Sequence Modeling with Multiresolution Convolutional Memory

Jiaxin Shi [1] **Ke Alexander Wang** [1] **Emily B. Fox** [1 2]

Abstract

Efficiently capturing the long-range patterns in sequential data sources salient to a given task—such as classification and generative modeling—poses a fundamental challenge. Popular approaches in the space tradeoff between the memory burden of brute-force enumeration and comparison, as in transformers, the computational burden of complicated sequential dependencies, as in recurrent neural networks, or the parameter burden of convolutional networks with many or large filters. We instead take inspiration from wavelet-based multiresolution analysis to define a new building block for sequence modeling, which we call a MULTIRESLAYER. The key component of our model is the *multiresolution convolution*, capturing multiscale trends in the input sequence. Our MULTIRESCONV can be implemented with *shared* filters across a dilated causal convolution tree. Thus it garners the computational advantages of convolutional networks and the principled theoretical motivation of wavelet decompositions. Our MULTIRESLAYER is straightforward to implement, requires significantly fewer parameters, and maintains at most a $\mathcal{O}(N \log N)$ memory footprint for a length N sequence. Yet, by stacking such layers, our model yields state-of-the-art performance on a number of sequence classification and autoregressive density estimation tasks using CIFAR-10, ListOps, and PTB-XL datasets.

1. Introduction

A key challenge in sequence modeling is summarizing, or *memorizing*, long-term patterns in data informative for a particular task, such as classification, forecasting, or clustering. By definition, patterns are higher-level structures in the data that arise from multiple timesteps. However, patterns can occur at multiple levels, corresponding to different timescales. For example, in studying energy consumption, patterned variations may occur within a day, between days, and quarterly. Similar salient multiscale trends appear in physiological time series such as dysfunctional glucose patterns in diabetic patients and anomalous heart beats in arrhythmic patients. Audio signals of speech may be described in terms of utterances, phonemes, and subphonemes. And, the multiscale structure of images and video has been well-studied. Even for data sources without an explicit multiscale interpretation, multiscale modeling approaches can provide an efficient mechanism for capturing long-range patterns.

In this paper, we propose a general and reusable building block for sequence modeling—MULTIRESLAYER—leveraging a multiscale approach to memorize past data. We view memory through the lens of multiresolution analysis (MRA) (Willsky, 2002), with a particular emphasis on wavelet analysis, a powerful tool from signal processing for compression, denoising, feature extraction, and more (Jawerth & Sweldens, 1994; Akansu et al., 2001). As discussed in Sec. 2, wavelet analyses can be computed in a computationally efficient manner and interpreted as a series of convolutions. However, our use of wavelets is a design choice and other MRA techniques could likewise be considered for memorizing patterns at different timescales.

Taking inspiration from wavelets, the key component of MULTIRESLAYER is a *multiresolution convolution* operation (MULTIRESCONV) that retains the overall tree-structure of MRA. We show that constructing a memory of the past at each timestep of the sequence using MULTIRESCONV can be collectively implemented as a stack of carefully-placed dilated causal convolutions with filters shared between levelsW. In contrast to traditional wavelet analysis, however, we *learn* the filters and do so end-to-end. When we fix the filters to pre-defined wavelet filters, the MULTIRESCONV reduces to a traditional discrete wavelet transform, though we show the benefits of learning the filters in Sec. 5.5. The basic MULTIRESCONV building block can be stacked in a multitude of ways that are common in deep learning models (e.g., across multiple channels, vertically as multiple layers, etc.). Our model resembles WaveNet (Oord et al., 2016a) in the use of tree-structured dilated convolutions. However, our principle-guided design has distinct skip-connection structures and filter sharing patterns, resulting in significantly

[1]Stanford University [2]CZ Biohub SF. Correspondence to: Jiaxin Shi <jiaxins@stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

better parameter efficiency and performance (see Sec. 4 for further details).

There is a rapidly growing literature on machine learning for sequence modeling. Popular classes of approaches include variants of recurrent networks (Hochreiter & Schmidhuber, 1997), self-attention networks (Vaswani et al., 2017), and state-space models (Gu et al., 2021). See Sec. 4 for further discussion. Our MULTIRESLAYER has key advantages over this body of past work:

- **Architecture simplicity:** The workhorse of our layer is simple dilated convolutions and linear transforms.
- **Efficient training:** Our layer parallelizes easily across hardware accelerators implementing convolutions.
- **Parameter efficiency:** Our layer reuses filters across the stack of depthwise dilated convolutions.

Likewise, by leveraging an MRA structure, we start from a principled and interpretable framework for thinking about memory in sequence modeling. Furthermore, we can lean on the vast MRA literature for modeling generalizations, such as shift-invariant wavelet transforms (Kingsbury, 1998; Selesnick et al., 2005) for shift-invariant representation learning, scaling to multiple input dimensions, etc.

Our empirical evaluation covers sequential image classification and autoregressive generative modeling (CIFAR-10), reasoning on syntax trees (ListOps), and multi-label classification of electrocardiogram (PTB-XL). We also note that our proposed MULTIRESCONVs can readily be applied and extended to other tasks such as representation learning and long-term forecasting. Likewise, although we focus on sequence analysis, the ideas we propose generalize to other data domains with multiresolution structure, such as images and videos. Exploring the application of MULTIRESLAYER in these settings is an exciting future direction.

2. Background: Wavelet Decompositions

In contrast to the frequency-domain analysis of Fourier transforms, *wavelets* provide a *time–frequency* analysis. In particular, wavelets are a finite-support basis with a multiresolution structure, i.e., basis functions are divided into groups with different resolutions—some focus on "local" function values at very short timescales, while others capture more "global" structures at longer timescales. In the following, we explain the idea of wavelet MRA with the simplest wavelet family—*Haar wavelets*. A formal treatment covering all orthogonal wavelets is in Appendix A.

Suppose we want to approximate a signal $f(t)$ over the time interval $[0, 1)$. The roughest approximation we can produce is $\hat{f}^{(0)}(t) \triangleq a_{0,0}\phi(t)$ where $\phi(t) = \mathbf{1}(0 \leq t < 1)$ and $a_{0,0} = \int_0^1 f(t)dt$ is the average value of f. We use superscript 0 to indicate that this is the lowest resolution

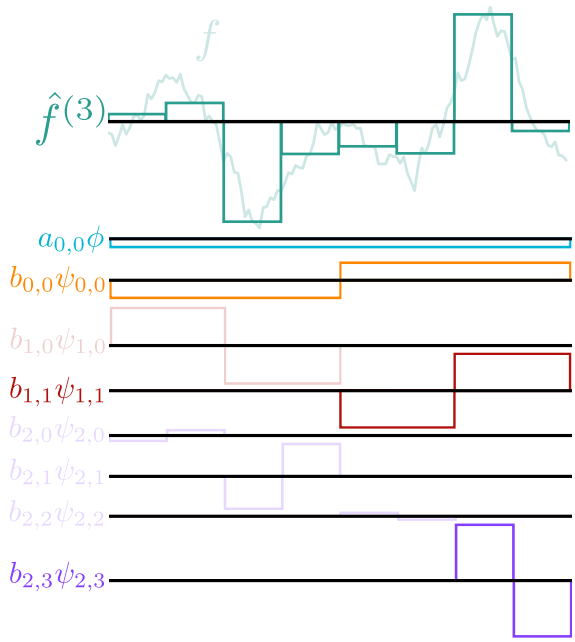

Figure 1. In standard MRA, we approximate the continuous signal f with $\hat{f}^{(j)}$. Here, we visualize $\hat{f}^{(3)}$ and its decomposition into a sum of functions that capture structures of increasing resolution over a binary-tree-partitioned input space, corresponding to Eq. (1). Components belonging to the same level have the same color. The non-transparent components of each resolution level retain the most recent information in the decomposition. Retaining only these components corresponds to our MULTIRESLAYER with a "resolution fading" TREESELECT.

approximation of f we make. We can better approximate f by dividing the unit interval in two and approximating f as: $f(t) \approx \hat{f}^{(1)}(t) \triangleq a_{1,0}\phi(2t) + a_{1,1}\phi(2t-1)$ where $a_{1,0} = \int_0^{1/2} f(t)dt$ and $a_{1,1} = \int_{1/2}^1 f(t)dt$.

