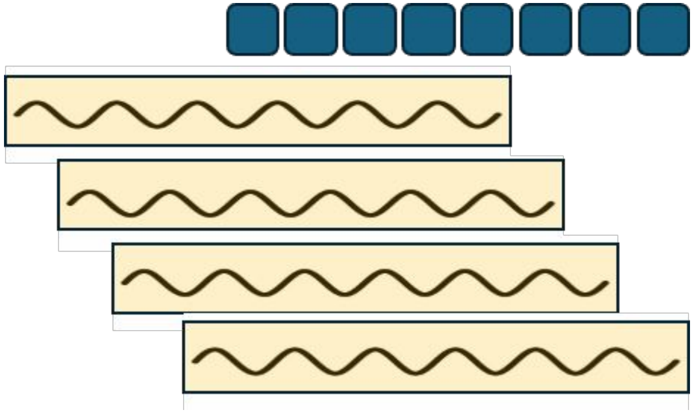
...through the lens of spectral filtering algorithms (Hazan et al., 2017b, 2018). Spectral filtering ...emerged as a robust method for learning linear dynamical systems when the system is unknown and the hidden state is unobserved. Beyond their theoretically sound properties, spectral filtering-based predictors have proven practical

Fast Inference

Generating 1 token:

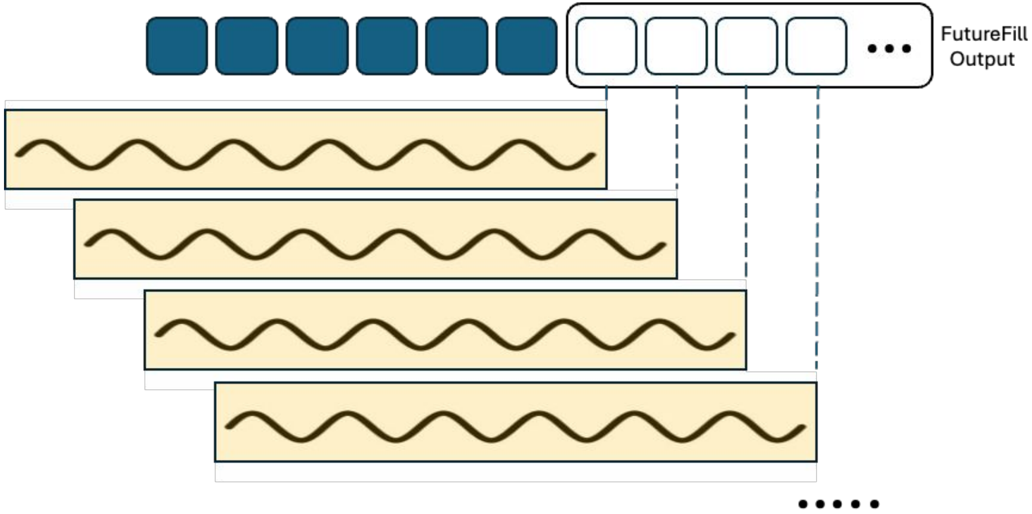
- **Transformers - O(prompt length + generation length)** - memory & compute
- **RNNs/SSMs - O(1) -** memory & compute
- **Convolutional Models -**
  - Naively same as attention

