Designing Sequence Models with Wavelets and Multiresolution Convolutions

Jiaxin Shi
2024/3/26 @ FIMI

jiaxins.io

Joint work with Alex Wang, Emily Fox

Outline

- Motivation
- Sequence Modeling: Background and Related Work
- Our Model: MultiresConv
- Results and Conclusions

The Generative Model Era

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

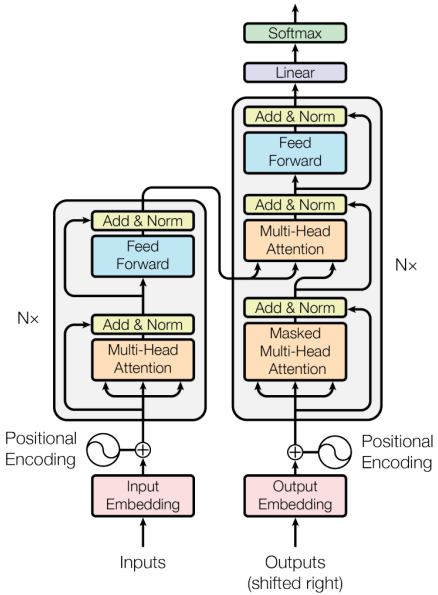
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

[Wei et al.]

[Brooks, Peebles et al.]

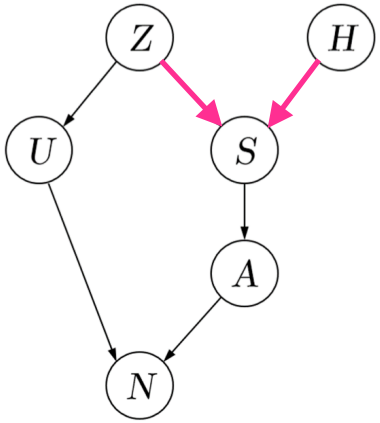


[Vaswani et al.]

"Generative Model" = Statistical Modeling + Neural Network Architecture

Why Algorithmic Modeling Research

- Traditional probabilistic ML focuses too much on structure of independence.
- To build powerful statistical models, we need algorithmic structure (architecture) that captures complex dependencies.
- Current algorithmic model development mostly by trial and error