We can repeat this procedure of halving the intervals, rescaling, and translating ϕ, to get finer approximations $\{\hat{f}^{(j)}\}_{j\in\mathbb{N}_0}$. Each $\hat{f}^{(j)}$ is a linear combinations of compactly supported basis functions, $\{\phi_{j,k}(t) \triangleq 2^{j/2}\phi(2^j t - k)\}_{k\in\mathbb{Z}}$, with their resolution levels indexed by j:

$$\hat{f}^{(j)}(t) = \sum_{k\in\mathbb{Z}} a_{j,k}\phi_{j,k}(t), \quad \text{where } a_{j,k} = \langle f, \phi_{j,k}\rangle.$$

For each level $j \in \mathbb{N}_0$, the subspace $V_j \triangleq \operatorname{span}(\{\phi_{j,k}\}_{k\in\mathbb{Z}})$ contains functions that are constant over intervals of length $1/2^j$. In other words, *basis functions in V_j describe structures in f no larger than the timescale of $\Delta t \sim 1/2^j$*. For sufficiently large j, V_j has the capacity to approximate any continuous time series arbitrarily well.

One may try to summarize or represent f by collecting the coefficients $\{a_{j,k}\}_{k\in\mathbb{Z}}$ into a vector. Though the coefficients altogether fully describe the approximation $\hat{f}^{(j)}$, each individual coefficient alone may be too local to be representative of structures in f. Each $a_{j,k}$ only summarizes the value of f

within a $1/2^j$ interval, while patterns may occur over larger intervals. We would need multiple $a_{j,k}$ to summarize these larger-scale structures. Is there a way to produce coefficients each of which summarizes a structure at a different scale?

Representing structure at disjoint resolutions. We can indeed produce this kind of representation by using tools from MRA. In MRA, we repeatedly decompose V_j into the sum of a lower-resolution subspace V_{j-1} and its orthogonal complement W_{j-1}: $V_j = V_{j-1} \oplus W_{j-1}$. Since basis functions in V_j and V_{j-1} describe structures at scales coarser than $\Delta t \sim 1/2^j$ and $\Delta t \sim 1/2^{j-1}$, respectively, basis functions in W_{j-1} represent structures *exactly at* the $1/2^j$ scale, summarized by the basis coefficients $\{b_{j,k}\}_{k\in\mathbb{Z}}$. Starting from some high-resolution level J and repeating this process, we have

$$V_J = V_{J-1} \oplus W_{J-1} = V_0 \oplus W_0 \oplus \ldots \oplus W_{J-2} \oplus W_{J-1}$$

and, as visualized in Fig. 1,

$$f(t) \approx \hat{f}^{(J)}(t) = a_{0,0}\phi(t) + \sum_{j'=0}^{J-1}\sum_{k\in\mathbb{Z}} b_{j',k}\psi_{j',k}(t). \quad (1)$$

The basis functions $\{\psi_{j,k}\}$ are called Haar wavelets and ϕ is called their scaling function; see Appendix A.1 for their functional forms. The coefficients[1] $\{a_{0,0}\} \cup \{b_{0,k}\}_{k\in\mathbb{Z}} \cup \ldots \cup \{b_{J-1,k}\}_{k\in\mathbb{Z}}$ now summarize the structures of f at multiple resolutions, ranging from $1/2^0$ to $1/2^{J-1}$.

Computing the representation. Our original problem of summarizing the multiresolution structures of f then comes down to computing the wavelet basis coefficients $a_{0,0}, \{b_{j,k}\}$ of the approximation $\hat{f}^{(J)} \in V_J$. See Appendix A.1 for how to compute these coefficients for Haar wavelets. In general, we can efficiently and recursively compute these coefficients for any wavelet family using the discrete wavelet transform (DWT; see Appendix A.3).

In Appendix D, we illustrate the representational power of wavelet transforms. In particular, we consider a raw audio waveform capturing 1 second of a recording at a sampling rate of 16,384. We use a 10-level wavelet tree with a total of 2068 coefficients used to reconstruct the audio signal. The wavelet transform is able to "memorize" many of the important patterns of the audio signal over this long sequence. This representational power motivates our MULTIRESLAYER outlined in Sec. 3.

3. Sequence Modeling with Multiresolution Convolutions

We leverage the computation structure of DWT to construct a multiresolution memory for sequences. Given a sequence $\mathbf{x} \in \mathbb{R}^N$ representing a discretely sampled signal, the DWT can be implemented by the following recursive computations for $\mathbf{a}_0$ and $\mathbf{b}_{0:J-1} \triangleq (\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{J-1})$ starting from $\mathbf{a}_J(n) = \mathbf{x}(n)$[2]:

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

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

where the filters $\mathbf{h}_0, \mathbf{h}_1 \in \mathbb{R}^K$ are determined by the class of wavelets. For Haar wavelets, we have $\mathbf{h}_0 = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $\mathbf{h}_1 = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. To decouple the underlying computation from the choice of filters, we define this procedure composed of convolution and downsampling at multiple scales as the *multiresolution convolution* operation:

$$\mathbf{a}_0, \mathbf{b}_{0:J-1} = \text{MULTIRESCONV}(\mathbf{x}, \mathbf{h}_0, \mathbf{h}_1, J). \quad (2)$$

Recall, the coefficients $\{\mathbf{a}_0(n)\}$ and $\{\mathbf{b}_j(n)\}$ serve as a multiresolution representation of $\mathbf{x}$. When the filter values come from wavelets, we can perfectly reconstruct the original sequence $\mathbf{x}$ from the coefficients $\mathbf{a}_0, \mathbf{b}_{0:J-1}$ by inverting the recursive procedure. In other words, this procedure is powerful enough to give us *perfect* memory of the past, summarized by the coefficients.

Instead of setting the filters to fixed values, however, we propose to use MULTIRESCONV as a building block for sequence models by letting $\mathbf{h}_0, \mathbf{h}_1$ be learnable. Learning the filters allows us to go beyond hand-designed wavelets while still keeping the multiresolution structure in our computation. Although we may lose the ability to perfectly reconstruct the input, we will see in our experiments that we gain significant predictive improvements in return.

3.1. A general sequence modeling layer

Most competitive sequence models have the structure of mapping a sequence $\mathbf{x} \in \mathbb{R}^N$ to another sequence $\mathbf{y} \in \mathbb{R}^N$. The output at each timestep should 1) be relevant to the input taken in at that step and 2) capture potentially long-range dependencies from previous timesteps. This can be achieved by brute-force enumeration and comparison (Vaswani et al., 2017), which has quadratic complexity with respect to the input length. One can avoid quadratic scaling by maintaining a fixed-length memory of the past, as is done in recurrent models. However, a fixed-length memory will lose information over time, so we must ensure that the memory always retains the most relevant historical information.

[1] $\{a_{j,k}\}$ and $\{b_{j,k}\}$ are called the approximation coefficients and detail coefficients in signal processing. Note that some sources like Lee et al. (2019) call V_J level 0, V_{J-1} level 1, and so on.