<u>New Result:</u> **O($\log^2 L$) –** compute



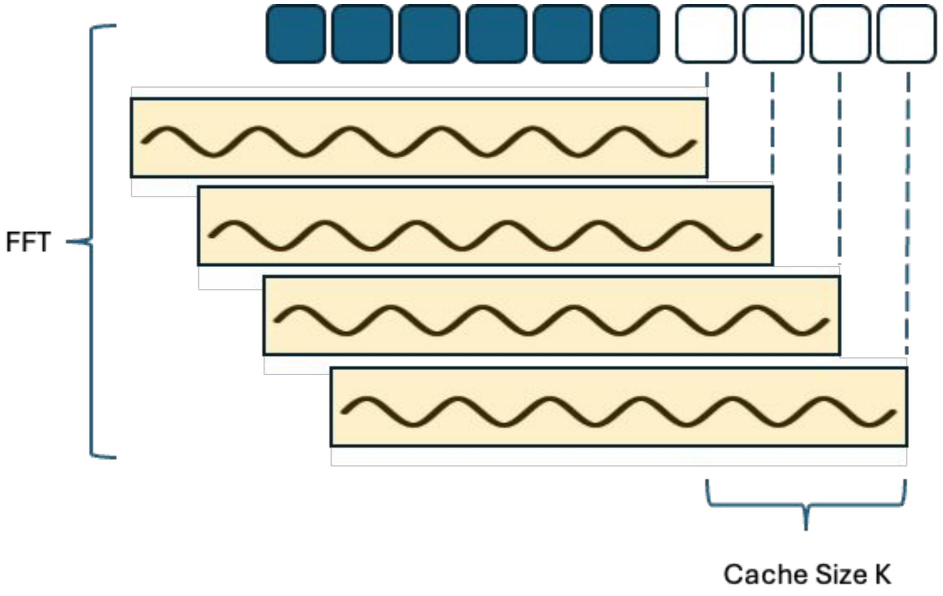
FutureFill

- The future contribution of the current token can be computed now
  - Not possible for attention



FutureFill

- The future contribution of the current token can be computed now
  - Not possible for attention



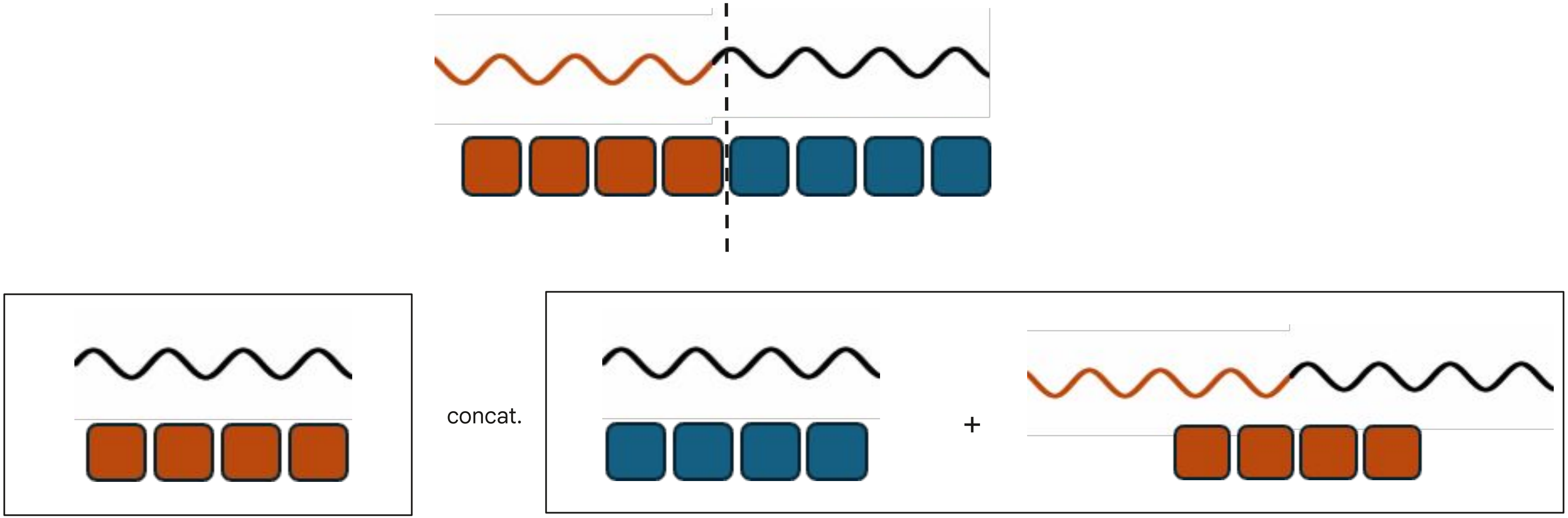
Use FFT to compute FutureFill in time L log(L)