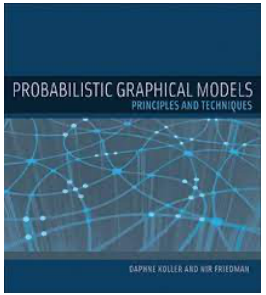


Statistical Science
2001, Vol. 16, No. 3, 199–231

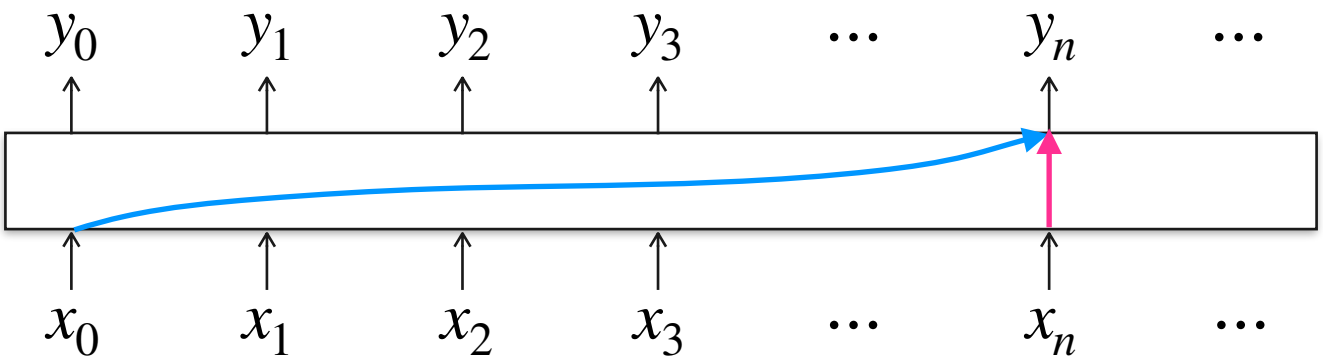
Statistical Modeling: The Two Cultures

Leo Breiman

Abstract. There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commit-



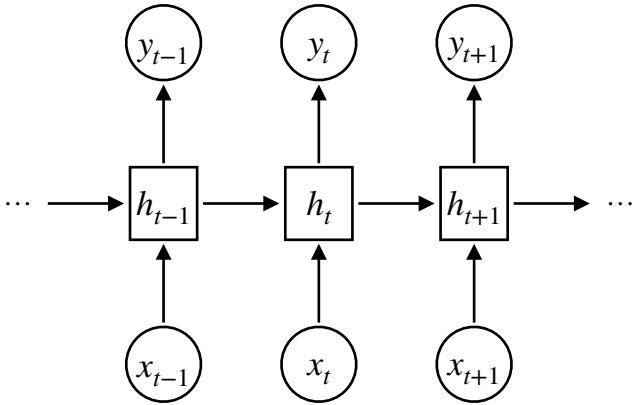
Sequence Modeling Setup



- Simple abstraction but very general
- Key challenge: A sequence model should capture potentially long range dependencies.

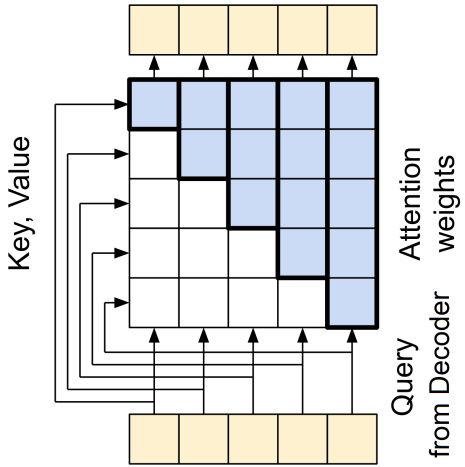
Capturing Long Range Dependencies

RNNs: Keep a memory of the past



Complicated sequential dependencies in computation

Transformers: Brute force enumeration



https://pylessons.com/transformer-attention

Attention is All You Need?

Pros

- Numerous empirical successes
- Suited for massively parallel computation

Cons

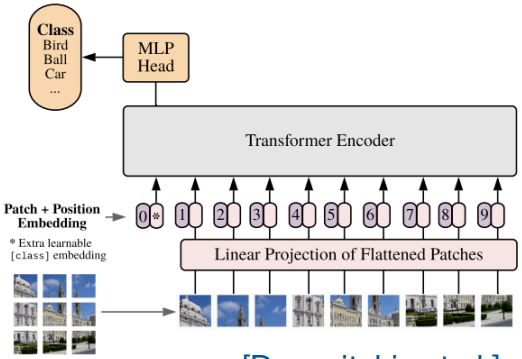
- Quadratic complexity in length—limiting further scaling
- Needs ad-hoc fix for dense input: audio, image, video, long text

https://www.isattentionallyouneed.com/

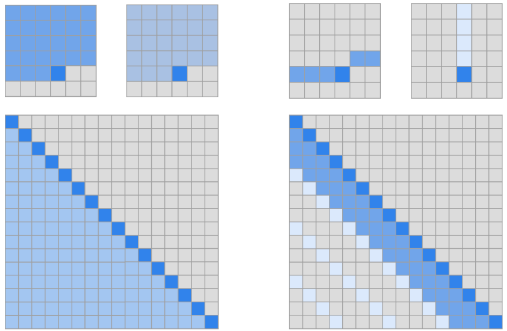
Is Attention All You Need?

Current Status: Yes

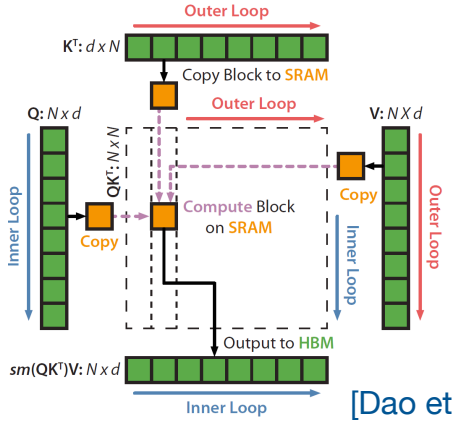
Time Remaining: 1127d 9h 17m 6s



[Dosovitskiy et al.]