[2] To deal with finite sequences, we pad $\mathbf{a}_{j+1}$ with zeros on the left before each iteration of the recursion, if necessary, to ensure that no elements of $\mathbf{a}_{j+1}$ are excluded.

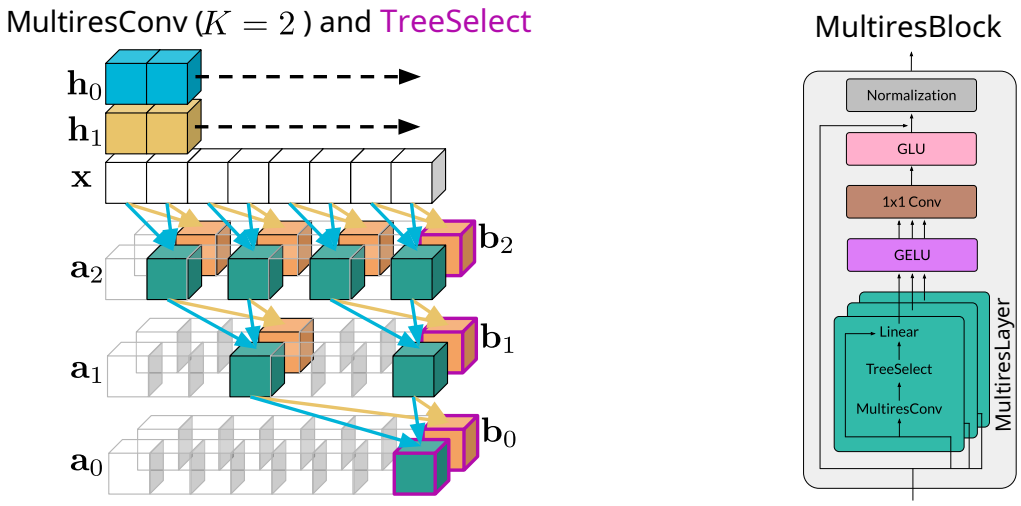

Figure 2. (Left) MULTIRESCONV consists of a sequence of dilated convolutions that share the same filters $\mathbf{h}_0, \mathbf{h}_1$ across all levels. Here we illustrate TREESELECT with the "resolution fading" strategy of keeping the right-most coefficient at each level of $\mathbf{a}_0, \mathbf{b}_{0:J-1}$, as indicated by magenta outlines. (Right) A schematic of our MULTIRESBLOCK architecture. Each channel of the input sequence is processed independently in the MULTIRESLAYER. The 1x1 convolution mixes the information across channels. We stack multiple MULTIRESBLOCKs to build our deep sequence models.

We use the MULTIRESCONV operation of Eq. (2) as a memory mechanism to capture historical patterns at multiple resolutions. The key idea is to perform MULTIRESCONV on the sequence up to the current timestep t, $\mathbf{x}(0:t)$, and *select* a relevant subset of the representation coefficients $(\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t)$ to form the memory vector $\mathbf{z}_t \in \mathbb{R}^M$ at step t:

$$\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t = \text{MULTIRESCONV}(\mathbf{x}(0:t), \mathbf{h}_0, \mathbf{h}_1, J), \quad (3)$$

$$\mathbf{z}_t = \text{TREESELECT}((\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t), M). \quad (4)$$

The t-step output $\mathbf{y}(t)$ is generated from the input at this step and the memory through a linear transform:

$$\mathbf{y}(t) = \mathbf{w}^\top \begin{bmatrix} \mathbf{z}_t \\ \mathbf{x}(t) \end{bmatrix}.$$

We repeat this procedure for $t = 0, 1, \ldots, N-1$ to get the entire output sequence $\mathbf{y} \in \mathbb{R}^N$. This architecture is schematically depicted in Fig. 2 and we discuss the coefficient selection process TREESELECT in Sec. 3.2.

Interestingly, when J is chosen to be the same for all t, the MULTIRESCONVs across time can be collectively viewed as multiple layers of dilated causal convolutions, similar to those in WaveNet (Oord et al., 2016a). Thus, all computations here can be done in time linear in the sequence length and are massively parallelizable on modern hardware that implement convolutions. By default, we set

$$J = \left\lceil \log_2 \left(\frac{N-1}{K-1} + 1 \right) \right\rceil \quad (5)$$

such that $\mathbf{a}_0^t$ has a single element $a_{0,0}^t$ and its receptive field covers the whole $x(0:t)$ for any $t \leq N$.

3.2. The memory mechanism

Ideally, we want to keep the entire sequence of $\mathbf{a}_0^t, \mathbf{b}_{0:J-1}^t$ for $t = 0, \ldots, N$. However, storing all of these coefficients requires space quadratic in N. Thus we need a way to *select* coefficients from our MULTIRESCONV tree that best summarizes the history. We call this operation TREESELECT.

We studied two strategies for determining which part of the MULTIRESCONV outputs to include in the memory:

(1) **Uniform over time:** Traditional usage of wavelets in signal compression and denoising thresholds out the time-localized, high-frequency components larger than a particular scale j. Here, we can do the same by keeping $\mathbf{a}_0^{(t)}$ and $\mathbf{b}_0^{(t)}, \ldots, \mathbf{b}_j^{(t)}$ for j below a threshold.

(2) **Resolution fading:** Another strategy is to gradually increase our approximation resolution as time approaches t. This can be achieved by keeping the entries of $\mathbf{a}_0^{(t)}$ and $\{\mathbf{b}_j^{(t)}\}$ at the highest index at each resolution level. We visualize this in Figs. 1 and 2. This approximation allows us to retain the coefficients that are the most important for reconstructing the recent input values.

Although our ablation study of Sec. 5.5 did not highlight statistically significant differences between the two strategies, we use resolution fading in all of our experiments to bias the model to focus on memorizing the most recent information and for a simpler implementation—we provide an example PyTorch implementation in Appendix B. Similar techniques of upweighting recent history proved critical in state-space models (Gu et al., 2020a; 2021).

In addition to the methods outlined above, an obvious next choice would be to dynamically select the relevant multiresolution components for the task at hand, perhaps via attention or L_1 regularization. We leave this as future work.

3.3. Multiple channels and the mixing layer

The above only describes a mapping between single-channel sequences. To deal with an input $\mathbf{x} \in \mathbb{R}^{d \times N}$ with d channels, we adopt a simple strategy of stacking d independent layers, which can be efficiently implemented using *depthwise* causal dilated convolutions. The d output sequences, after passing through a nonlinear activation function, are then mixed by a 1×1 convolution. Here, we use the GELU activation (Hendrycks & Gimpel, 2016). We note that this use of depthwise convolution followed by a mixing across channels has proven successful in a number of sequence modeling practices (Wu et al., 2018; Gu et al., 2021).

3.4. Stacking into deep models

We wrap the sequence modeling layer and the mixing layer in a residual network block (He et al., 2016) with gated linear activations (Dauphin et al., 2017) and identity skip connections (see Fig. 2). We observed that the gating mechanism improves our model performance, which is consistent with observations made in a number of other sequence models (Dauphin et al., 2017; Oord et al., 2016b;a; Gu et al., 2021). We then stack multiple residual blocks into a deep model. Depending on the task, layer normalization (Ba et al., 2016) or batch normalization (Ioffe & Szegedy, 2015) is applied after each residual block. We refer to this model as MULTIRESNET in our experiments.

3.5. Optimization and regularization

We use the Adam optimizer with default hyperparameters and decoupled weighted decay (Loshchilov & Hutter, 2018). Dropout is applied after the GELU and gated linear activation functions whenever overfitting is observed.