- FutureFill cost – $O(L \log L)$
- Compute k^{th} token - $O(k)$
- Overall:

$$\frac{\sum_{i=1}^{K} i + L\log L}{K} = \frac{K^2 + L\log L}{K} \sim O(\sqrt{L\log L})$$

Recursive FutureFill

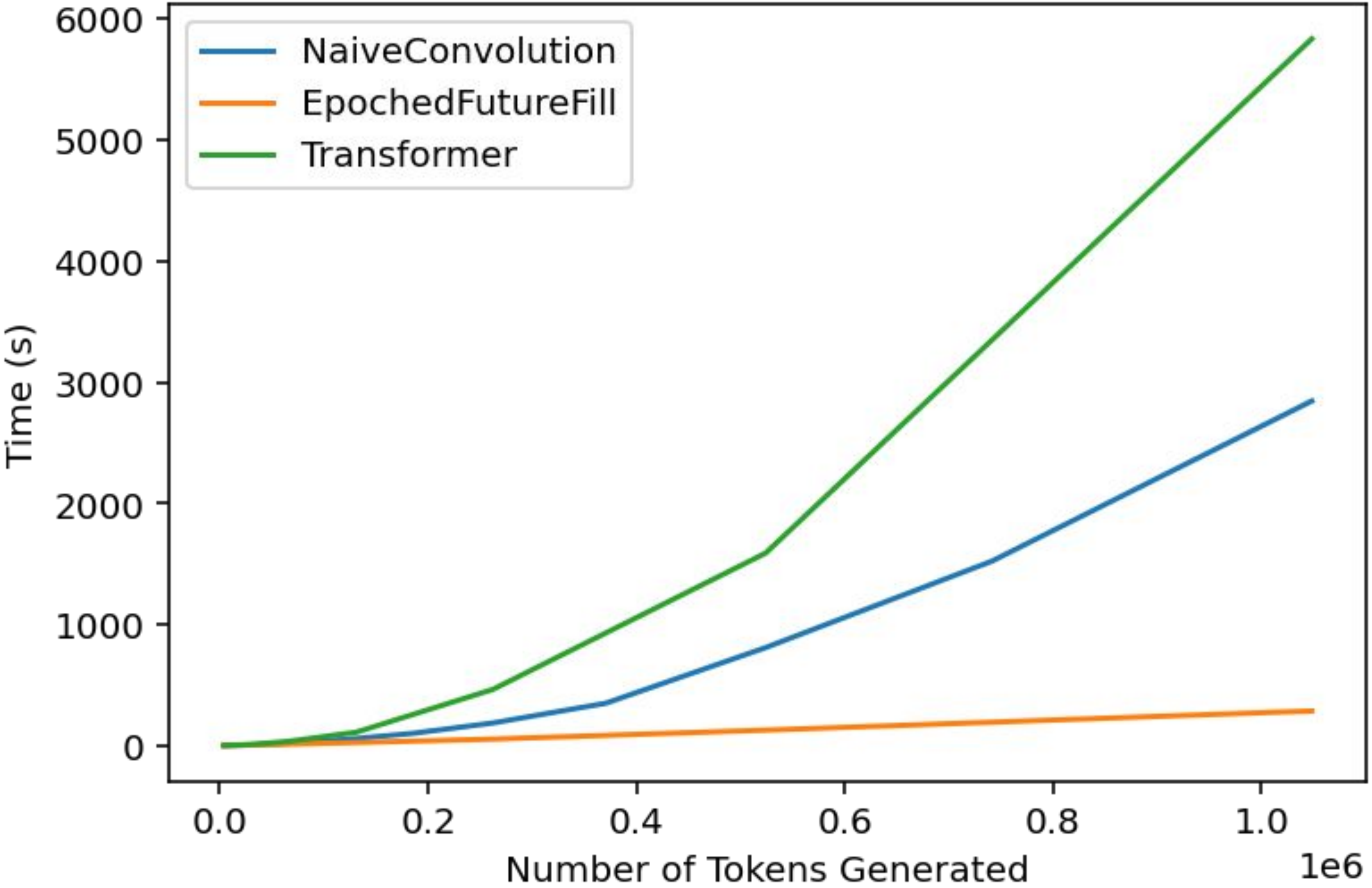
Idea can be applied recursively to achieve **O(log² L)** compute



