Structured State Space Models (S4) for Long Sequence Modeling

Your Name

April 6, 2025

Abstract

This report presents an overview of Structured State Space Models (S4), a novel approach for efficiently modeling long-range dependencies in sequential data. We focus on the S4 architecture's key innovations, including its diagonal-plus-low-rank parameterization and HiPPO initialization, which enable linear-time complexity while maintaining state-of-the-art performance on benchmarks like Long Range Arena (LRA) and sequential CIFAR-10.

1 Introduction

Structured State Space Models (SSMs) address the fundamental challenge of modeling long-range dependencies in sequences. Traditional approaches like RNNs, CNNs, and Transformers struggle with sequences exceeding 10,000 steps. The S4 model [**?**] introduces:

- HiPPO-LegS initialization for long-term memorization
- Diagonal-plus-low-rank (DPLR) parameterization
- FFT-based convolutional computation

2 Technical Approach

2.1 State Space Model Fundamentals

The continuous SSM is defined by:

$$
\begin{aligned}
x'(t) &= \mathbf{A}x(t) + \mathbf{B}u(t) \\
y(t) &= \mathbf{C}x(t) + \mathbf{D}u(t)
\end{aligned}
$$

Discretized using bilinear transform:

$$\overline{\mathbf{A}} = (\mathbf{I} - \Delta/2\mathbf{A})^{-1}(\mathbf{I} + \Delta/2\mathbf{A})$$

2.2 S4 Innovations

Key components of S4:

- **HiPPO Initialization**: Specialized $\mathbf{A}$ matrices for continuous-time memorization
- **DPLR Parameterization**: $\mathbf{A} = \mathbf{\Lambda} - \mathbf{P}\mathbf{Q}^*$
- **FFT Acceleration**: Convolution via Cauchy kernel computation

Algorithm 1 S4 Convolution Kernel Computation

1: Compute Vandermonde product $\mathbf{K} = \mathbf{C}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$
2: Apply Woodbury identity for low-rank correction
3: Compute FFT-based convolution using Cauchy kernel

3 Implementation

PyTorch implementation highlights:

```
class S4Layer(nn.Module):
    def __init__(self, d_model, n, l_max):
        self.A = nn.Parameter(hippo_init(n))  # HiPPO-LegS
        self.B, self.C = nn.Parameter(...)     # Low-rank factors

    def forward(self, x):
        # FFT-based convolution
        K = compute_cauchy_kernel(self.A, self.B, self.C)
        return fft_conv(x, K)
```

4 Experimental Results

S4 demonstrates strong performance across benchmarks:

Task	S4 Accuracy	Previous Best
LRA Path-X (16k)	88.0%	50.0%
sCIFAR-10	91.1%	84.7%
Raw Speech (SC10)	98.3%	96.3%

Table 1: Performance comparison on long-range tasks

5 Conclusion

S4 establishes new state-of-the-art results while maintaining $O(N \log N)$ complexity. The combination of theory-guided initialization and computational optimizations makes it a promising foundation for general sequence modeling.

References