[Child et al.]



[Dao et al.]

Resurgence of RNNs

$$x_k = \overline{A} x_{k-1} + \overline{B} u_k$$

$$y_k = \overline{C} x_k$$

S4: Explicitly Unroll Linear RNNs
→ Convolutions

$$\overline{K} \in \mathbb{R}^L := (\overline{CB}, \overline{CAB}, \ldots, \overline{CA}^{L-1}\overline{B})$$

$$y = \overline{K} * u$$

Model	Accuracy (%)
Attention:	
Transformer (Trinh et al., 2018)	62.2
RNN:	
LSTM (Hochreiter & Schmidhuber, 1997)	63.01
r-LSTM (Trinh et al., 2018)	72.2
UR-GRU (Gu et al., 2020b)	74.4
HiPPO-RNN (Gu et al., 2020a)	61.1
LipschitzRNN (Erichson et al., 2021)	64.2
State Space Models:	
S4 (Gu et al., 2021)	91.80
S4D (Gu et al., 2022)	90.69
S5 (Smith et al., 2023)	90.10
Liquid-S4 (Hasani et al., 2022)	92.02

Gu et al. (2021). Efficiently modeling long sequences with structured state spaces.

Resurgence of RNNs

Why convolutions are great:

- Avoid sequential computations, highly parallelizable
- Scale sub-quadratically with length

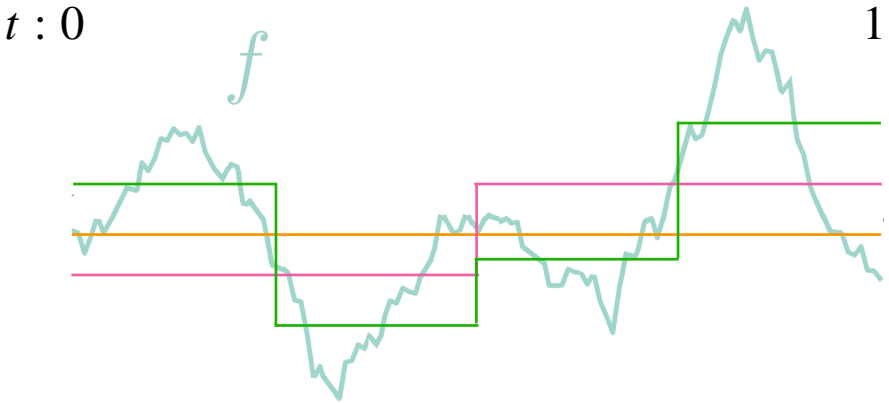
Weaknesses of state space-unrolled convolution (e.g., Gu et al., 22, Smith et al. 22, Li et al., 23):

- Kernel as long as input
- Requires complex parameterization schemes and FFT (or parallel scan)
- Specialized initialization such as HiPPO

Gu et al. (2021). Efficiently modeling long sequences with structured state spaces.
Li et al. (2022). What makes convolutional models great on long sequence modeling?
Smith et al. (2022). Simplified state space layers for sequence modeling.

Wavelets and Multiresolution Analysis

$$\phi(t) = \mathbf{1}(0 \le t < 1)$$



$$\hat{f}^{(0)}(t) = a_{00} \cdot \phi(t)$$

$$\hat{f}^{(1)}(t) = a_{10} \cdot 2^{1/2}\phi(2t) + a_{11} \cdot 2^{1/2}\phi(2t-1)$$

$$\hat{f}^{(2)}(t) = a_{20} \cdot 2\phi(4t) + a_{21} \cdot 2\phi(4t-1) + a_{22} \cdot 2\phi(4t-2) + a_{23} \cdot 2\phi(4t-3)$$