Compute Recursively

Compute Recursively

Future Fill via FFT


NaiveConvolution
EpochedFutureFill
Transformer
Time (s)
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Number of Tokens Generated

Fast Inference

Generating 1 token:

- **Transformers - O(prompt length + generation length)** - memory & compute
- **RNNs/SSMs - O(1) -** memory & compute
- **Flash STU - O(k log L) ≈ O(log^2 L)** - compute
- **Ongoing: Flash Distilled STU - O(1)** - compute

- **After training a Flash STU (Language) model, we distill the STU layers into LDS layers**
    - **O(1) token generation**
    - **Allows for (very) fast language models**

Distilling STU layers into LDS layers

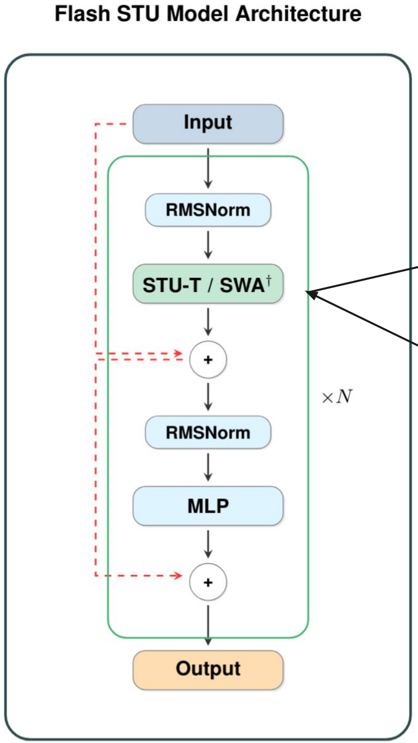
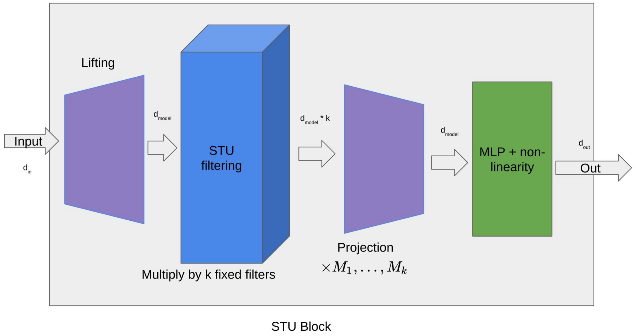


Figure 12: Flash STU Model Architecture, alternating between STU-T and (sliding window) attention†.



$$x_t = Ax_{t-1} + Bu_t$$
$$y_t = Cx_t + Du_t$$

But isn't learning the LDS layers hard?

Learning optimal weights for factors of *A, B, C, D* is **hard** (non-convex)

$$y_t^{\mathrm{LDS}} = (CB + D)u_t + CABu_{t-1} + CA^2Bu_{t-2} + CA^3Bu_{t-3} + \ldots$$

Learning optimal weights for relaxed parameterizations is **easy** (convex)

$$\hat{y}_t = M_0 u_t + M_1 u_{t-1} + M_2 u_{t-2} + \ldots$$

Realizations

In practice, fitting an LDS becomes feasible when we learn the signal from an STU.

We hypothesize the STU denoises the signal and, because it represents similar functions as an LDS, changes the signal to something an LDS can learn. **(Ongoing)**

However, this is a slight oversimplification – despite trained STUs being learnable by an LDS, a randomly initialized STU cannot be learned by an LDS.

What makes the STUs trained from data “special”? **(Ongoing)**

Recap

- STU, a **new deep neural network architecture!**
- Subquadratic time complexity
- Can provably learn symmetric marginally stable LDS
  - Can still learn more difficult settings in practice
- FutureFill - subquadratic inference speed, memory
- Provable length generalization (new!)
- STU to LDS Distillation - O(1) Language Models
- Robust to hyperparameter changes, “just works”
- Can scale up all the way to LLM size

Thanks!

Motivation
Spectral filtering
STU
Experiments
Current work