4. Related Work

There has been significant activity building convolutional neural networks for sequential data, inspired by applications in modeling audio (Waibel et al., 1989; Oord et al., 2016a) and natural language (Collobert et al., 2011; Kalchbrenner et al., 2016). Recent work aims to improve the performance of such models through the use of gating units (Dauphin et al., 2017), depthwise convolution (Kaiser et al., 2018), input-dependent weights (Shen et al., 2018; Wu et al., 2018), weight tying across depth (Bai et al., 2019), and adaptive filter size (Romero et al., 2022a). Our MULTIRESCONV shares many ingredients of these works, but is unique in two aspects. First, our method explicitly defines memory units $\mathbf{z}_t$, which are important for modeling long-range dependencies. Second, we have better theoretical underpinnings to our MULTIRESCONV since it collapses to the standard DWT when the filters are specified as wavelet filters.

Among all convolutional sequence models, the closest to our work is the WaveNet architecture proposed in Oord et al. (2016a). As shown in Fig. 2, when the filter size is 2, our computation graph for a_js shares the same connection pattern as WaveNet. Both models are implemented with tree-structured dilated causal convolutions. However, there are three main differences between the two models: (1) our explicit memory construction via TREESELECT creates a distinct skip-connection structure compared to WaveNet; (2) our model uses the same filters for all timescales; (3) we do not use nonlinear activation functions between convolutions at different timescales. As a result, our architecture is simpler and uses significantly fewer parameters than WaveNet. Additionally, the link we establish between wavelets and tree-structured dilated causal convolutions offers the first principled justification for the effectiveness of WaveNet in modeling raw audio waveforms, an exemplary case of lengthy sequences with multiscale structure.

Recurrent neural networks and their linear variants (e.g., state-space models) also explicitly maintain an memory of the past. Models like S4 (Gu et al., 2021) are particularly relevant since they can be implemented as convolutions thanks to the linear recurrence. However, the convolution kernel for such models is as long as the input sequence, and efficiently computing these kernels relies on sophisticated parameterization and approximation techniques. Although this issue can be circumvented by some recent advances (Gupta et al., 2022; Gu et al., 2022; Smith et al., 2022), initializing these models still requires special effort. The initialization mechanism shared by many of them, called HiPPO (Gu et al., 2020a), aims to memorize historical data via projection to orthogonal polynomials. Although this is related to wavelets (i.e., a different basis in function space), we see in Table 5 that our method is insensitive to initialization—standard initialization schemes (Glorot & Bengio, 2010) perform equally well as wavelet initialization. Some empirical motivation for the benefits of the wavelet basis over the HiPPO polynomial basis is given in Appendix D for an audio signal reconstruction task.

5. Experiments

We empirically test our model on classification and density estimation tasks that involve images, symbolic sequences, and physiological signals. Unless otherwise mentioned, we used standard Xavier uniform initialization for our MULTIRESCONV filters. All experimental details are presented in Appendix C. PyTorch code can be found in `https://github.com/thjashin/multires-conv`.

Table 1. Performance of pixel-level sequential classification on the sCIFAR dataset. Bold indicates the best performing model and underline the second best. Results are taken from either the citation or Hasani et al. (2022).

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., 2022)	90.10
Liquid-S4 (Hasani et al., 2022)	<u>92.02</u>
Convolution:	
TrellisNet (Bai et al., 2019)	73.42
CKConv (Romero et al., 2022b)	63.74
FlexConv (Romero et al., 2022a)	80.82
MULTIRESNET (Ours)	**93.15**

5.1. Pixel-level sequential image classification

We first consider image classification tasks where images are treated as a 1D sequence of pixels. The models are not allowed to use any 2D bias from the image. Therefore, the model must be able to capture patterns at multiple different timescales, including pixels that are near in the original image but far from each other in its sequence representation.

We evaluate our model on the Sequential CIFAR-10 dataset, which has long been used as a standard benchmark for modeling long-range dependencies in RNNs. We use the standard train and test split of the CIFAR-10 dataset and leave out 10% of the training set as the validation set. For the classification task, we perform a mean-pooling of all timesteps on the output sequences ($\mathbf{y}_{[\text{batchsize},256,N]} \rightarrow \overline{\mathbf{y}}_{[\text{batchsize},256]}$) and pass the result into a fully-connected layer to generate the class logits. We report the test accuracy from the model that has the highest validation accuracy.

The results are reported in Table 1. As we see, our model yields state-of-the-art performance (best test accuracy) on this benchmark sequence classification task, outperforming many recent strong competitors including transformers (Vaswani et al., 2017), RNNs, state space models, and other convolutional models. In particular, MULTIRESNET outperforms previous purely convolution-based models by more than 10 percentage points.

Surprisingly, our model achieves this new performance benchmark with an almost embarrassingly simple architecture comprised primarily of dilated causal convolutions with length-2 filters shared between tree levels, and levels connected by linear links. In particular, our best model uses 10 MULTIRESBLOCKS, each containing a MULTIRESCONV layer that amounts to 10 layers of tied-weight dilated causal convolutions. Together, that amounts to 100 layers of dilated causal convolutions, but the total number of parameters is only 1.4M. In contrast, the best S4 model uses around 7.9M parameters (Gu et al., 2021), but still underperforms our 5x smaller model (and likewise relies on fancy initializations).

Table 2. Performance of predicting outcomes of list operations in the long ListOps dataset of Tay et al. (2021). Bold indicates the best-performing model and underlines the second best. *SGConv is built to mimic the global convolution interpretation of S4. Results are taken from either the citation or Hasani et al. (2022).

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., 2020)	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., 2022)	<u>62.15</u>
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**

5.2. Hierarchical reasoning on symbolic sequences

In order to test our model's capability of reasoning about hierarchical structures, we conduct experiments on the long ListOps dataset from Tay et al. (2021). The dataset consists of sequences that represent a composition of multiple list operations including `MAX`, `MEAN`, `MED` (median) and `SM` (sum and mod). For example, the input can be

```
[MAX 1 [MAX 2 3 ] 5 6 [MIN 7 8 ] ].
```

The output in this case is 7. This is a far shorter version of the examples in the dataset which have length up to 2048. The prediction is formulated as a ten-way classification problem. We use the train, validation, and test split specified

Table 3. AUROC for ECG multi-label/multi-class classification on the PTB-XL dataset. Bold indicates the best performing model and underline the second best. Results for other models taken from Zhang et al. (2023) and Strodthoff et al. (2021).

Model (AUROC)	All	Diag	Sub-diag	Super-diag	Form	Rhythm
MULTIRESNET (Ours)	**0.938**	0.939	**0.934**	**0.934**	0.897	0.975
Spacetime (Zhang et al., 2023)	0.936	**0.941**	0.933	0.929	0.883	0.967
S4 (Gu et al., 2021)	**0.938**	0.939	0.929	0.931	0.895	**0.977**
InceptionTime (Ismail Fawaz et al., 2020)	0.925	0.931	0.930	0.921	**0.899**	0.953
LSTM (Hochreiter & Schmidhuber, 1997)	0.907	0.927	0.928	0.927	0.851	0.953
Transformer (Vaswani et al., 2017)	0.857	0.876	0.882	0.887	0.771	0.831
Wavelet features (Strodthoff et al., 2021)	0.849	0.855	0.859	0.874	0.757	0.890

by Tay et al. (2021). Following the practice in prior work, we pad the sequences to maximum length (2048) to form minibatches, and average over only the actual inputs when pooling over output sequences. Our model for this experiment has 12 MULTIRESBLOCKS. We found a larger filter size (4) is beneficial for this task [3].