Wavelets and Multiresolution Analysis

$$\phi(t) = \mathbf{1}(0 \le t < 1)$$

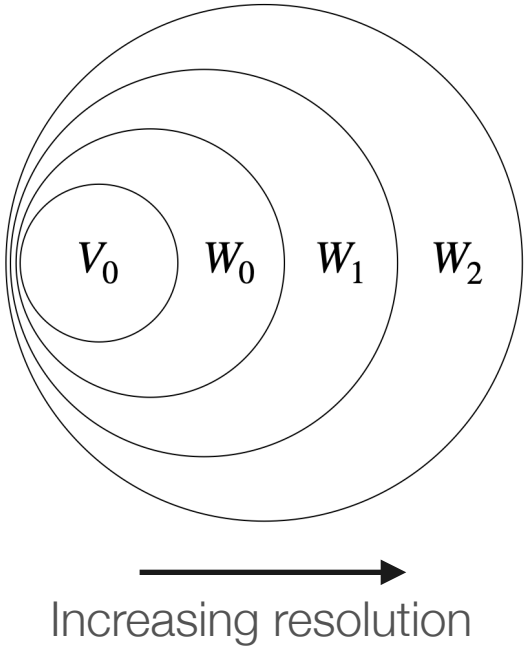
$$\hat{f}^{(j)}(t) = \sum_{k \in \mathbb{Z}} a_{j,k} \phi_{j,k}(t),$$

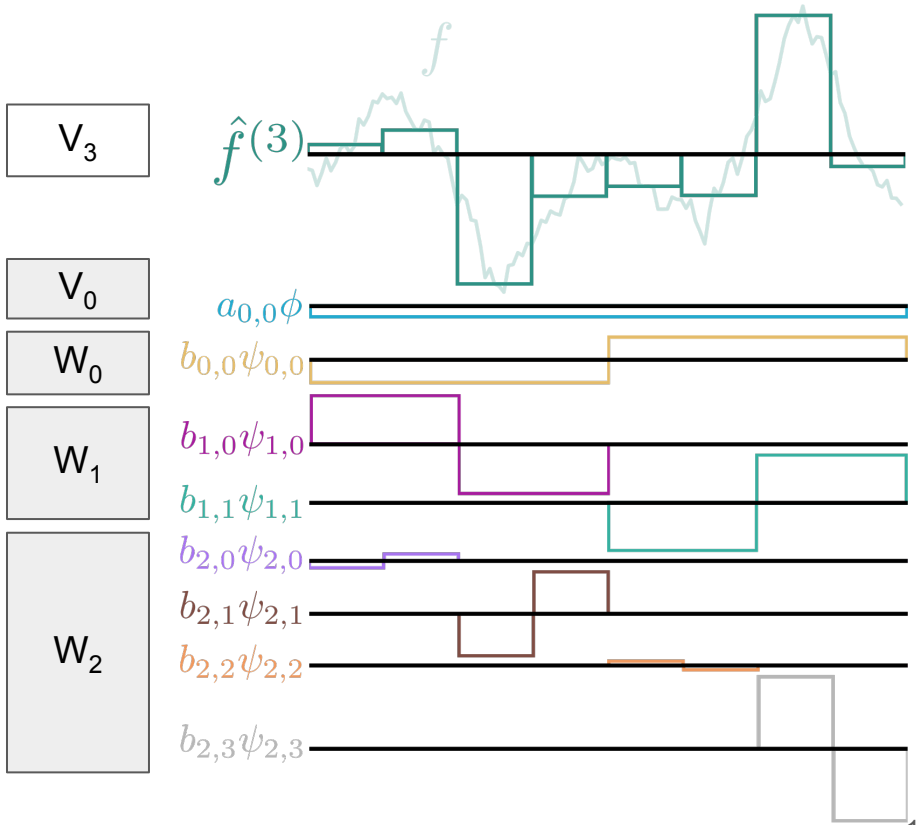
$$a_{j,k} = \langle f, \phi_{j,k} \rangle$$

- Represent f by the vector $\{a_{j,k}\}_{k \in \mathbb{Z}}$ for a sufficiently large j
- Problem: Each feature $a_{j,k}$ may be too local to be representative

Wavelets and Multiresolution Analysis

$$V_3 = W_2 \oplus V_2 = W_2 \oplus W_1 \oplus W_0 \oplus V_0$$





Wavelets and Multiresolution Analysis

A multiresolution representation of $f(t)$ can be computed as:

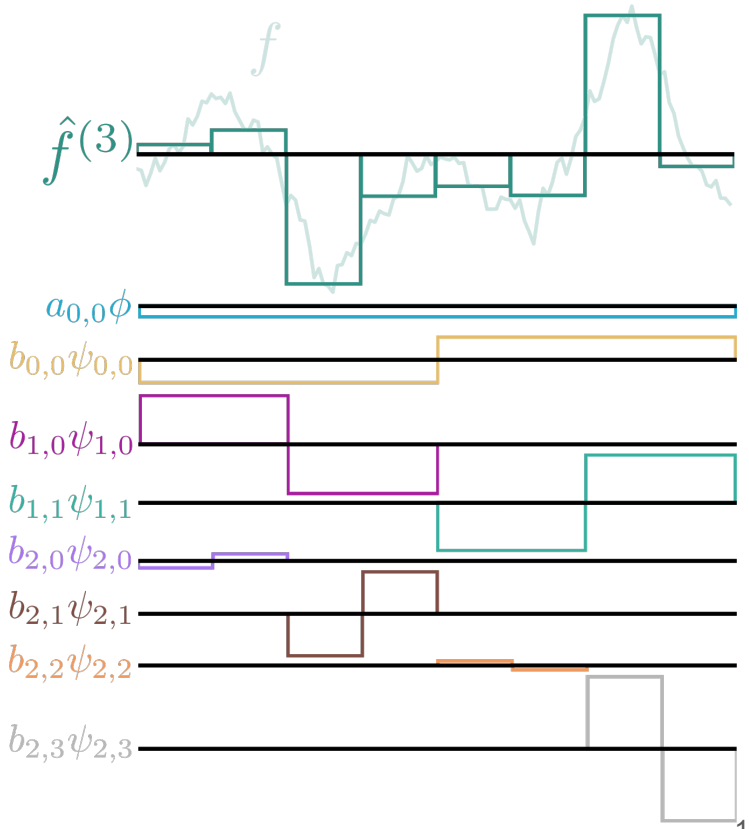
$$f^{(J)}(t) = \sum_{k\in\mathbb{Z}} a_{0k}\phi_{0,k}(t) + \sum_{j=0}^{J-1}\sum_{k\in\mathbb{Z}} b_{j,k}\psi_{j,k}(t)$$

where

$$\mathbf{a}_j(n) \triangleq a_{j,n} = \sum_{k=0}^{K-1} \mathbf{a}_{j+1}(2n+k)\mathbf{h}_0(k),$$

$$\mathbf{b}_j(n) \triangleq b_{j,n} = \sum_{k=0}^{K-1} \mathbf{a}_{j+1}(2n+k)\mathbf{h}_1(k)\,.$$

Discrete Wavelet Transform (DWT)



A Wavelet Memory

- Memorize the history up to time t

$$\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t = \mathrm{DWT}(x(0:t), \mathbf{h}_0, \mathbf{h}_1)$$

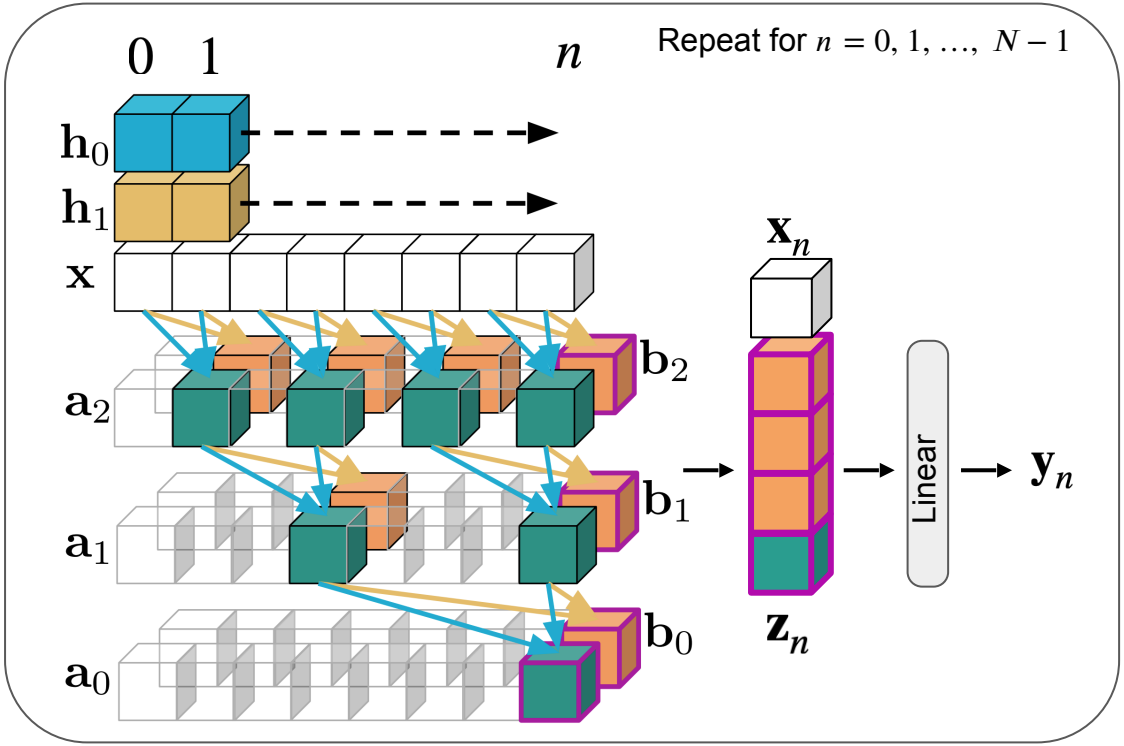
- Forgetting: Drop fine details in the past

$$\mathbf{b}_{j,<f(j)}^t = 0$$

- Retrieve data from a past time s

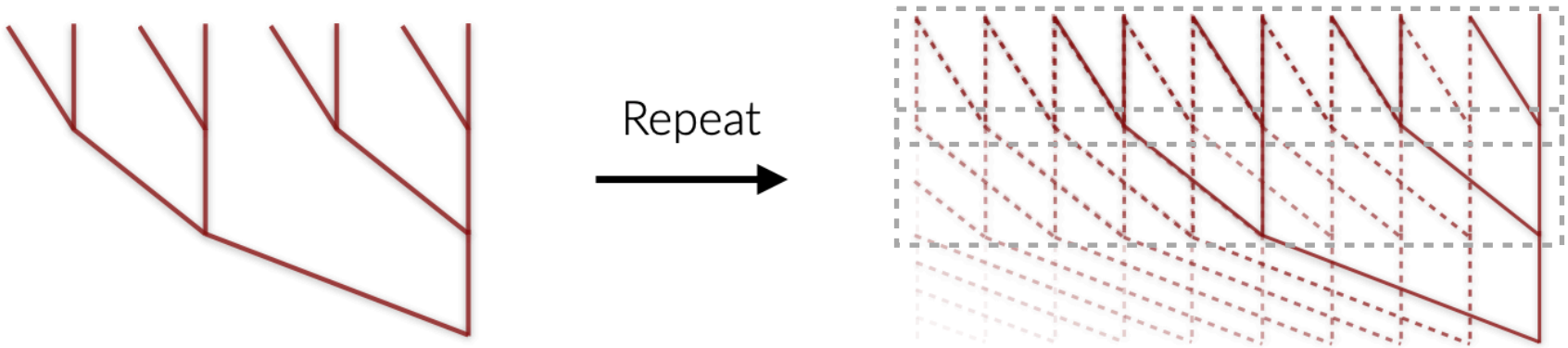
$$\hat{x}(s) = \langle \mathbf{w}, (\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t) \rangle \quad \text{where} \quad \mathbf{w} = (\phi(s), \psi_{0,0}(s), \psi_{1,0}(s), \psi_{1,1}(s), \cdots)$$

Multiresolution Convolutional Memory



- The output ($\boldsymbol{y}_n$) at step n is generated from a multiresolution convolutional memory $\boldsymbol{z}_n$.
- Architectures with larger kernel sizes can be derived from Daubechies wavelets.