Results are reported in Table 2. MULTIRESNET achieves the overall best result, outperforming all attention-based models, S4 and its diagonal variants. It is only matched by Liquid-S4 which additionally introduced inner-product structures between inputs into state space models.

MULTIRESNET is the first model with small kernel convolutions to achieve competitive performance with state space models on the long ListOps dataset. Although SGConv (Li et al., 2022) is advertised as a convolutional neural network, the model is built is to mimic the global convolution interpretation of S4. Therefore, it shares the same limitation of a sophisticated parameterization and relies on an FFT for computing the large kernel convolution.

5.3. Classifying physiological signals

PTB-XL (Wagner et al., 2020) is a publicly available dataset of electrocardiogram (ECG) time series. The dataset contains 21,837 12-lead ECG recordings for 10 seconds from 18,885 patients. Each recording is labeled with at least one of the 71 ECG labels from the SCP-ECG standard. The dataset is partitioned into six subsets: "all", "diagnostic", "diagnostic subclass", "diagnostic superclass", "form", and "rhythm", each containing a different subset of the full 71 labels. "Diagnostic superclass" is a multi-class classification task while the others are multi-label classification tasks.

We use the 100Hz version of the dataset, where each time series has 1K timesteps and 12 channels. We use the train, validation, and test split specified by Strodthoff et al. (2021). We transferred the architecture and learning rate from our Sequential CIFAR-10 experiments. We tuned the dropout rate on the validation set of "diagnostic superclass" and used the same setting for all subsets afterwards.

We show the results in Table 3. On all six subsets of PTB-XL, our model either attains the best or second-best AUROC out of all models presented. It significantly outperforms a neural network trained on fixed wavelet features (Strodthoff et al., 2021), showing the benefits of combining multiresolution computation with end-to-end learning.

[3]This corresponds to switching from the tree structure of the Haar DWT to those of more general wavelets, such as the Daubechies wavelets (Akansu et al., 2001).

Table 4. Autoregressive generative modeling on CIFAR-10 dataset. Results are reported as bits per dimension (bpd) (lower is better). Bold indicates the best-performing model and underlines the second best. Results are taken from the citation.

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

5.4. Autoregressive generative modeling

In this experiment, we go beyond classification and evaluate our sequence modeling layer on autoregressive generative modeling for CIFAR-10. A detailed description of the settings is provided in Appendix C.4. We used 48 MULTIRESLAYERS with 512 channels and filter size 4. As the nature of task requires more parameters than the former experiments, we added an additional 1×1 convolutional layer after the GLU activation in the residual block.

Table 5. Results of ablation study performed on the long ListOps dataset. We used a smaller model than the one in Sec. 5.2 and report the average performance over three random seeds.

ID	Filters	Initialization	Memory mechanism	Filter size	MultiresConv depth	Accuracy
1	Fixed	wavelet	Fading	2	7	50.15±0.60
2	Trainable	wavelet	Fading	2	7	52.08±0.35
3	Trainable	Xavier	Fading	2	7	51.70±0.16
4	Trainable	Xavier	Fading	2	11	61.07±0.26
5	Trainable	Xavier	Uniform	2	7	51.58±0.57
6	Trainable	wavelet	Fading	4	7	59.23±0.13

We hold out 2K examples from the training set for validation and report test negative log-likelihood (in bits per dimension) with best validation performance. We benchmark our results with a number of previously reported results in Table 4. With no 2D bias and significantly fewer parameters, our model achieved 2.838 bits per dimension on the test set, outperforming 2D convolution-based models by a large margin and even surpasses their improved variant (PixelSNAIL) that leverages both convolution and attention. Our model also outperforms S4, which did not report details of its architecture, but mentioned they used downsampling and a U-Net structure. Our model has no downsampling or U-Net structure, demonstrating its memory efficiency and the ability to capture long-range dependencies. It also beats Image Transformer with 1D attention, and is only outperformed by Sparse Transformer, which additionally leveraged 2D biases and used significantly more parameters.

5.5. Ablation study

We reuse the long ListOps dataset from Sec. 5.2 and conduct six experiments to study the effect of model architecture, hyperparameters and initialization schemes. Each experiment consists of three independent runs with different random seeds. The results are reported in Table 5.

Fixed wavelet filters vs. trainable filters. By comparing Experiment 1 and 2, we can see that training the wavelet filters improves the performance, which justifies our design choice of decoupling the MULTIRESCONV operation from the wavelet transform.

Sensitivity to initialization. Next, we examine the effect of initializing the filters as wavelet filters. From Experiment 2 to 3, we switched from wavelet initialization to standard Xavier initialization of neural networks (Glorot & Bengio, 2010), fixing all the other parts of the model. We do not observe statistically significant differences between the approaches. This demonstrates the advantage of our model over S4-related methods that require careful initialization.

Memory mechanism. We do not notice a statistically significant difference between uniform and resolution fading, though resolution fading provides a simpler implementation.

Importance of receptive fields. Finally, we show that we can significantly improve the performance of this model by increasing either the filter size (Experiment 2 vs. 6) or the depth of the MULTIRESCONV (Experiment 3 vs. 4). We believe this is because both changes increase the receptive field size of the MULTIRESCONV operation, which is particularly important for reasoning tasks like ListOps.

6. Conclusion

We presented MULTIRESLAYER for robust and efficient memorization of long-term patterns in sequential data sources. It takes inspiration from the multiresolution analysis (MRA) literature, building on wavelet decompositions, to memorize patterns occurring at multiple timescales. In particular, our memory is generated by *multiresolution convolution*s, implemented as dilated causal convolutions with *learned* filters shared between tree levels that are connected via purely linear operations. To create the memory, all multiresolution values may be maintained, or more emphasis can be placed on more recent time points by leveraging the time-localized nature of wavelet transforms.

The resulting MULTIRESNET garners the computational advantages of convolutional networks while being defined by dramatically fewer parameters than competitor models, all while achieving state-of-the-art performance in a number of benchmark sequence modeling tasks. These experiments demonstrate the portability of our multiresolution memory structure to a number of tasks, even in cases where a given task may not intuitively be viewed in a multiscale fashion (e.g., syntax tree parsing in ListOps).

By taking inspiration from the wavelet literature, we built an effective convolutional layer with dramatically fewer parameters without taking a performance hit. The principled underpinnings of the MULTIRESCONV ensure it possesses a configuration with strong reconstruction capabilities (e.g., when our filters equal the wavelet filters); however, as we showed, predictive performance can be improved by learning the filters.

Another potential benefit of starting from the wavelet framework is the ability to leverage that vast literature in that

domain for future modeling advances. In particular, we plan to explore the utility of MULTIRESCONV in representation learning and long-term forecasting. For representation learning, we can consider the structure of *shift-invariant wavelet transforms* (Kingsbury, 1998; Selesnick et al., 2005) to target representations that are invariant to shifts of the input signals. For example, we may want to cluster individuals with similar ECG signals even if the key signatures are shifted relative to one another. Wavelets may also be extended to image analysis, enabling video analysis in our sequential setting.

Acknowledgements

This work was supported in part by AFOSR Grant FA9550-21-1-0397, ONR Grant N00014-22-1-2110, the National Science Foundation under grant 2205084, and the Stanford Institute for Human-Centered Artificial Intelligence (HAI). EBF is a Chan Zuckerberg Biohub – San Francisco Investigator. KAW was partially supported by Stanford Data Science as a Stanford Data Science Scholar.

References

Akansu, A. N., Haddad, R. A., and Haddad, P. A. *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic press, 2001.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bai, S., Kolter, J. Z., and Koltun, V. Trellis networks for sequence modeling. In *International Conference on Learning Representations*, 2019.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. PixelSNAIL: An improved autoregressive generative model. In *International Conference on Machine Learning*, pp. 864–872. PMLR, 2018.

Cheng, L., Khalitov, R., Yu, T., Zhang, J., and Yang, Z. Classification of long sequential data using circular dilated convolutional neural networks. *Neurocomputing*, 518:50–59, 2023.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(ARTICLE):2493–2537, 2011.

Daubechies, I. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pp. 933–941. PMLR, 2017.

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

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Iintelligence and Statistics*, pp. 249–256, 2010.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. HiPPO: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33: 1474–1487, 2020a.

Gu, A., Gulcehre, C., Paine, T., Hoffman, M., and Pascanu, R. Improving the gating mechanism of recurrent neural networks. In *International Conference on Machine Learning*, pp. 3800–3809. PMLR, 2020b.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Hasani, R., Lechner, M., Wang, T.-H., Chahine, M., Amini, A., and Rus, D. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

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

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, November 2020. ISSN 1573-756X. doi: 10.1007/s10618-020-00710-y.

Jawerth, B. and Sweldens, W. An overview of wavelet based multiresolution analyses. *SIAM review*, 36(3):377–412, 1994.

Kaiser, L., Gomez, A. N., and Chollet, F. Depthwise separable convolutions for neural machine translation. In *International Conference on Learning Representations*, 2018.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A., and Kavukcuoglu, K. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kingsbury, N. G. The dual-tree complex wavelet transform: a new technique for shift invariance and directional filters. In *IEEE digital signal processing workshop*, volume 86, pp. 120–131. Citeseer, 1998.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Lee, G. R., Gommers, R., Waselewski, F., Wohlfahrt, K., and O'Leary, A. PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software*, 4 (36):1237, April 2019. ISSN 2475-9066. doi: 10.21105/joss.01237.

Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. Fnet: Mixing tokens with fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4296–4313, 2022.

Li, Y., Cai, T., Zhang, Y., Chen, D., and Dey, D. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*, 2022.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Ma, X., Kong, X., Wang, S., Zhou, C., May, J., Ma, H., and Zettlemoyer, L. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34: 2441–2453, 2021.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Oord, A. v. d., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems*, 29, 2016b.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756. PMLR, 2016c.

Romero, D. W., Bruintjes, R., Bekkers, E. J., Tomczak, J. M., Hoogendoorn, M., and van Gemert, J. FlexConv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a.

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Selesnick, I. W., Baraniuk, R. G., and Kingsbury, N. C. The dual-tree complex wavelet transform. *IEEE signal processing magazine*, 22(6):123–151, 2005.

Shen, D., Min, M. R., Li, Y., and Carin, L. Learning context-sensitive convolutional filters for text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1839–1848, 2018.

Smith, J. T., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Strang, G. and Nguyen, T. *Wavelets and filter banks*. SIAM, 1996.

Strodthoff, N., Wagner, P., Schaeffter, T., and Samek, W. Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL. *IEEE Journal of Biomedical and Health Informatics*, 25(5):1519–1528, May 2021. ISSN 2168-2208. doi: 10.1109/JBHI.2020.3022989.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021.

Trinh, T., Dai, A., Luong, T., and Le, Q. Learning longer-term dependencies in RNNs with auxiliary losses. In *International Conference on Machine Learning*, pp. 4965–4974. PMLR, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Wagner, P., Strodthoff, N., Bousseljot, R.-D., Kreiseler, D., Lunze, F. I., Samek, W., and Schaeffter, T. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, May 2020. ISSN 2052-4463. doi: 10.1038/s41597-020-0495-6.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Willsky, A. S. Multiresolution Markov models for signal and image processing. *Proceedings of the IEEE*, 90(8): 1396–1458, 2002.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2018.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zhang, M., Saab, K. K., Poli, M., Dao, T., Goel, K., and Re, C. Effectively modeling time series with simple discrete state spaces. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=2EpjkjzdCAa.

Zhu, Z. and Soricut, R. H-transformer-1d: Fast one-dimensional hierarchical attention for sequences. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3801–3815, 2021.

A. Background: Wavelet Theory

In this section, we provide a formal treatment of the background on wavelet theory, expanding upon the introduction presented in Sec. 2. We begin by going through the derivation of Haar wavelets using the multiresolution analysis (MRA) framework. We then demonstrate how the same framework can be leveraged to derive a wide array of orthogonal wavelets.

A.1. Haar wavelets

We first introduce the definition of Haar *scaling function*. The scaling function is also known as the mother wavelet.

Definition A.1 (Haar scaling function)**.** The Haar scaling function is defined as

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

Let V_0 be the space spanned by translations of the scaling function:

$$V_0 \triangleq \left\{ s_0 \middle| s_0(t) = \sum_{k\in\mathbb{Z}} a_{0,k}\phi(t-k), a_{0,k} \in \mathbb{R} \right\}.$$

It contains functions that are constant over intervals of length 1. Next, we construct a larger space V_1 that subsumes V_0 by dividing the unit interval into halves:

$$V_1 \triangleq \left\{ s_1 \middle| s_1(t) = \sum_{k\in\mathbb{Z}} a_{1,k}\sqrt{2}\phi(2t-k), a_{1,k} \in \mathbb{R} \right\}.$$

The coefficient $\sqrt{2}$ is introduced such that the basis function $\sqrt{2}\phi(2t-k)$ is normalized, i.e., $\int_{\mathbb{R}} \sqrt{2}\phi(2t-k)\ dt = 1$. These functions are constant over intervals of length $1/2$. We could repeat such a process and create a sequence of function spaces for $j \geq 0$:

$$V_j \triangleq \left\{ s_j \middle| s_1(t) = \sum_{k\in\mathbb{Z}} a_{j,k}\phi_{j,k}(t), a_{j,k} \in \mathbb{R} \right\}, \text{ where } \phi_{j,k}(t) \triangleq 2^{j/2}\phi(2^j t - k). \tag{6}$$

These function spaces come with increasing richness and expressive power:

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_j.$$

It is easy to see that V_j with a sufficiently large j will have the capacity to approximate any continuous time series f with arbitrary precision. Let $\hat{f}^{(j)}(t) = \sum_{k\in\mathbb{Z}} a_{j,k}\phi_{j,k}(t)$ denote the approximation of f in V_j. The best approximation is given by the coefficients

$$a_{j,k} = \int_{\mathbb{R}} f(t)\phi_{j,k}(t)\ dt = \int_{k/2^j}^{(k+1)/2^j} f(t)\ dt. \tag{7}$$

One can represent f with the vector of coefficients $\{a_{j,k}\}_{k\in\mathbb{Z}}$. However, as we have explained in Sec. 2, each individual coefficient is too localized to capture patterns over larger intervals. One way to remedy this is applying MRA to generate representations that summarize patterns of f at multiple different timescales, resulting in Haar wavelets.

The wavelet MRA has the following steps. First, observing that $V_{j-1} \subset V_j$, we decompose V_j into the sum of V_{j-1} and its orthogonal complement W_{j-1}. Repeating this process, we get

$$V_j = V_{j-1} \oplus W_{j-1} = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{j-1}. \tag{8}$$

Because the functions in V_j and V_{j-1} represent structures at timescales coarser than $1/2^j$ and $1/2^{j-1}$, respectively, we can expect the basis functions in W_{j-1} uniquely represent structures at the $1/2^j$ timescale. The next step is to find an orthonormal basis for W_{j-1}. Theorem A.3 shows that this basis can be obtained from rescaling and translations of a function ψ, the collection of functions known as the *Haar wavelets*.

Definition A.2 (Haar wavelets)**.** The Haar wavelets are a family of functions defined as

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \text{ where } \psi(t) \triangleq \phi(2t) - \phi(2t-1). \tag{9}$$

Theorem A.3 (Haar wavelets as orthonormal bases)**.** *Let V_j be defined as in (6) and W_{j-1} be the orthogonal complement of V_{j-1} in V_j. Then, $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ is an orthonormal basis for W_j.*

Proof. We first notice that $\psi_{j,k} \in V_{j+1}$ and $\langle \psi_{j,k}, \psi_{j',k'} \rangle = \delta_{j,j'}\delta_{k,k'}$. To show the result, we need to prove (1) $\psi_{j,k} \in W_j$, i.e., $\psi_{j,k}$ is orthogonal to any function in V_j, and (2) any function in W_j can be represented with $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$. For the first requirement, we have that, for any $k' \in \mathbb{Z}$,

$$\langle \psi_{j,k}, \phi_{j,k'} \rangle = \langle \frac{1}{\sqrt{2}}(\phi_{j+1,2k} - \phi_{j+1,2k+1}), \frac{1}{\sqrt{2}}(\phi_{j+1,2k'} + \phi_{j+1,2k'+1}) \rangle = \frac{1}{2}(\delta_{k,k'} - \delta_{k,k'}) = 0.$$

Therefore, $\psi_{j,k}$ is orthogonal to any function in V_j, which implies $\psi_{j,k} \in W_j$. To prove (2), we notice that for some $s_{j+1} = \sum_{k\in\mathbb{Z}} a_{j+1,k}\phi_{j+1,k} \in V_{j+1}$, $s_{j+1} \in W_j$ implies for any $k' \in \mathbb{Z}$,

$$\begin{aligned}
\langle s_{j+1}, \phi_{j,k'} \rangle &= \sum_{k\in\mathbb{Z}} a_{j+1,k} \langle \phi_{j+1,k}, \phi_{j,k'} \rangle \\
&= \sum_{k\in\mathbb{Z}} a_{j+1,k} \langle \phi_{j+1,k}, \frac{1}{\sqrt{2}}(\phi_{j+1,2k'} + \phi_{j+1,2k'+1}) \rangle \\
&= \frac{1}{\sqrt{2}}(a_{j+1,2k'} + a_{j+1,2k'+1}) = 0.
\end{aligned}$$

Therefore, we can write any $s_{j+1} \in W_j$ using $a_{j+1,2k+1} = -a_{j+1,2k}$ as

$$s_{j+1} = \sum_{k\in\mathbb{Z}} a_{j+1,2k}(\phi_{j+1,2k} - \phi_{j+1,2k+1}) = \sum_{k\in\mathbb{Z}} \sqrt{2} a_{j+1,2k}\psi_{j,k}.$$

Thus, any function in W_j can be represented with $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$. Combining (1) and (2) proves the original statement. □

Now that we have the wavelet basis, we can rewrite the approximation of f in V_j to reflect the decomposition in (8):

$$f(t) \approx \hat{f}^{(j)}(t) = \sum_{k\in\mathbb{Z}} a_{0,k}\phi(t-k) + \sum_{j'=0}^{j-1}\sum_{k'\in\mathbb{Z}} b_{j',k'}\psi_{j',k'}(t), \tag{10}$$

where $\{\phi(t-k)\}_{k\in\mathbb{Z}}$ is the basis of V_0 and $\{\psi_{j',k'}\}_{k'\in\mathbb{Z}}$ is the basis of $W_{j'}$ for $j' = 0, 1, \cdots, j-1$. Therefore, we can summarize the multiresolution structure of f using the coefficients $\{a_{0,k}\}_{k\in\mathbb{Z}}$ and $\{b_{0,k}\}_{k\in\mathbb{Z}} \cup \cdots \cup \{b_{j-1,k}\}_{k\in\mathbb{Z}}$.

A.2. Generalization to all orthogonal wavelets

The multiresolution analysis framework described in the previous section can be generalized beyond Haar wavelets to all orthogonal wavelets. The starting point is that we can use a different scaling function than the Haar scaling function that satisfies the following equation:

$$\phi(t) = \sum_{k=0}^{K-1} \mathbf{h}_0(k)\sqrt{2}\phi(2t-k). \tag{11}$$

This condition guarantees that the associated function spaces satisfy $V_{j-1} \subset V_j$. The sequence $\mathbf{h}_0(\cdot) \in \mathbb{R}^K$ here is referred to as a (low-pass) *filter*, with potentially different lengths for the various wavelets. For example, the Haar wavelet filter is a 2D vector with $\mathbf{h}_0(0) = \mathbf{h}_0(1) = \frac{1}{\sqrt{2}}$. The Daubechies-4 wavelet (Daubechies, 1988) filter has 4 elements:

$$\left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}\right).$$

Similarly, we have the following generalized form of the mother wavelet:

$$\psi(t) = \sum_{k=0}^{K-1} \mathbf{h}_1(k)\sqrt{2}\phi(2t-k), \tag{12}$$

where $\mathbf{h}_1(\cdot)$ is known as the *high-pass* filter. Using the orthogonality between $\phi(t-k)$ and $\psi(t)$ for any k, one can show (see, e.g., Strang & Nguyen, 1996):

$$\mathbf{h}_1(k) = (-1)^k \mathbf{h}_0(K-1-k),$$

where N is the length of the filter $\mathbf{h}_0$. For Haar wavelets, $\mathbf{h}_1(0) = \mathbf{h}_0(1) = \frac{1}{\sqrt{2}}$, $\mathbf{h}_1(1) = -\mathbf{h}_0(0) = -\frac{1}{\sqrt{2}}$. For Daubechies-4 wavelets, the corresponding $\mathbf{h}_1$ filter is

$$\left(\frac{1-\sqrt{3}}{4\sqrt{2}}, -\frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, -\frac{1+\sqrt{3}}{4\sqrt{2}}\right).$$

In this work, except for ablation purposes, we do not fix the values of the filters $(\mathbf{h}_0, \mathbf{h}_1)$ or enforce coupling between them. Instead, we allow both filters to be learned via gradient descent. We also found that initializing the filters with Haar or Daubechies wavelets does not provide advantages over standard Xavier initialization. This demonstrates the ability of the model to discover effective parameterizations through optimization.

A.3. Discrete wavelet transform

We have already shown that a continuous time series can be represented with a vector of coefficients that capture its structure at multiple different timescales. The remaining challenge is determining these coefficients from a discretely sampled signal $\mathbf{x} \in \mathbb{R}^N$. From (7), we know that each coefficient $a_{j,k}$ summarizes the average value of the time series in a $1/2^j$ interval. Therefore, if we choose a sufficiently large J, the coefficients $\mathbf{a}_J$ can be directly approximated with the discretely sampled $\mathbf{x}$. We can obtain the coefficients needed for the wavelet expansion (10) recursively from $\mathbf{a}_J$ using a process known as the discrete wavelet transform (DWT).

To derive the DWT recursion, we start from $s_{j+1} = \sum_m a_{j+1,m}\phi_{j+1,m} \in V_{j+1}$ and compute the coefficents for its decomposition in V_j and W_j:

$$s_{j+1} = \sum_n a_{j,n}\phi_{j,n} + \sum_\ell b_{j,\ell}\psi_{j,\ell}.$$

Due to the orthogonality of $\{\phi_{j,n}\}_{n\in\mathbb{Z}}$ and $\{\psi_{j,n}\}_{n\in\mathbb{Z}}$,

$$\begin{aligned}
a_{j,n} &= \langle s_{j+1}, \phi_{j,n}\rangle \\
&= \langle \sum_m a_{j+1,m}\phi_{j+1,m}, \sum_k \mathbf{h}_0(k)\phi_{j+1,2n+k}\rangle \\
&= \sum_k a_{j+1,2n+k}\mathbf{h}_0(k), \text{ and similarly,} \\
b_{j,\ell} &= \langle s_{j+1}, \psi_{j,\ell}\rangle = \sum_k a_{j+1,2\ell+k}\mathbf{h}_1(k).
\end{aligned}$$

The DWT is repeating the above computations from $j = J$ to 0. We can see each step of this process is a convolution between $\mathbf{a}_{j+1}$ and the filter ($\mathbf{h}_0(-\cdot)$ or $\mathbf{h}_1(-\cdot)$), and then applying a downsampler ($\mathbf{y}(n) = \mathbf{x}(2n)$). Note that although the convolution is done after a negation of the filter $\mathbf{h}_0, \mathbf{h}_1$, this is exactly equivalent to a 1-dimensional convolutional layer in neural networks with the convolution kernel $\mathbf{h}_0$ or $\mathbf{h}_1$, where the convolution operation is standard convolution plus negation of the filter.

B. MultiresLayer Implementation

We provide an example PyTorch implementation of a MULTIRESLAYER with the "resolution fading" TreeSelect strategy (see Sec. 3.2) in Fig. 3.

```python
import math
from torch.nn.functional import pad, conv1d

def multires_layer(x, h0, h1, w, depth=None):
    """
    Args:
        x: input of shape (batch_size, n_channels, sequence_length).
        h0, h1: convolution filters of shape (n_channels, 1, filter_size).
        w: weights of the linear layer after TreeSelect. Shape: (n_channels, depth + 2).
        depth: depth of MultiresConv, i.e., J.

    Returns:
        y: output of shape (batch_size, n_channels, sequence_length).
    """
    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
```

Figure 3. PyTorch code for implementing a MULTIRESLAYER with the "resolution fading" TreeSelect strategy.

C. Experiment Details

C.1. Sequential CIFAR-10 classification

The inputs are sequences of length 1024 with three channels corresponding to RGB values. We rescaled and centered the inputs between $[-1, 1]$, and used a 1×1 convolutional layer to encode them into 256 channels. For this task, we used 10 MULTIRESBLOCKS, each with filter size 2, 256 channels, and the resolution fading memory mechanism. They were followed by a mean-pooling over timesteps and a linear layer to generate classification logits. We trained the network for 250 epochs with batch size 50. We used the AdamW optimizer (Loshchilov & Hutter, 2018) with default hyperparameters and a weight decay rate 0.01. We use a dropout rate 0.25, layer normalization and an initial learning rate 0.0045. The learning rate followed a cosine annealing procedure (Loshchilov & Hutter, 2017).

C.2. ListOps

The inputs are sequences of token IDs (integers) with maximum length 2048. We padded all sequences to this maximum length and used an embedding layer to encode them into 128 channels. We used 10 MULTIRESBLOCKS, each with filter size 4, 128 channels, and the resolution fading memory mechanism. We performed mean-pooling only on timesteps that were actual inputs to generate classification logits. We trained the network for 100 epochs with batch size 50. We used the AdamW optimizer with a weight decay rate 0.03. The learning rate was set to 0.003 after 1 epoch of linear warmup and then followed by a cosine annealing. We used a dropout rate 0.1 and batch normalization instead of layer normalization in this experiment.

C.3. PTB-XL

The inputs are time series with 1000 timesteps and 12 channels. We transferred the architecture and learning rate from our Sequential CIFAR-10 experiments. We used layer normalization, a dropout rate 0.2, and the AdamW optimizer with weight decay rate 0.06. We trained the network for 5 epochs of linear warmup followed by 95 epochs of cosine learning rates.

C.4. Autoregressive generative modeling

CIFAR-10 images were reshaped into sequences of length 1024, each with three channels. Let $x_i \in \mathbb{R}^3$ denote the RGB values (rescaled and centered between $[-1, 1]$) of the i-th pixel in the sequence. To model the conditional distributions $p(x_i|x_1, \cdots, x_{i-1})$ for $i = 1, 2, \cdots, 1024$, the inputs of the network were chosen as $(0, x_1, x_2, \cdots, x_{1023})$, with the target output sequence being the parameters for distributions of $x_1, x_2, \cdots, x_{1024}$. We used the same discretized mixture of Logistics distribution and linear correlation structure from Salimans et al. (2017) to model the RGB values. We used 48 MULTIRESLAYERS with filter size 4, 512 channels, and the resolution fading memory mechanism. Additionally, we added a 1x1 convolutional layer close to the output end of the residual block. The causal structure of MULTIRESLAYER ensures that the i-th output only depends on the first i input elements, which keeps the conditional dependency pattern required for autoregressive modeling. We trained the network for 250 epochs with batch size 64. We used Adam optimizer with default hyperparameters. We used a dropout rate 0.1, layer normalization, and an initial learning rate 0.001. The learning rate followed a cosine annealing procedure.

D. Additional Results

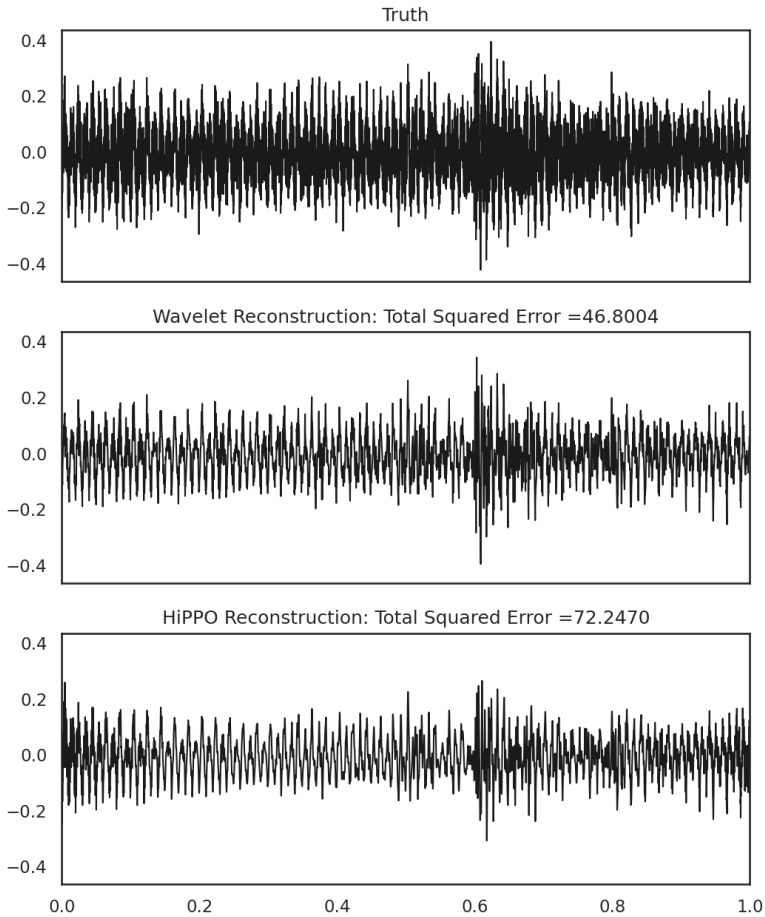

Figure 4. Audio signal reconstruction from wavelet and polynomial bases. *Top:* 1 second of a raw audio waveform with 16,384 time points. *Middle:* Reconstruction from a 10-level Daubechies-4 wavelet decomposition with 2068 coefficients. *Bottom:* Reconstruction from HiPPO (Gu et al., 2020a) via projection to orthogonal polynomials using 2068 coefficients (see Sec. 4).