Fast Implementation as Causal Dilated Convolutions

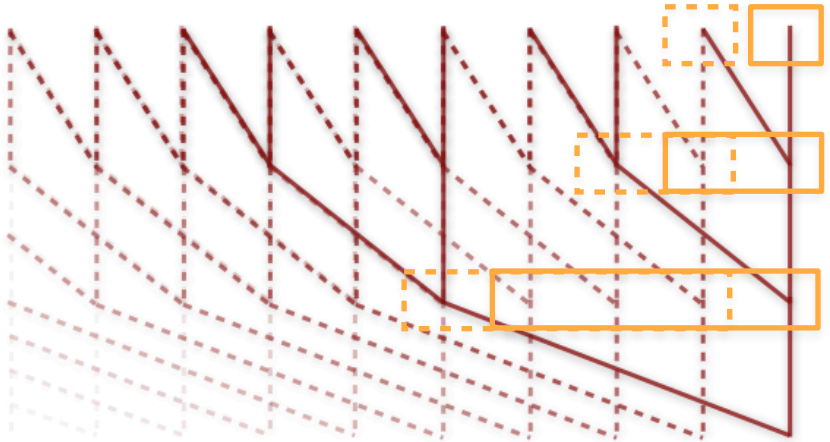


Complexity: Fixed depth— $O(N)$ At most— $O(N \log N)$

- Principled justification of WaveNet*-style causal dilated convolutions
- **Differences** (suggested improvements)
  - Explicit memory construction
  - Filter sharing for all timescales
  - No nonlinearities across timescales

*van den Oord et al. (2016). WaveNet: A generative model for raw audio

$O(1)$ Autoregressive Sampling



At each scale, maintain a queue of size (kernel size - 1) x dilation

Deep Learning with MultiresConv

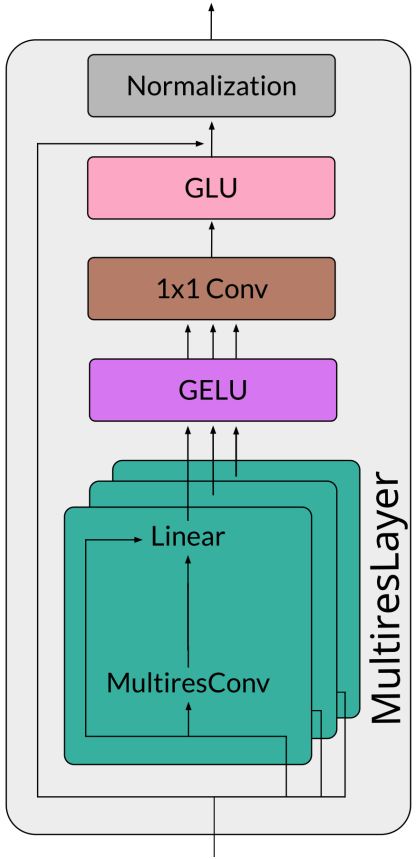
Our layer is linear and parameter efficient, thus straightforward to integrate into residual blocks and stack into deep architectures.

Model		Accuracy (%)
Attention:		
Transformer (Trinh et al., 2018)		62.2
RNN:		
LSTM (Hochreiter & Schmidhuber, 1997)		63.01
r-LSTM (Trinh et al., 2018)		72.2
UR-GRU (Gu et al., 2020b)		74.4
HiPPO-RNN (Gu et al., 2020a)		61.1
LipschitzRNN (Erichson et al., 2021)		64.2
State Space Models:		
S4 (Gu et al., 2021)	7.9M Params	91.80
S4D (Gu et al., 2022)		90.69
S5 (Smith et al., 2023)		90.10
Liquid-S4 (Hasani et al., 2022)		92.02
Convolution:		
TrellisNet (Bai et al., 2019)		73.42
CKConv (Romero et al., 2022b)		63.74
FlexConv (Romero et al., 2022a)		80.82
MULTIRESNET (Ours)	1.4M Params	**93.15**

Better than a 5x larger S4

Outperforming prior ConvNets by >10pps

Benchmark: Pixel-level Image Classification (Sequential CIFAR-10)



Ablations on Cross-Layer Filter Sharing

	Tied	Untied
SCIFAR	93.15%	92.16%
Long ListOps	62.75%	61.85%

More Results

Syntax Reasoning (Long ListOps)

First competitive small-kernel ConvNets*

Model	Accuracy (%)
Attention:	
Local Attention (Tay et al., 2021)	15.82
Linear Trans. (Katharopoulos et al., 2020)	16.13
Linformer (Wang et al., 2020)	16.13
Sparse Transformer (Child et al., 2019)	17.07
Performer (Choromanski et al.)	18.01
Transformer (Vaswani et al., 2017)	36.37
Sinkhorn Transformer (Tay et al., 2020)	33.67
FNet (Lee-Thorp et al., 2022)	35.33
Longformer (Beltagy et al., 2020)	35.63
BigBird (Zaheer et al., 2020)	36.05
Nyströmformer (Xiong et al., 2021)	37.15
Luna-256 (Ma et al., 2021)	37.25
Reformer (Kitaev et al., 2020)	37.27
H-Transformer-1D (Zhu & Soricut, 2021)	49.53
State Space Models:	
S4 (Gu et al., 2022)	59.60
DSS (Gupta et al., 2022)	57.6
S4D (Gu et al., 2022)	60.52
S5 (Smith et al., 2023)	62.15
Liquid-S4 (Hasani et al., 2022)	**62.75**
Convolution:	
CDIL (Cheng et al., 2023)	44.05
SGConv* (Li et al., 2022)	61.45
MULTIRESNET (Ours)	**62.75**

Autoregressive Generative Modeling (CIFAR-10)

The best method without 2D bias

Model	#params	Test bpd.
RNN + 2D bias:		
PixelRNN (Oord et al., 2016c)		3.00
2D Convolution:		
PixelCNN (Oord et al., 2016c)		3.14
Gated PixelCNN (Oord et al., 2016b)		3.03
PixelCNN++ (Salimans et al., 2017)	53M	2.92
2D Convolution + Attention:		
PixelSNAIL (Chen et al., 2018)	46M	2.85
1D Attention:		
Image Trans. (Trinh et al., 2018)		2.90
2D Attention:		
Sparse Trans. (Child et al., 2019)	59M	**2.80**
State-Space Models + U-Net structure:		
S4 (Gu et al., 2021)		2.85
Convolution (no 2D bias):		
MULTIRESNET (Ours)	38M	2.84

*State-space models (S4, SGConv, etc.) can be equivalently represented as convolutions with an input-length kernel. However, these models rely on sophisticated parameterization & initialization techniques.

Implementation in 15 Lines of Code

```
def multires_layer(x, h0, h1, w, depth=None):
    if depth is None:
        depth = math.ceil(math.log2((x.shape[-1] - 1) / (self.h0.shape[-1] - 1) + 1))
    y = 0.
    a = x
    dilation = 1
    for i in range(depth, 0, -1):
        padding = dilation * (kernel_size - 1)
        a = pad(a, (padding, 0), "constant", 0)
        b = conv1d(a, h1, dilation=dilation, groups=x.shape[1])
        a = conv1d(a, h0, dilation=dilation, groups=x.shape[1])
        y += w[:, i:i+1] * b
        dilation *= 2
    y += w[:, :1] * a
    y += w[:, -1:] * x
    return y
```

Takeways

- It's possible to "derive" a performant neural network architecture.
- A new **sequence modeling layer** grounded in wavelet theory.
- **Simple** and **parameter efficient**
  - Implemented with small-kernel convolutions
  - No specialised init or parameterization schemes, nor FFT
- **Strong performance**: Remains SOTA on long sequence modeling benchmark

Future Work

- Is the layer a universal sequence function approximator?
- MultiresConv for 2D, 3D and general discrete topologies
- Continuous MultiresConv
- Large-scale MultiresConv generative models

Thanks

Paper: Sequence Modeling with Multiresolution Convolutional Memory

Code: github.com/thjashin/multires-conv

Contact: `ishijiaxin@gmail.com`

Collaborators:

Alex Wang

Emily Fox

References

Shi et al. (2023). Sequence modeling with multiresolution convolutional memory.
Gu et al. (2021). Efficiently modeling long sequences with structured state spaces.
Li et al. (2022). What makes convolutional models great on long sequence modeling?
Smith et al. (2022). Simplified state space layers for sequence modeling.
van den Oord et al. (2016). WaveNet: A generative model for raw audio.
Dosovitskiy et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.
Child et al. (2019). Generating long sequences with sparse transformers.
Dao et al. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness.